

# **DASAR PEMROGRAMAN**

**PYTHON**



**UNIT PENGEMBANG AKADEMIK**

## Capaian Pembelajaran Matakuliah

### Dasar Pemrograman :

1. Mahasiswa mampu Menganalisis Tools
2. Mahasiswa mampu membuat Dokumen Kode Program
3. Mahasiswa mampu melakukan Debugging
4. Mahasiswa Menulis Kode Dengan Prinsip Sesuai Guidelines dan Best Practice
5. Mahasiswa Menerapkan Pemecahan Permasalahan Menjadi Subrutin

#### Mahasiswa mampu Menganalisis *Tools*

Unit ini menentukan kompetensi, pengetahuan dan sikap kerja yang diperlukan untuk menganalisis tools yang diperlukan untuk mengembangkan perangkat lunak aplikasi sesuai dengan kebutuhan.

#### Mahasiswa mampu membuat dokumen kode program

Kompetensi ini berhubungan dengan sikap, pengetahuan, dan keterampilan yang Diperlukan untuk membuat dokumentasi dari kode program yang telah ditulis secara hardcopy termasuk identifikasi penjelasan dari dokumen tersebut.

#### Mahasiswa mampu melakukan Debugging

Unit kompetensi ini berhubungan dengan sikap, pengetahuan, dan keterampilan yang dibutuhkan dalam memeriksa kode program dari kesalahan (bug).

#### Mahasiswa menulis kode dengan prinsip sesuai Guidelines dan Best Practice

Menentukan kompetensi, pengetahuan dan Sikap kerja yang diperlukan dalam menerapkan Prinsip penulisan kode yang baik agar kode tersebut dapat dirawat (maintainability).

#### Mahasiswa menerapkan pemecahan Permasalahan menjadi Subrutin

Kompetensi ini berhubungan dengan sikap, pengetahuan, dan keterampilan yang dibutuhkan dalam memecah permasalahan menjadi permasalahan –permasalahan yang lebih kecil dan menyelesaikan permasalahan lebih kecil tersebut berupa fungsi, prosedur, library, atau representasi yang lain sesuai paradigma bahasa pemrograman yang digunakan.

## Uji Kompetensi

1. Individu
2. Final Project (Berkelompok)

### Uji Kompetensi Individu

1. Untuk Matakuliah Dasar Pemrograman Tidak Ada UTS Dan UAS, digantikan Uji kompetensi (Individu dan Final Project)
2. Uji Kompetensi Individu dilaksanakan pada pertemuan 12. Masing-masing mahasiswa diminta mengerjakan soal yang sudah ditentukan. Wajib membawa laptop.

### Uji Kompetensi (Final Project)

1. Final Project dilakukan di pertemuan 13-15 dengan ketentuan sebagai berikut:
  - a. Isi dari final project :
    - Nilai Running Program diambil berdasarkan: (Logika Program, oop, Debuging, penulisan Kode Program, Tampilan output Program)
  - b. Masing-masing kelompok membuat paper laporan pembuatan final project
  - c. Program, Paper dan Presentasi di Burning Kedalam CD
  - d. Masing-masing kelompok mempresentasikan hasil final projectnya.
  - e. Presentasi disajikan dengan media presentasi yang isinya berupa alur logika program dan eksekusi running program.
  - f. Penilaian di tentukan oleh dosen pengajar di ruang kelas.
2. Tema Project di serahkan ke dosen pengajar di Pertemuan ke 2
3. Project sudah bisa di kerjakan setelah di lakukan penyerahan tema kepada dosen pengajar
4. Penilaian dilakukan oleh dosen pengajar ketika presentasi

### Tema Project UAS :

1. Berbasis Bisnis (Optional) Contoh :
  - a. Penjualan dan Pembelian
  - b. Pengadaan barang
2. Berbasis Science
  - a. Science (Bidang Matematika, Fisika, Kimia atau IPA)
  - b. Animasi Edukasi
  - c. Berbasis Kesehatan (Diagnosa Penyakit)
3. Kreatifitas tampilan
4. Tema Harus Menarik
5. Penilaian di tentukan oleh Dosen Pengajar.

## Pertemuan 1

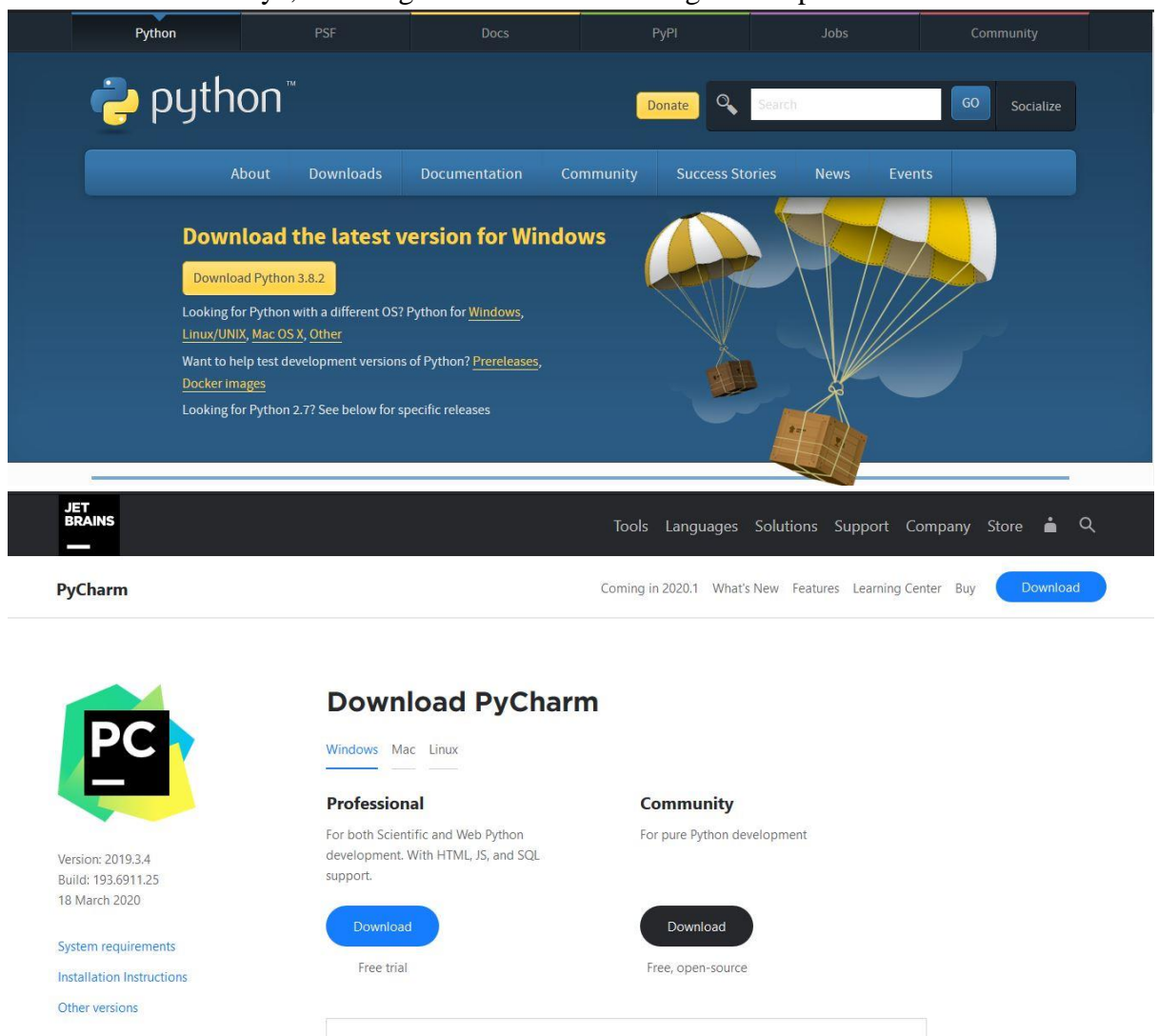
1. Instalasi Program Phyton
2. Instalasi Program PyCharm
3. Sejarah perkembangan Phyton
4. Pengenalan Phyton
5. Pengenalan Struktur Program Phyton
6. Pengenalan Model Data

## Link Download Program

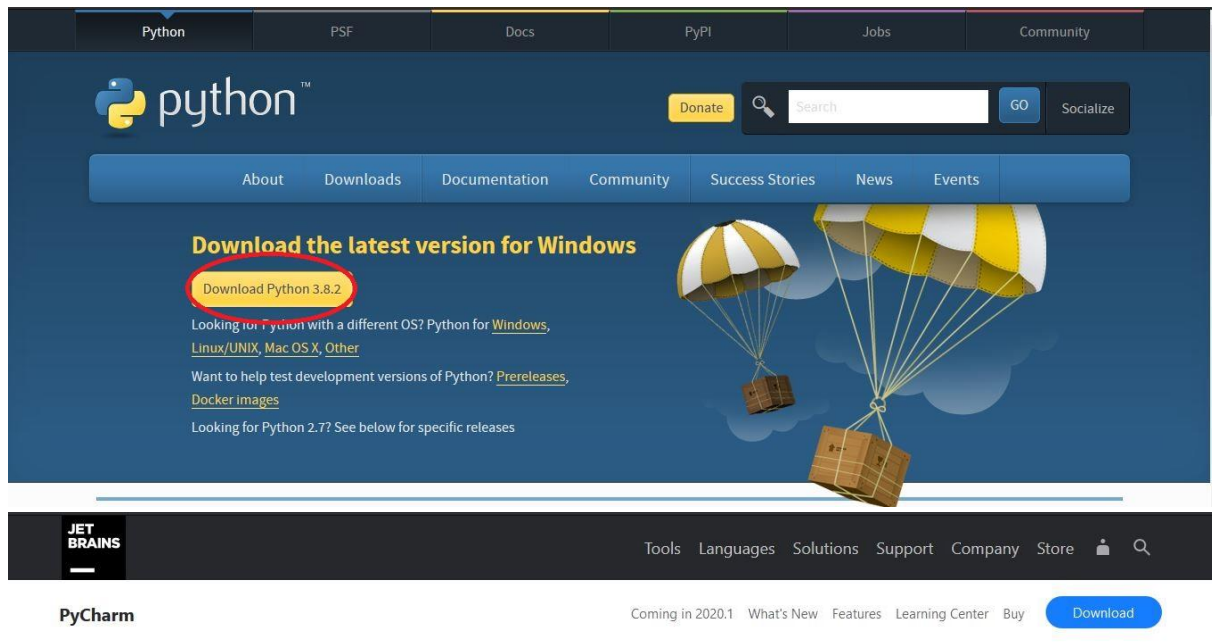
- Link Download Phyton <http://www.pyhton.org/download/>
- Link Download PyCharm <http://www.jetbrains.com/pycharm/download/>

## Cara install Phyton 3.8.2 dan PyCharm 5.8.0

1. Siapkan Laptopnya, Bisa menggunakan windows 7, 8 atau windows 10, untuk proses instalasi nya tidak berbeda.
2. Master Pyhton / Aplikasi mentahan bisa di download melalui halaman : <http://www.pyhton.org/download/>
3. Master Pyhton / Aplikasi mentahan bisa di download melalui halaman : <http://www.jetbrains.com/pycharm/download/>
4. Jika sudah klik linknya, berikut gambar untuk cara mengunduh aplikasi tersebut.

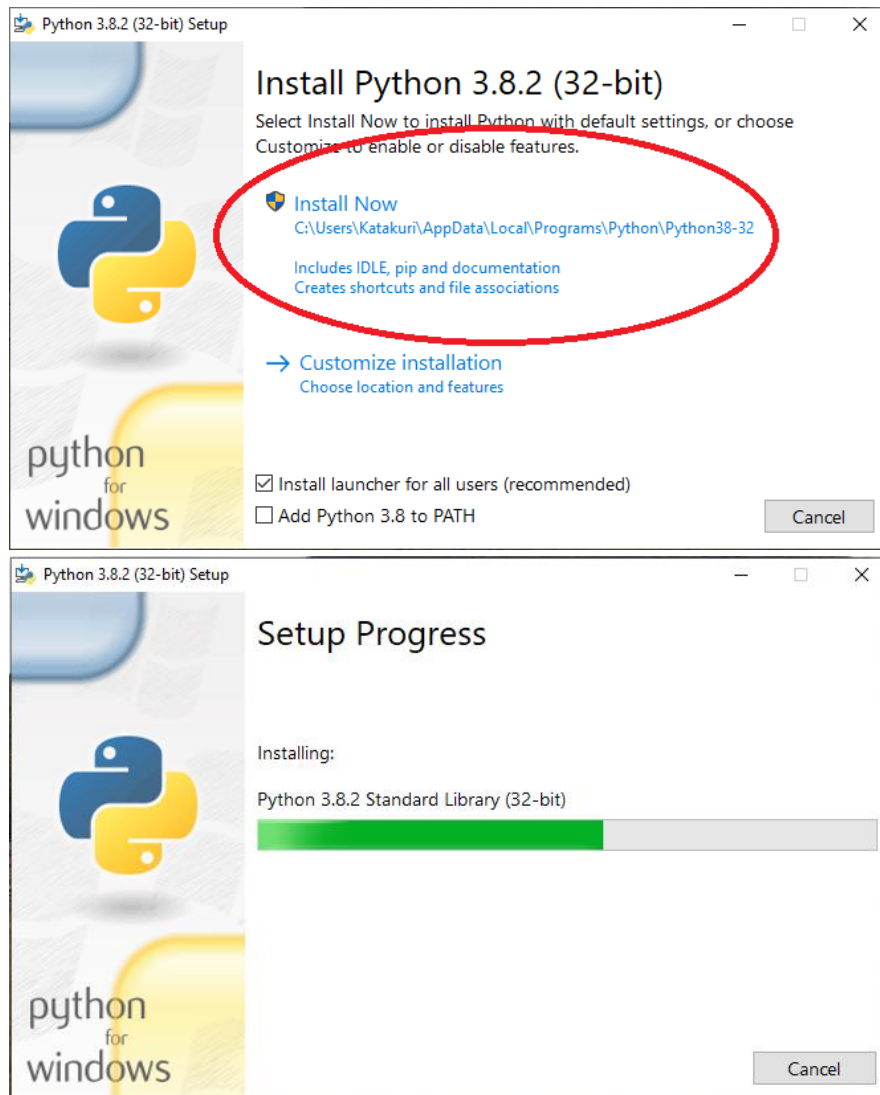


5. Selanjutnya klik tombol download seperti pada gambar berikut :

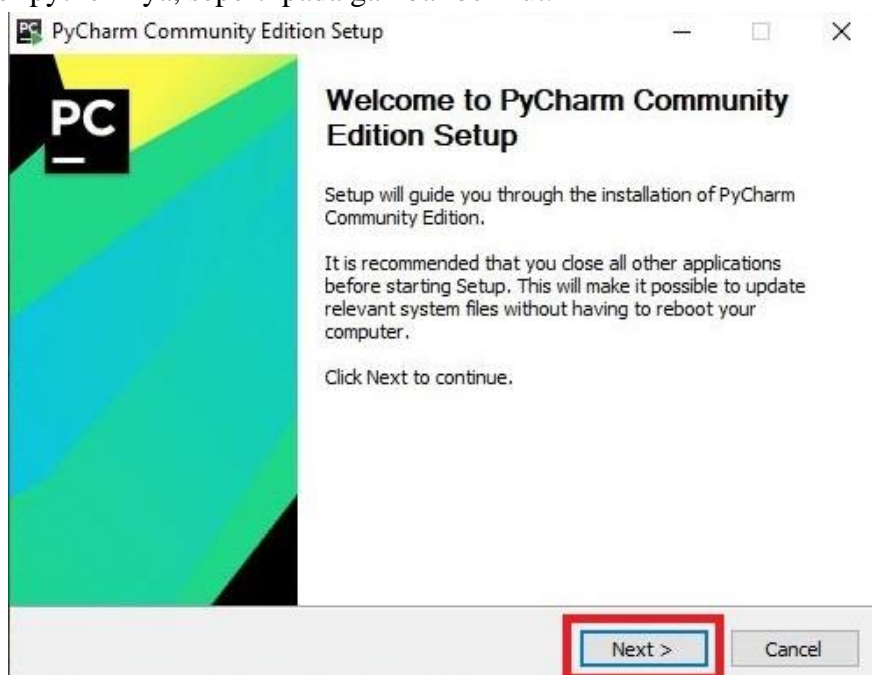


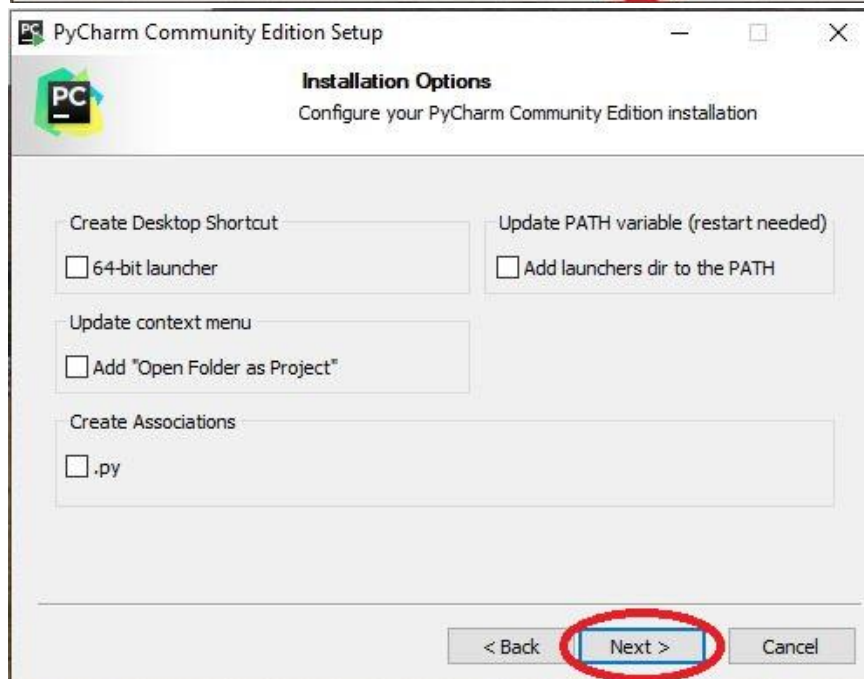
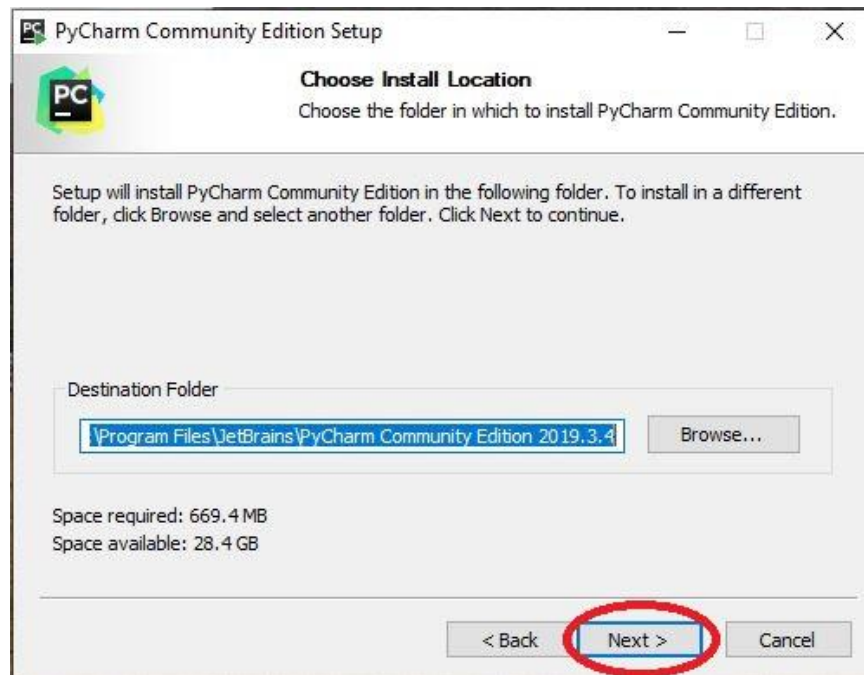
6. Setelah selesai mengunduh kedua aplikasi tersebut, kita double click aplikasi Phyton yang sudah kita unduh terlebih dahulu untuk proses instalasi, seperti pada gambar berikut :



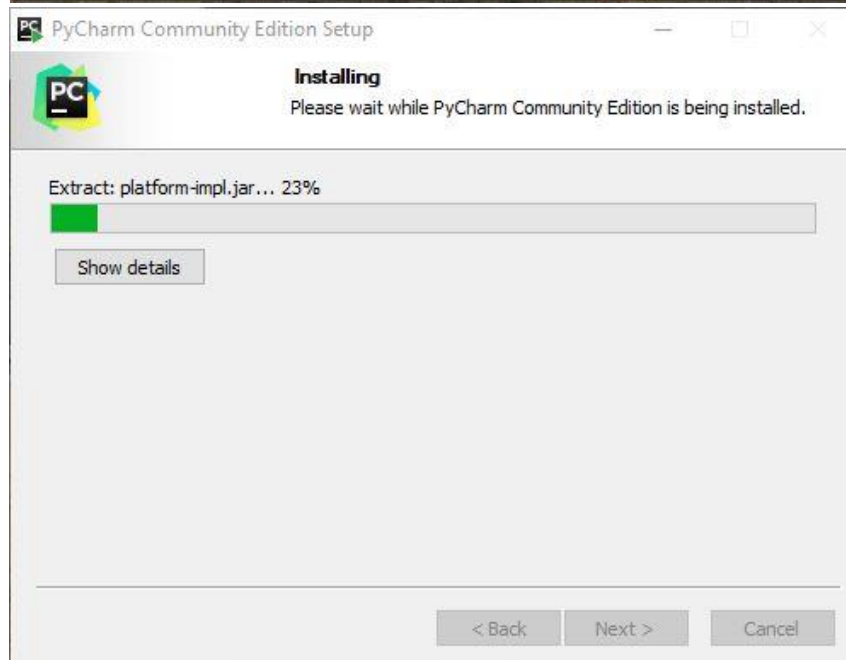
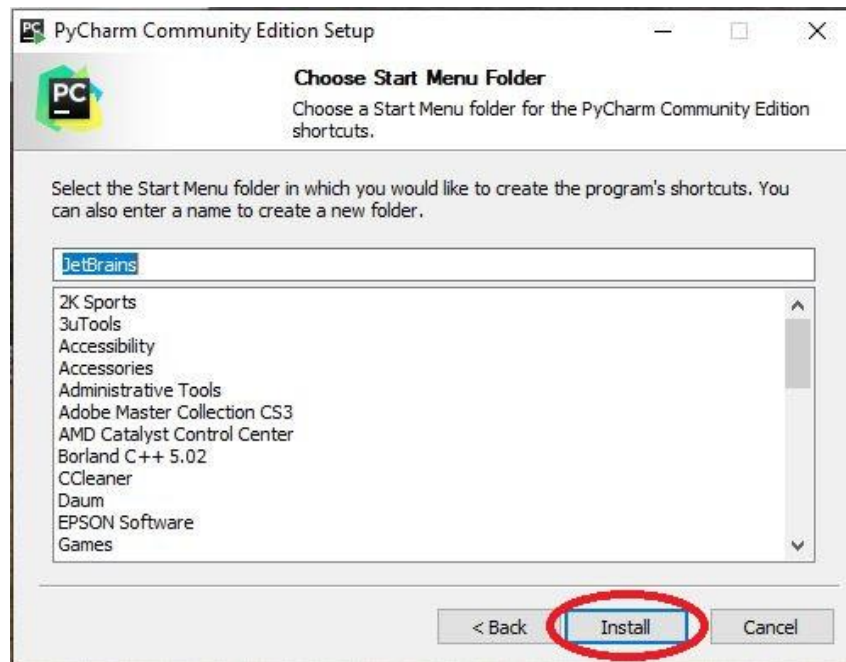


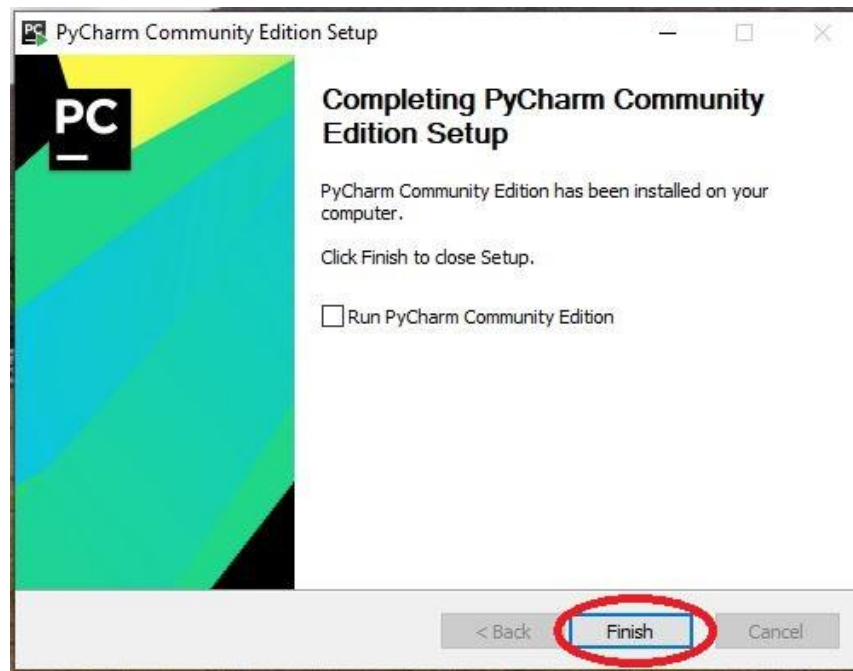
7. Setelah selesai proses instalasi Python, berikutnya kita install PyCharm untuk aplikasi text editor python nya, seperti pada gambar berikut:



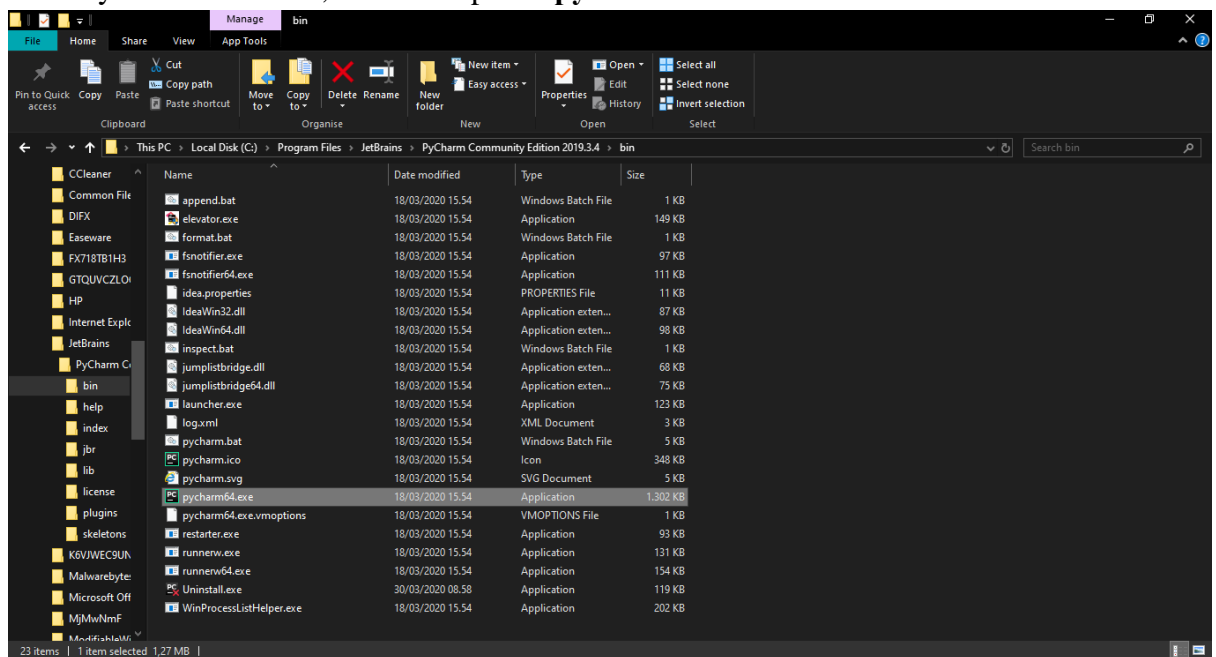




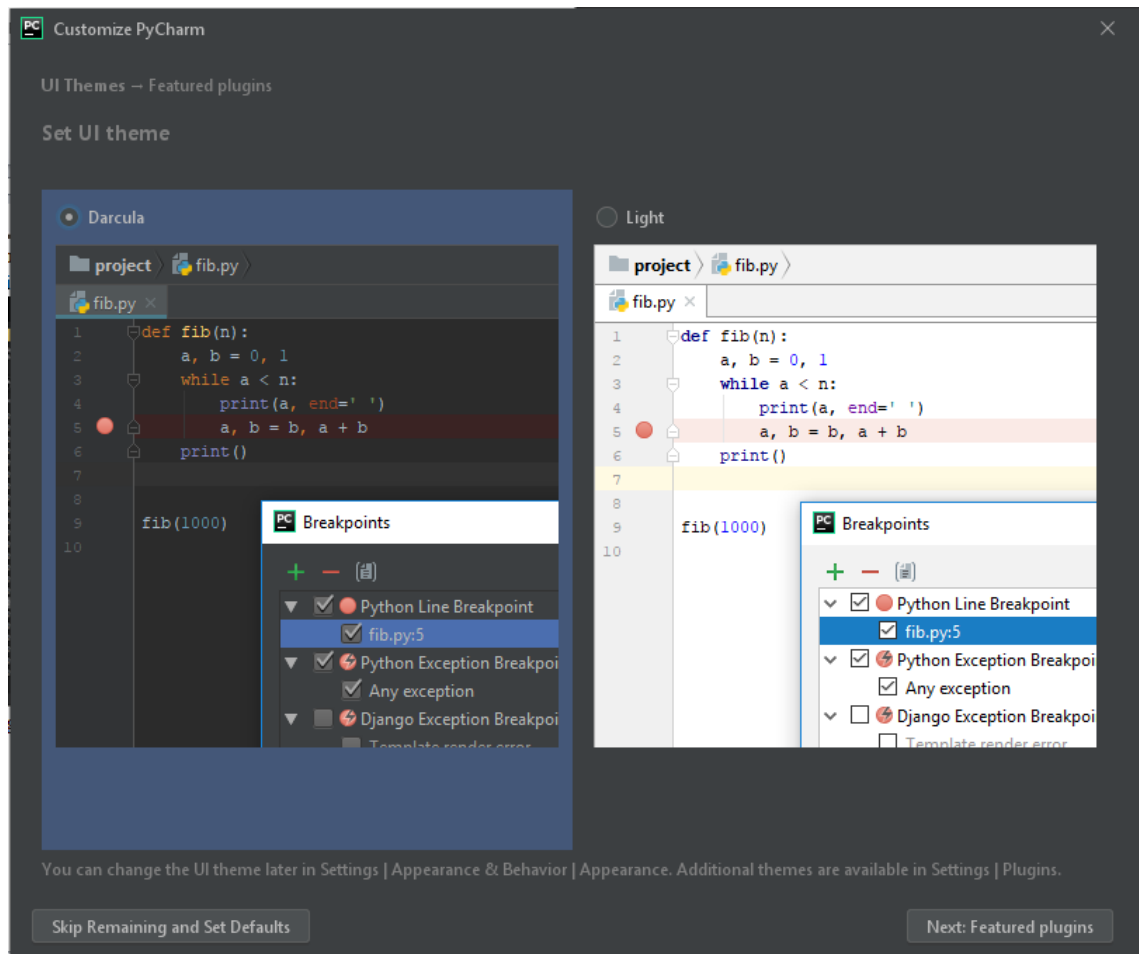




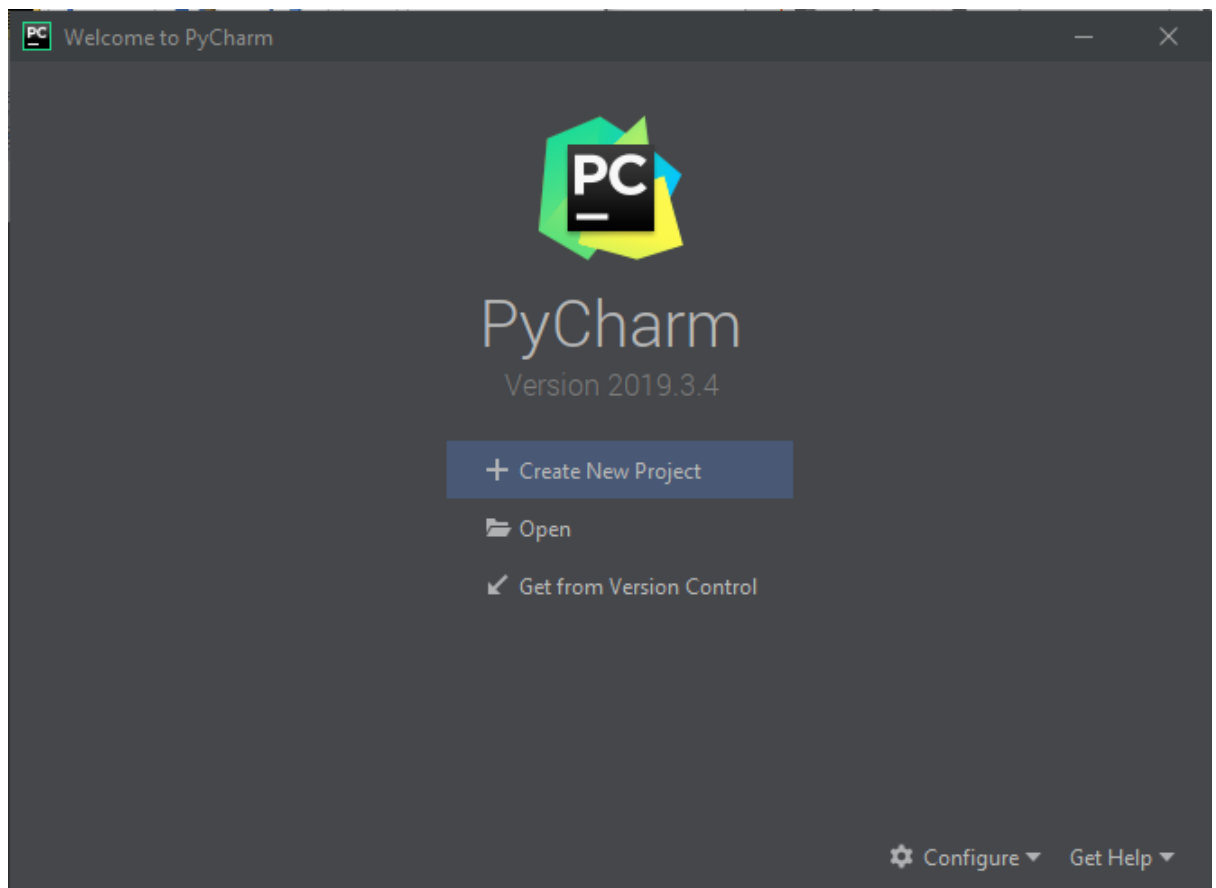
8. Setelah instalasi berhasil, jika ingin mengoperasikan PyCharm, kita pergi ke folder dimana PyCharm terinstall, kemudian pilih **“pycharm64.exe”**



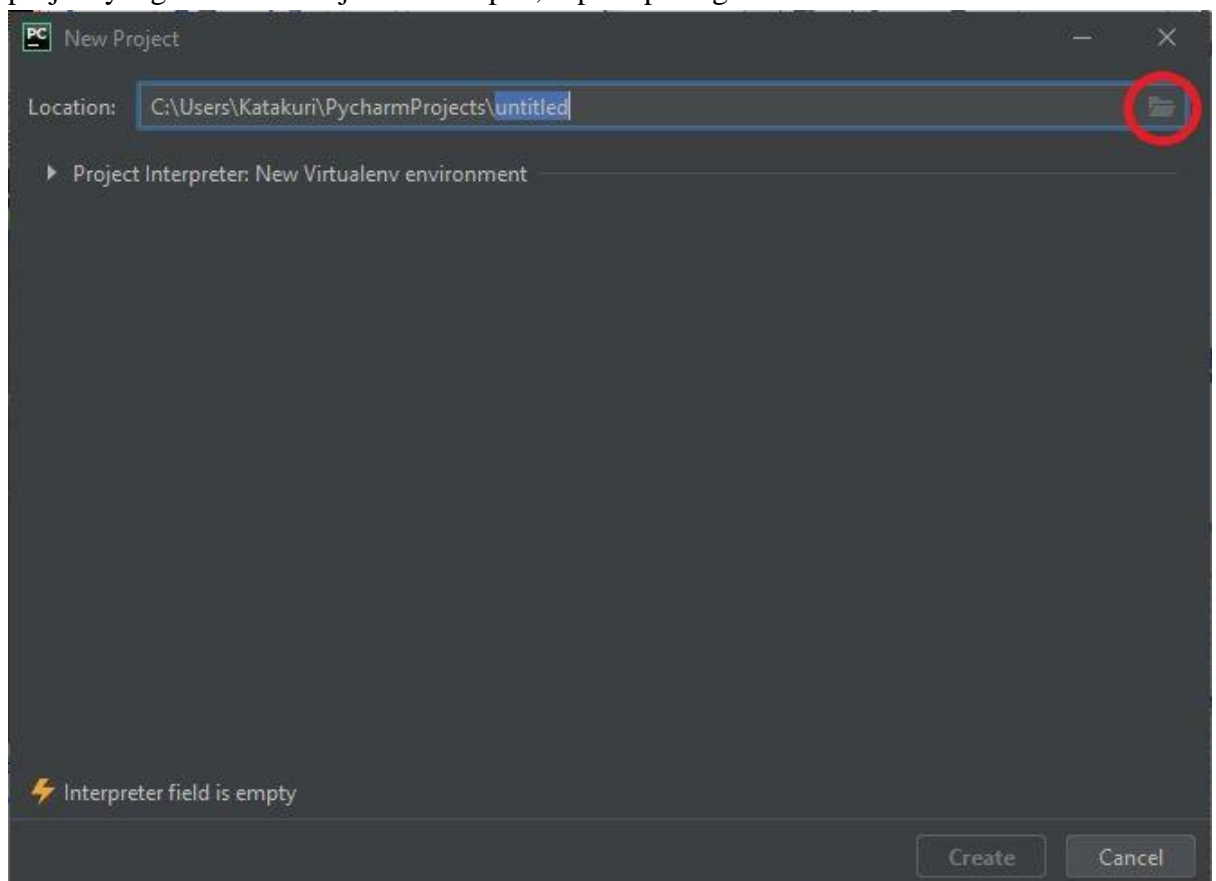
9. Sebelum aplikasi terbuka kita juga bisa menyesuaikan tema PyCharm sesuai yang kita inginkan, seperti gambar berikut :



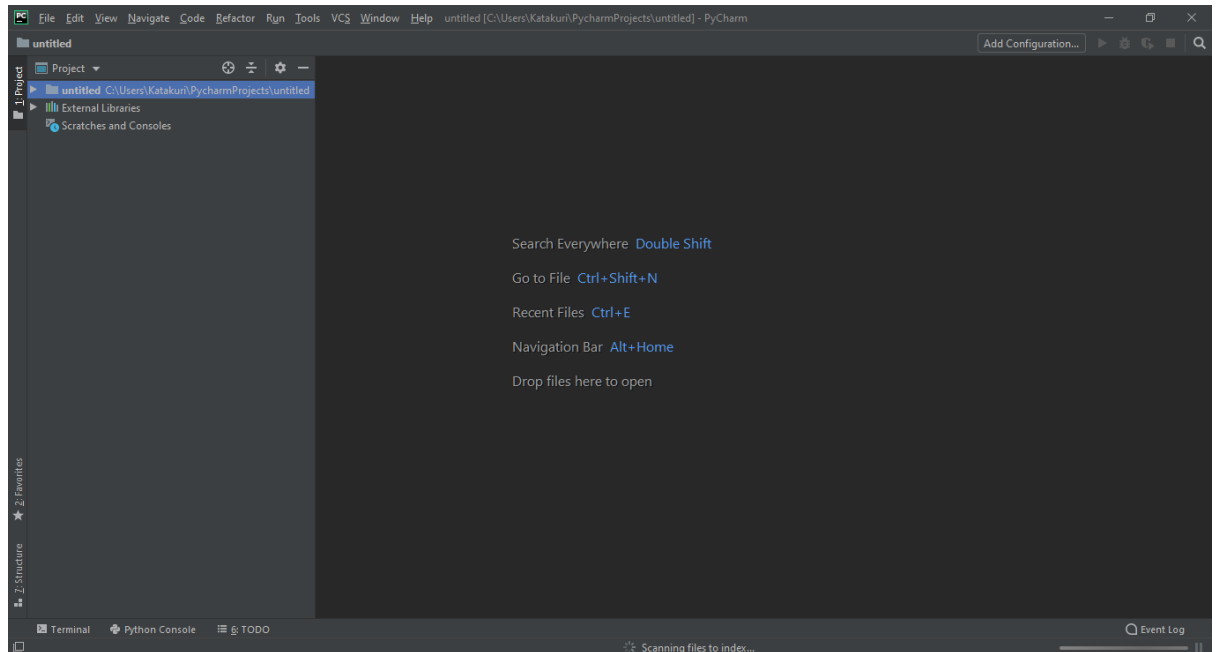
10. Kemudian pilih “Create new project” untuk memulai menggunakan aplikasi PyCharm, seperti gambar berikut :



11. Setelah kita klik “Create new project”, kita harus atur terlebih dahulu di folder mana project yang akan kita kerjakan disimpan, seperti pada gambar berikut :



12. Setelah ditentukan tempat penyimpanan projek nya, selanjutnya klik tombol “create”, dan seperti inilah tampilah PyCharm



13. Untuk membuat file baru, kita bisa menggunakan beberapa cara, bisa dengan klik menu “New” kemudian pilih “Python File”, bisa juga dengan klik kanan pada project kita, dan pilih “Python File”.

## Pengenalan Bahasa Python

### 1.1. Sejarah singkat perkembangan bahasa pemrograman Python

#### 1.1.1. Sekilas Perkembangan Bahasa Python

Python diciptakan oleh Guido van Rossum pertama kali di Centrum Wiskunde & Informatica (CWI) di Belanda pada awal tahun 1990-an. Bahasa python terinspirasi dari bahasa pemrograman ABC. Sampai sekarang, Guido masih menjadi penulis utama untuk python, meskipun bersifat open source sehingga ribuan orang juga berkontribusi dalam mengembangkannya.

Di tahun 1995, Guido melanjutkan pembuatan python di Corporation for National Research Initiative (CNRI) di Virginia Amerika, di mana dia merilis beberapa versi dari python.

Pada Mei 2000, Guido dan tim Python pindah ke BeOpen.com dan membentuk tim BeOpen PythonLabs. Di bulan Oktober pada tahun yang sama, tim python pindah ke Digital Creation (sekarang menjadi Perusahaan Zope). Pada tahun 2001, dibentuklah Organisasi Python yaitu Python Software Foundation (PSF). PSF merupakan organisasi nirlaba yang dibuat khusus untuk semua hal yang berkaitan dengan hak intelektual Python. Perusahaan Zope menjadi anggota sponsor dari PSF.

Semua versi python yang dirilis bersifat open source. Dalam sejarahnya, hampir semua rilis python menggunakan lisensi GFL-compatible. Berikut adalah versi mayor dan minor python berikut tanggal rilisnya.

- Python 1.0 – Januari 1994
- Python 1.2 – 10 April 1995
- Python 1.3 – 12 Oktober 1995
- Python 1.4 – 25 Oktober 1996
- Python 1.5 – 31 Desember 1997
- Python 1.6 – 5 September 2000
- Python 2.0 – 16 Oktober 2000
- Python 2.1 – 17 April 2001
- Python 2.2 – 21 Desember 2001
- Python 2.3 – 29 Juli 2003
- Python 2.4 – 30 Nopember 2004
- Python 2.5 – 19 September 2006
- Python 2.6 – 1 Oktober 2008
- Python 2.7 – 3 Juli 2010
- Python 3.0 – 3 Desember 2008
- Python 3.1 – 27 Juni 2009
- Python 3.2 – 20 Februari 2011
- Python 3.3 – 29 September 2012
- Python 3.4 – 16 Maret 2014
- Python 3.5 – 13 September 2015
- Python 3.6 – 23 Desember 2016
- Python 3.7 – 27 Juni 2018

Nama python sendiri tidak berasal dari nama ular yang kita kenal. Guido adalah penggemar grup komedi Inggris bernama Monty Python. Ia kemudian menamai bahasa ciptaannya dengan nama Python.

#### 1.1.2. Mengapa Harus Python

Mengapa harus Python? Bukankah masih banyak bahasa pemrograman lain di luar sana? Apa kelebihan Python?

Pertanyaan – pertanyaan tersebut sering menjadi pertanyaan yang muncul sebelum seseorang mempelajari Python. Berikut adalah beberapa di antara kelebihan Python:

1. Python adalah bahasa pemrograman yang populer. Per September 2018, Python berada di urutan ke 3 bahasa program yang paling populer di dunia.
2. Python relatif lebih mudah dipelajari dan digunakan dibandingkan bahasa pemrograman lain. Sintaksnya sederhana, mudah dibaca dan diingat karena filosofi python sendiri menekankan pada aspek kemudahan dibaca (readability). Kode python mudah ditulis dan mudah dibaca, sehingga lebih mudah diperbaiki kalau ada kesalahan, dan juga mudah untuk dipelihara.
3. Selain lebih mudah dibaca, python juga lebih efisien dibandingkan bahasa lain seperti C, C++, maupun Java. Untuk melakukan sesuatu dengan 5 baris kode pada bahasa lain, bisa jadi di python hanya diperlukan 1 baris kode. Hal ini menyebabkan

pembuatan program dalam Python menjadi lebih ringkas dan lebih cepat dibandingkan bahasa lain.

4. Python merupakan bahasa multifungsi. Dengan python Anda bisa melakukan berbagai hal mulai dari memproses teks, membuat website, membuat program jaringan, robotika, data mining, sampai dengan kecerdasan buatan. Dengan python Anda bisa membuat aplikasi berbasis desktop maupun berbasis smartphone.

5. Python kaya akan dukungan library (pustaka) standar. Tersedia banyak sekali modul-modul dan ekstensi program yang sudah siap Anda pakai untuk membuat program sesuai kebutuhan Anda. Komunitas python adalah komunitas yang sangat aktif mengembangkan python sehingga menjadi bahasa yang sangat handal.

6. Python bisa berinteraksi dengan bahasa lain. Kode python bisa memanggil oleh bahasa C, C++, dan sebaliknya juga bisa dipanggil dari bahasa lain.

Tapi, itu hanya kelebihanannya. Terus, apa kekurangannya? Python adalah bahasa interpreter. Kekurangan python dibanding bahasa lain yang menggunakan kompiler adalah 'sedikit' lebih lambat pada saat dijalankan bila dibandingkan bahasa C maupun C++. Tapi hal inipun sangat bersifat relatif. Tergantung dari besar ukuran program yang dibuat.

Untuk program besar yang membutuhkan kecepatan pemrosesan tinggi mungkin Python kalah cepat dari bahasa C, tapi untuk hal selain itu Python lebih mudah dan lebih baik dari bahasa lain. Selain itu, kode sumber sekarang sudah dioptimasi menggunakan bahasa C, sehingga kecepatannya juga sudah sangat mendekati kecepatan bahasa C. Spesifikasi komputer juga sekarang ini sudah semakin tinggi sehingga bisa memproses program dengan cepat, sehingga sering kali ini tidak menjadi hal penting dan bisa diabaikan.

### 1.1.3. Sekilas Tentang Bahasa Python

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai pemrograman yang menggabungkan kapabilitas bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif.

Python mendukung multi paradigma pemrograman, utamanya namun tidak dibatasi pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis. Seperti halnya pada pemrograman dinamis lainnya. Python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan untuk berbagai pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi.

## 1.2. Pengenalan IDE Python

### A. Pengenalan IDE PyCharm

IDE merupakan singkatan dari Integrated Development, merupakan lembar kerja terpadu untuk pengembangan program. IDE dari Python yaitu PyCharm IDE, dapat digunakan untuk :

1. Menulis naskah program.
2. Mengkompilasi Program (Compile)
3. Melakukan Pengujian Program (Debugging)
4. Mengaitkan Object dan Library ke program (Linking)
5. Menjalankan Program (Running)

Untuk mengaktifkan aplikasi PyCharm, lakukanlah langkah-langkah berikut ini:  
Klik tombol pencarian pada windows lalu ketikan PyCharm.

PyCharm IDE pada Python, terbagi menjadi 4 bagian yaitu :

a. Baris Menu (Menu Bar)

Menu utama terdiri dari : File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, dan Help.

b. Baris Peralatan (Tools Bar)

Baris yang menampilkan shortcuts (icons) untuk mempermudah pengguna dalam pembuatan program-program python, seperti icon open, save, compiler, run dan lain-lain.

c. Jendela Editor

Tempat untuk pengetikan program dan membuat program.

d. Jendela Message

Tempat untuk menampilkan pesan-pesan pada proses kompilasi dan link program. Jika ada kesalahan syntax program maupun variabel dan objek, maka akan diberikan pesan kesalahannya yang kemudian dapat didouble klik pada pesan tersebut untuk mendapatkan petunjuk di baris yang mana terdapat kesalahannya.

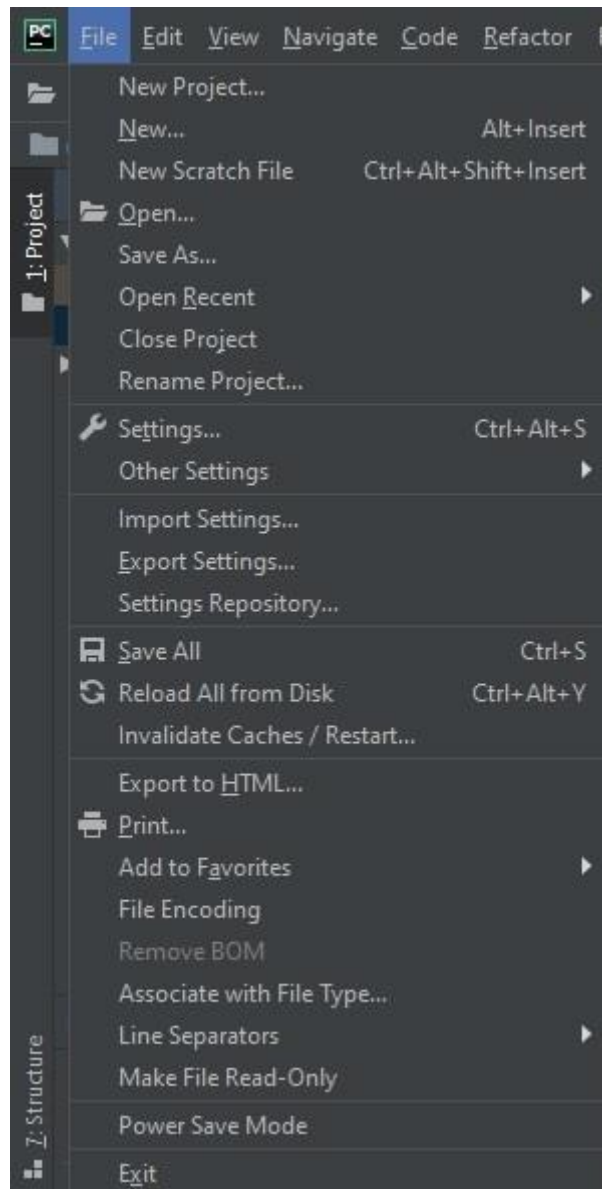
e. Baris Status

Baris yang akan menampilkan keterangan-keterangan pada saat mengaktifkan menu bar dan sub menu serta keterangan-keterangan lain (seperti petunjuk baris dan kolom, waktu yang sedang berjalan).

Secara garis besar, aplikasi sama saja fungsi-fungsinya, menu dan icon-icon sebagai alat mempermudah kita dalam melakukan perintah, dan windows yang warna hitam, adalah text editor, tempat kita membuat aplikasi melalui coding- coding yang akan kita pelajari pada bab berikutnya.

B. Tools Program





Gambar 1.1 Tools Program

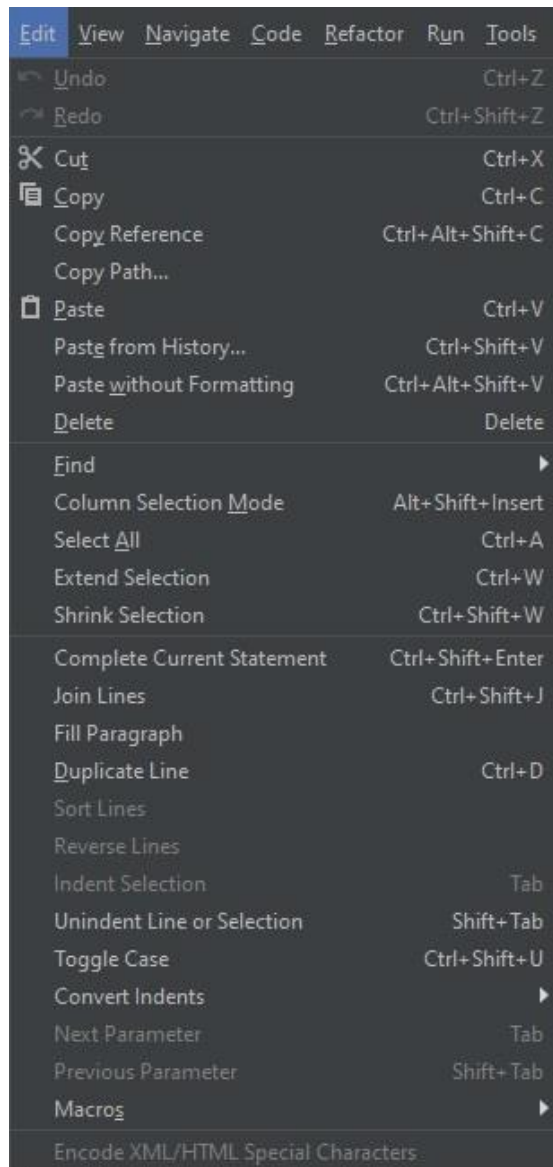
## 1. Menu File

Fungsi :

- a. **New Project** untuk membuat sebuah proyek baru yang belum pernah dibuat sebelumnya.
- b. **New** untuk membuat sebuah file baru yang belum pernah dibuat sebelumnya
- c. **New Scratch File** untuk membuat file scratch, file scratch berfungsi penuh, runnbale, dan debuggable yang mendukung penyorotan sintaksis, penyelsaian kode, dan semua fitur lainnya untuk tipe yang sesuai.
- d. **Open** membuka file atau proyek yang sudah pernah dibuat sebelumnya.
- e. **Save as** untuk menyimpan aplikasi dengan nama baru
- f. **Open Recent** untuk membuka kembali file atau proyek yang sudah pernah dibuka
- g. **Close Project** untuk keluar dari proyek yang sedang dikerjakan

- h. **Rename Project** untuk mengganti nama proyek yang sedang dikerjakan
- i. **Setting** untuk mengubah setelan pada pycharm sesuai keinginan
- j. **Other Setting** berisikan menu Setting for new projects dan Run configuration template for new projects.
- k. **Import Setting** untuk menyimpan pengaturan pada pycharm agar bisa di export ketika menggunakan aplikasi pycharm di komputer / laptop yang berbeda.
- l. **Export Setting** untuk merubah pengaturan secara instan dengan pengaturan yang sudah di import dari komputer/laptop yang berbeda.
- m. **Setting Repository** untuk membagikan pengaturan IDE di antara berbagai contoh produk PyCharm (atau basis platform IntelliJ lainnya) yang di install pada komputer yang berbeda
- n. **Save all** untuk menyimpan keseluruhan program aplikasi yang kita buat.
- o. **Reload all from disk** untuk memuat ulang proyek atau file dalam disk pada komputer
- p. **Invalidate Caches / Restart** PyCharm memiliki sejumlah besar caches sistem, oleh karena itu caches mungkin suatu hari menjadi beban yang berlebihan, invalidate caches berfungsi untuk membersihkan caches sistem pada PyCharm.
- q. **Export to HTML** untuk menyimpan file aplikasi dengan merubah ekstensi nya menjadi ekstensi HTML
- r. **Print** untuk mencetak coding aplikasi.
- s. **Add to Favorite** untuk menambahkan sejumlah file atau folder ke menu favorit / yang sering digunakan, untuk memudahkan pencarian file yang sering digunakan oleh seorang pemrogram.
- t. **File Encoding** Encoding memiliki pengaruh pada PyCharm membaca atau menulis File. Jika file telah dimodifikasi tetapi belum disimpan, setiap perubahan dalam pengkodean memengaruhi penulisan file.
- u. **Remove BOM** penggunaan BOM adalah opsional. Kehadirannya mengganggu penggunaan UTF-8 oleh perangkat lunak yang tidak mengharapkan byte non-ASCII pada awal file tetapi yang sebaliknya dapat mengenali aliran teks.
- v. **Associate with file type** untuk menghindari salah penamaan ekstensi bisa menggunakan associate with file type.
- w. **Line Separators** untuk mengatur pemisah garis (akhiran garis) untuk file yang baru dibuat, dan mengubah gaya pemisah garis untuk file yang ada
- x. **Make File Read Only** untuk membuat file yang sudah kita buat menjadi tidak bisa di rubah / edit.
- y. **Power Save Mode** untuk mengaktifkan mode hemat daya.
- z. **Exit** untuk Keluar dari aplikasi.'

## 2. Menu Edit

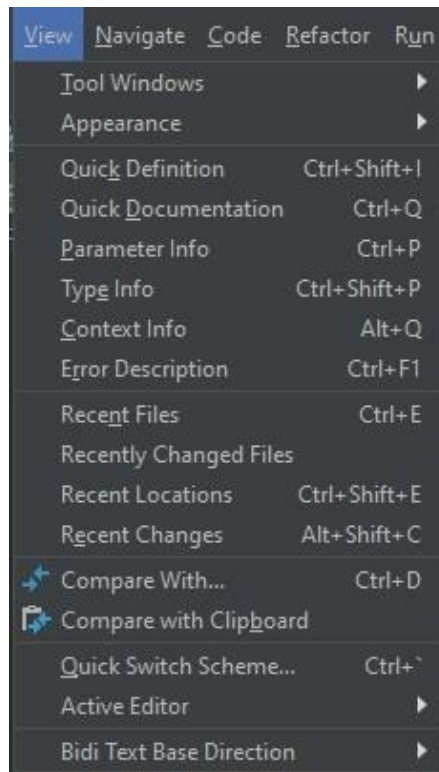


Gambar 1.2 Menu Edit

- a. **Undo** menu yang berisi perintah untuk membatalkan suatu perintah yang sudah dilakukan sebelumnya.
- b. **Redo** ialah kebalikan dari undo atau cara untuk mengulang perintah yang sudah dibatalkan sebelumnya.
- c. **Cut** memotong objek / teks yang terpilih.
- d. **Copy** menyalin objek / teks yang terpilih.
- e. **Copy Reference** menyalin objek / teks ke beberapa modul dan jika tindakan itu di panggil dari editor maka no baris akan disertakan.
- f. **Copy Path** menyalin jalur penyimpanan file.
- g. **Paste** menempelkan objek yang telah disalin.
- h. **Paste from History** menempelkan objek yang telah disalin berdasarkan riwayat yang telah dilakukan.
- i. **Paste without formatting** menempelkan objek yang telah disalin tanpa mengatur formatnya
- j. **Delete** untuk menghapus suatu objek / teks yang ingin dihapus.

- k. **Find** untuk mencari kata yang di inginkan
- l. **Column Selection Mode** pemilihan kolom dapat digunakan untuk memilih area file.
- m. **Select All** untuk memilih semua objek / teks.
- n. **Extend Selection** untuk secara berturut-turut memilih memperluas blok kode yang logis sehingga dapat memilih ekspresi apapun dalam kode dengan menempatkan tanda sisipan disuatu tempat beberapa kali.
- o. **Shrink Selectoin** untuk secara berturut-turut memilih mempersempit blok kode yang logis sehingga dapat memilih ekspresi apapun dalam kode dengan menempatkan tanda sisipan disuatu tempat beberapa kali.
- p. **Complete Curent Statement** memasukan elemen sintaks yang diperlukan (tanda kurung, kurung kurawal, titik koma dan sebagainya).
- q. **Join Lines** untuk menambahkan baris berikutnya setelah bergabung dengan beberapa baris yang dipilih menjadi satu.
- r. **Fill Paraggraph** untuk menyesuaikan jeda baris komentar secara otomatis.
- s. **Duplicate line** untuk menduplikasi baris atau pilihan saati ini dengan cepat.
- t. **Sort Lines** untuk mengurutkan baris atau seluruh file yang dipilih jika seleksi kosong.
- u. **Reverse Line** membalikan baris teks yang dipilih.
- v. **Indent Selection** untuk memilih baris teks yang akan di indentasi.
- w. **Unindent line or selection** untuk mengulang perintah indentasi baris atau teks.
- x. **Toggle Case** huruf kecil pada bagian depan tiap-tiap kata.
- y. **Convert indent** untuk mengubah indentasi.
- z. **Macros** menyediakan cara untuk mengotomatiskan prosedur yang sering dilakukan saat menulis kode.

### 3. Menu View

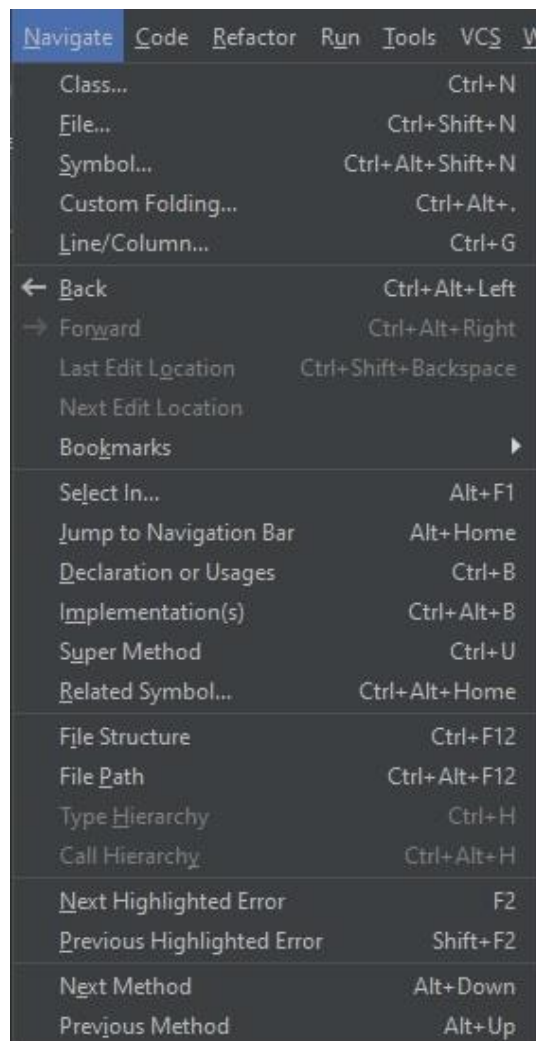


Gambar 1.3 Menu View

- a. **Tool Windows** jendela-jendela sekunder ini memungkinkan melihat proyek dari berbagai perspektif dan menyediakan tugas-tugas pengembang yang khas.
- b. **Appearance** untuk merubah menu tampilan pada aplikasi.
- c. **Quick Definition** untuk melihat dimana dan bagaimana simbol seperti, tag, kelas, bidang, metode atau fungsi didefinisikan dalam proyek.
- d. **Quick Documentation** untuk membantu mendapatkan informasi dengan cepat mengenai simbol apapun yang disertakan dengan komentar dokumentasi dalam format yang berlaku.
- e. **Parameter Info** menunjukkan nama-nama parameter dalam panggilan metode dan fungsi.
- f. **Type Info** menyediakan berbagai cara untuk membantu mengamati dan memeriksa jenis objek dalam skrip.
- g. **Context Info** menyediakan berbagai cara untuk membantu mengamati dan memeriksa hubungan objek dalam skrip.
- h. **Error Description** untuk memberikan daftar solusi masalah dan solusi yang diketahui
- i. **Recently Changed File** untuk melacak perubahan yang telah dilakukan terhadap suatu file
- j. **Recent Locations** untuk melacak perubahan yang telah dilakukan terhadap lokasi penyimpanan
- k. **Recent Changes** untuk melacak perubahan yang telah dilakukan.
- l. **Compare With** untuk meninjau perbedaan antara dua file, folder, sumber teks, atau objek basis data apapun.

- m. **Compare with Clipboard** untuk meninjau perbedaan antara dua file, folder, sumber teks, atau objek basis data apapun, serta antara file lokal dan versi repositori mereka.
- n. **Quick Switch Scheme** untuk beralih di antara berbagai skema warna, dan tata letak keyboard dengan cepat.
- o. **Active Editor** editor PyCharm adalah bagian utama yang digunakan untuk membuat, membaca, dan memodifikasi kode.
- p. **Bidi Text Base Direction** memungkinkan untuk memilih arah dasar untuk merender string dan token, seperti campuran bahasa inggris dan bahasa arab.

#### 4. Menu Navigate



Gambar 1.4 Menu Navigate

- a. **Class** merupakan sebuah cetakan untuk menciptakan suatu isntant dari objek.
- b. **File** merupakan kumpulan berbagai informasi yang berhubungan dan juga tersimpan dalam secondary strorage.
- c. **Symbol** gambar, bentuk, atau benda yang mewakili suatu gagasan, benda, ataupun jumlah sesuatu
- d. **Custom Folding** untuk merentangkan semua fragmen yang diciutkan dalam file hingga tingkat bersarang yang ditentukan

- e. **Line/Column** sebuah baris / kolom pada skrip.
- f. **Back** untuk kembali ke proses sebelumnya.
- g. **Forward** untuk meneruskan ke proses berikutnya.
- h. **Bookmarks** untuk menampilkan bookmark berikutnya atau sebelumnya, di menu utama
- i. **Select In** secara otomatis menghapus pilihan kejadian saat ini dan memilih yang berikutnya.
- j. **Jump to Navigation Bar** untuk menampilkan kembali bar navigasi yang telah disembunyikan.
- k. **Declaration or Usage** untuk menavigasi ke deklarasi simbol dari penggunaan simbol apapun.
- l. **Implementation(s)** untuk menavigasi ke implementasi.
- m. **Super Method** untuk mengganti metode apapun dari kelas utama dengan menghasilkan kode yang diperlukan dari template yang telah ditentukan.
- n. **Related Symbol** untuk menampilkan simbol terkait.
- o. **File Structure** untuk memperlihatkan semua kelas, metode dan elemen lain dari file saat ini.
- p. **File Path** untuk memperlihatkan lokasi penyimpanan file dari file saat ini.
- q. **Type Hierarchy** untuk memeriksa hierarki kelas, metode, di jendela hierarki.
- r. **Call Hierarchy** untuk memanggil hierarki kelas, metode, di jendela hierarki.
- s. **Next Highlighted Error** untuk menavigasi error kode di editor menggunakan berbagai tindakan.
- t. **Previous Highlighted Error** untuk menavigasi error kode di editor menggunakan berbagai tindakan.
- u. **Next Method** untuk menavigasi cara di editor menggunakan berbagai tindakan.
- v. **Previous Method** untuk menavigasi cara di editor menggunakan berbagai tindakan.

## 5. Menu Code



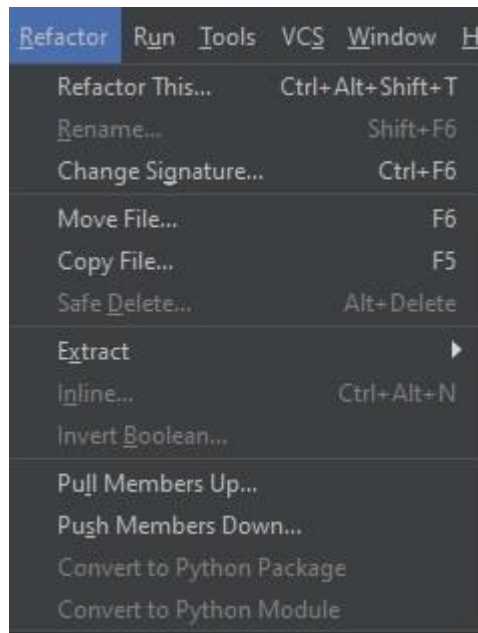
Gambar 1.5 Menu Code

- a. **Generate** untuk menghasilkan konstruksi kode umum dan elemen berulang, yang membantu meningkatkan produktivitas.
- b. **Surround With** template standar untuk fragmen kode disekitarnya dengan berbagai konstruksi berdasarkan sumber bahasa pemrograman.
- c. **Unwrap/Remove** untuk membuka atau mengekstraksi ekspresi dengan cepat dari melampirkan pernyataan.
- d. **Completion** untuk mencakup berbagai teknik penyelesaian konteks kode yang memungkinkan untuk mempercepat proses pengkodean
- e. **Insert Live Template** untuk menggambarkan cara membuat template untuk komentar TODO dengan tanggal dan nama pengguna saat ini.
- f. **Comment with Line Comment** untuk membuat komentar dengan komentar garis.
- g. **Reformat Code** untuk memformat ulang kode sesuai dengan persyaratan yang telah ditentukan dalam pengaturan Style Code.
- h. **Show Reformat File Dialog** untuk menampilkan tampilan menu reformat code.



- i. **Auto-Indent Lines** untuk mengatur indentasi secara otomatis sesuai dengan persyaratan yang telah ditentukan dalam pengaturan adjust indentation.
- j. **Optimizing Imports** untuk mengimpor satu kelas atau seluruh paket, tergantung pengaturan yang telah disesuaikan sebelumnya.
- k. **Move Statement Down** untuk memindahkan pernyataan ke bawah.
- l. **Move Statement Up** untuk memindahkan pernyataan ke atas.
- m. **Move Line Down** untuk mengatur ulang ekspresi, pernyataan, dan elemen lainnya ke bawah
- n. **Move Line Up** untuk mengatur ulang ekspresi, pernyataan, dan elemen lainnya ke atas.
- o. **Inspect Code** untuk menganalisis kode dalam file yang dibuka di editor dan meyoroti kode animal saat mengetik.
- p. **Code Clean Up** untuk secara instan menghilangkan kesalahan gaya kode dalam satu atau lebih file, dalam suatu proyek.
- q. **Run Inspection by Name** untuk menganalisis kode dalam file yang dibuka di editor dan meyoroti kode animal saat mengetik berdasarkan nama nya.
- r. **Configure Current File Analysis** beberapa inspeksi mungkin melaporkan masalah yang saat ini tidak ingin dilihat. Dalam hal ini, anda dapat menonaktifkannya.
- s. **View Offline Inspection Result** ketika bekerja di editor PyCharm, ia terus-menerus menjalankan inspeksi kode untuk menemukan dan meyoroti kesalahan sintksis, kemungkinan bug, dan sebagainya. Hasil inspeksi dapat disimpan sebagai file XML, JSON, atau teks biasa dengan laporan.

## 6. Menu Refactor

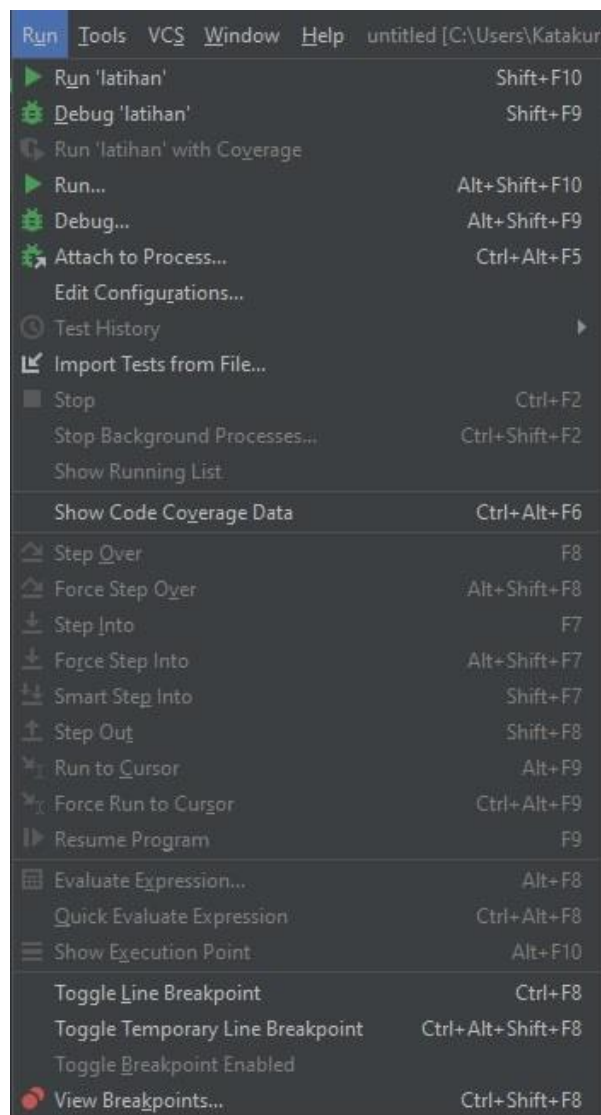


Gambar 1.6 Menu Refactor

- a. **Refactor This** untuk menunjukkan beberapa refactoring yang tersedia di PyCharm, menggunakan contoh sederhana yang memanfaatkan bilangan rasional.

- b. **Change Signature** untuk menggabungkan beberapa modifikasi berbeda yang dapat diterapkan pada fungsi Change Signature.
- c. **Move File** memungkinkan untuk memindahkan kelas, fungsi, modul, file, dan direktori dalam suatu proyek.
- d. **Copy File** memungkinkan untuk menyalin kelas, fungsi, modul, file, dan direktori dalam suatu proyek.
- e. **Extract** memungkinkan untuk mengambil sebuah fragmen kode yang dapat dikelompokkan bersama, memindahkannya ke metode yang terpisah dan mengganti kode lama dengan panggilan ke metode tersebut.
- f. **Pull Members Up** memungkinkan anda untuk memindahkan anggota kelas ke superclass.
- g. **Push Members Down** membantu membersihkan hierarki kelas dengan anggota kelas ke subclass.

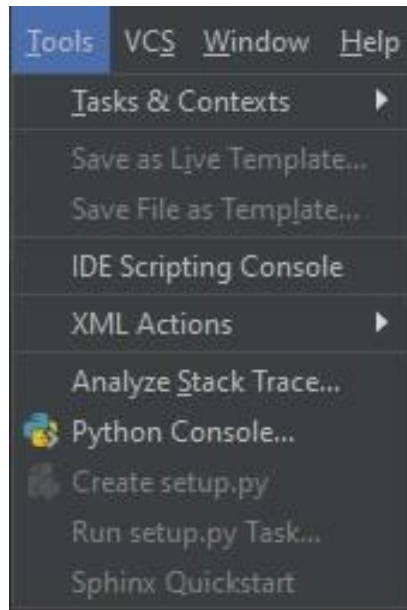
## 7. Menu Run



Gambar 1.7 Menu Run

- a. **Run** untuk menampilkan output yang dihasilkan oleh aplikasi.
- b. **Debug** proses pencarian di dalam baris kode yang menyebabkan eror terjadi.
- c. **Attach to Process** PyCharm memungkinkan untuk melampirkan ke proses lokal Python, saat menjalankan skrip Python dijalankan baik dari sistem operasi atau menggunakan terminal PyCharm, tetapi bukan dalam mode debug.
- d. **Edit Configuration** membuat profil konfigurasi atau mengubah yang default.
- e. **Test History** untuk melihat statistik pengujian, menavigasi tumpukan jejak, menunjukkan atau menyembunyikan tes yang berhasil, dan banyak lagi.
- f. **Import Test from File** untuk mengimpor hasil suatu tes atau beberapa test dari sebuah file.
- g. **Stop** untuk dapat menghentikan program, atau menjeda hasilnya.
- h. **Stop Background Process** untuk menghentikan program, yang masih bekerja dalam dalam background process.
- i. **Show Running List** PyCharm memungkinkan menjalankan seluruh aplikasi serta skrip tertentu.
- j. **Show Code Coverage Data** melihat cakupan kode membantu anda mendeteksi bagian-bagian kode sumber yang tidak terpengaruh oleh simulasi.
- k. **Step Over** langkah-langkah di atas baris kode saat ini dan membawa anda ke baris berikutnya bahkan jika jalur yang disorot memiliki panggilan metode di dalamnya.
- l. **Force Step Over** untuk menghentikan secara paksa proses step over.
- m. **Step Info** untuk menunjukkan apa yang terjadi di dalamnya.
- n. **Force Step Info** untuk menghentikan secara paksa proses step info.
- o. **Smart Step Info** sangat membantu ketika ada beberapa panggilan metode pada suatu saluran, dan anda ingin lebih spesifik tentang metode yang akan dimasukan.
- p. **Step Out** langkah keluar dari metode saat ini dan membawa anda ke metode pemanggil.
- q. **Run to Cursor** menunjukkan eksekusi sampai posisi caret tercapai.
- r. **Force run to Cursor** menunjukkan eksekusi sampai posisi caret tercapai. Semua breakpoint di jalan diabaikan.
- s. **Resume Program** aplikasi dijeda secara manual, sesi debug ditunda.
- t. **Evaluate Expression** dengan PyCharm, anda tidak hanya dapat melihat nilai variabel yang terpisah, tetapi juga mengevaluasi ekspresi yang lebih kompleks, seperti panggilan metode, ekspresi operator, ekspresi lambda, dan kelas anonim.

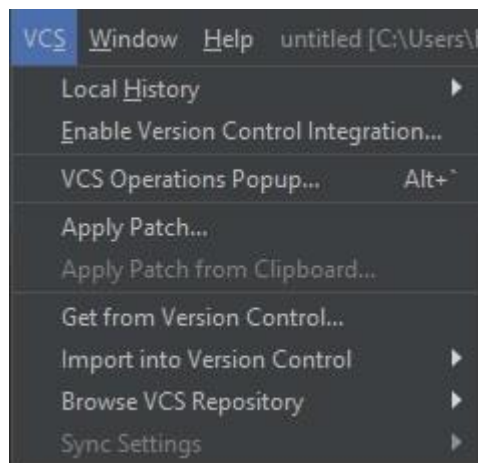
## 8. Menu Tools



Gambar 1.8 Menu Tools

- a. **Tasks & Context** saat anda mengerjakan suatu proyek, anda dapat mengatur pekerjaan anda dalam tugas-tugas kecil yang harus anda selesaikan.
- b. **IDE Scripting Console** dapat digunakan untuk menulis skrip sederhana yang mengotomatisasi fitur PyCharm dan mengekstrak berbagai informasi.
- c. **Analyze Stack Trace** dengan PyCharm, anda dapat menyalin pengecualian atau dump thread penuh, menempelkannya ke stack trace analyzer, dan menavigasi ke kode sumber yang sesuai.
- d. **Python Console** memungkinkan menjalankan perintah dan skrip Python baris demi baris, mirip dengan Python Shell.

## 9. Menu VCS

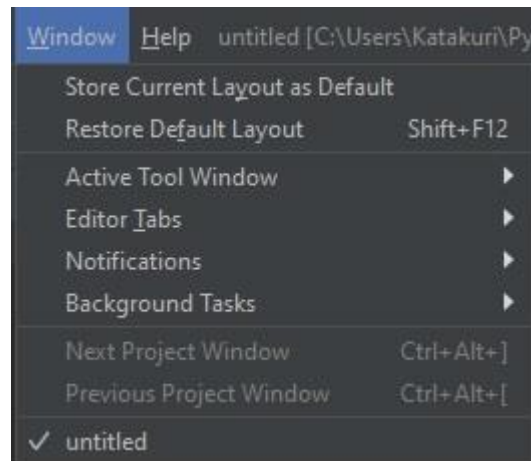


Gambar 1.9 Menu VCS

- a. **Local History** membantu anda melacak semua perubahan yang dibuat untuk memproyeksikan file dan strukturnya, terlepas dari kontrol versi.

- b. **Enable Version Control Integration** PyCharm memungkinkan anda mengaktifkan integrasi projek anda dengan cepat dengan sistem kontrol versi, dan mengaitkannya dengan root projek.
- c. **VCS Operations Popup** untuk secara cepat memohon perintah yang berhubungan dengan VCS.
- d. **Apply Patch** anda dapat mengimpor patch yang dibuat di dalam atau diluar PyCharm dan menerapkannya sebagai perubahan yang ditangguhkan.
- e. **Get from Version Control** memungkinkan anda untuk memeriksa repositori yang ada dan membuat projek baru berdasarkan data yang telah anda unduh.
- f. **Import into Version Control** memungkinkan anda untuk mengimpor projek yang ada berdasarkan data yang telah anda buat.
- g. **Browse VCS Repository** browser subversion repository memungkinkan anda untuk menambah atau membuang lokasi repositori, melihat riwayat file dan folder, menavigasi ke sumber kode, menelusuri perubahan, dan sebagainya.

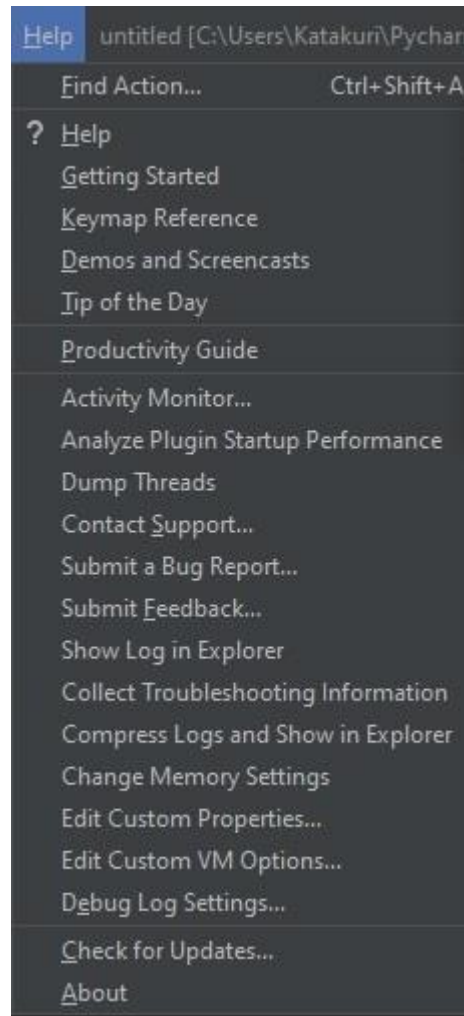
## 10. Menu Window



Gambar 1.10 Menu Window

- a. **Store Current Layout as Default** untuk menyimpan tampilan aplikasi yang telah dirubah sebagai tampilan defaultnya.
- b. **Restore Default Layout** untuk mengembalikan tampilan aplikasi menjadi tampilan default yang sudah di terapkan oleh pengembang aplikasi.
- c. **Active Tool Window** untuk melihat mengedit tampilan window yang aktif pada aplikasi.
- d. **Editor Tabs** anda dapat menutup, menyembunyikan, dan melepaskan editor tab.
- e. **Notifications** untuk mengaktifkan dan menonaktifkan pemberitahuan tentang peristiwa tertentu, mengubah presentasi mereka, dan secara opsional mengkatifkan pencatatan nyaa.
- f. **Background Tasks** ketika tugas yang panjang sedang berjalan, mialnya, mencari dan mengganti, pembaruan VCS, dan sebagainya, PyCharm menampilkan bilah kemajuan. Anda membawa pelaksanaan tugas-tugas tersebut ke latar belakang, dengan mengklik tombol latar belakang.

## 11. Menu Help



Gambar 1.11 Menu Help

- a. **Find Action** adalah perintah terpenting yang memungkinkan Anda mencari perintah dan pengaturan di semua menu dan alat.
- b. **Help** Menu bantuan pada aplikasi PyCharm
- c. **Getting Started** adalah menu panduan yang disediakan oleh PyCharm untuk memudahkan pengguna baru menjelajahi fungsi-fungsi yang terdapat pada aplikasi.
- d. **Keymap Reference** referensi peta kunci default untuk Windows / Linux dan untuk macOS
- e. **Check for Update** Cek update tentang aplikasi PyCharm
- f. **About** tentang aplikasi PyCharm

### 1.3. Struktur Program Python

Struktur bahasa pemrograman python berbeda dengan struktur bahasa pemrograman lainnya, dalam bahasa pemrograman python kita tidak harus mengetikkan bagian header, footer dan lain sebagainya, tetapi kita bisa langsung mengetikkan perintah apa yang kita inginkan.

#### C++ "Hello World"

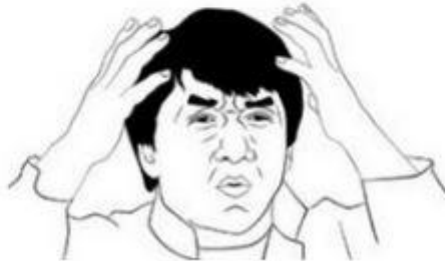
```
#include <iostream.h>
main()
{
    cout << "Hello World! ";
}
return 0;
```

#### Java "Hello World"

```
class HelloWorldApp
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
    }
}
```

#### Python

```
print "Hello world"
```

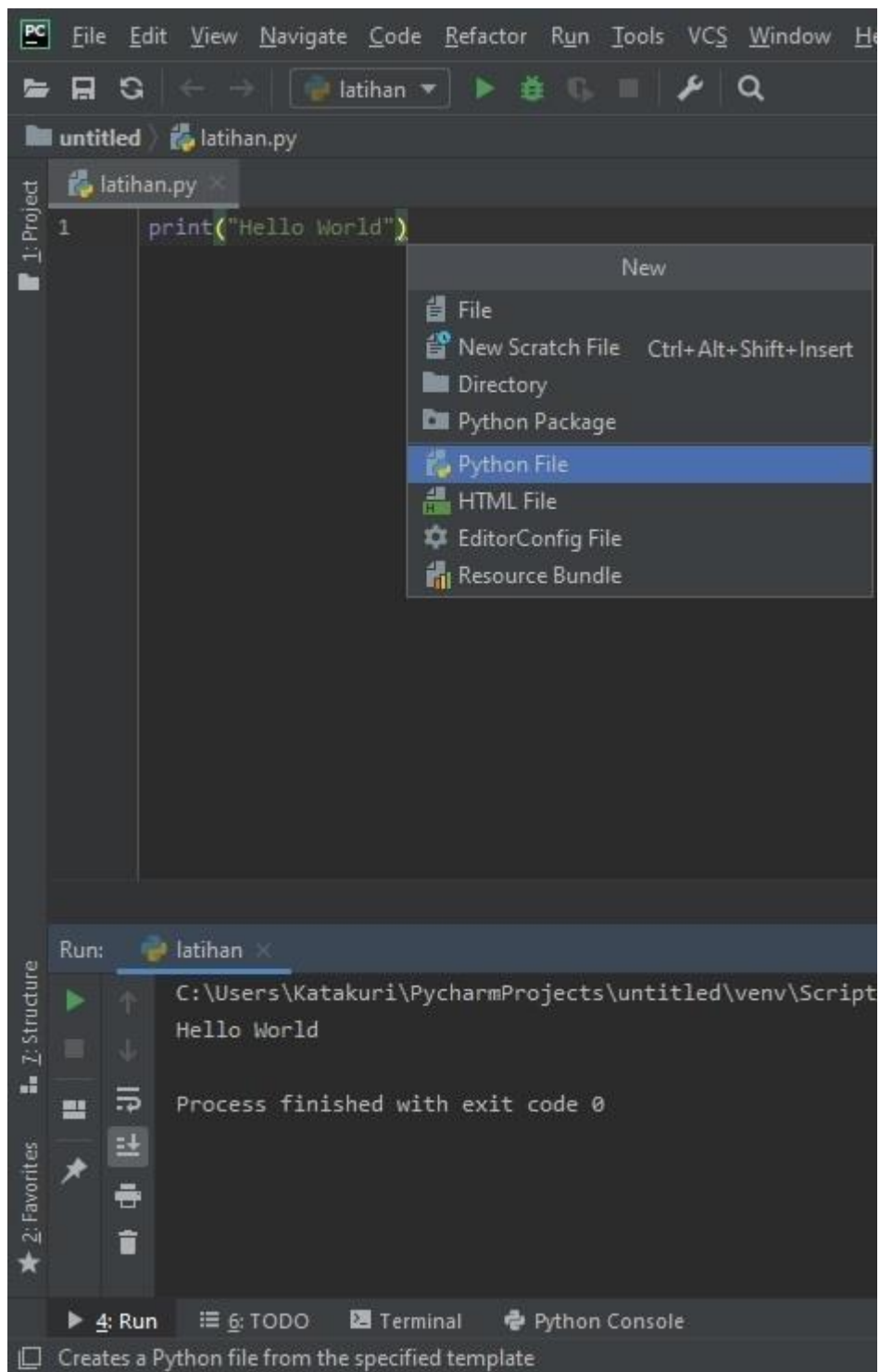


Gambar 1.12 Perbandingan Python dengan bahasa pemrograman lainnya.

### 1.4. Membuat File Editor

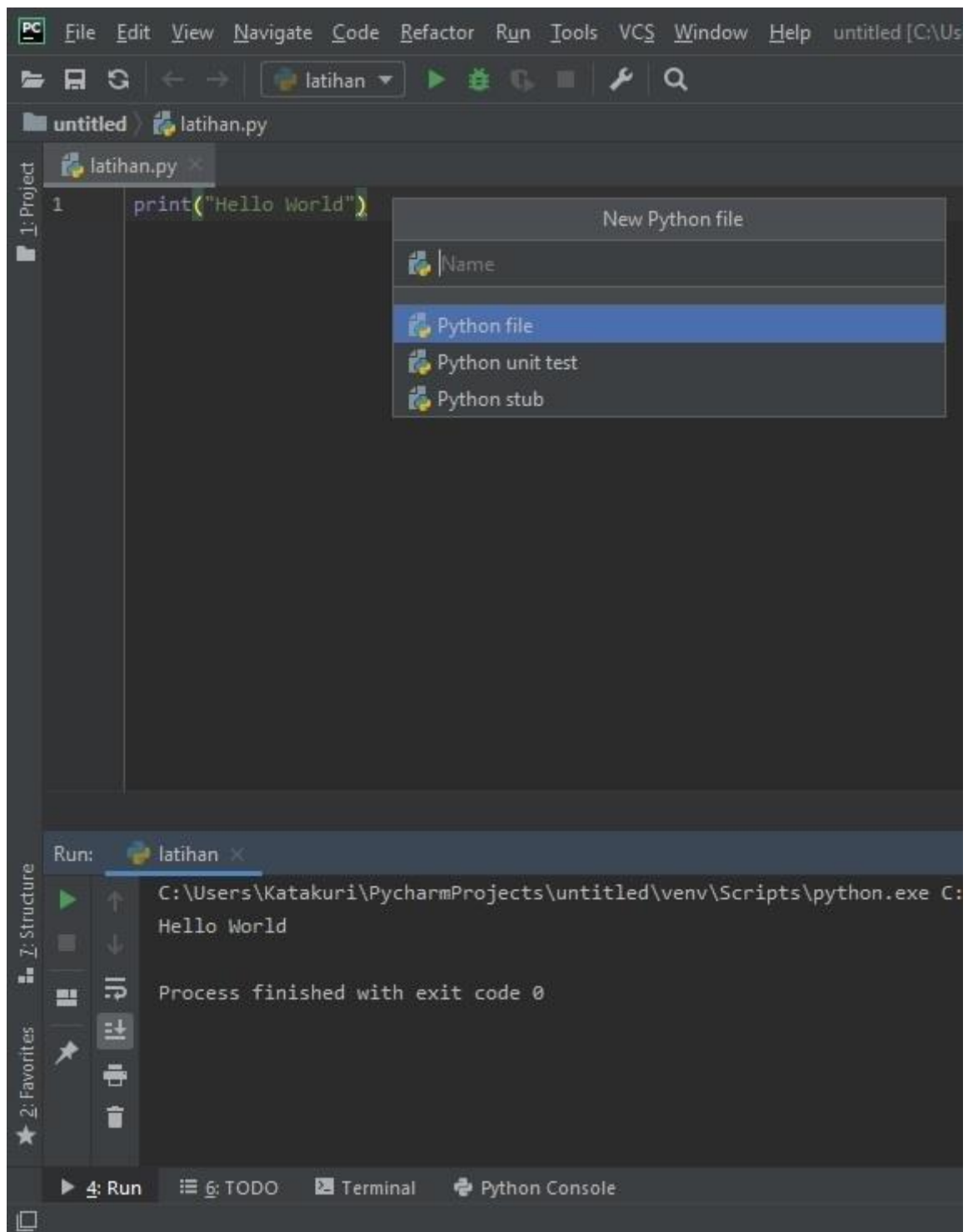
File Editor merupakan file kode program yang dapat dikompilasi, kemudian dijalankan untuk menampilkan hasilnya yang mempunyai ekstensi **.PY**.

Cara megkatifkan nya klik menu File Klik New → Klik Pyhton File → Beri Nama File → Tekan tombol Enter

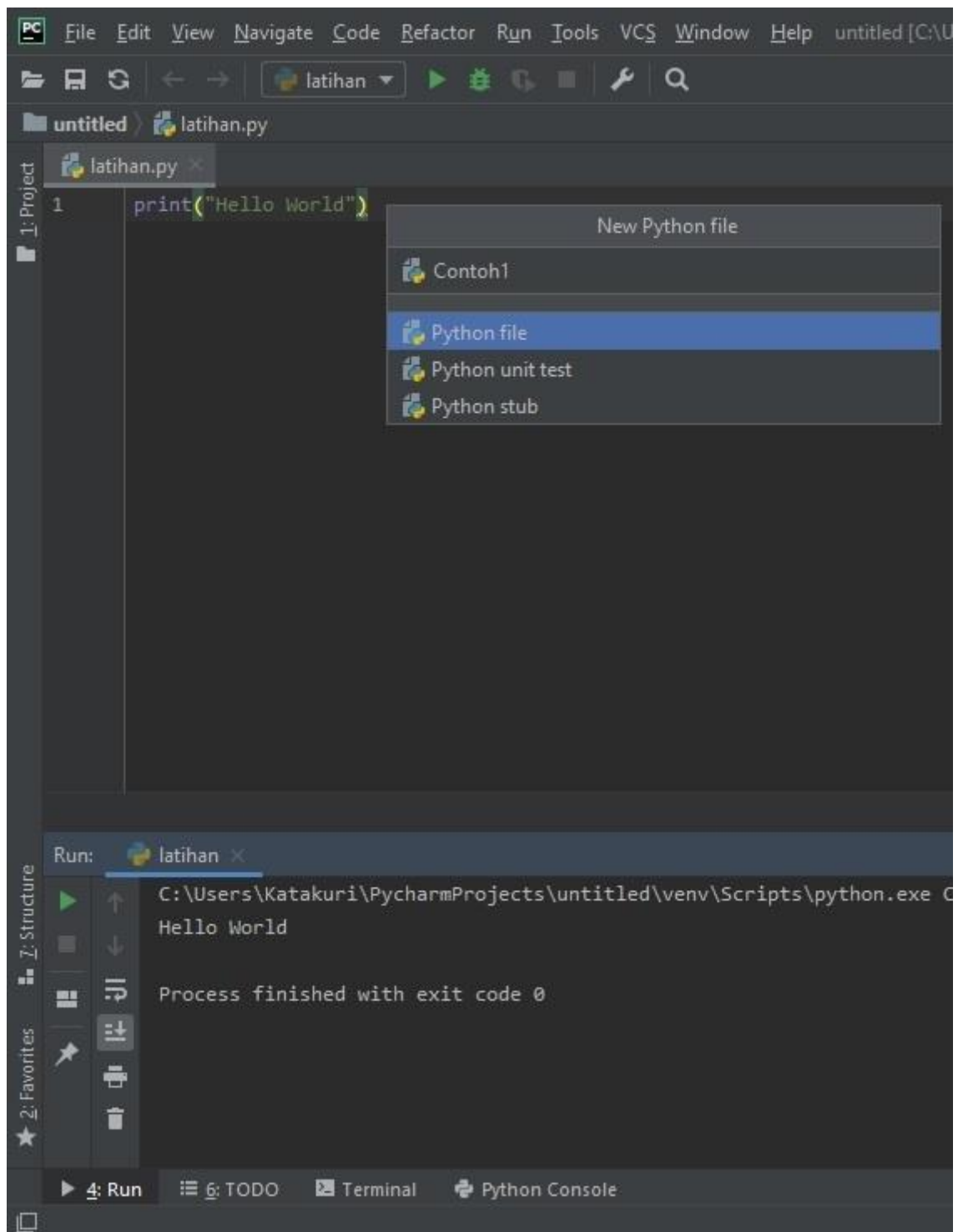


Gambar 1.13 Membuat File Baru





Gambar 1.14 Membuat File Baru



Gambar 1.15 Membuat File Baru

### 1.5. Menyimpan File Editor

Setelah selesai mengetikkan naskah program yang baru pada jendela Tex Editor, maka selanjutnya disimpan dengan cara :

- a. Klik Menu File → Save All.

Pada PyCharm 2019.3 terdapat 2 cara menyimpan file editor, diantaranya yaitu :

**Save All** digunakan untuk menyimpan semua file Python pada file proyek yang sedang aktif.

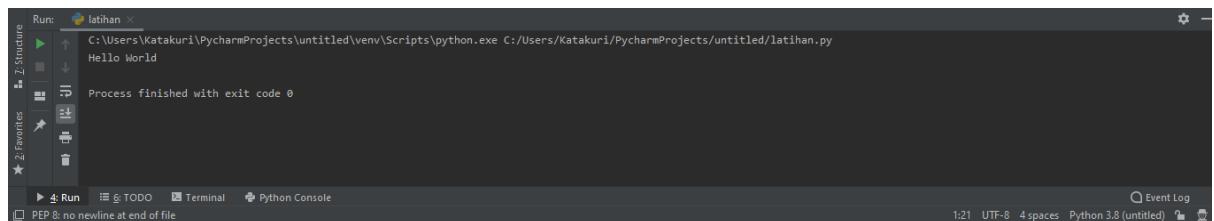
**Save As** digunakan untuk menyimpan file Python pada file proyek yang sedang aktif dengan nama file Python yang berbeda

## 1.6. Menjalankan Program

Proses Run merupakan suatu proses menerjemahkan program, melakukan proses linking, dan sekaligus menjalankan program, yaitu dengan cara :

- a. Klik Menu Run → Run 'Nama File'
- b. Menekan hotkey Shift + F10

Setelah proses menerjemahkan program, proses linking, selanjutnya tampil hasil seperti gambar 1.16 dibawah ini:

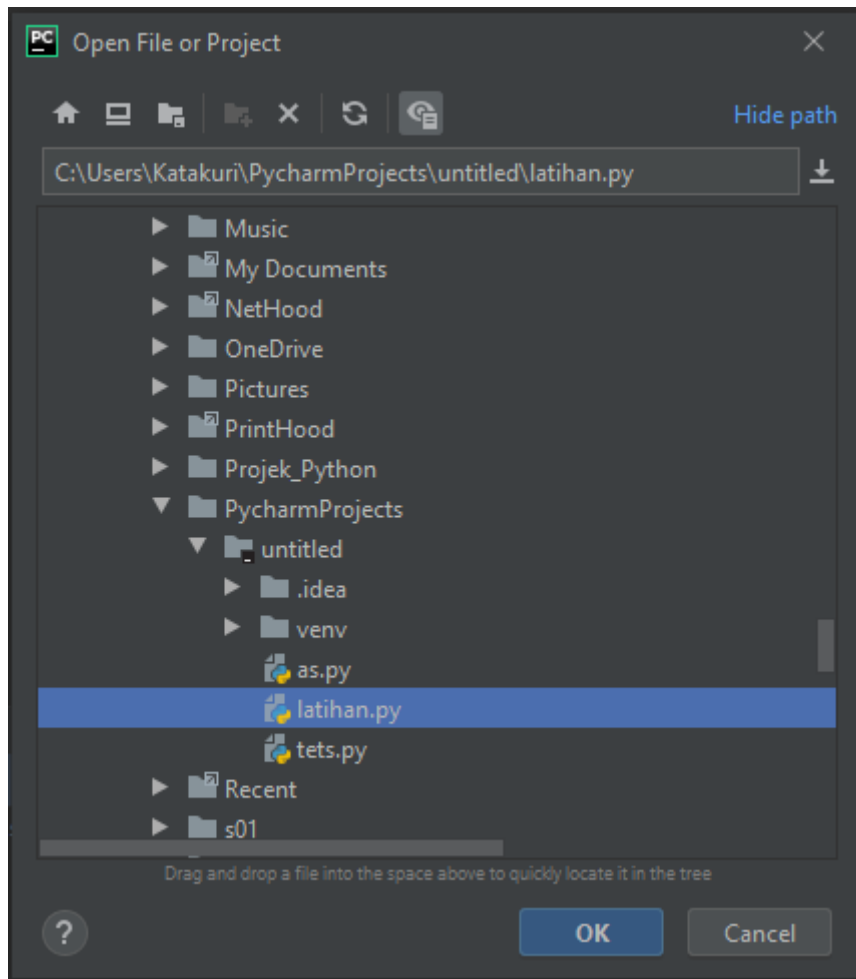


Gambar 1.16 Contoh Hasil Keluaran Program

## 1.7. Membuka File Editor

Penjelasan membuka atau memanggil file editor yang sudah dibuat, dengan cara, Klik Menu → File Open

Selanjutnya tampil jendela Open, seperti dibawah ini :

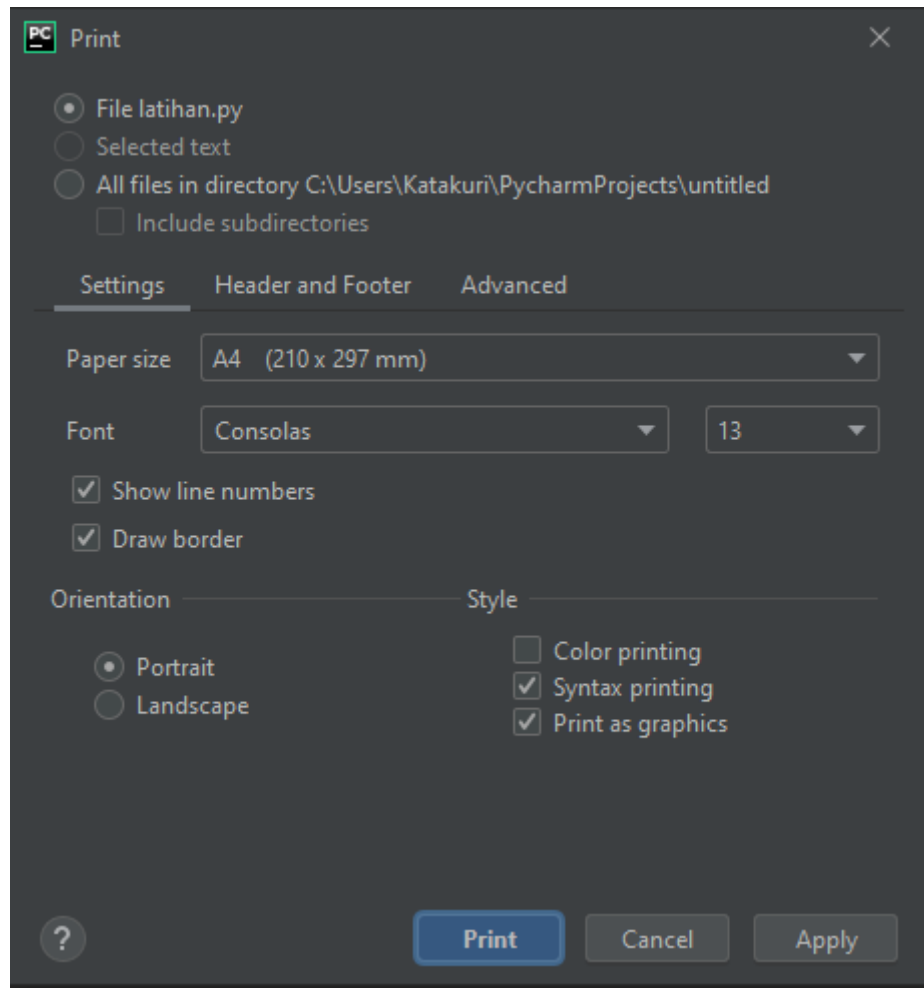


Gambar 1.17 Jendela Open File pada PyCharm 2019.3

### 1.8. Mencetak File Editor

Penjelasan mencetak file program pada jendela yang sedang aktif dengan klik menu File → Print

Selanjutnya akan tampil jendela print option, seperti dibawah ini:

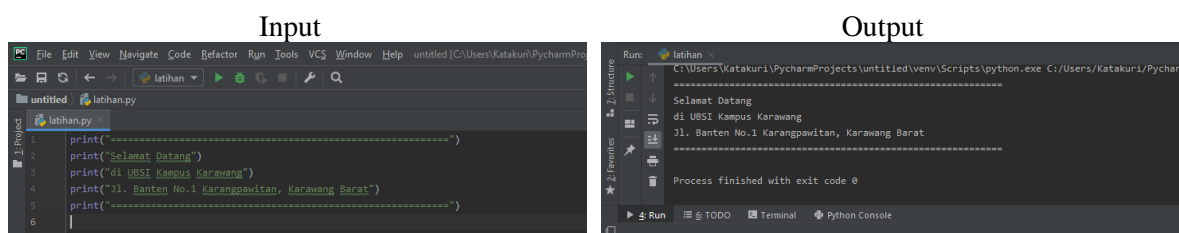


Gambar 1.18 Jendela Print Option

### 1.9. Keluar dari PyCharm 2019.3

Keluar dari aplikasi PyCharm 2019.3, dengan cara klik Menu File → Exit

Contoh struktur sederhana dalam PyCharm 2019.3

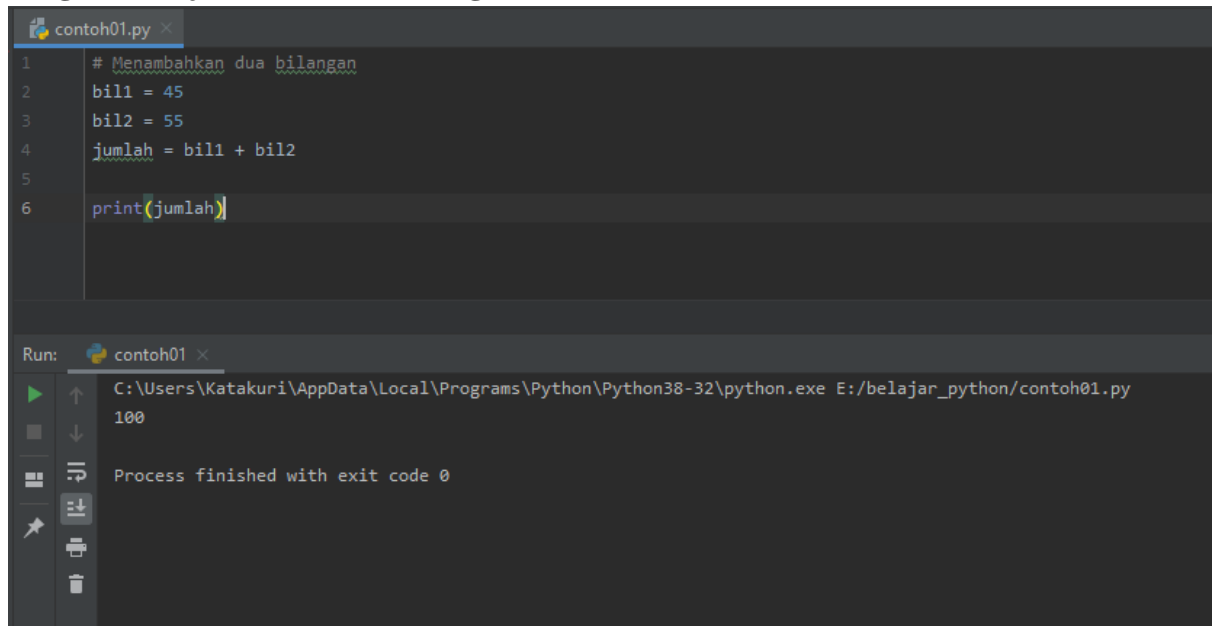


Gambar 1.19 Input dan Output Program

### 1.10. Program Pertama dengan Python

Seringkali, program “Hello World!” digunakan untuk mengenalkan suatu bahasa pemrograman ke pemula. Program “Hello World!” adalah sebuah program sederhana yang menampilkan kalimat “Hello World!” di monitor. Akan tetapi, Python adalah bahasa yang paling mudah dipelajari, dan membuat program “Hello World!” hanya sesederhana menuliskan perintah `print("Hello World!")`. Oleh karena itu, kita akan lebih memilih untuk membuat program penjumlahan dua bilangan.

## Program Penjumlahan Dua Bilangan



```
1 # Menambahkan dua bilangan
2 bil1 = 45
3 bil2 = 55
4 jumlah = bil1 + bil2
5
6 print(jumlah)
```

Run: contoh01 x

C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar\_python/contoh01.py  
100

Process finished with exit code 0

### Penjelasan Program

Python adalah bahasa pemrograman yang menggunakan interpreter. Pada interpreter program akan dieksekusi baris perbaris. Bila ada error maka program akan terhenti, kecuali dengan menggunakan metode penanganan eksepsi. Pada program di atas, baris 1 adalah komentar program. Interpreter tidak memproses komentar. Komentar hanya untuk penjelasan kode agar dipahami oleh manusia yang membacanya.

Pada baris 2 kita menciptakan sebuah objek bilangan yaitu 15. Kita membuat variabel bil1 menunjuk ke objek 15. Dengan kata lain, objek 15 ditugaskan ke variabel bil1. Penjelasan untuk baris 3 sama dengan penjelasan untuk baris ke 2.

Selanjutnya pada baris ke 4, objek yang ditunjuk oleh bil1 yaitu 15 dan yang ditunjuk oleh bil2 yaitu 25 dijumlahkan. Hasilnya ditugaskan ke variabel jumlah. Di baris terakhir, kita menggunakan fungsi bawaan (builtin) python, yaitu print() untuk menampilkan variabel jumlah ke monitor. Demikian program sederhana menggunakan python.

## Pertemuan 2

### Sintaks Dasar Python

Python merupakan bahasa pemrograman yang memiliki sintaks yang sederhana dan mudah dimengerti. Python memiliki filosofi bahwa kode program harus mudah dibaca.

**Statement (Pernyataan) di Python**, Semua perintah yang bisa dieksekusi oleh Python disebut statement. Misalnya, `a = 1` adalah sebuah statement penugasan. Selain statement penugasan ada statement lain seperti statement if, statement for, dan lain sebagainya.

**Statement Multibaris**, Pada Python, akhir dari sebuah statement adalah karakter baris baru (newline). Kita dapat membuat sebuah statement terdiri dari beberapa baris dengan menggunakan tanda backslash (`\`). Misalnya:

```
a = panjang1 + panjang2 + \
panjang3 + \
panjang4
```

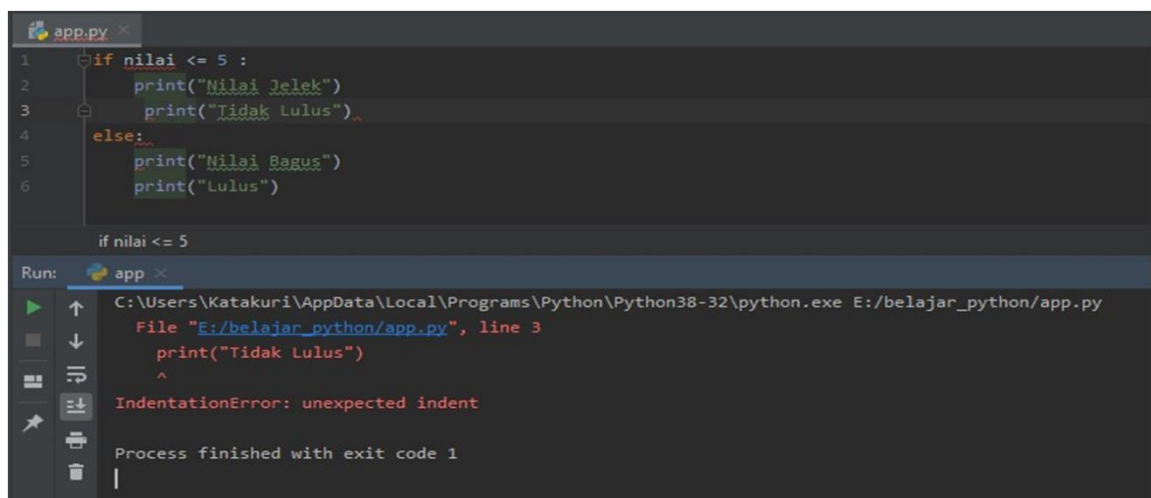
Statement yang ada di dalam tanda kurung `[ ]`, `{ }`, dan `( )` tidak memerlukan tanda `\`. Contohnya:

```
nama_bulan = ['Januari', 'Februari', 'Maret', 'April', 'Mei',
              'Juni']
```

**5.1. Baris dan Indentasi**, Python tidak menggunakan tanda `{ }` untuk menandai blok / grup kode. Blok kode di python menggunakan tanda indentasi (spasi). Jumlah spasi untuk setiap baris yang ada dalam satu blok kode harus sama. Contoh yang benar adalah sebagai berikut:

```
if nilai <= 5 :
    print("Nilai Jelek")    print("Tidak Lulus")
else:
    print("Nilai Bagus")    print("Lulus")
```

Bila indentasi dalam satu grup kode tidak sama, python akan menampilkan sintaks error.



**5.2. Tanda Kutip di Python**

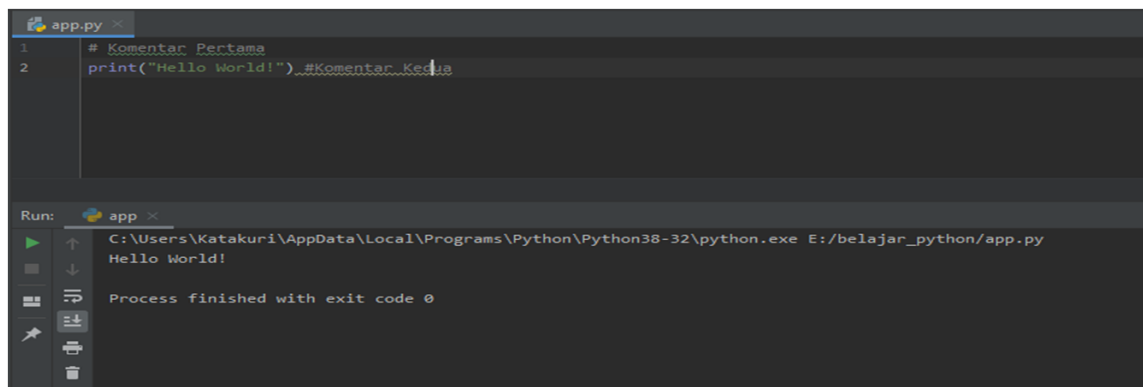
Python menggunakan tanda kutip tunggal ('), ganda ("), maupun triple (""" atau """) untuk menandai string, sepanjang stringnya diawali oleh tanda kutip yang sama di awal dan akhir string. Tanda kutip tiga digunakan untuk string multibaris. Ketiga contoh berikut, semuanya adalah benar.

```
kata = 'kata'
```

```
kalimat = "Ini adalah kalimat"    paragraf = """Ini adalah  
Paragraf terdiri dari beberapa baris"""
```

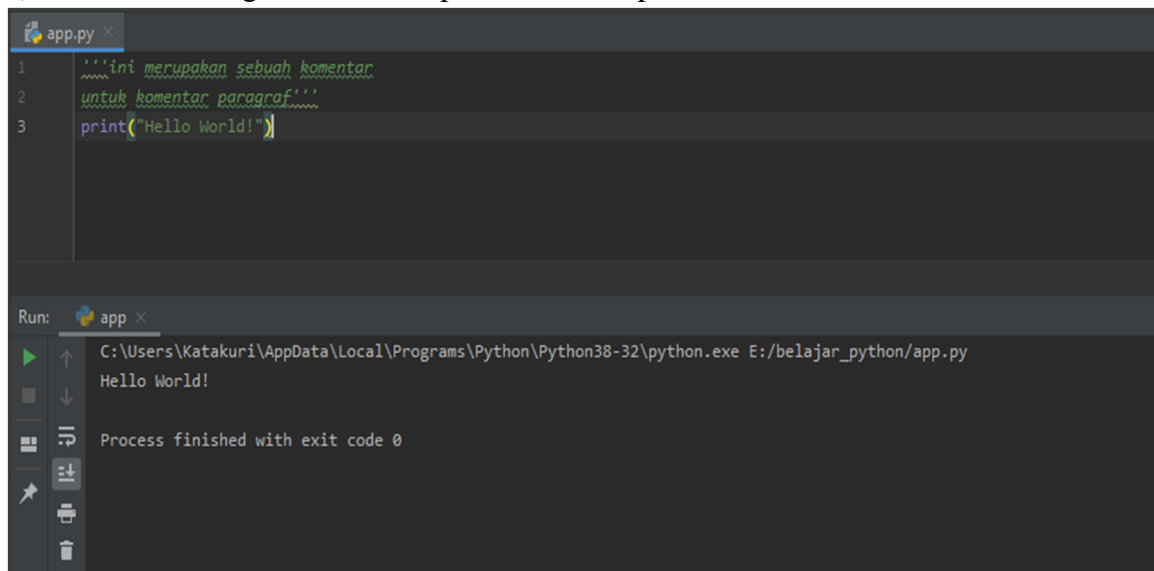
### 5.3. Komentar di Python

Tanda pagar ( # ) digunakan untuk menandai komentar di python. Komentar tidak akan diproses oleh interpreter Python. Komentar hanya berguna untuk programmer untuk memudahkan memahami maksud dari kode.



The screenshot shows a Python IDE with a file named `app.py`. The code contains a single-line comment: `# Komentar Pertama` followed by `print("Hello World!")`. The comment is highlighted in green. Below the code editor, the Run window shows the command `C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/app.py` and the output `Hello World!`. The process finished with exit code 0.

Python juga memiliki fitur komentar multibaris dengan perintah triple tanda kutip ("""). Kita harus mengomentari satu persatu baris seperti berikut:

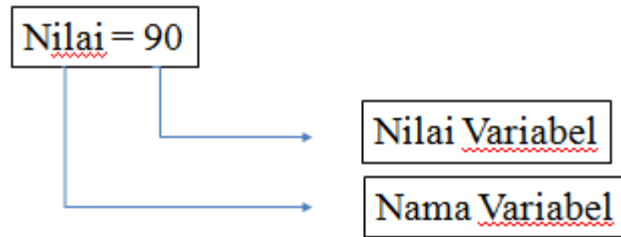


The screenshot shows a Python IDE with a file named `app.py`. The code contains a multi-line comment using triple quotes: `'''ini merupakan sebuah komentar  
untuk komentar paragraf'''` followed by `print("Hello World!")`. The comment is highlighted in green. Below the code editor, the Run window shows the command `C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/app.py` and the output `Hello World!`. The process finished with exit code 0.

### 5.4. Variabel dan Tipe Data Python

Variabel adalah lokasi di memori yang digunakan untuk menyimpan nilai. Memberi Nilai Variabel, Di python, variabel tidak perlu dideklarasikan secara eksplisit. Deklarasi atau pembuatan variabel terjadi secara otomatis pada saat kita memberi (menugaskan) suatu nilai ke variabel. Tanda sama dengan ( = ) digunakan untuk memberikan nilai ke variabel. Operand di sebelah kiri tanda = adalah nama variabel dan di sebelah kanan tanda = adalah nilai yang disimpan di dalam variabel.





### 5.5. Tipe Data Python

Data yang disimpan di memori memiliki tipe yang berbeda – beda. Misalnya untuk panjang, akan disimpan dengan tipe bilangan. Nama orang akan disimpan dalam tipe string/karakter. Suhu akan disimpan dalam bentuk bilangan berkoma. Dan lain sebagainya. Tipe data adalah suatu media atau memori pada komputer yang digunakan untuk menampung informasi. Python sendiri mempunyai tipe data yang cukup unik bila kita bandingkan dengan bahasa pemrograman yang lain. Berikut adalah tipe data dari bahasa pemrograman Python :

Tabel 2.1 Tipe data Python

Tipe Data	Contoh	Penjelasan
Boolean	True atau False	Menyatakan benar True yang bernilai 1, atau salah False yang bernilai 0
String	"Ayo belajar Python"	Menyatakan karakter/kalimat bisa berupa huruf angka, dll (diapit tanda " atau ')
Integer	25 atau 1209	Menyatakan bilangan bulat
Float	3.14 atau 0.99	Menyatakan bilangan yang mempunyai koma
Hexadecimal	9a atau 1d3	Menyatakan bilangan dalam format heksa (bilangan berbasis 16)
Complex	1 + 5j	Menyatakan pasangan angka real dan imajiner
List	['xyz', 786, 2.23]	Data untaian yang menyimpan berbagai tipe data dan isinya bisa diubah-ubah

Tuple	('xyz', 768, 2.23)	Data untaian yang menyimpan berbagai tipe data tapi isinya tidak bisa diubah
Dictionary	{'nama': 'adi', 'id': 2}	Data untaian yang menyimpan berbagai tipe data berupa pasangan penunjuk dan nilai

### 5.6. Input dan Output pada Python

Python menyediakan banyak fungsi *built-in* yang bisa kita gunakan. Salah satunya adalah yang berkenaan dengan fungsi i/o atau input output. Fungsi bawaan untuk melakukan operasi output adalah **print()**, dan fungsi untuk melakukan operasi input adalah fungsi **input()**.

### 5.7. Input pada Python

Python 2 memiliki dua fungsi built-in untuk membaca data dari input standar, yang secara default berasal dari keyboard. Fungsi ini adalah `input()` dan `raw_input()`. Dengan Python 3, fungsi `raw_input()` tidak digunakan lagi. Selain itu, `input()` berfungsi membaca data dari keyboard sebagai string, terlepas dari apakah itu tertutup dengan tanda kutip (‘ atau ’) atau tidak. Perhatikan contoh dibawah ini :

#### 1. Jika variable dengan tipe data string

```
nim=input("masukkan NIM anda")
```

#### 2. Jika variable dengan tipe data Int

```
angka=int(input("Masukkan angka : "))
```

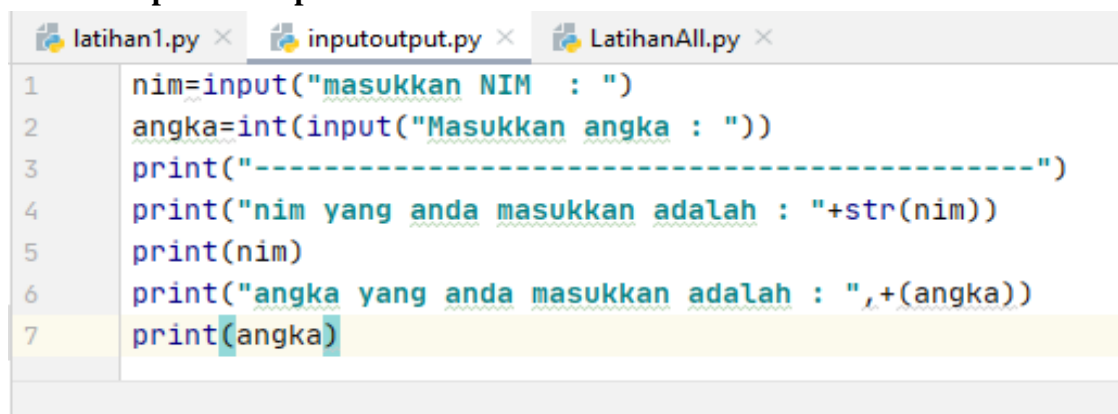
### 5.8. Output pada Python

Cara termudah untuk menghasilkan output adalah dengan menggunakan pernyataan cetak di mana Anda bisa melewati nol atau lebih banyak ekspresi yang dipisahkan dengan koma. Fungsi ini mengubah ekspresi yang Anda berikan ke string dan menulis hasilnya ke output standar sebagai berikut :

```
print ("Python adalah bahasa pemrograman yang hebat")
```

### Mencetak Nilai Variabel pada Python (Print)

### 5.9. Latihan Input & Output



```

latihan1.py x inputoutput.py x LatihanAll.py x
1  nim=input("masukkan NIM : ")
2  angka=int(input("Masukkan angka : "))
3  print("-----")
4  print("nim yang anda masukkan adalah : "+str(nim))
5  print(nim)
6  print("angka yang anda masukkan adalah : ",+(angka))
7  print(angka)

```

## Data Diri Mahasiswa

---

Masukkan NIM anda : <input>

Masukkan Nama anda : <input>

Jurusan : <input>

Alamat : <input>

```
"E:\OneDrive - Bina Sarana Informatika\0 - KAPRODI SI\Buku Ajar\Taha
Data Diri Mahasiswa
```

```
-----
masukkan NIM anda : 12204455
```

```
Masukkan Nama anda : sandi
```

```
Jurusan      : sistem informasi
```

```
Alamat       : karawang
```

```
-----
Hasil cetak data diatas adalah :
```

```
Nim anda adalah : 12204455 dengan nama : sandi
```

```
Jurusan anda : sistem informasi anda berasal dari : karawang
```

```
Process finished with exit code 0
```

```
|
```

## Pertemuan 3

### Operator Logika

Operator adalah simbol tertentu yang digunakan untuk melakukan operasi aritmatika maupun logika. Nilai yang padanya dilakukan operasi disebut operand. Misalnya adalah  $2 + 3$ . Di sini tanda  $+$  adalah operator penjumlahan. 2 dan 3 adalah operand.

Python memiliki sejumlah operator, yaitu:

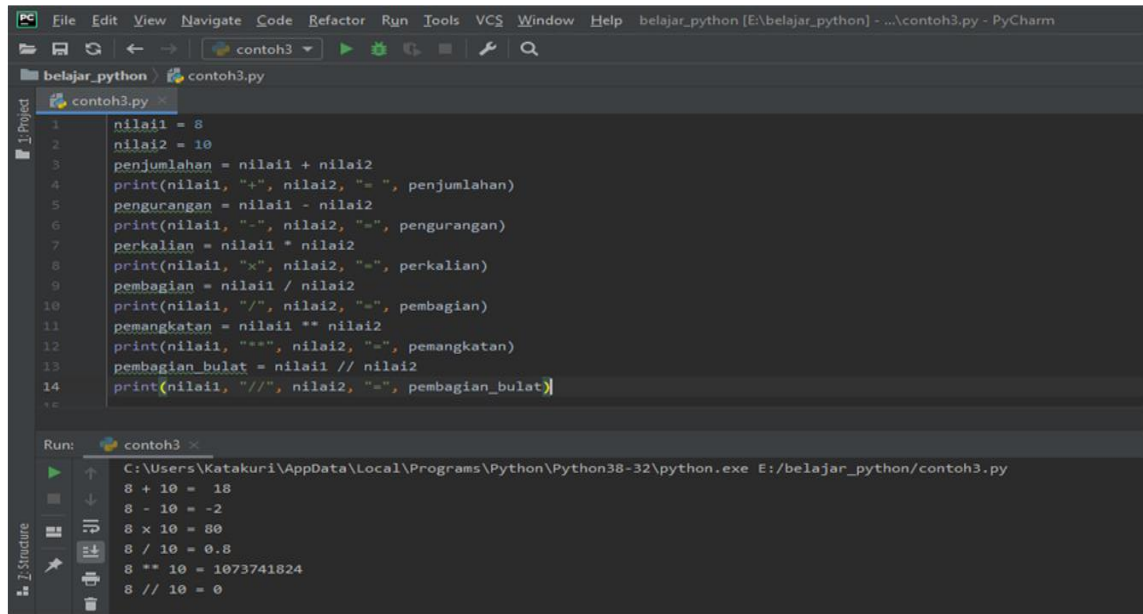
- Operator Aritmatika
- Operator Perbandingan
- Operator Penugasan
- Operator Logika
- Operator Bitwise
- Operator Identitas
- Operator Keanggotaan

#### 5.1. Operator Aritmatika

Operator aritmatika adalah operator yang digunakan untuk melakukan operasi matematika, seperti penjumlahan, pengurangan, perkalian, pembagian, dan sebagainya. Tabel berikut menunjukkan jenis operator aritmatika.

Tabel 3.1 Operator Aritmatika

Operator	Nama dan Fungsi	Contoh
+	Penjumlahan, menjumlahkan 2 buah operand	$a + b$
-	Pengurangan, mengurangi 2 buah operand	$a - b$
*	Perkalian, mengalikan 2 buah operand	$a * b$
/	Pembagian, membagi 2 buah operand	$a / b$
**	Pemangkatan, mengangkat bilangan	$a ** b$
//	Pembagian bulat, menghasilkan hasil bagi tanpa koma	$a // b$



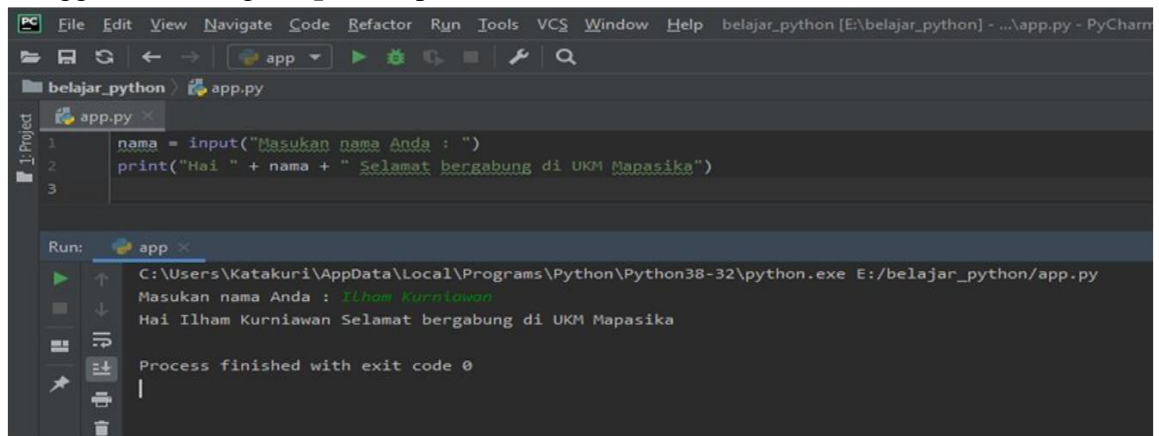
The screenshot shows the PyCharm IDE with a file named `contoh3.py` open. The code performs various arithmetic operations on two variables, `nilai1` and `nilai2`, which are both initialized to 8 and 10 respectively. The operations include addition, subtraction, multiplication, division, exponentiation, and floor division. The results are printed to the console. Below the code editor, the 'Run' window shows the command used to execute the script and the corresponding output.

```
1 nilai1 = 8
2 nilai2 = 10
3 penjumlahan = nilai1 + nilai2
4 print(nilai1, "+", nilai2, "=", penjumlahan)
5 pengurangan = nilai1 - nilai2
6 print(nilai1, "-", nilai2, "=", pengurangan)
7 perkalian = nilai1 * nilai2
8 print(nilai1, "x", nilai2, "=", perkalian)
9 pembagian = nilai1 / nilai2
10 print(nilai1, "/", nilai2, "=", pembagian)
11 pemangkatan = nilai1 ** nilai2
12 print(nilai1, "**", nilai2, "=", pemangkatan)
13 pembagian_bulat = nilai1 // nilai2
14 print(nilai1, "//", nilai2, "=", pembagian_bulat)
```

Run: `contoh3`

```
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh3.py
8 + 10 = 18
8 - 10 = -2
8 x 10 = 80
8 / 10 = 0.8
8 ** 10 = 1073741824
8 // 10 = 0
```

Kita juga bisa menentukan suatu variabel sesuai dengan keinginan kita dengan menggunakan fungsi **input()** seperti contoh berikut :



The screenshot shows the PyCharm IDE with a file named `app.py` open. The code prompts the user to enter their name and then prints a greeting message. The 'Run' window shows the command used to execute the script and the output, including the user's input.

```
1 nama = input("Masukan nama Anda : ")
2 print("Hai " + nama + " Selamat bergabung di UKM Mapasika")
3
```

Run: `app`

```
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/app.py
Masukan nama Anda : Ilham Kurniawan
Hai Ilham Kurniawan Selamat bergabung di UKM Mapasika
Process finished with exit code 0
```

Jika karakter yang akan kita input merupakan sebuah integer maka kita juga harus menambahkan fungsi **int()** seperti pada contoh berikut :

```

1  nilai1 = int(input("Masukan Nilai Pertama : "))
2  nilai2 = int(input("Masukan Nilai Kedua : "))
3  penjumlahan = nilai1 + nilai2
4  print(nilai1, "+", nilai2, "=", penjumlahan)
5  pengurangan = nilai1 - nilai2
6  print(nilai1, "-", nilai2, "=", pengurangan)
7  perkalian = nilai1 * nilai2
8  print(nilai1, "x", nilai2, "=", perkalian)
9  pembagian = nilai1 / nilai2
10 print(nilai1, "/", nilai2, "=", pembagian)
11 pemangkatan = nilai1 ** nilai2
12 print(nilai1, "**", nilai2, "=", pemangkatan)
13 pembagian_bulat = nilai1 // nilai2
14 print(nilai1, "//", nilai2, "=", pembagian_bulat)

```

Run: contoh3 ×

```

C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh3.py
Masukan Nilai Pertama : 8
Masukan Nilai Kedua : 8
8 + 8 = 16
8 - 8 = 0
8 x 8 = 64
8 / 8 = 1.0
8 ** 8 = 16777216
8 // 8 = 1

```

## 5.2. Operator Perbandingan

Operator perbandingan adalah operator yang digunakan untuk membandingkan 2 buah nilai. Hasil perbandingannya adalah True atau False tergantung kondisi.

Tabel 3.2 Operator Perbandingan

Operator	Nama dan Fungsi	Contoh
>	Lebih besar dari – Hasilnya True jika nilai sebelah kiri lebih besar dari nilai sebelah kanan	$a > b$
<	Lebih kecil dari – Hasilnya True jika nilai sebelah kiri lebih kecil dari nilai sebelah kanan	$a < b$
==	Sama dengan – Hasilnya True jika nilai sebelah kiri sama dengan nilai sebelah kanan	$a == b$
!=	Tidak sama dengan – Hasilnya True jika nilai sebelah kiri tidak sama dengan nilai sebelah kanan	$a != b$

>=	Lebih besar atau sama dengan – Hasilnya True jika nilai sebelah kiri lebih besar atau sama dengan nilai sebelah kanan	a >= b
<=	Lebih kecil atau sama dengan – Hasilnya True jika nilai sebelah kiri lebih kecil atau sama dengan nilai sebelah kanan	a <= b

The screenshot shows the PyCharm IDE with a file named `contoh4.py`. The code defines two variables, `nilai1` and `nilai2`, and performs several comparisons using relational operators. The output window shows the results of these comparisons for the input values 8 and 9.

```

1  nilai1 = int(input("Masukan Nilai Pertama : "))
2  nilai2 = int(input("Masukan Nilai Kedua : "))
3  operator_lebih_besar = nilai1 > nilai2
4  print(nilai1, ">", nilai2, "adalah", operator_lebih_besar)
5  operator_lebih_kecil = nilai1 < nilai2
6  print(nilai1, "<", nilai2, "adalah", operator_lebih_kecil)
7  operator_sama_dengan = nilai1 == nilai2
8  print(nilai1, "==", nilai2, "adalah", operator_sama_dengan)
9  operator_tidak_sama_dengan = nilai1 != nilai2
10 print(nilai1, "!= ", nilai2, "adalah", operator_tidak_sama_dengan)

```

Run: contoh4

```

C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh4.py
Masukan Nilai Pertama : 8
Masukan Nilai Kedua : 9
8 > 9 adalah False
8 < 9 adalah True
8 == 9 adalah False
8 != 9 adalah True

```

### 5.3. Operator Penugasan

Operator penugasan adalah operator yang digunakan untuk memberi nilai ke variabel. `a = 7` adalah contoh operator penugasan yang memberi nilai 7 di kanan ke variabel `a` yang ada di kiri.

Tabel 3.3 Operator Penugasan

Operator	Penjelasan	Contoh
=	Menugaskan nilai yang ada di kanan ke operand yang ada di sebelah kiri	<code>c = a + b</code> menugaskan <code>a + b</code> ke <code>c</code>
+=	Menambahkan operand yang di kanan dengan operand yang ada di kiri dan hasilnya di tugaskan ke operand yang	<code>c += a</code> sama dengan

	di kiri	$c = c + a$
<code>-=</code>	Mengurangi operand yang di kanan dengan operand yang ada di kiri dan hasilnya di tugaskan ke operand yang di kiri	$c -= a$ sama dengan $c = c - a$
<code>*=</code>	Mengalikan operand yang di kanan dengan operand yang ada di kiri dan hasilnya di tugaskan ke operand yang di kiri	$c *= a$ sama dengan $c = c * a$
<code>/=</code>	Membagi operand yang di kanan dengan operand yang ada di kiri dan hasilnya di tugaskan ke operand yang di kiri	$c /= a$ sama dengan $c = c / a$
<code>**=</code>	Memangkatkan operand yang di kanan dengan operand yang ada di kiri dan hasilnya ditugaskan ke operand yang di kiri	$c **= a$ sama dengan $c = c ** a$
<code>//=</code>	Melakukan pembagian bulat operand di kanan terhadap operand di kiri dan hasilnya disimpan di operand yang di kiri	$c //= a$ sama dengan $c = c // a$
<code>%=</code>	Melakukan operasi sisa bagi operand di kanan dengan operand di kiri dan hasilnya di simpan di operand yang di kiri	$c \% = a$ sama dengan $c = c \% a$

#### 5.4. Operator Logika

Operator logika adalah operator yang digunakan untuk melakukan operasi logika

Tabel 3.4 Operator Logika

Operator	Penjelasan	Contoh
<code>and</code>	Hasilnya adalah True jika kedua operandnya bernilai benar	<code>a and b</code>



or	Hasilnya adalah True jika salah satu atau kedua operandnya bernilai benar	a or b
not	Hasilnya adalah True jika operandnya bernilai salah (kebalikan nilai)	not a

### 5.5. Operator Bitwise

Operator bitwise adalah operator yang melakukan operasi bit terhadap operand. Operator ini beroperasi bit per bit sesuai dengan namanya. Sebagai misal, angka 2 dalam bit ditulis 10 dalam notasi biner dan angka 7 ditulis 111. Pada tabel di bawah ini, misalkan a = 10 (0000 1010) dalam biner dan b = 4 (0000 0100) dalam biner.

Tabel 3.5 Operator Bitwise

Operator	Nama	Contoh
&	Bitwise AND	a & b = 0 (0000 0000)
	Bitwise OR	a   b = 14 (0000 1110)
~	Bitwise NOT	~a = -11 (1111 0101)
^	Bitwise XOR	a ^ b = 14 (0000 1110)
>>	Bitwise right shift	a >> 2 = 2 (0000 0010)
<<	Bitwise left shift	b << 2 = 40 (0010 1000)

### 5.6. Operator Identitas

Operator identitas adalah operator yang memeriksa apakah dua buah nilai ( atau variabel ) berada pada lokasi memori yang sama.

Tabel 3.6 Operator Identitas

Operator	Penjelasan	Contoh
----------	------------	--------

is	True jika kedua operand identik (menunjuk ke objek yang sama)	a is True
is not	True jika kedua operand tidak identik (tidak merujuk ke objek yang sama)	a is not True

### 5.7. Operator Keanggotaan

Operator keanggotaan adalah operator yang digunakan untuk memeriksa apakah suatu nilai atau variabel merupakan anggota atau ditemukan di dalam suatu data (string, list, tuple, set, dan dictionary)

Tabel 3.7 Operator Keanggotaan

Operator	Penjelasan	Contoh
in	True jika nilai/variabel ditemukan di dalam data	5 in a
not in	True jika nilai/variabel tidak ada di dalam data	5 not in a

### 5.8. Latihan 1

- Buatlah 1 Contoh Operator Penugasan
- Buatlah 1 Contoh Operator Logika
- Buatlah 1 Contoh Operator Bitwise
- Buatlah 1 Contoh Operator Identitas
- Buatlah 1 Contoh Operator Keanggotaan

### 5.9. Latihan 2

- Buatlah program seperti gambar dibawah ini

```

Run: Tugas01 x
E:\projek_pyhton\Dasar_Pemrograman\venv\Scripts\python.exe E:/proj
TOKO MAINAN ANAK
*****
Masukan Nama Pembeli : Ilham Kurniawan
Masukan Kode Mainan : MAIN-0909
Masukan Harga : 30000
Masukan Jumlah Beli : 10
=====
Nama Pembeli = Ilham Kurniawan
Kode Kue   = MAIN-0909
Harga      = 30000
Jumlah Beli = 10
Total      = 300000

```

## Pertemuan 4

### Percabangan

Percabangan adalah cara yang digunakan untuk mengambil keputusan apabila di dalam program dihadapkan pada kondisi tertentu. Jumlah kondisinya bisa satu, dua atau lebih. Percabangan mengevaluasi kondisi atau ekspresi yang hasilnya benar atau salah. Kondisi atau ekspresi tersebut disebut ekspresi boolean.

Hasil dari pengecekan kondisi adalah True atau False. Bila benar (True), maka pernyataan yang ada di dalam blok kondisi tersebut akan dieksekusi. Bila salah (False), maka blok pernyataan lain yang dieksekusi.

Di Python ada 3 jenis pernyataan yang digunakan untuk percabangan, yaitu sebagai berikut:

Tabel 4.1 Tabel Percabangan

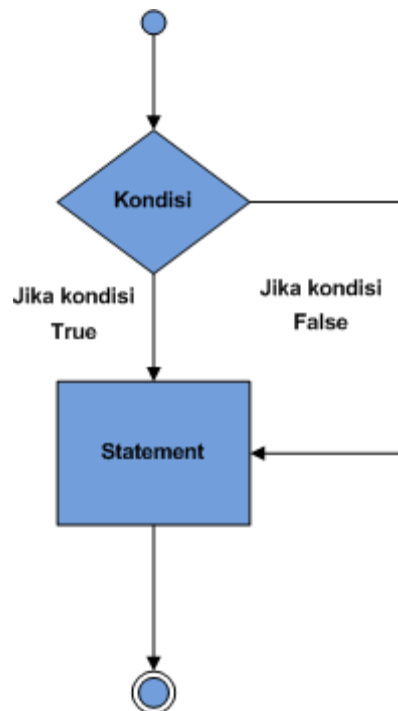
No	Pernyataan	Deskripsi
1	if	Pernyataan <b>if</b> terdiri dari ekspresi <i>boolean</i> diikuti oleh satu baris atau lebih pernyataan.
2	if...else	Bila pernyataan <b>if</b> benar, maka blok pernyataan <b>if</b> dieksekusi. Bila salah, maka blok pernyataan <b>else</b> yang dieksekusi.
3	if...elif...else	Disebut juga if bercabang. Bila ada kemungkinan beberapa kondisi bisa benar maka digunakan pernyataan <b>if...elif</b> atau <b>if...elif...else</b>

#### 3.1. Pernyataan if

Pernyataan if menguji satu buah kondisi. Bila hasilnya benar maka pernyataan di dalam blok if tersebut dieksekusi. Bila salah, maka pernyataan tidak dieksekusi. Sintaksnya adalah seperti berikut:

```
if tes kondisi:  
    blok pernyataan if
```

Gambar diagram alir untuk pernyataan if adalah seperti berikut:



Gambar 4.1 Diagram Alir Pernyataan if

The screenshot shows the PyCharm IDE interface. The top toolbar includes icons for File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The main editor window displays a file named `contoh05.py` with the following Python code:

```

1 # Bila bilangan positif, tampilkan pesan
2
3 angka = 5
4 if angka > 0:
5     print(angka, "adalah bilangan positif.")
6
7 angka = -1
8 # yang berikut akan bernilai False sehingga tidak dieksekusi
9 if angka > 0:
10    print(angka, "adalah bilangan positif.")
  
```

Below the editor, the 'Run' console shows the execution of `contoh05.py`. The command executed is `C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh05.py`. The output is `5 adalah bilangan positif.`, and the process finished with exit code 0.

Pada contoh di atas, awalnya angka berisi 5. Pada saat if yang pertama dieksekusi maka kondisinya adalah apakah  $5 > 0$ ? Karena hasilnya benar/True, maka statement di grup if ini dieksekusi dan menampilkan pesan 5 adalah bilangan positif.

Selanjutnya angka sudah diubah jadi -1. Untuk if yang kedua, hasil pengujian kondisinya menjadi apakah  $-1 > 0$ ? Hasilnya salah/False. Oleh karena itu, pernyataan di dalam grupnya tidak dijalankan.

### 3.2. Pernyataan if...else

Pernyataan if...else menguji 2 kondisi. Kondisi pertama kalau benar, dan kondisi kedua kalau salah. Sintaksnya adalah seperti berikut:

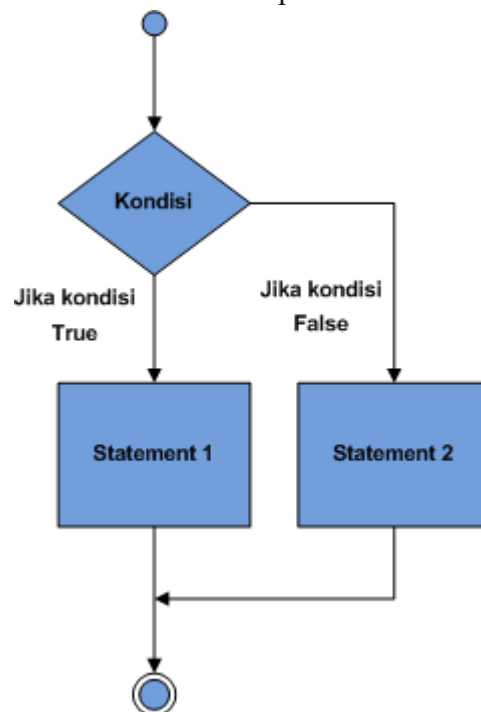
```
if tes kondisi:
```

```
    blok pernyataan if
```

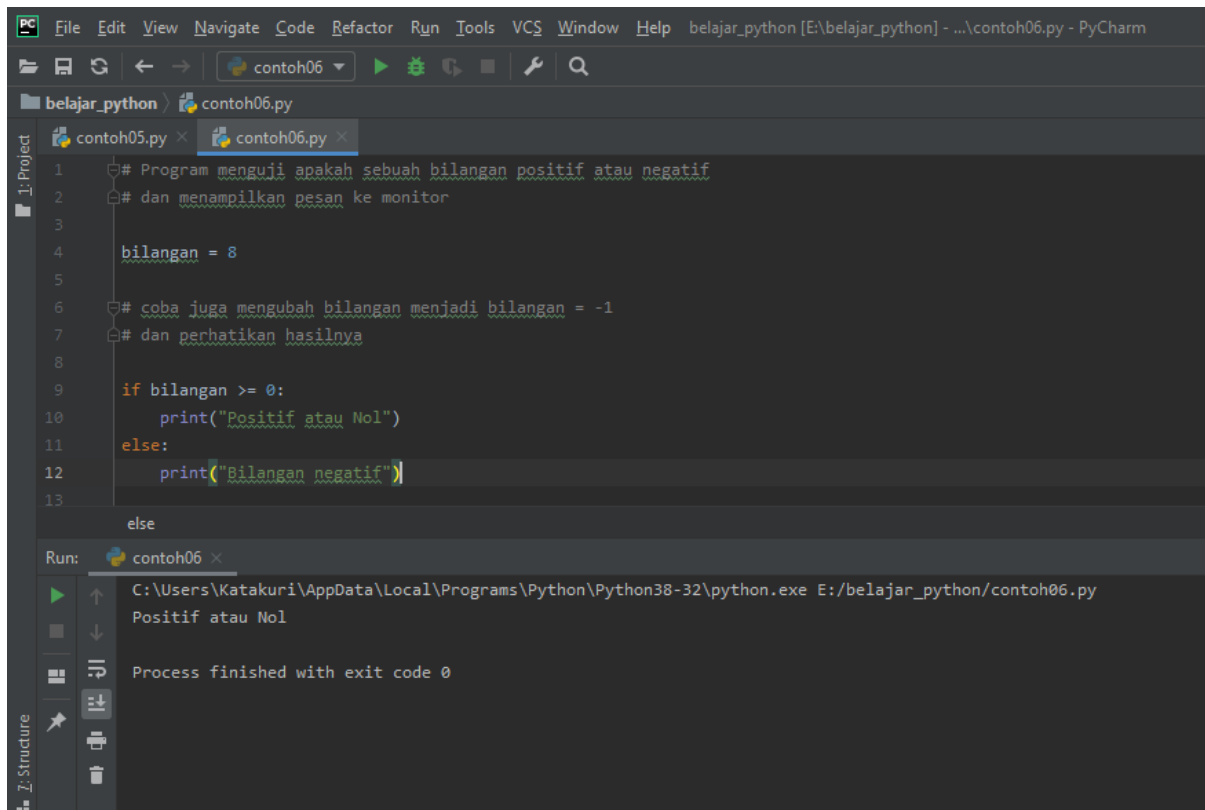
```
else:
```

```
    blok pernyataan else
```

Diagram alir untuk pernyataan if...else adalah seperti berikut:



Gambar 4.2. Diagram Alir Pernyataan if...else



Pada contoh di atas, bilangan kita beri nilai 8. Kemudian pada pengujian if, kondisinya adalah apakah bilangan  $\geq 0$  ? Hasilnya adalah benar, maka hasil yang ditampilkan adalah Positif atau Nol. Seandainya kita ganti bilangan jadi -1, maka hasil pengujian if nya akan salah/False dan blok else yang akan dijalankan, yaitu menampilkan pesan Bilangan negatif.

### 3.3. Pernyataan if...elif...else...

Pernyataan if...elif...else digunakan untuk menguji lebih dari 2 kondisi. Bila kondisi pada if benar, maka pernyataan di dalamnya yang dieksekusi. Bila salah, maka masuk ke pengujian kondisi elif. Terakhir bila tidak ada if atau elif yang benar, maka yang dijalankan adalah yang di blok else. Sintaksnya adalah seperti berikut:

```
if tes kondisi:
```

```
    blok pernyataan if
```

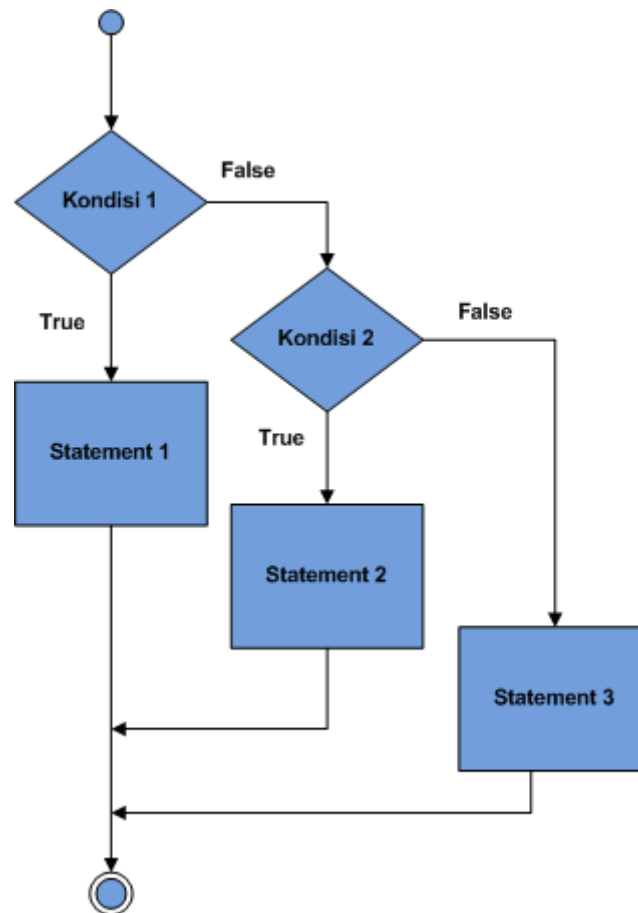
```
elif tes kondisi:
```

```
    blok pernyataan elif
```

```
else:
```

```
    blok pernyataan else
```

Diagram alir if...else...if adalah sebagai berikut:



Gambar 4.3 Diagram Alir Pernyataan if...elif...else

The screenshot shows the PyCharm IDE with a file named `contoh07.py`. The code in the editor is as follows:

```
1 '''Di sini kita menguji apakah sebuah bilangan adalah bilangan positif, nol, atau negatif -  
2 dan menampilkan hasilnya ke layar '''  
3  
4 bilangan = 5.5  
5  
6 '''Coba juga mengganti bilangan jadi  
7 bilangan = 0  
8 bilangan = -5.5 '''  
9  
10 if bilangan > 0:  
11     print("Bilangan positif")  
12 elif bilangan == 0:  
13     print("Nol")  
14 else:  
15     print("Bilangan negatif")
```

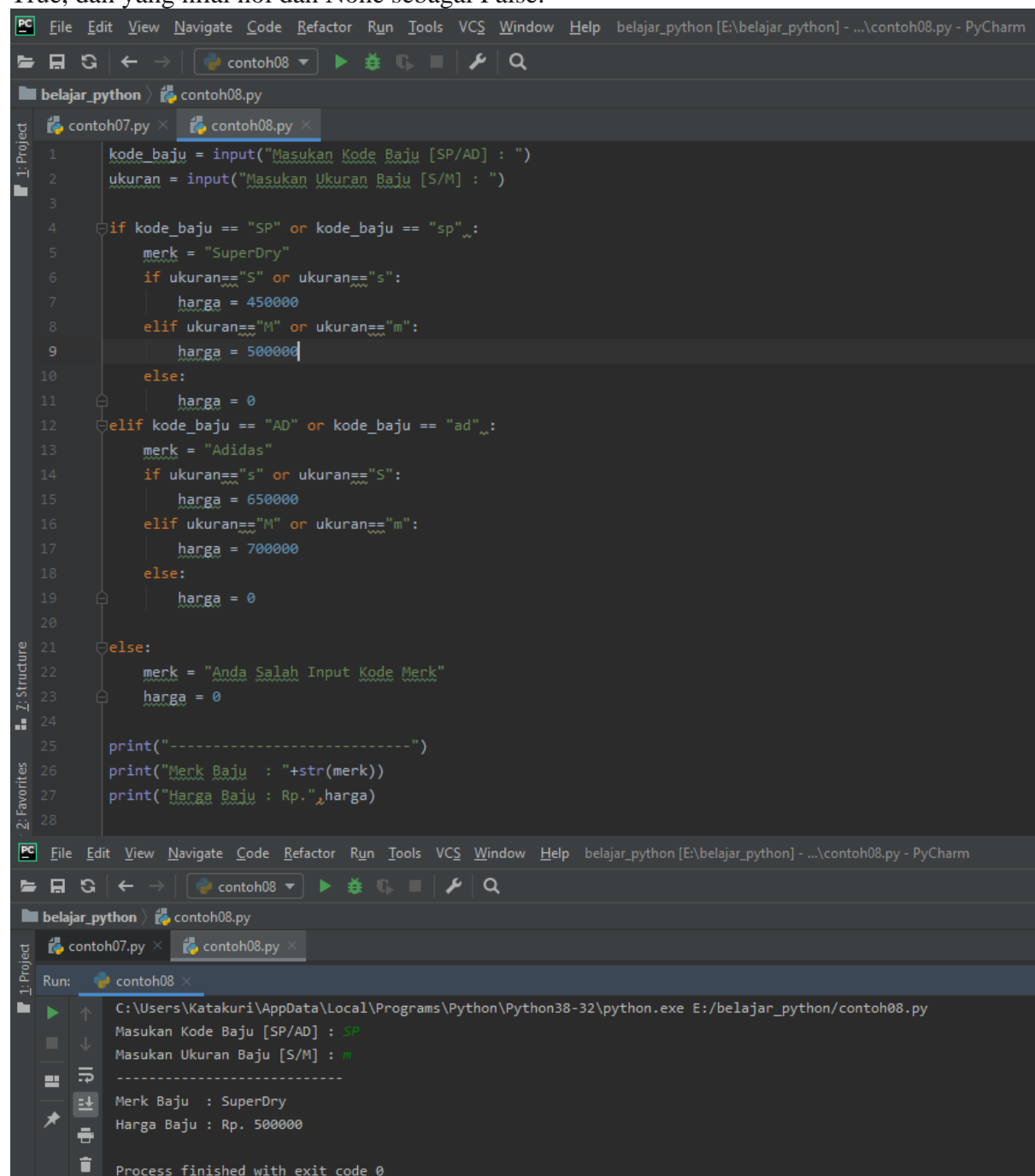
Below the editor, the Run console shows the execution of the script:

```
Run: contoh07 x  
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh07.py  
Bilangan positif  
Process finished with exit code 0
```

Pada contoh di atas, bilangan kita beri nilai 5.5. Pada pengujian if, kondisinya adalah apakah bilangan  $> 0$ ? Hasilnya benar, maka yang ditampilkan adalah pesan Bilangan positif.

Bila nilai bilangan kita ganti menjadi 0, maka yang akan bernilai benar adalah pernyataan elif. Bila kita mengganti bilangan jadi minus, maka kondisi if dan elif salah, dan yang dijalankan adalah blok else.

Catatan: Python mengasumsikan bahwa nilai selain nol dan selain tipe None sebagai nilai True, dan yang nilai nol dan None sebagai False.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh08.py - PyCharm
contoh08
1 kode_baju = input("Masukan Kode Baju [SP/AD] : ")
2 ukuran = input("Masukan Ukuran Baju [S/M] : ")
3
4 if kode_baju == "SP" or kode_baju == "sp":
5     merk = "SuperDry"
6     if ukuran=="S" or ukuran=="s":
7         harga = 450000
8     elif ukuran=="M" or ukuran=="m":
9         harga = 500000
10    else:
11        harga = 0
12 elif kode_baju == "AD" or kode_baju == "ad":
13     merk = "Adidas"
14     if ukuran=="s" or ukuran=="S":
15         harga = 650000
16     elif ukuran=="M" or ukuran=="m":
17         harga = 700000
18     else:
19         harga = 0
20
21 else:
22     merk = "Anda Salah Input Kode Merk"
23     harga = 0
24
25 print("-----")
26 print("Merk Baju : "+str(merk))
27 print("Harga Baju : Rp. "+str(harga))
28
Run: contoh08
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh08.py
Masukan Kode Baju [SP/AD] : SP
Masukan Ukuran Baju [S/M] : M
-----
Merk Baju : SuperDry
Harga Baju : Rp. 500000
Process finished with exit code 0
```

## Studi Kasus : Penjualan Tiket

Perusahaan XYZ bergerak dibidang penjualan tiket bus dengan detail tiket sebagai berikut :



1. setiap transaksi perlu menginput data pembeli seperti Nama Pembeli, jumlah tiket yang akan dibeli, no\_Hp, dan memilih jurusan sesuai kategori yang diinginkan.
2. Potongan 10% didapat jika jumlah beli  $\geq 3$ .
3. Totalharga=(jumlahbeli\*harga)-potongan.

Data harga tiket sebagai berikut :

Kodejurusan	Nama Kota	Harga
SBY	surabaya	300,000
BL	Bali	350,000
LMP	lampung	500,000

Buatlah program input dan hasil output sesuai perintah diatas menggunakan Bahasa pemrograman Python.

Masukan

```

penjualantiket x
"E:\OneDrive - Bina Sarana Informatika\Python\penjualantiket.py"
Input Nama Pembeli : sopandi
Input No. Handphone : 082213445567
Input Jurusan [SBY/BL/LMP] : SBY
Masukkan Jumlah Beli : 3

```

Keluaran

```

penjualantiket x
-----
                PENJUALAN TIKET BUS
                XYZ
-----
Nama Pembeli   : sopandi
No. Handphone  : 082213445567
Kode Jurusan yang dipilih : SBY
Nama Kota Tujuan : Surabaya
Harga          : 300000
Jumlah Beli    : 3
-----
potongan yang didapat : 90000.0
Total Bayar       : 810000.0
Masukkan Uang Bayar : 1000000
Uang Kembali      : 190000.0

Process finished with exit code 0
|

```

```

#Input
pembeli=input("Input Nama Pembeli : ")
no_hp=input("Input No. Handphone : ")
jurusan=input("Input Jurusan [SBY/BL/LMP] : ")
#Proses
if jurusan=="SBY":
    namajurusan="Surabaya"
    harga=300000
elif jurusan=="BL":
    namajurusan="Bali"
    harga=350000
else :
    namajurusan="Lampung"
    harga=500000

#Input Jumlah Beli
jumlah=int(input("Masukkan Jumlah Beli : "))

#proses potongan
if jumlah>=3 :
    potongan=(jumlah*harga)*0.1
else:
    potongan=0

total=(jumlah*harga)-potongan

#Cetak Hasil
print("-----")
print("          PENJUALAN TIKET BUS")
print("          XYZ")
print("-----")
print("Nama Pembeli   : "+str(pembeli))
print("No. Handphone  : "+str(no_hp))
print("Kode Jurusan yang dipilih : "+str(jurusan))
print("Nama Kota Tujuan : "+str(namajurusan))
print("Harga          : ",+(harga))
print("Jumlah Beli    : ",+(jumlah))
print("-----")
print("potongan yang didapat : ",+(potongan))
print("Total Bayar    : ",+(total))
ubay=int(input("Masukkan Uang Bayar : "))
uangkembali=ubay-total
print("Uang Kembali   : ",+uangkembali)

```

## Latihan Studi Kasus : Pendaftaran Mahasiswa Baru

1. setiap transaksi perlu menginput data calon mahasiswa seperti NIS,nama,jurusan
2. Data biaya kuliah sebagai berikut :

Kodejurusan	Nama Prodi	Harga
SI	Sistem Informasi	2,400,000
SIA	Sistem Informasi AKuntansi	2,000,000

Buatlah program input dan hasil output sesuai perintah diatas menggunakan Bahasa pemrograman Python.

### Tugas 1

PT. DINGIN DAMAI, memberi gaji pokok kepada karyawan kontraknya sebesar Rp. 300,000 perbulan, dengan memperoleh tunjangan-tunjangan sebagai berikut :

- **Tunjangan Jabatan**

Golongan	Persentase
1	5%
2	10%
3	15%

Logikanya : Jika seorang karyawan tersebut dengan golongan 3, maka mendapatkan tunjangan sebesar  $15\% \times \text{Rp. } 300,000$

- **Tunjangan Pendidikan**

Tingkat Pendidikan	Persentase
SMA	2.5%
D1	5%
D3	20%
S1	30%

Jika seorang karyawan tersebut dengan Tingkat Pendidikan S1, maka mendapatkan tunjangan pendidikan sebesar  $30\% \times \text{Rp. } 300,000$

### Honor Lembur

Jumlah jam kerja normal sebanyak 8 jam, Honor lembur diberikan jika jumlah jam kerja lebih dari 8 jam, maka kelebihan jam kerja tersebut dikalikan dengan Rp. 3500 untuk setiap kelebihan jam kerja karyawan tersebut. Tampilan yang diinginkan sebagai berikut :

## Layar Masukkan

### PROGRAM HITUNG GAJI KARYAWAN

Nama Karyawan: ...  
Golongan Jabatan : ...  
Pendidikan : ...  
Jumlah jam kerja : ...

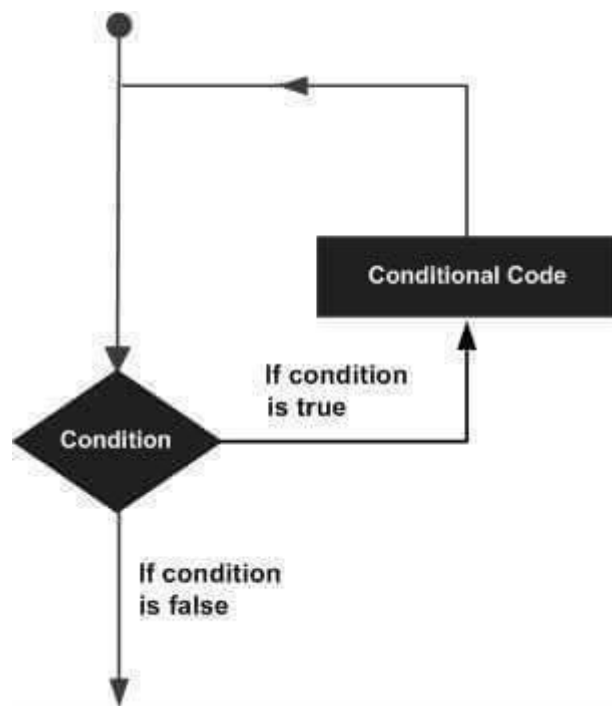
## Layar Keluaran

Karyawan yang bernama .....	
Honor yang diterima	
Tunjangan Jabatan	Rp ...
Tunjangan Pendidikan	Rp ...
Honor Lembur	Rp .....
	<hr/>
Total Gaji	Rp ...
(Gaji pokok + tunjangan + lembur)	

## Pertemuan 5

### Perulangan

Secara umum, Python mengeksekusi program baris perbaris. Mulai dari baris satu, dua, dan seterusnya. Ada kalanya, kita perlu mengeksekusi satu baris atau satu blok kode program beberapa kali. Hal ini disebut dengan perulangan atau biasa disebut looping atau iterasi. Untuk lebih jelasnya perhatikan gambar berikut:



Gambar 5.1 Diagram Alir Perulangan (looping)

Pada gambar bisa dilihat bahwa perulangan juga memerlukan tes kondisi. Bila hasil tes kondisi True, maka blok kode kembali dieksekusi. Tapi jika False, maka keluar dari perulangan. Pada python, perulangan bisa dilakukan dengan dua cara atau metode, yaitu:

1. Menggunakan for
2. Menggunakan while

#### 1.1. Perulangan dengan Menggunakan For

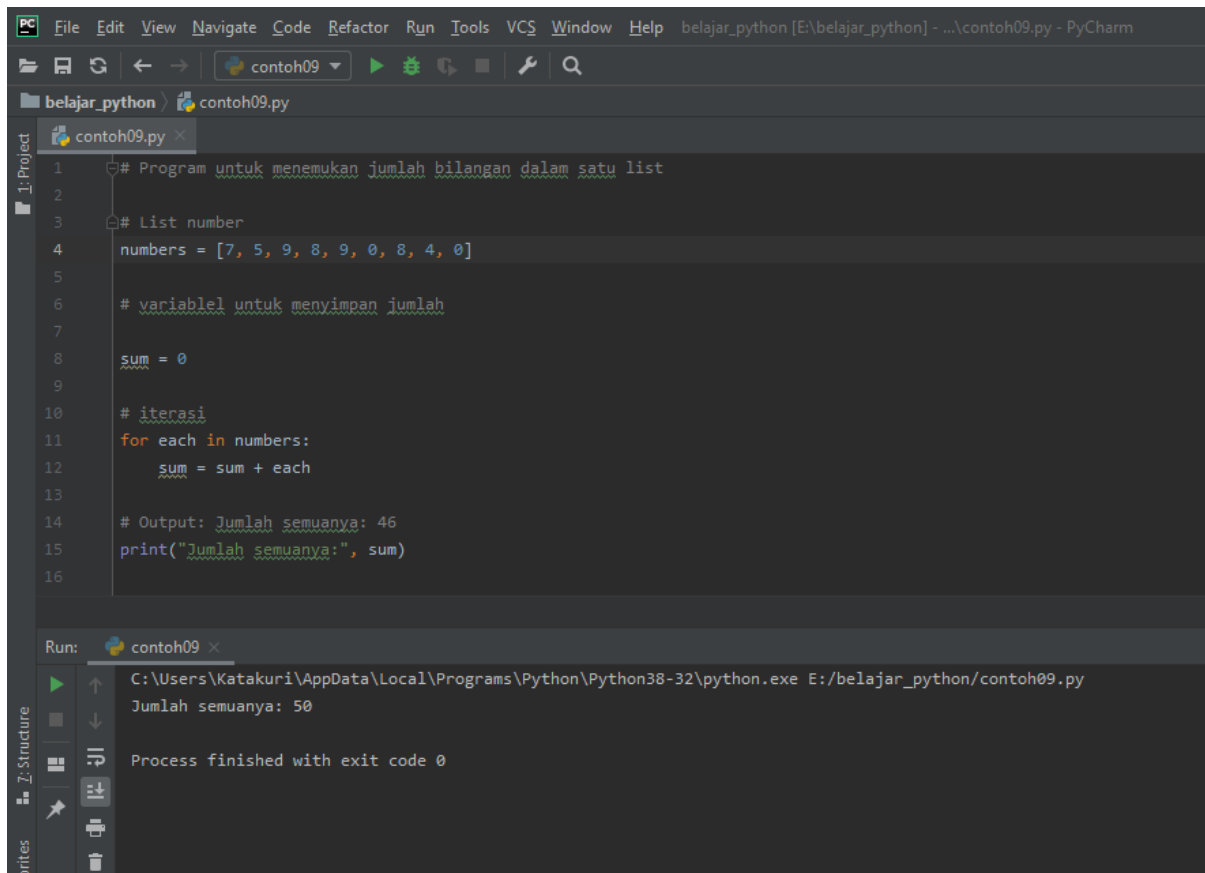
Perulangan dengan menggunakan for memiliki sintaks seperti berikut:

```
for var in sequence:
```

```
    body of for
```

var adalah variabel yang digunakan untuk penampung sementara nilai dari sequence pada saat terjadi perulangan. Sequence adalah tipe data berurut seperti string, list, dan tuple.

Perulangan terjadi sampai looping mencapai elemen atau anggota terakhir dari sequence. Bila loop sudah sampai ke elemen terakhir dari sequence, maka program akan keluar dari looping.



The screenshot shows the PyCharm IDE interface. The main editor window displays a Python file named `contoh09.py` with the following code:

```
1  # Program untuk menemukan jumlah bilangan dalam satu list
2
3  # List number
4  numbers = [7, 5, 9, 8, 9, 0, 8, 4, 0]
5
6  # variabel untuk menyimpan jumlah
7
8  sum = 0
9
10 # iterasi
11 for each in numbers:
12     sum = sum + each
13
14 # Output: Jumlah semuanya: 46
15 print("Jumlah semuanya:", sum)
16
```

The Run window at the bottom shows the execution of the script:

```
Run: contoh09 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh09.py
Jumlah semuanya: 50
Process finished with exit code 0
```

## 5.2. Fungsi range

Fungsi `range()` dapat digunakan untuk menghasilkan deret bilangan. `range(10)` akan menghasilkan bilangan dari 0 sampai dengan 9 (10 bilangan). Kita juga bisa menentukan batas bawah, batas atas, dan interval dengan format `range(batas bawah, batas atas, interval)`. Bila interval dikosongkan, maka nilai default 1 yang akan digunakan.

Fungsi `range` tidak menyimpan semua nilai dalam memori secara langsung. Ia hanya akan mengingat batas bawah, batas atas, dan interval dan membangkitkan hasilnya satu persatu hanya bila dipanggil. Untuk membuat fungsi ini langsung menampilkan semua item, kita bisa menggunakan fungsi `list()`. Untuk jelasnya perhatikan contoh berikut:

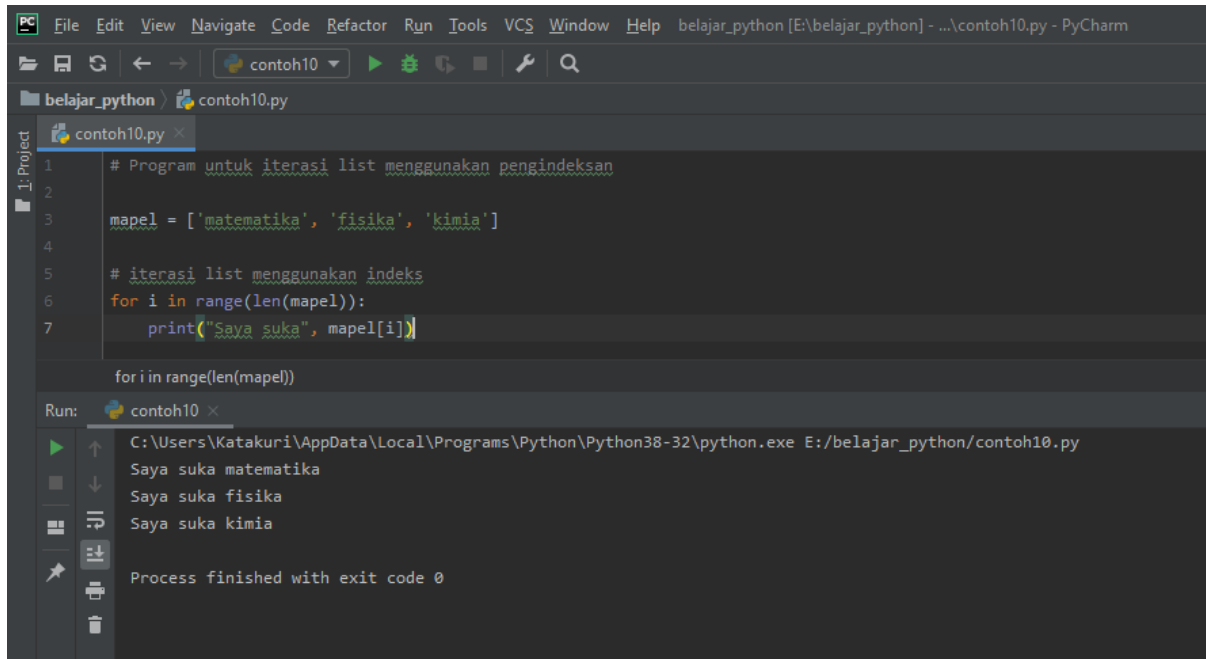
```
# Output: range(0,10)
print(range(10))
```

```
# Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(list(range(10)))
```

```
# Output: [2, 3, 4, 5, 6, 7]
print(list(range(2,8)))
```

```
# Output: [2, 5, 8, 11, 14, 17]
print(list(range(2, 20, 3)))
```

Kita bisa menggunakan fungsi `range()` dalam perulangan menggunakan `for` untuk iterasi bilangan berurut. Hal ini dengan cara mengkombinasikan fungsi `range()` dengan fungsi `len()`. Fungsi `len()` berfungsi untuk mendapatkan panjang atau jumlah elemen suatu data sekuensial atau berurut.



The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script named `contoh10.py` with the following code:

```
1 # Program untuk iterasi list menggunakan pengindeksan
2
3
4 mapel = ['matematika', 'fisika', 'kimia']
5
6 # iterasi list menggunakan indeks
7 for i in range(len(mapel)):
8     print("Saya suka", mapel[i])
```

Below the editor, the Run console shows the output of the script:

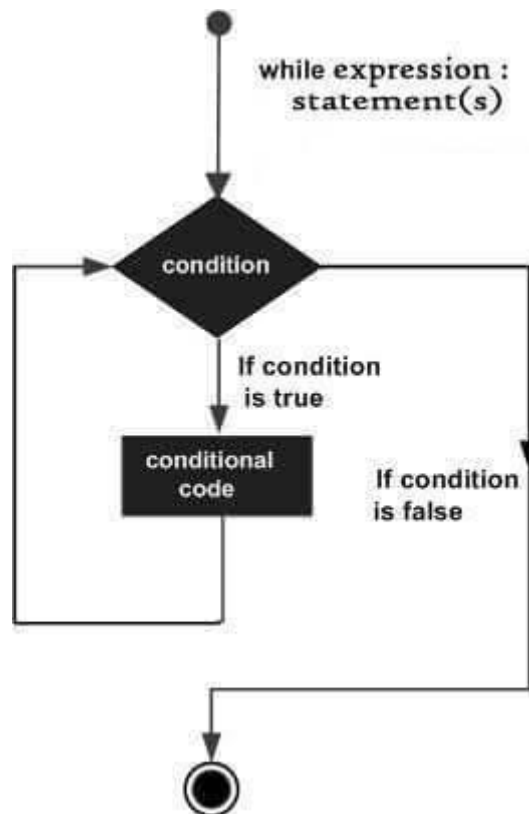
```
Run: contoh10 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh10.py
Saya suka matematika
Saya suka fisika
Saya suka kimia
Process finished with exit code 0
```

### 5.3. Perulangan Menggunakan While

Perulangan menggunakan `while` akan menjalankan blok pernyataan terus menerus selama kondisi bernilai benar. Adapun sintaks dari perulangan menggunakan `while` adalah:

```
while expression:
    statement (s)
```

Di sini, `statement (s)` bisa terdiri dari satu baris atau satu blok pernyataan. `Expression` merupakan ekspresi atau kondisi apa saja, dan untuk nilai selain nol dianggap `True`. Iterasi akan terus berlanjut selama kondisi benar. Bila kondisi salah, maka program akan keluar dari `while` dan lanjut ke baris pernyataan di luar `while`. Adapun diagram alir `while` adalah seperti gambar berikut:



Gambar 5.2. Diagram Alir Perulangan While

Perhatikan bahwa bila kondisi yang diuji bernilai salah, maka loop tidak akan pernah dieksekusi.

The screenshot shows the PyCharm IDE interface. The top toolbar includes icons for File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The main editor window displays a Python file named "contoh11.py" with the following code:

```
1 count = 0
2 while (count < 5):
3     print('The count is:', count)
4     count = count + 1
5     print('Good bye!')
```

Below the editor, the "Run" console shows the output of the program:

```
Run: contoh11 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh11.py
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
Good bye!
```

Di sini, blok pernyataan `print('The count is:', count)`, dijalankan terus selama `count` masih lebih kecil dari 5. `Count` ditambah 1 setiap kali iterasi. Pada saat nilai `count` mencapai 5, maka kondisi menjadi `False` dan program keluar dari looping `while` dan melanjutkan baris selanjutnya yaitu `print("Good bye")`.

#### 5.4. Infinite Loop



Sebuah kondisi dimana loop selalu benar dan tidak pernah salah disebut loop tidak terbatas (infinite loop). Terkadang hal ini menjadi masalah. Tapi sering juga infinite loop berguna, misalnya untuk program client/server dimana server perlu menjaga komunikasi tetap hidup dan tidak terputus. Pada contoh program while di atas, bila kita lupa menuliskan kode `count = count + 1`, maka akan jadi infinite loop. Hasilnya akan jadi seperti berikut:

```
The count is: 0
```

```
The count is: 0
```

```
The count is: 0
```

```
The count is: 0
```

```
The count is: 0
```

```
Traceback (most recent call last):
```

```
File "<pyshell#4>", line 2, in <module>
```

```
print('The count is:', count)
```

```
File "C:\Python34\lib\idlelib\PyShell.py", line 1344, in write
```

```
return self.shell.write(s, self.tags)
```

```
KeyboardInterrupt
```

Kita perlu menekan CTRL+C untuk menghentikan program.

### 5.5. Kendali Looping

Looping umumnya akan berhenti bila kondisi sudah bernilai salah. Akan tetapi, seringkali kita perlu keluar dari looping di tengah jalan tergantung keperluan. Hal ini bisa kita lakukan dengan menggunakan kata kunci **break** dan **continue**.

Statement **break** memaksa program keluar dari blok looping di tengah jalan. Sedangkan statement **continue** menyebabkan program langsung melanjut ke step / interval berikutnya dan mengabaikan (skip) baris kode di bawahnya (yang satu blok). Jelasnya perhatikan contoh berikut:

```
1 # contoh penggunaan statement break
2 for letter in "PythonProgramming":
3     if letter == "g":
4         break
5     print("Huruf sekarang:", letter)
6 print("Good bye")
```

for letter in "PythonProgrammin...

Run: contoh12 x

C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar\_python/contoh12.py

Huruf sekarang: P  
Huruf sekarang: y  
Huruf sekarang: t  
Huruf sekarang: h  
Huruf sekarang: o  
Huruf sekarang: n  
Huruf sekarang: P  
Huruf sekarang: r  
Huruf sekarang: o  
Good bye

Bila pada program di atas kita ganti kode **break** menjadi **continue**, maka hasilnya akan jadi seperti berikut:

```
1 # contoh penggunaan statement break
2 for letter in "PythonProgramming":
3     if letter == "g":
4         continue
5     print("Huruf sekarang:", letter)
6 print("Good bye")
```

for letter in "PythonProgrammin...

Run: contoh12 x

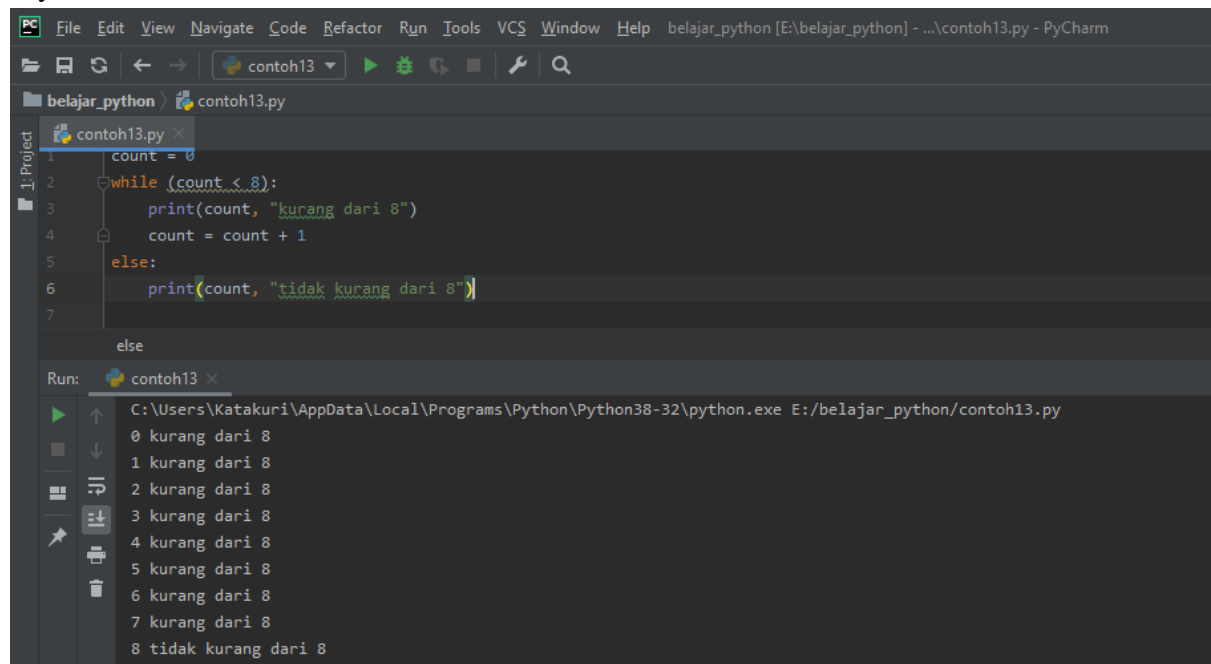
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar\_python/contoh12.py

Huruf sekarang: P  
Huruf sekarang: y  
Huruf sekarang: t  
Huruf sekarang: h  
Huruf sekarang: o  
Huruf sekarang: n  
Huruf sekarang: P  
Huruf sekarang: r  
Huruf sekarang: o  
Huruf sekarang: r  
Huruf sekarang: a  
Huruf sekarang: m  
Huruf sekarang: m  
Huruf sekarang: i  
Huruf sekarang: n  
Good bye

Perhatikan bahwa huruf **g** tidak pernah ditampilkan karena diabaikan karena kode **continue**.

## 5.6. While else

Python mendukung penggunaan else sebagai pasangan dari while. Blok pernyataan else hanya akan dieksekusi bila kondisi while bernilai salah.



The screenshot shows the PyCharm IDE with a file named `contoh13.py` open. The code in the editor is as follows:

```
1 count = 0
2 while (count < 8):
3     print(count, "kurang dari 8")
4     count = count + 1
5 else:
6     print(count, "tidak kurang dari 8")
7
8 else
```

Below the editor, the 'Run' window shows the execution of the script. The output is:

```
0 kurang dari 8
1 kurang dari 8
2 kurang dari 8
3 kurang dari 8
4 kurang dari 8
5 kurang dari 8
6 kurang dari 8
7 kurang dari 8
8 tidak kurang dari 8
```

## Latihan

Lakukan pengulangan input data sebanyak 2 kali dengan data dibawah ini :

Data Ke- <berulang>

Masukkan NIM anda : <Input Data Ke 1>

Masukkan Nilai UTS : <Input Data Ke 1>

Masukkan Nilai UAS : <Input Data Ke 1>

Nim anda adalah <outputnim1> nilai UTS anda <outpututs1> nilai UTS anda <outputuas1>

## Kode Program

```
ulang=2
for i in range(ulang):
    print ("data Ke - " + str(i+1))
    nama=input("Masukkan Nim anda : ")
    uts=int(input("Masukkan Nilai UTS anda : "))
    uas=int(input("Masukkan Nilai UAS : "))
    print("Nim anda adalah %s nilai UTS anda %i nilai UTS anda
%i" % (nama,uts,uas))
    print("-----\n")
```

## Output

```
perulangan x
"E:\OneDrive - Bina Sarana Informatika\0 - KAPRODI SI\Buku Ajar\Tah:
data Ke - 1
Masukkan Nim anda : 12101707
Masukkan Nilai UTS anda : 90
Masukkan Nilai UAS : 90
Nim anda adalah 12101707 nilai UTS anda 90 nilai UTS anda 90
-----

data Ke - 2
Masukkan Nim anda : 12181918
Masukkan Nilai UTS anda : 80
Masukkan Nilai UAS : 80
Nim anda adalah 12181918 nilai UTS anda 80 nilai UTS anda 80
-----

Process finished with exit code 0
|
```

## Tugas 2

Sebuah perusahaan ayam goreng dengan nama “**GEROBAK FRIED CHICKEN**” yang telah lumayan banyak pelanggannya, ingin dibantu dibuatkan program untuk membantu kelancaran usahanya.

“**GEROBAK FRIED CHICKEN**” mempunyai daftar harga ayam sebagai berikut :

Kode JenisPotong Harga

```
-----
D    Dada Rp. 2500
P    Paha Rp. 2000
S    Sayap Rp. 1500
-----
```

Buatlah programnya dengan ketentuan:

- Setiap pembeli dikenakan pajak sebesar 10% dari pembayaran.
- Banyak Jenis, Jenis Potong dan Banyak Beli diinput.
- Tampilan yang diinginkan sebagai berikut:

### Layar Masukkan

#### GEROBAK FRIED CHICKEN

Kode JenisPotong    Harga

D	Dada	Rp. 2500
P	Paha	Rp. 2000
S	Sayap	Rp. 1500

Banyak Jenis : ... <diinput>

Jenis Ke - ... <proses counter>

Kode Potong [D/P/S] : ... <diinput>

Banyak Potong : ... <diinput>

<<Terus berulang tergantung Banyak Jenis>>

### Layar Keluaran

#### GEROBAK FIRED CHICHEN

No.	Jenis Potong	Harga Satuan	Bayak Beli	Jumlah Harga
-----	--------------	--------------	------------	--------------

...	.....	....	....	Rp ....
-----	-------	------	------	---------

...	.....	....	....	Rp ....
-----	-------	------	------	---------

Jumlah Bayar Rp ....

Pajak 10%    Rp ....

Total Bayar    Rp ....

## Pertemuan 6

### List & Tuple

#### 3.1. List

Python menyediakan sejumlah tipe data yang dikenal dengan tipe data berurut (sequence). List adalah tipe data yang berisi satu atau beberapa nilai di dalamnya. Nilai – nilai ini sering juga disebut item, elemen, atau anggota list. List dibuat dengan menempatkan semua item di dalam tanda kurung [ ], dipisahkan oleh tanda koma. Anggota list bisa berisi satu tipe data, atau campuran.

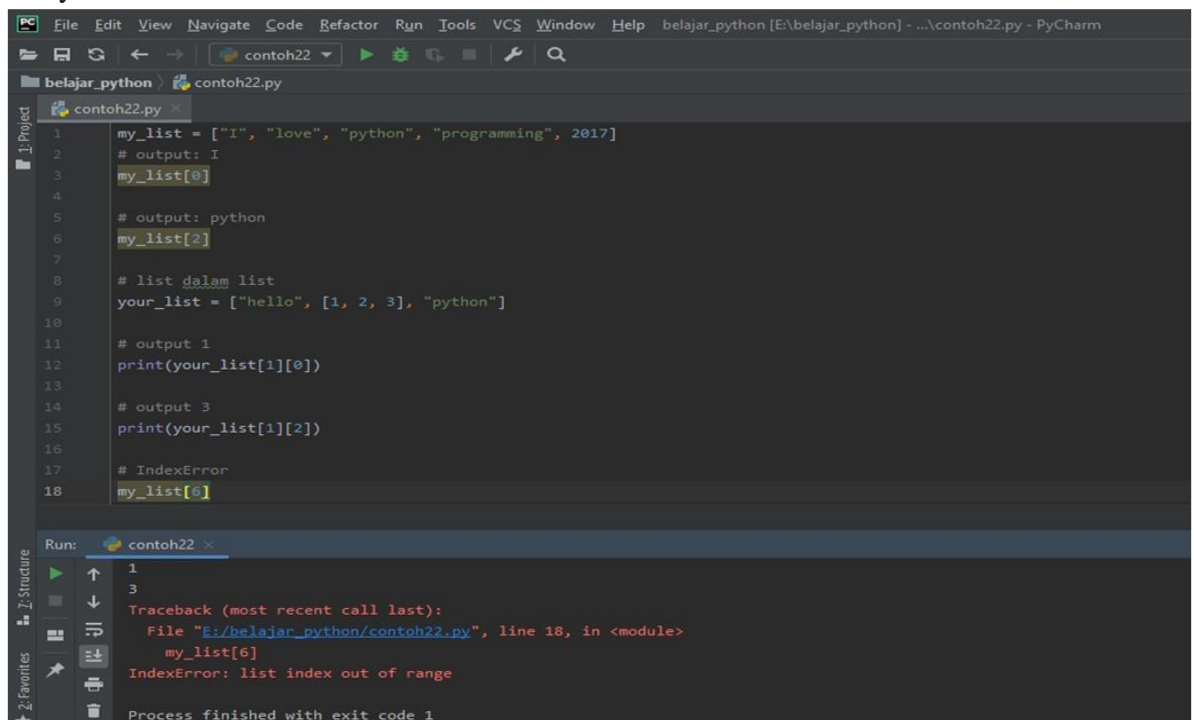
```
# list kosong
my_list = []
# list berisi integer my_list = [1,2,3,4,5]
# list berisi tipe campuran my_list = [1, 3.5, "Hello"]
```

List juga bisa berisi list lain. Ini disebut list bersarang

```
# list bersarang
my_list = ["hello", [2,4,6], ['a','b']]
```

#### 3.2. Mengakses anggota List

Kita bisa mengakses anggota list dengan menggunakan indeksinya dengan format `namalist[indeks]`. Indeks list dimulai dari 0. List yang memiliki 5 anggota akan memiliki indeks mulai dari 0 s/d 4. Mencoba mengakses anggota list di luar itu akan menyebabkan error `IndexError`.



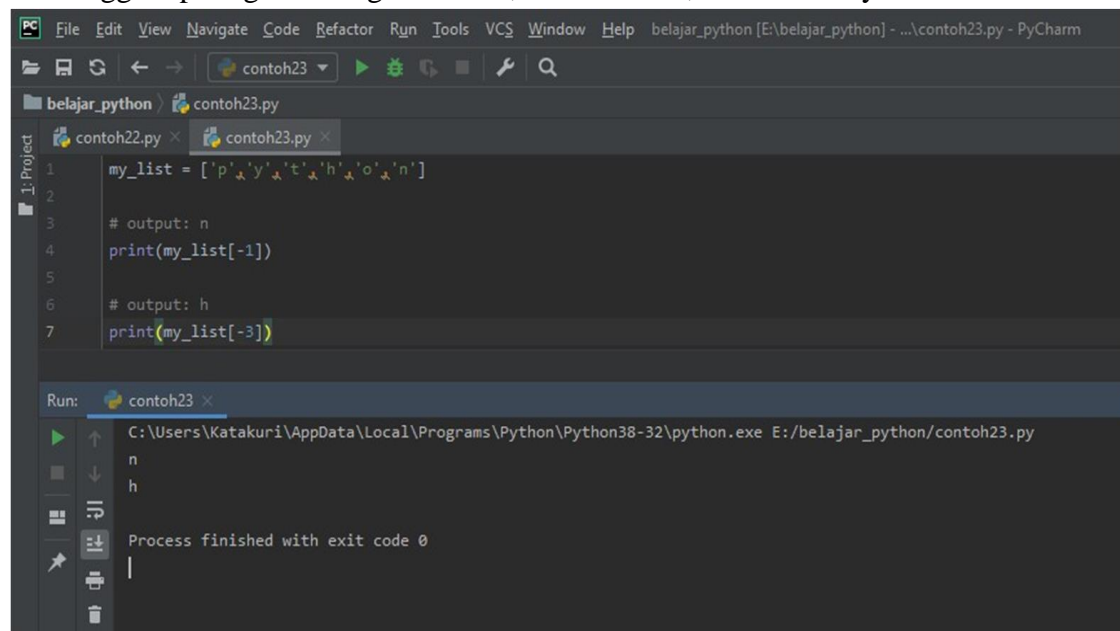
```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh22.py - PyCharm
contoh22
belajar_python > contoh22.py
contoh22.py
1 my_list = ["I", "love", "python", "programming", 2017]
2 # output: I
3 my_list[0]
4
5 # output: python
6 my_list[2]
7
8 # list dalam list
9 your_list = ["hello", [1, 2, 3], "python"]
10
11 # output 1
12 print(your_list[1][0])
13
14 # output 3
15 print(your_list[1][2])
16
17 # IndexError
18 my_list[6]
```

Run: contoh22 x

```
1
3
Traceback (most recent call last):
  File "E:/belajar_python/contoh22.py", line 18, in <module>
    my_list[6]
IndexError: list index out of range
Process finished with exit code 1
```

### 3.3. List dengan Indeks Negatif

Python mendukung indeks negatif, yaitu urutan dimulai dari anggota terakhir. Indeks anggota paling belakang adalah -1, kemudian -2, dan seterusnya.



The screenshot shows a PyCharm IDE with a project named 'belajar\_python'. The file 'contoh23.py' is open and contains the following code:

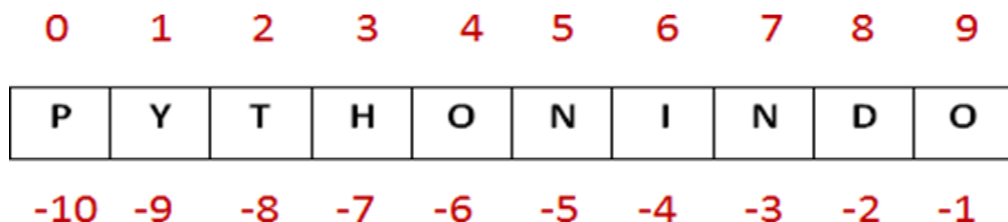
```
1 my_list = ['p','y','t','h','o','n']
2
3 # output: n
4 print(my_list[-1])
5
6 # output: h
7 print(my_list[-3])
```

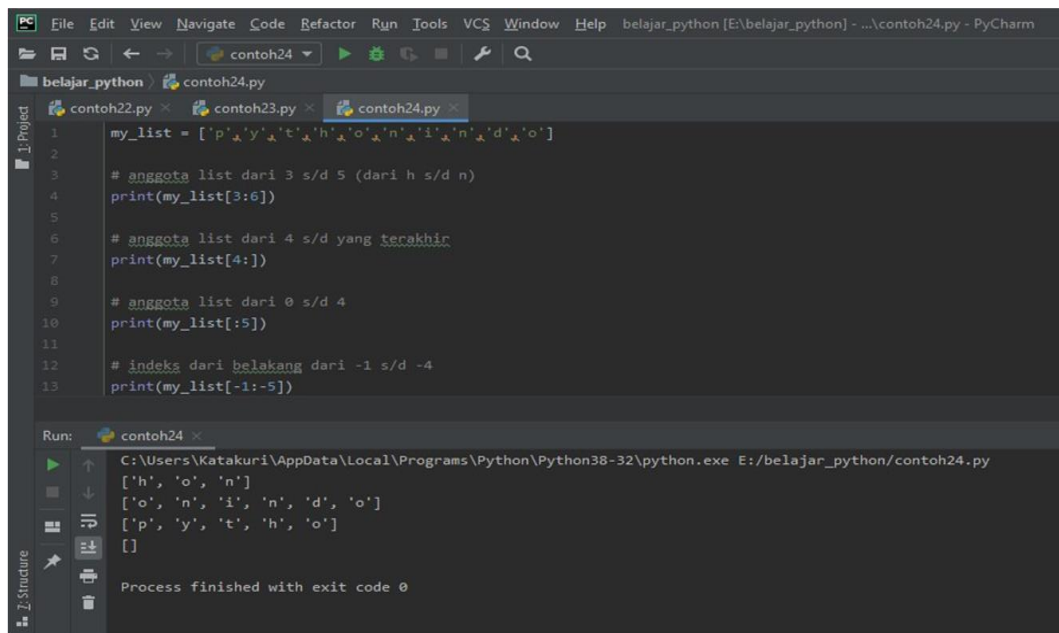
The Run window shows the execution of the script, with the following output:

```
Run: contoh23 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh23.py
n
h
Process finished with exit code 0
```

### 3.4. Memotong (Slicing) List

Kita bisa mengakses anggota list dari range tertentu dengan menggunakan operator slicing titik dua ( : ). Slicing akan lebih mudah bila kita memahami indeks dengan baik. Perhatikan gambar berikut:



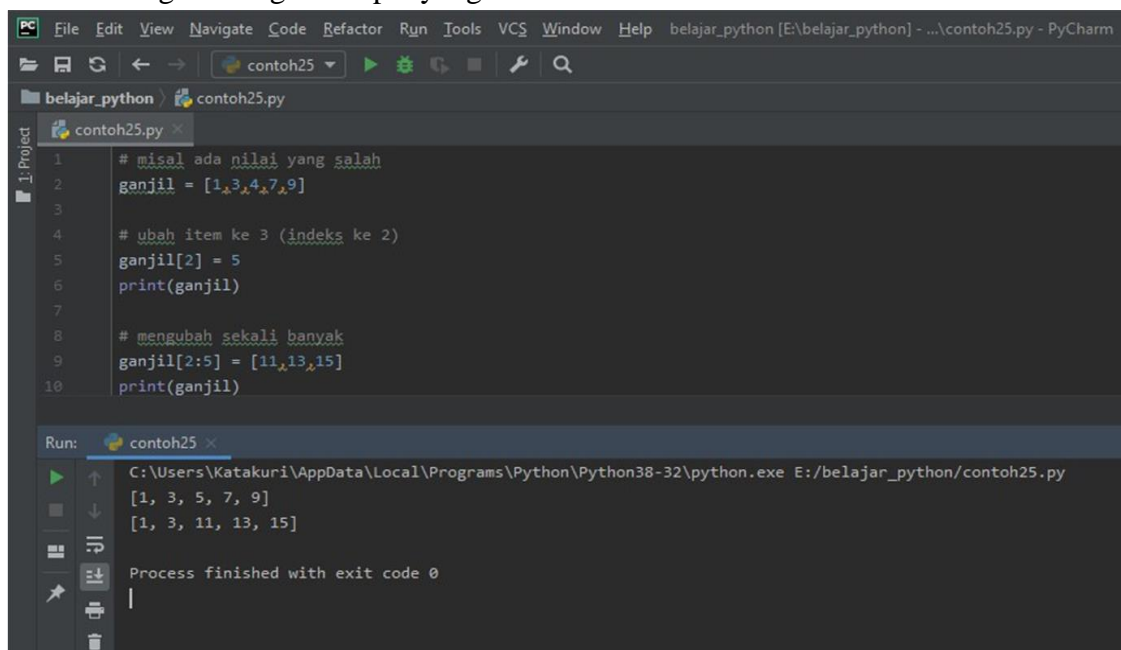


```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh24.py - PyCharm
contoh24
belajar_python > contoh24.py
contoh22.py x contoh23.py x contoh24.py x
1 my_list = ['p','y','t','h','o','n','i','n','d','o']
2
3 # anggota list dari 3 s/d 5 (dari h s/d n)
4 print(my_list[3:6])
5
6 # anggota list dari 4 s/d yang terakhir
7 print(my_list[4:])
8
9 # anggota list dari 0 s/d 4
10 print(my_list[:5])
11
12 # indeks dari belakang dari -1 s/d -4
13 print(my_list[-1:-5])

Run: contoh24 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh24.py
['h', 'o', 'n']
['o', 'n', 'i', 'n', 'd', 'o']
['p', 'y', 't', 'h', 'o']
[]
Process finished with exit code 0
```

### 3.5. Mengubah Anggota List

List adalah tipe data yang bersifat mutable, artinya anggotanya bisa diubah. Ini berbeda dengan string dan tuple yang bersifat immutable.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh25.py - PyCharm
contoh25
belajar_python > contoh25.py
contoh25.py x
1 # misal ada nilai yang salah
2 ganjil = [1,3,4,7,9]
3
4 # ubah item ke 3 (indeks ke 2)
5 ganjil[2] = 5
6 print(ganjil)
7
8 # mengubah sekali banyak
9 ganjil[2:5] = [11,13,15]
10 print(ganjil)

Run: contoh25 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh25.py
[1, 3, 5, 7, 9]
[1, 3, 11, 13, 15]
Process finished with exit code 0
```

### 3.6. Metode List

List memiliki banyak metode untuk operasi seperti menambahkan anggota, menghapus, menyisipkan, menyortir, dan lain sebagainya. Mereka bisa diakses menggunakan format list.metode().

### 3.7. Menambahkan Anggota List

Fungsi append() berguna untuk menambahkan anggota ke dalam list. Selain itu, ada metode extend() untuk menambahkan anggota list ke dalam list.

```
>>> ganjil = [1,3,5,7]
```

```
>>> ganjil.append(9)
```



```
>>> print(ganjil)    [1,3,5,7,9]
```

```
>>> ganjil.extend([11,13,15])
```

```
>>> print(ganjil)    [1,3,5,7,9,11,13,15]
```

Kita juga bisa menggunakan operator + untuk menggabungkan dua list, dan operator \* untuk melipatgandakan list.

```
>>> genap = [2, 4, 6]
```

```
>>> print(genap + [8, 10, 12])
[2, 4, 6, 8, 10, 12]
```

```
>>> print(['p','y'] * 2)
['p','y','p','y']
```

### 3.8. Menyisipkan Anggota List

Fungsi insert() berfungsi untuk menyisipkan anggota list pada indeks tertentu.

```
>>> ganjil = [5,7,11,13,15]
```

```
>>> # kita akan menyisipkan 9 setelah angka 7
```

```
>>> ganjil.insert(2,9)
```

```
>>> print(ganjil)    [5,7,9,11,13,15]
```

### 3.9. Menghapus Anggota List

Kita bisa menggunakan metode remove(), pop(), atau kata kunci del untuk menghapus anggota list. Selain itu kita bisa menggunakan clear() untuk mengosongkan list. Fungsi pop() selain menghapus anggota list, juga mengembalikan nilai indeks anggota tersebut. Hal ini berguna bila kita ingin memanfaatkan indeks dari anggota yang terhapus untuk digunakan kemudian.

### 3.10. Mengurutkan anggota List

Pada saat kita perlu mengurutkan atau menyortir anggota list, kita bisa menggunakan metode sort(). Untuk membalik dengan urutan sebaliknya bisa dengan menggunakan argumen reverse=True.

```
>>> alfabet = ['a','b','d','f','e','c','h','g','j','i']
```

```
>>> alfabet.sort()
```

```
>>> print(alfabet)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

```
>>> alfabet.sort(reverse=True)
```

```
>>> print(alfabet)
```

```
['j', 'i', 'h', 'g', 'f', 'e', 'd', 'c', 'b', 'a']
```

### 3.11. Membalik Urutan List

Selain mengurutkan, kita juga bisa membalikkan urutan list dengan menggunakan metode `reverse()`.

```
>>> alfabet = ['a','c','d','e','b']

>>> alfabet.reverse()
>>> print(alfabet)  ['b','e','d','c','b','a']
```

### 3.12. Tuple

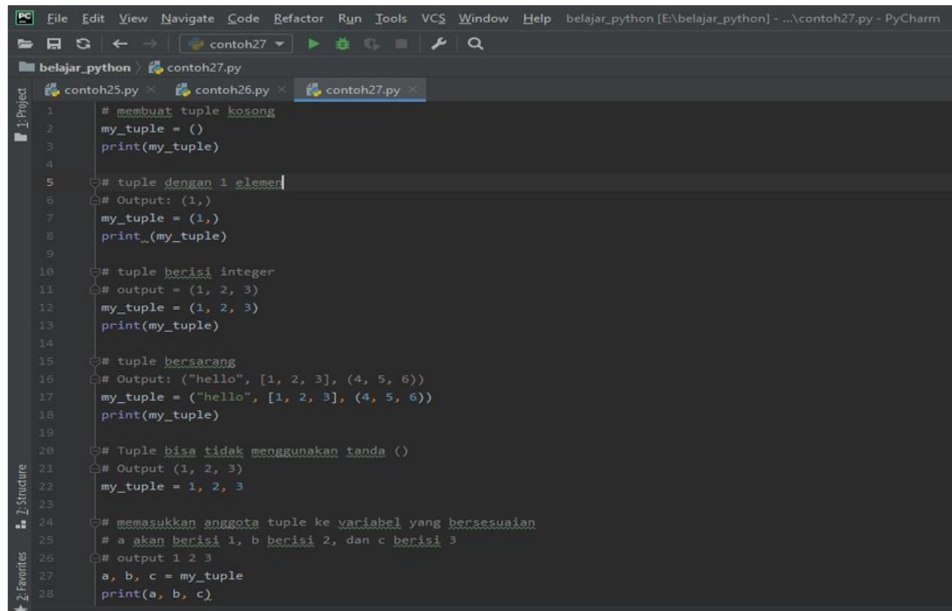
Tuple mirip dengan list. Bedanya, tuple bersifat immutable, sehingga anggotanya tidak bisa diubah. Kalau mirip, mengapa harus menggunakan tuple?

Kita menggunakan tuple tergantung kebutuhan. Untuk beberapa hal, tuple memiliki kelebihan sebagai berikut:

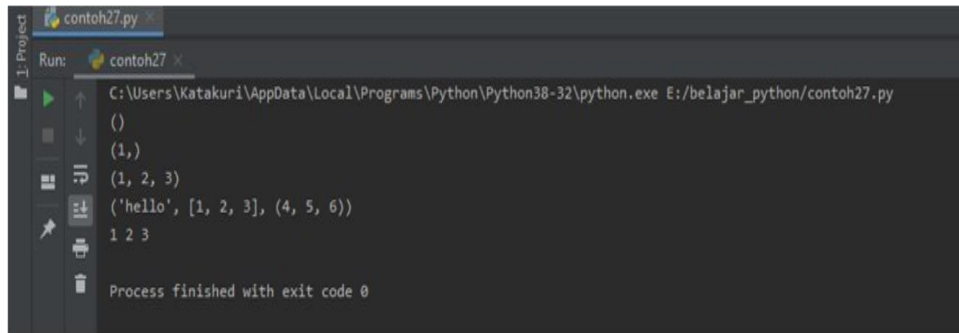
- Karena tuple adalah immutable, maka iterasi pada tuple lebih cepat dibandingkan list.
- Tuple bisa berisi anggota yang immutable yang dapat digunakan sebagai key untuk dictionary. List tidak bisa dipakai untuk itu.
- Kalau kita memerlukan data yang memang tidak untuk diubah, maka menggunakan tuple bisa menjamin bahwa data tersebut akan write-protected.

### 3.13. Membuat Tuple

Tuple dibuat dengan meletakkan semua anggota di dalam tanda kurung ( ), masing-masing dipisahkan oleh tanda koma. Menggunakan tanda kurung sebenarnya hanya opsional, tapi kita sebaiknya tetap menggunakannya untuk kemudahan pembacaan kode. Tuple dapat berisi tipe data yang sama maupun campuran.



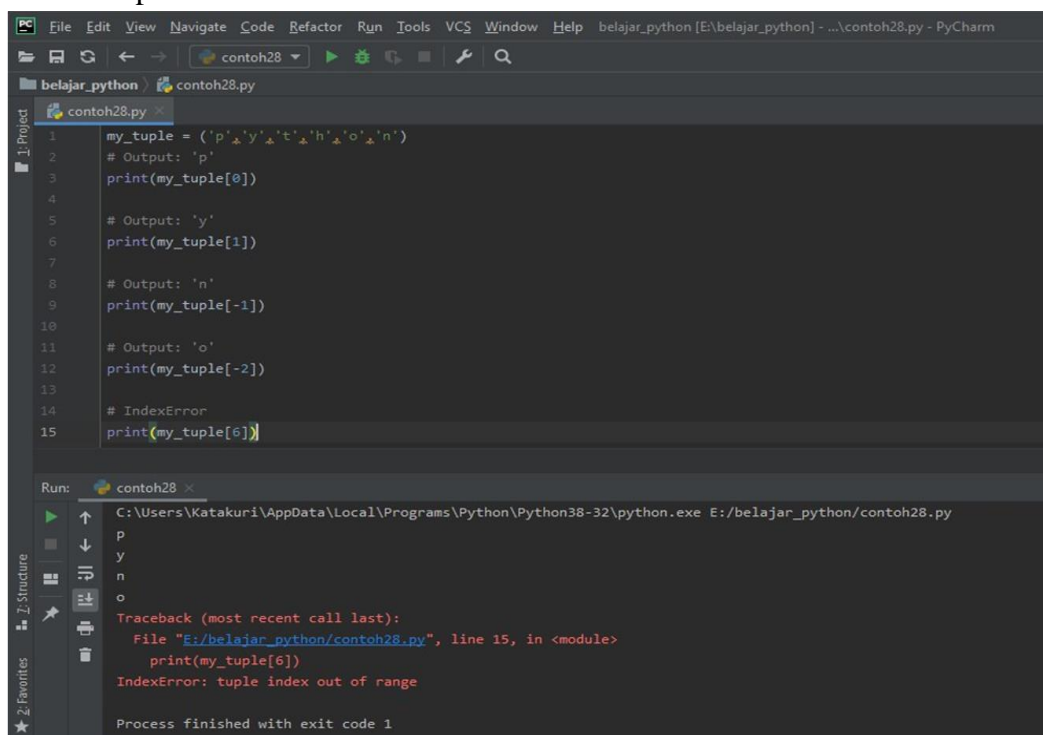
```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh27.py - PyCharm
contoh27
belajar_python > contoh27.py
contoh25.py x contoh26.py x contoh27.py x
1 # membuat tuple kosong
2 my_tuple = ()
3 print(my_tuple)
4
5 # tuple dengan 1 element
6 # Output: (1,)
7 my_tuple = (1,)
8 print(my_tuple)
9
10 # tuple berisi integer
11 # output = (1, 2, 3)
12 my_tuple = (1, 2, 3)
13 print(my_tuple)
14
15 # tuple bersarang
16 # Output: ("hello", [1, 2, 3], (4, 5, 6))
17 my_tuple = ("hello", [1, 2, 3], (4, 5, 6))
18 print(my_tuple)
19
20 # Tuple bisa tidak menggunakan tanda ( )
21 # Output (1, 2, 3)
22 my_tuple = 1, 2, 3
23
24 # memasukkan anggota tuple ke variabel yang bersesuaian
25 # a akan berisi 1, b berisi 2, dan c berisi 3
26 # output 1 2 3
27 a, b, c = my_tuple
28 print(a, b, c)
```



```
Run: contoh27 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh27.py
()
(1,)
(1, 2, 3)
('hello', [1, 2, 3], (4, 5, 6))
1 2 3
Process finished with exit code 0
```

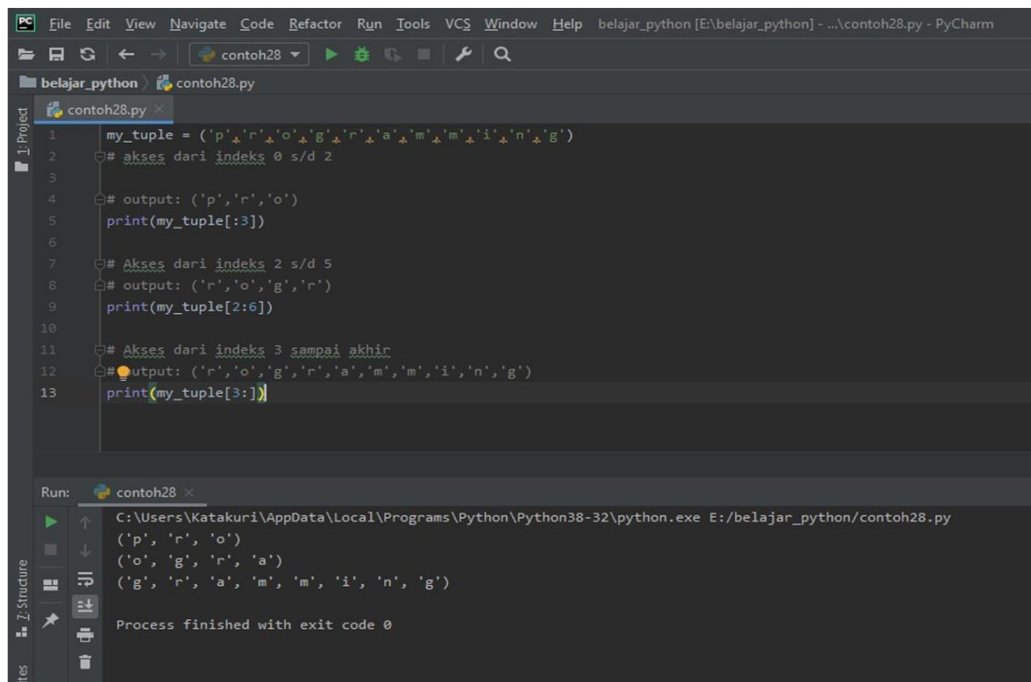
### 3.14. Mengakses anggota Tuple

Seperti halnya list, kita bisa mengakses anggota tuple lewat indeksnya menggunakan format `namatuple[indeks]`. Indeks dimulai dari 0 untuk anggota pertama. Selain itu, indeks negatif juga bisa dipakai mulai dari -1 untuk anggota terakhir tuple.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh28.py - PyCharm
belajar_python > contoh28.py
contoh28.py x
1 my_tuple = ('p','y','t','h','o','n')
2 # Output: 'p'
3 print(my_tuple[0])
4
5 # Output: 'y'
6 print(my_tuple[1])
7
8 # Output: 'n'
9 print(my_tuple[-1])
10
11 # Output: 'o'
12 print(my_tuple[-2])
13
14 # IndexError
15 print(my_tuple[6])
Run: contoh28 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh28.py
p
y
n
o
Traceback (most recent call last):
  File "E:/belajar_python/contoh28.py", line 15, in <module>
    print(my_tuple[6])
IndexError: tuple index out of range
Process finished with exit code 1
```

Sama seperti list, kita bisa mengakses satu range anggota tuple dengan menggunakan operator titik dua (`:`).

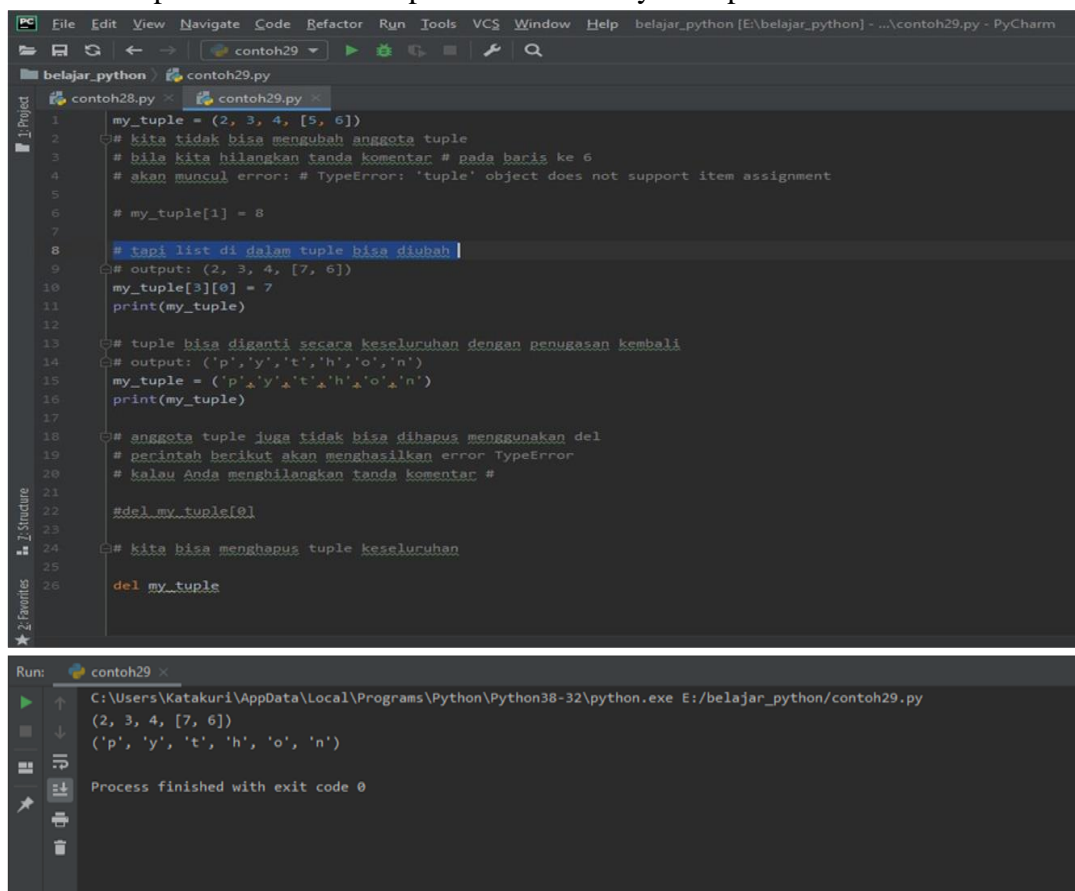


```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh28.py - PyCharm
contoh28.py
1 my_tuple = ('p','r','o','g','r','a','m','m','i','n','g')
2 # akses dari indeks 0 s/d 2
3
4 # output: ('p','r','o')
5 print(my_tuple[:3])
6
7 # Akses dari indeks 2 s/d 5
8 # output: ('r','o','g','r')
9 print(my_tuple[2:6])
10
11 # Akses dari indeks 3 sampai akhir
12 # output: ('r','o','g','r','a','m','m','i','n','g')
13 print(my_tuple[3:])

Run: contoh28 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh28.py
('p', 'r', 'o')
('o', 'g', 'r', 'a')
('g', 'r', 'a', 'm', 'm', 'i', 'n', 'g')
Process finished with exit code 0
```

### 3.15. Mengubah Anggota Tuple

Setelah tuple dibuat, maka anggota tuple tidak bisa lagi diubah atau dihapus. Akan tetapi, bila anggota tuple-nya adalah tuple bersarang dengan anggota seperti list, maka item pada list tersebut dapat diubah. Jelasnya ada pada contoh berikut:

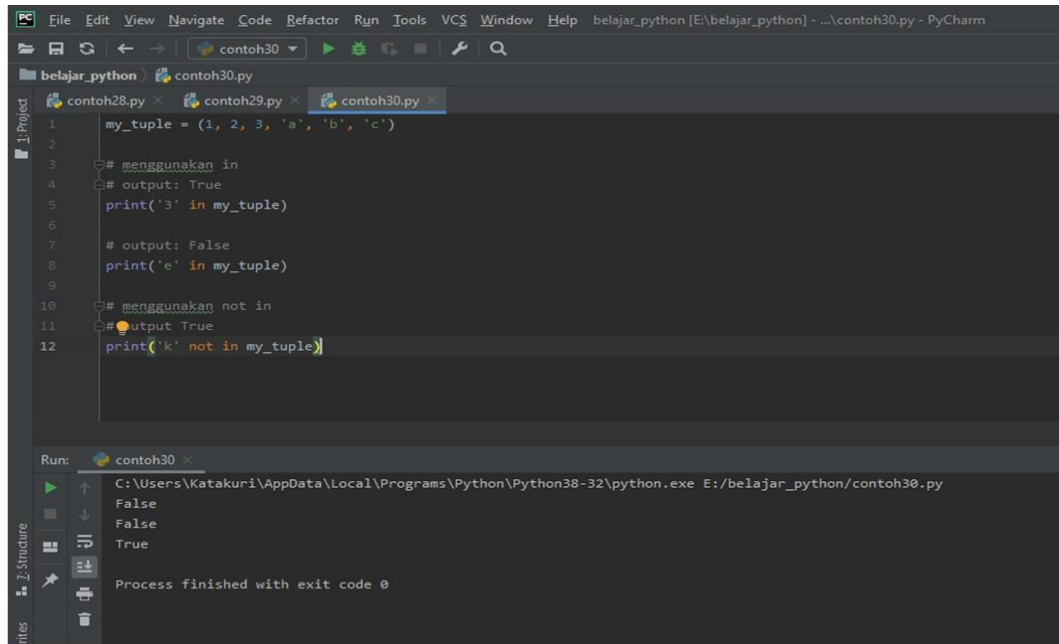


```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh29.py - PyCharm
contoh29.py
1 my_tuple = (2, 3, 4, [5, 6])
2 # kita tidak bisa mengubah anggota tuple
3 # bila kita hilangkan tanda komentar # pada baris ke 6
4 # akan muncul error: # TypeError: 'tuple' object does not support item assignment
5
6 # my_tuple[1] = 8
7
8 # tapi list di dalam tuple bisa diubah
9 # output: (2, 3, 4, [7, 6])
10 my_tuple[3][0] = 7
11 print(my_tuple)
12
13 # tuple bisa diganti secara keseluruhan dengan penugasan kembali
14 # output: ('p','y','t','h','o','n')
15 my_tuple = ('p','y','t','h','o','n')
16 print(my_tuple)
17
18 # anggota tuple juga tidak bisa dihapus menggunakan del
19 # perintah berikut akan menghasilkan error TypeError
20 # kalau Anda menghilangkan tanda komentar #
21
22 #del my_tuple[0]
23
24 # kita bisa menghapus tuple keseluruhan
25
26 del my_tuple

Run: contoh29 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh29.py
(2, 3, 4, [7, 6])
('p', 'y', 't', 'h', 'o', 'n')
Process finished with exit code 0
```

### 3.16. Menguji Keanggotaan Tuple

Seperti halnya string dan list, kita bisa menguji apakah sebuah objek adalah anggota dari tuple atau tidak, yaitu dengan menggunakan operator in atau not in untuk kebalikannya.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh30.py - PyCharm
belajar_python > contoh30.py
1 my_tuple = (1, 2, 3, 'a', 'b', 'c')
2
3 # menggunakan in
4 # output: True
5 print('3' in my_tuple)
6
7 # output: False
8 print('e' in my_tuple)
9
10 # menggunakan not in
11 # output: True
12 print('k' not in my_tuple)
```

Run: contoh30 x

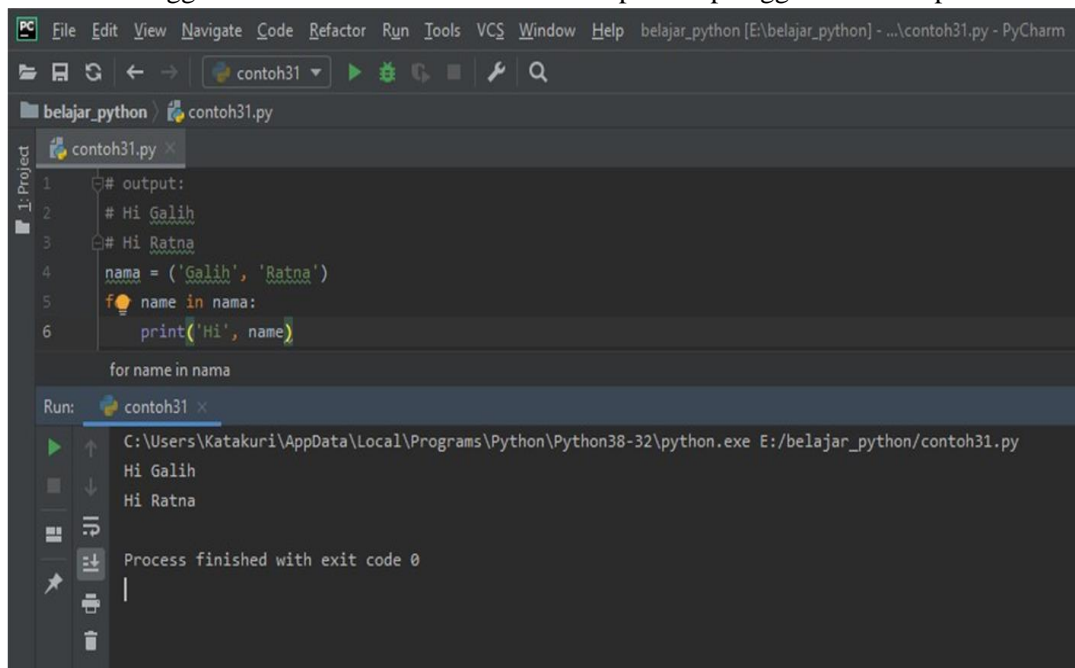
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar\_python/contoh30.py

False  
False  
True

Process finished with exit code 0

### 3.17. Iterasi pada Tuple

Kita bisa menggunakan for untuk melakukan iterasi pada tiap anggota dalam tuple.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh31.py - PyCharm
belajar_python > contoh31.py
1 # output:
2 # Hi Galih
3 # Hi Ratna
4 nama = ('Galih', 'Ratna')
5 for name in nama:
6     print('Hi', name)
```

Run: contoh31 x

C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar\_python/contoh31.py

Hi Galih  
Hi Ratna

Process finished with exit code 0

### 3.18. Metode dan Fungsi Bawaan Tuple

Tuple hanya memiliki dua buah metode yaitu count() dan index().

- Metode count(x) berfungsi mengembalikan jumlah item yang sesuai dengan x pada tuple
- Metode index(x) berfungsi mengembalikan indeks dari item pertama yang sama dengan x.

Walaupun hanya memiliki dua metode, banyak fungsi bawaan python yang berfungsi untuk melakukan operasi pada tuple. Berikut adalah daftarnya:

Tabel 6.1 Operasi Pada Tuple

Fungsi	Deskripsi
<code>all()</code>	Mengembalikan <b>True</b> jika semua anggota tuple adalah benar ( tidak ada yang kosong )
<code>any()</code>	Mengembalikan <b>True</b> jika salah satu atau semua bernilai benar. Jika tuple kosong, maka akan mengembalikan <b>False</b> .
<code>enumerate()</code>	Mengembalikan objek enumerasi. Objek enumerasi adalah objek yang terdiri dari pasangan indeks dan nilai.
<code>len()</code>	Mengembalikan panjang (jumlah anggota) tuple
<code>max()</code>	Mengembalikan anggota terbesar di tuple
<code>min()</code>	Mengembalikan anggota terkecil di tuple
<code>sorted()</code>	Mengambil anggota tuple dan mengembalikan list baru yang sudah diurutkan
<code>sum()</code>	Mengembalikan jumlah dari semua anggota tuple
<code>tuple()</code>	Mengubah sequence (list, string, set, dictionary) menjadi tuple

## 3.1. Studi Kasus :

Buatlah sebuah list (array) 2 dimensi dimana terdapat baris dan kolom dengan nilai sebagai berikut :

<b>1</b>	<b>2</b>	<b>3</b>
4	5	6
7	8	9

0		
---	--	--

Tampilkan lah :

- Baris Pertama, Kolom Pertama
- Baris Pertama, Kolom Kedua
- Baris Pertama, Kolom Ketiga
- Baris Ketiga, Kolom Ketiga
- Baris Ke Empat, Kolom Pertama

Hasil :

```

Run: latihan x
D:\IMK\projek_python\venv\Scripts\python.exe D:/IMK/projek_python/latihan.py
Baris Pertama, Kolom Pertama Adalah :
1
Baris Pertama, Kolom Kedua Adalah :
2
Baris Pertama, Kolom Ketiga Adalah :
3
Baris Ketiga, Kolom Ketiga Adalah :
9
Baris Ke Empat, Kolom Pertama Adalah :
0
Process finished with exit code 0

```

## Latihan

Buatlah input, proses dan output secara berulang dengan memanfaatkan fungsi matriks/list seperti pada koding dibawah ini :

*#variable yg berulang menggunakan List/matriks*

```
list_nim=[]
```

```
list_uts=[]
```

```
list_uas=[]
```

```
list_total=[]
```

```
ulang=2
```

```
for i in range(ulang):
```

```
    print ("data Ke - " + str(i+1))
```

```
    list_nim.append(input("Masukkan Nim anda : "))
```

```
    list_uts.append(int(input("Masukkan Nilai UTS anda :")))

```

```
    list_uas.append(int(input("Masukkan Nilai UAS : ")))
```

*#proses*

```
for i in range(ulang):
```

```
    list_total.append((list_uas[i] + list_uts[i]) / 2)
```

*#Cetak*

```
print("=====")
```

```
print("Nim    Nilai Uts    Nilai UAS    Total")
```

```
print("=====")
```

```

for i in range(ulang):
    print ("%s \t %i \t %i \t %i" % (list_nim[i],list_uts[i],list_uas[i],list_total[i]))

print("=====")

```

### Hasil Tampilan Input

```

LatihanAll
"E:\OneDrive - Bina Sarana Informa
data Ke - 1
Masukkan Nim anda : 12181707
Masukkan Nilai UTS anda : 90
Masukkan Nilai UAS : 90
data Ke - 2
Masukkan Nim anda : 12134567
Masukkan Nilai UTS anda : 80
Masukkan Nilai UAS : 90

```

### Hasil Tampilan Output

```

=====
Nim      Nilai Uts      Nilai UAS      Total
=====
12181707      90            90            90
12134567      80            90            85
=====

```

S



## Pertemuan 7

### String dan Bilangan

String adalah tipe data yang paling sering digunakan di Python. Kita bisa membuat string dengan meletakkan karakter di dalam tanda kutip. Tanda kutipnya bisa kutip tunggal maupun kutip ganda. Contohnya adalah sebagai berikut:

```
var1 = 'Hello Python'
var2 = 'Programming with Python'
```

#### 3.1. Mengakses Nilai String

Untuk mengakses nilai atau substring dari string, kita menggunakan indeks dalam tanda [ ].

```
var1 = 'Hello Python!'
var2 = "I love Python"
print("var1[0]", var1[0])
print("var2[2:6] :", var2[2:6])
```

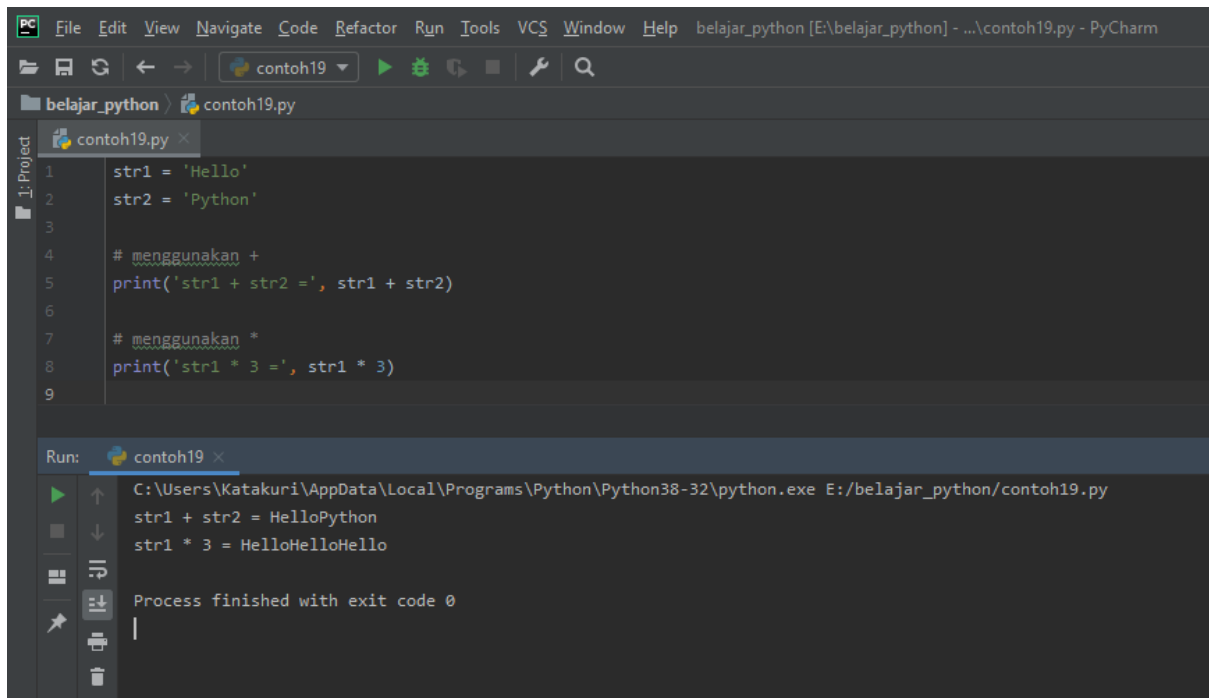
#### 3.2. Mengupdate String

String adalah tipe data immutable, artinya tidak bisa diubah. Untuk mengupdate string, kita perlu memberikan nilai variabel string lama ke string yang baru. Nilai yang baru adalah nilai string lama yang sudah diupdate.

```
var1 = 'Hello Python!'
var2 = var1[:6]
print("String Update: - ", var1[:6] + 'World')
```

#### 3.3. Menggabung String

Kita bisa menggabungkan dua atau lebih string menjadi satu dengan menggunakan operator +. Selain itu kita juga bisa melipatgandakan string menggunakan operator \*.



### 3.4. Mengetahui Panjang String

Untuk mengetahui panjang dari string, kita bisa menggunakan fungsi `len()`.

```
>>> string = 'I love Python'
>>> len(string)
18
```

### 3.5. Karakter Escape

Kalau kita hendak mencetak string: He said, "What's there?" kita tidak bisa menggunakan tanda kutip tunggal maupun ganda. Bila kita melakukannya, akan muncul pesan error `SyntaxError` karena teks berisi kutip tunggal dan ganda.

```
>>> print("He said, "What's there?")
...
SyntaxError: invalid syntax
>>> print('He said, "What's there?")
...
SyntaxError: invalid syntax
```

Untuk hal seperti ini kita bisa menggunakan tanda kutip tiga atau menggunakan karakter escape. Karakter escape dimulai dengan tanda backslash `\`. Interpreter akan menerjemahkannya dengan cara berbeda dengan string biasa. Solusi untuk error di atas adalah sebagai berikut:

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The project name is 'belajar\_python' and the current file is 'contoh20.py'. The code editor contains the following Python code:

```
1 # menggunakan kutip tiga
2 print('He said, "What's there?")
3
4 # menggunakan karakter escape untuk tanda kutip tunggal
5 print('He said, "What\'s there?")
6
7 # menggunakan karakter escape untuk tanda kutip ganda
8 print("He said, \"What's there?\")
9
```

Below the code editor, the 'Run' tab shows the execution command: `C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh20.py`. The output of the program is displayed as:

```
He said, "What's there?"
He said, "What's there?"
He said, "What's there?"
```

The process finished with exit code 0.

Berikut adalah daftar karakter escape yang didukung oleh Python.

Tabel 9.1 Karakter Escape yang didukung Python

Karakter Escape	Deskripsi
\newline	Backslash dan newline diabaikan
\\	Backslash
\'	Kutip tunggal
\"	Kutip ganda
\a	ASCII bel
\b	ASCII backspace
\f	ASCII formfeed

<code>\n</code>	ASCII linefeed
<code>\r</code>	ASCII carriage return
<code>\t</code>	ASCII tab horizontal
<code>\v</code>	ASCII tab horizontal
<code>\ooo</code>	karakter dengan nilai oktal oo
<code>\xHH</code>	karakter dengan nilai heksadesimal HH

Berikut ini adalah beberapa contohnya:

```
>>> print("C:\\Python34\\Lib")
```

```
C:\Python34\Lib
```

```
>>> print("Ini adalah baris pertama\nDan ini baris dua")
```

```
Ini adalah baris pertama
```

```
Dan ini baris dua
```

```
>>> print("Ini adalah \x48\x45\x58")
```

```
Ini adalah HEX
```

### 3.6. Raw String untuk Mengabaikan Karakter Escape

Kadang kala kita perlu untuk mengabaikan karakter escape yang ada dalam string. Kita bisa melakukannya dengan meletakkan huruf r atau R sebelum tanda kutip string.

```
>>> print("This is \x61 \ngood example")
```

```
This is a good example
```

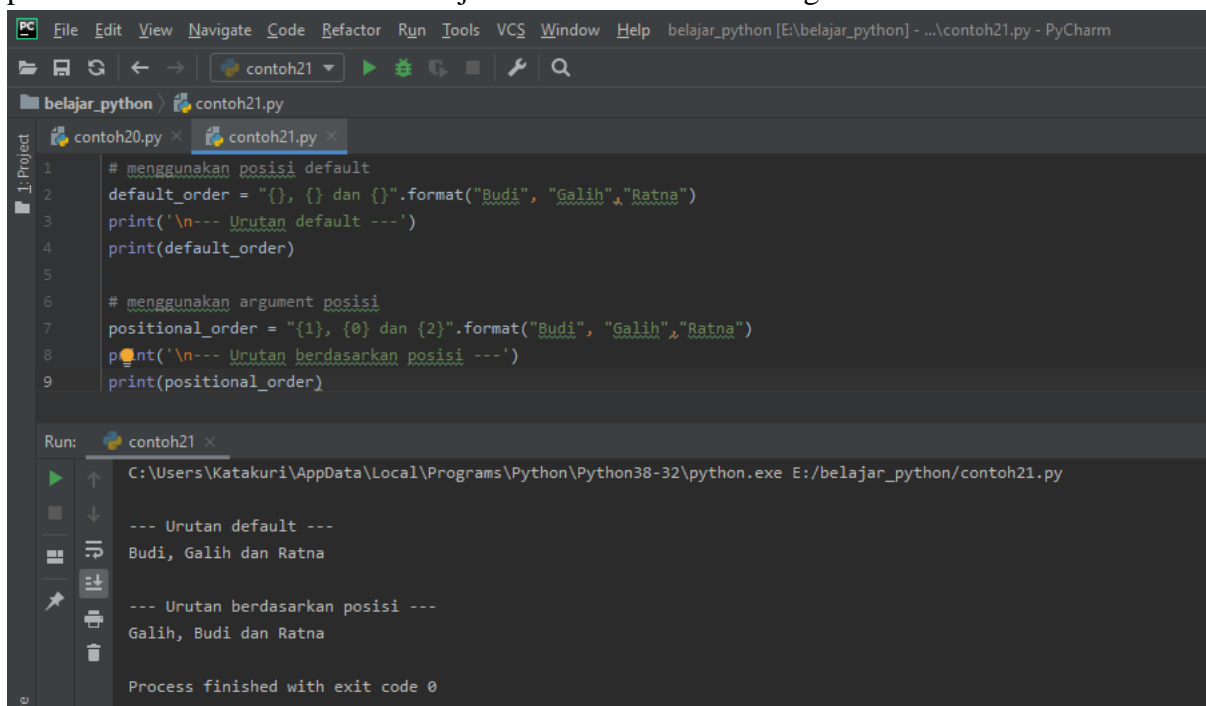
```
>>> print(r"This is \x61 \ngood example")
This is \x61 \ngood example
```

### 3.7. Mengatur Format String

Ada dua cara melakukan format pada string. Pertama dengan menggunakan fungsi format(), dan kedua dengan menggunakan cara lama (menggunakan %).

#### Metode format()

Memformat string dengan fungsi format() dibuat dengan menggunakan tanda {} sebagai placeholder atau posisi substring yang akan digantikan. Kita biasa menggunakan argumen posisi atau kata kunci untuk menunjukkan urutan dari substring.



The screenshot shows the PyCharm IDE with a file named `contoh21.py` open. The code in the editor is as follows:

```
1 # menggunakan posisi default
2 default_order = "{}, {} dan {}".format("Budi", "Galih", "Ratna")
3 print('\n--- Urutan default ---')
4 print(default_order)
5
6 # menggunakan argument posisi
7 positional_order = "{1}, {0} dan {2}".format("Budi", "Galih", "Ratna")
8 print('\n--- Urutan berdasarkan posisi ---')
9 print(positional_order)
```

Below the editor, the Run console shows the output of the script:

```
Run: contoh21 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh21.py

--- Urutan default ---
Budi, Galih dan Ratna

--- Urutan berdasarkan posisi ---
Galih, Budi dan Ratna

Process finished with exit code 0
```

Metode format() dapat memiliki spesifikasi format opsional. Misalnya, kita bisa menggunakan tanda < untuk rata kiri, > untuk rata kanan, ^ untuk rata tengah, dan sebagainya.

```
>>> # format integer
>>> "{0} bila diubah jadi biner menjadi {0:b}".format(12)
'12 bila diubah jadi biner menjadi 1100'
```

```
>>> # format float
>>> "Format eksponensial: {0:e}".format(1566.345)
'Format eksponensial: 1566345e+03'
```

```
>>> # pembulatan
>>> "Sepertiga sama dengan: {0:.3f}".format(1/3)
'Sepertiga sama dengan: 0.333'
```

```
>>> # Meratakan string
```

```
>>> "|{:<10}|{: ^10}|{:>10}|".format('beras', 'gula', 'garam')
'|beras      |      gula      |      garam|'
```

**Format cara lama dengan %**

Kita bisa menggunakan operator % untuk melakukan format string.

```
>>> nama = 'Budi'
```

```
>>> print('Nama saya %s' %s)
```

```
Nama saya Budi
```

```
>>> x = 12.3456789
```

```
>>> print('Nilai x = %3.2f' %x)
```

```
Nilai x = 12.35
```

```
>>> print('Nilai x = %3.4f' %x)
```

```
Nilai x = 12.3456
```

### 3.8. Metode / Fungsi Bawaan String

String memiliki banyak fungsi bawaan. `format()` yang kita bahas di atas hanya salah satu darinya. Fungsi atau metode lainnya yang sering dipakai adalah `join()`, `lower()`, `upper()`, `split()`, `startswith()`, `endswith()`, `replace()` dan lain sebagainya.

```
>>> "Universitas Bina Sarana Informatika".lower()
'universitas bina sarana informatika'
>>> "Universitas Bina Sarana Informatika ".upper()
'UNIVERSITAS BINA SARANA INFORMATIKA'
>>> "I love programming in Python".split()
['I', 'love', 'programming', 'in', 'Python']
>>> "I love Python".startswith("I")
True
>>> "Saya belajar Python".endswith("on")
True
>>> ' - '.join(['I', 'love', 'you'])
'I - love - you'
>>> "Belajar Java di BSI".replace('Java', 'Python')
'Belajar Python di BSI'
```

### 3.9. Bilangan (Number)

Bilangan (number) adalah salah satu tipe data dasar di Python. Python mendukung bilangan bulat (integer), bilangan pecahan (float), dan bilangan kompleks (complex). Masing – masing diwakili oleh kelas int, float, dan complex. Integer adalah bilangan bulat, yaitu bilangan yang tidak mempunyai koma. Contohnya 1, 2, 100, -30, -5, 99999, dan lain sebagainya. Panjang integer di python tidak dibatasi jumlah digitnya. Selama memori masih cukup, maka sepanjang itulah jumlah digit yang akan ditampilkan.

Float adalah bilangan pecahan atau bilangan berkoma. Contohnya adalah 5.5, 3.9, 72.8, -1.5, -0.7878999, dan lain sebagainya. Panjang angka di belakang koma untuk float ini adalah 15 digit. Bilangan kompleks (complex) adalah bilangan yang terdiri dari dua bagian, yaitu bagian yang real dan bagian yang imajiner. Contohnya adalah  $3 + 2j$ ,  $9 - 5j$ , dan lain sebagainya.

### 3.10. Konversi Jenis Bilangan

Kita bisa mengubah jenis bilangan dari int ke float, atau sebaliknya. Mengubah bilangan integer ke float bisa menggunakan fungsi `int(num)` dimana num adalah bilangan float.

```
>>> int(2.5)
2
```

```
>>> int(3.8)
3
```

```
>>> float(5)
5.0
```

Pada saat kita mengubah float ke integer, bilangan dibulatkan ke bawah. Sebaliknya saat kita mengubah integer ke float, maka bilangan bulat akan menjadi bilangan berkoma.

### 3.11. Python Decimal

Ada kalanya perhitungan menggunakan float di Python membuat kita terkejut. Kita tahu bahwa  $1.1 + 2.2$  hasilnya adalah  $3.3$ . Tapi pada saat kita lakukan dengan Python, maka hasilnya berbeda.

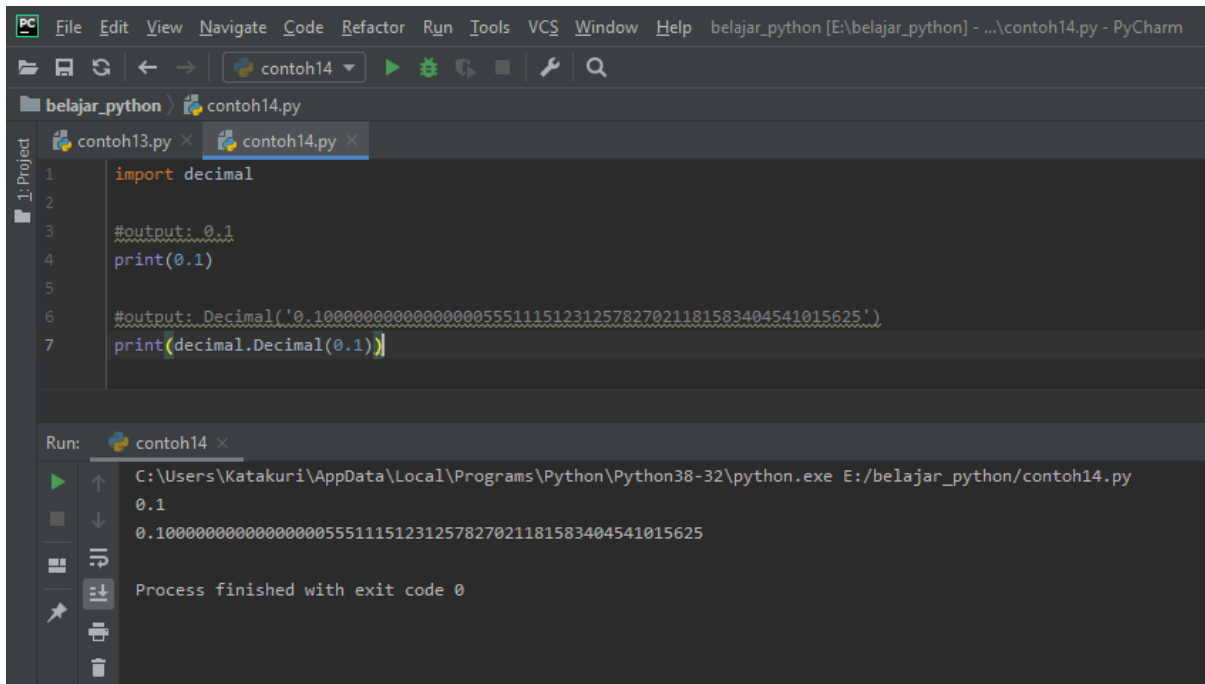
```
>>> (1.1 + 2.2) == 3.3
False
```

```
>>> 1.1 + 2.2
3.3000000000000003
```

Mengapa terjadi demikian?

Hal ini terjadi karena bilangan dalam komputer disimpan dalam bentuk digit 0 atau 1. Bila padanan digitnya tidak sesuai, maka bilangan float seperti 0.1 dalam bilangan biner akan menjadi pecahan yang sangat panjang yaitu 0.000110011001100110011... dan komputer kita hanya akan menyimpan panjang yang terbatas. Hal inilah yang menyebabkan terjadinya masalah seperti pada contoh di atas.

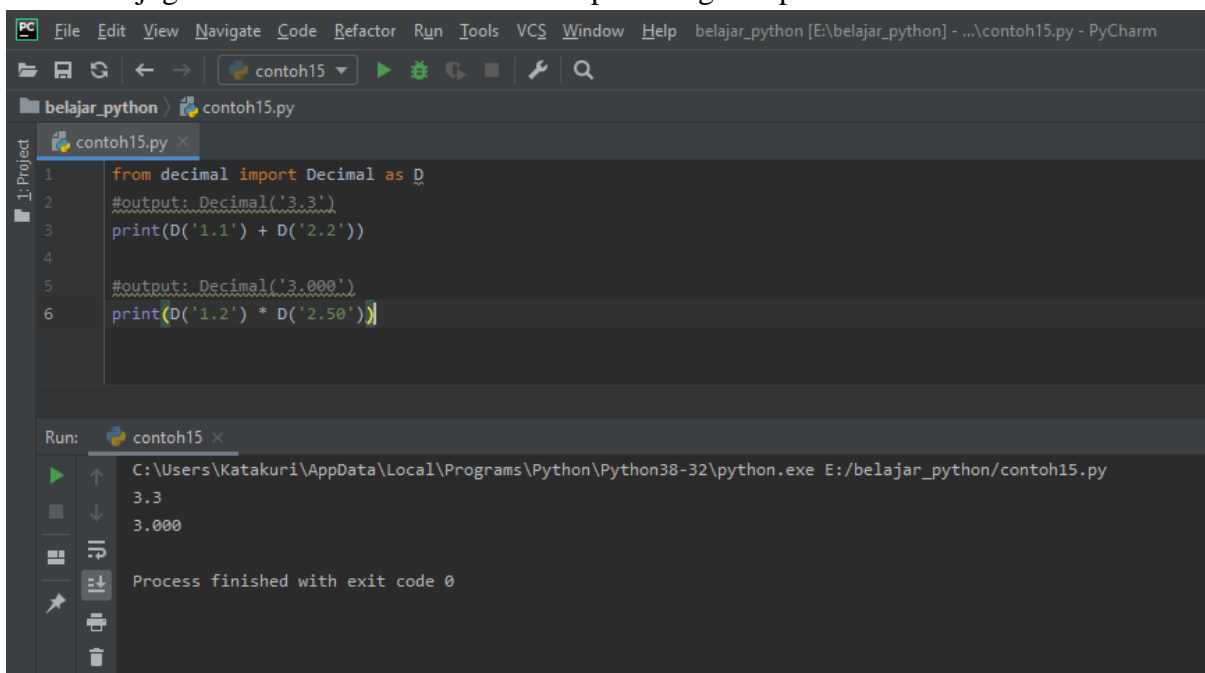
Untuk menangani hal seperti itu, kita bisa menggunakan modul bawaan Python yaitu modul decimal. Float hanya memiliki presisi sampai 15 digit di belakang koma, sementara dengan modul decimal kita bisa mengatur presisi jumlah digit di belakang koma.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh14.py - PyCharm
contoh14
belajar_python > contoh14.py
I:Project
contoh13.py x contoh14.py x
1 import decimal
2
3 #output: 0.1
4 print(0.1)
5
6 #output: Decimal('0.100000000000000055511151231257827021181583404541015625')
7 print(decimal.Decimal(0.1))

Run: contoh14 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh14.py
0.1
0.100000000000000055511151231257827021181583404541015625
Process finished with exit code 0
```

Modul ini juga membuat kita bisa melakukan perhitungan seperti di sekolah.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh15.py - PyCharm
contoh15
belajar_python > contoh15.py
I:Project
contoh15.py x
1 from decimal import Decimal as D
2 #output: Decimal('3.3')
3 print(D('1.1') + D('2.2'))
4
5 #output: Decimal('3.000')
6 print(D('1.2') * D('2.50'))

Run: contoh15 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh15.py
3.3
3.000
Process finished with exit code 0
```

Kapan Saatnya Menggunakan Decimal Dibanding Float?

Kita lebih baik menggunakan Decimal dalam kasus:

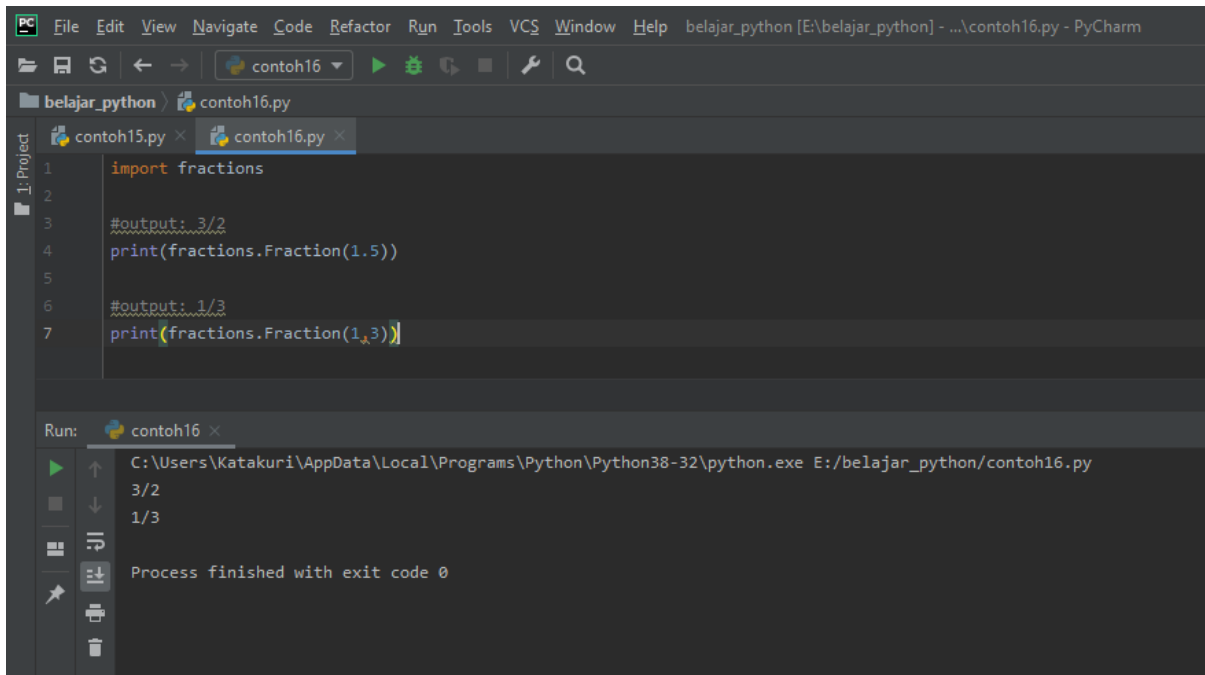
- Saat kita ingin membuat aplikasi keuangan yang membutuhkan presisi desimal yang pasti
- Saat kita ingin mengontrol tingkat presisi yang diperlukan
- Saat kita ingin menerapkan perkiraan berapa digit decimal yang signifikan



Saat kita ingin melakukan operasi perhitungan sama persis dengan yang kita lakukan di sekolah

### 3.12. Bilangan Pecahan

Python menyediakan modul `fractions` untuk mengoperasikan bilangan pecahan. Pecahan adalah bilangan yang memiliki pembilang dan penyebut, misalnya  $3/2$ . Perhatikan contoh berikut:



```
1 import fractions
2
3 #output: 3/2
4 print(fractions.Fraction(1.5))
5
6 #output: 1/3
7 print(fractions.Fraction(1,3))
```

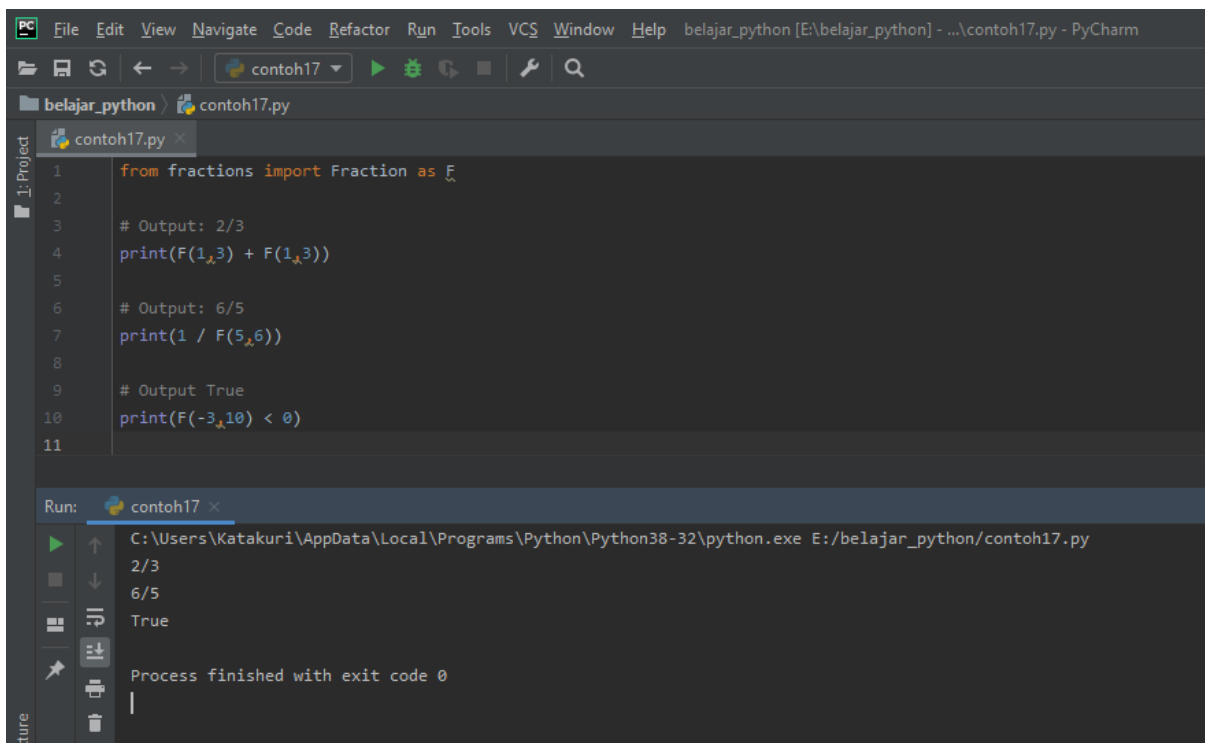
Run: contoh16 x

C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar\_python/contoh16.py

3/2  
1/3

Process finished with exit code 0

Operasi dasar seperti penjumlahan atau pembagian pecahan juga bisa dilakukan dengan modul `fractions` ini



```
1 from fractions import Fraction as F
2
3 # Output: 2/3
4 print(F(1,3) + F(1,3))
5
6 # Output: 6/5
7 print(1 / F(5,6))
8
9 # Output True
10 print(F(-3,10) < 0)
11
```

Run: contoh17 x

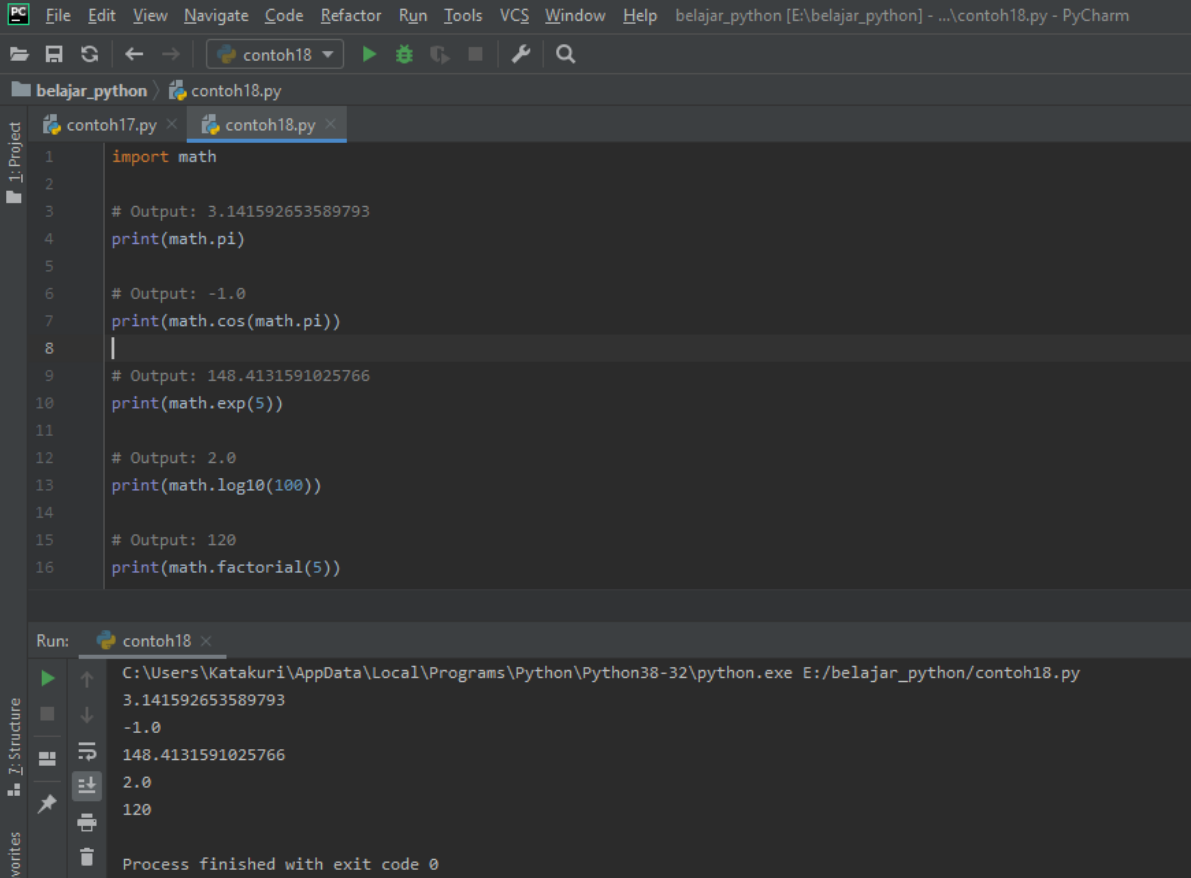
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar\_python/contoh17.py

2/3  
6/5  
True

Process finished with exit code 0

### 3.13. Matematika dengan Python

Python menyediakan modul `math` melakukan hal yang berbaur matematis seperti trigonometri, logaritma, probabilitas, statistik, dan lain – lain.



The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The toolbar contains icons for file operations, navigation, and execution. The main editor window displays a Python script in `contoh18.py` with the following code:

```
1 import math
2
3 # Output: 3.141592653589793
4 print(math.pi)
5
6 # Output: -1.0
7 print(math.cos(math.pi))
8
9 # Output: 148.4131591025766
10 print(math.exp(5))
11
12 # Output: 2.0
13 print(math.log10(100))
14
15 # Output: 120
16 print(math.factorial(5))
```

Below the editor, the Run tool window shows the execution of `contoh18`. The command executed is `C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh18.py`. The output is as follows:

```
3.141592653589793
-1.0
148.4131591025766
2.0
120
```

The process finished with exit code 0.

**UTS**

## **Pertemuan 8**

## Pertemuan 9

### Fungsi

Fungsi adalah grup/blok program untuk melakukan tugas tertentu yang berulang. Fungsi membuat kode program menjadi reusable, artinya hanya di definisikan sekali saja, dan kemudian bisa digunakan berulang kali dari tempat lain di dalam program.

Fungsi memecah keseluruhan program menjadi bagian – bagian yang lebih kecil . Dengan semakin besarnya program, maka fungsi akan membuatnya menjadi lebih mudah diorganisir dan dimanage.

Sejauh ini, kita sudah menggunakan beberapa fungsi, misalnya fungsi `print()`, `type()`, dan sebagainya. Fungsi tersebut adalah fungsi bawaan dari Python. Kita bisa membuat fungsi kita sendiri sesuai kebutuhan.

#### 3.1. Mendefinisikan Fungsi

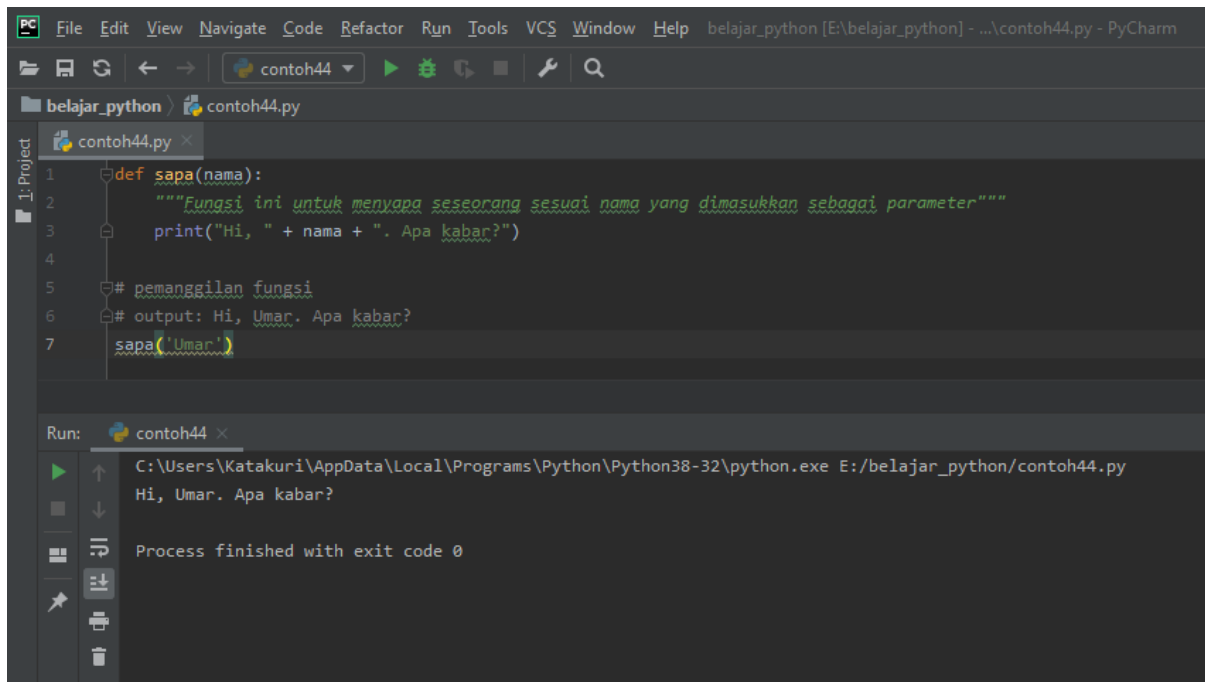
Berikut adalah sintaks yang digunakan untuk membuat fungsi:

```
def function_name(parameters) :  
    """function_docstring"""  
    statement(s)  
  
    return [expression]
```

Penjelasannya dari sintaks fungsi di atas:

1. Kata kunci `def` diikuti oleh `function_name` (nama fungsi), tanda kurung dan tanda titik dua (`:`) menandai header (kepala) fungsi.
2. Parameter / argumen adalah input dari luar yang akan diproses di dalam tubuh fungsi.
3. `"function_docstring"` bersifat opsional, yaitu sebagai string yang digunakan untuk dokumentasi atau penjelasan fungsi. `"function_docstring"` diletakkan paling atas setelah baris `def`.
4. Setelah itu diletakkan baris – baris pernyataan (statements). Jangan lupa indentasi untuk menandai blok fungsi.
5. `return` bersifat opsional. Gunanya adalah untuk mengembalikan suatu nilai `expression` dari fungsi.

Berikut adalah contoh fungsi untuk menyapa seseorang.



The screenshot shows the PyCharm IDE interface. The top toolbar includes icons for File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The main editor window displays a file named `contoh44.py` with the following Python code:

```
1 def sapa(nama):
2     """Fungsi ini untuk menyapa seseorang sesuai nama yang dimasukkan sebagai parameter"""
3     print("Hi, " + nama + ". Apa kabar?")
4
5 # pemanggilan fungsi
6 # output: Hi, Umar. Apa kabar?
7 sapa('Umar')
```

Below the editor, the Run console shows the execution of the script. The command executed is `C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh44.py`, and the output is `Hi, Umar. Apa kabar?`. The console also indicates that the process finished with exit code 0.

### 3.2. Memanggil Fungsi

Bila fungsi sudah didefinisikan, maka ia sudah bisa dipanggil dari tempat lain di dalam program. Untuk memanggil fungsi caranya adalah dengan mengetikkan nama fungsi berikut paramaternya. Untuk fungsi di atas, kita bisa melakukannya seperti contoh berikut:

```
>>> sapa('Galih')
Hi, Galih. Apa kabar?

>>> sapa('Ratna')
Hi, Ratna. Apa kabar?
```

### 3.3. Docstring

Docstring adalah singkatan dari documentation string. Ini berfungsi sebagai dokumentasi atau keterangan singkat tentang fungsi yang kita buat. Meskipun bersifat opsional, menuliskan docstring adalah kebiasaan yang baik. Untuk contoh di atas kita menuliskan docstring. Cara mengaksesnya adalah dengan menggunakan format `namafungsi.__doc__`

```
>>> print(sapa.__doc__)
"""Fungsi ini untuk menyapa seseorang sesuai nama yang
dimasukkan sebagai parameter"""
```

### 3.4. Pernyataan Return

Pernyataan `return` digunakan untuk keluar dari fungsi dan kembali ke baris selanjutnya dimana fungsi dipanggil.

Adapun sintaks dari `return` adalah:

```
return [expression_list]
```

return bisa berisi satu atau beberapa ekspresi atau nilai yang dievaluasi dan nilai tersebut akan dikembalikan. Bila tidak ada pernyataan return yang dibuat atau ekspresi dikosongkan, maka fungsi akan mengembalikan objek None. Perhatikan bila hasil keluaran dari fungsi sapa kita simpan dalam variabel.

```
>>> keluaran = sapa('Gani')
>>> print(keluaran) None
```

### 3.5. Argumen Fungsi

Kita bisa memanggil fungsi dengan menggunakan salah satu dari empat jenis argumen berikut:

- Argumen wajib (required argument)

Argumen wajib adalah argumen yang dilewatkan ke dalam fungsi dengan urutan posisi yang benar. Di sini, jumlah argumen pada saat pemanggilan fungsi harus sama persis dengan jumlah argumen pada pendefinisian fungsi. Pada contoh fungsi sapa() di atas, kita perlu melewatkan satu argumen ke dalam fungsi sapa(). Bila tidak, maka akan muncul error.

```
>>> sapa('Umar')
Hi Umar. Apa kabar?
```

```
>>> # akan muncul error
>>> sapa()
```

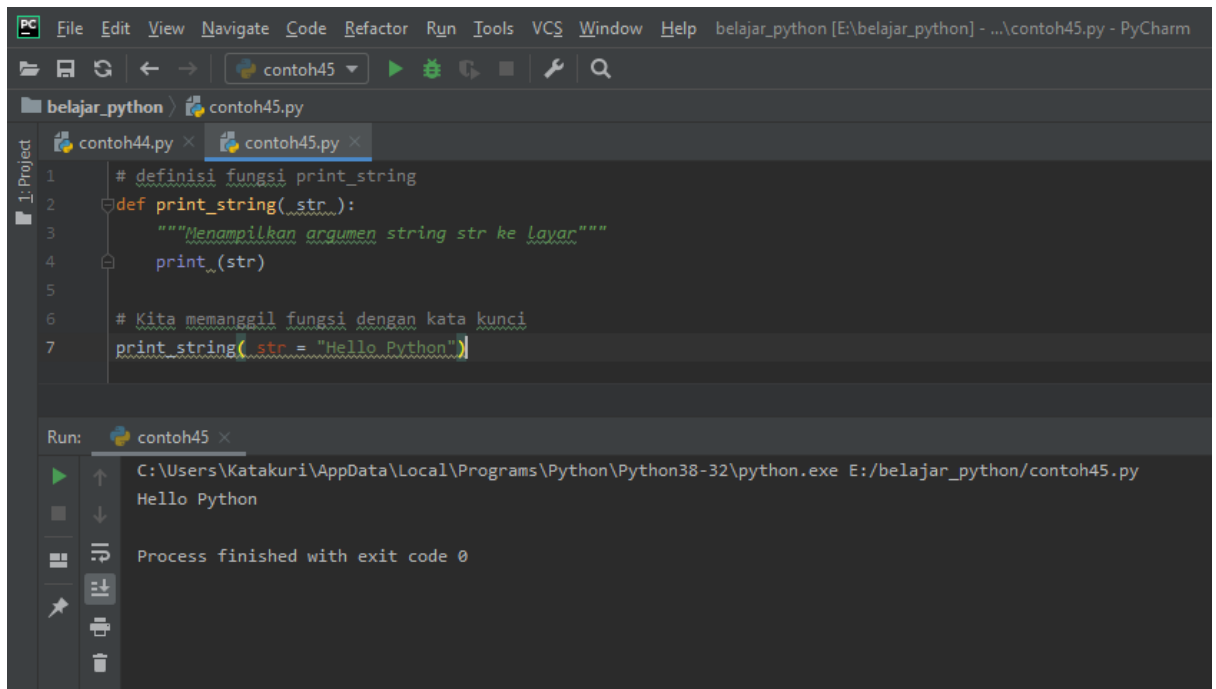
```
Traceback (most recent call last):
```

```
File "<pyshell#5>", line 1, in <module>
    sapa()
```

```
TypeError: sapa() missing 1 required positional
argument: 'nama'
```

- Argumen kata kunci (keyword argument)

Argumen dengan kata kunci berkaitan dengan cara pemanggilan fungsi. Ketika menggunakan argumen dengan kata kunci, fungsi pemanggil menentukan argumen dari nama parameternya. Hal ini membuat kita bisa mengabaikan argumen atau menempatkannya dengan sembarang urutan. Python dapat menggunakan kata kunci yang disediakan untuk mencocokkan nilai sesuai dengan parameternya. Jelasnya ada pada contoh berikut:

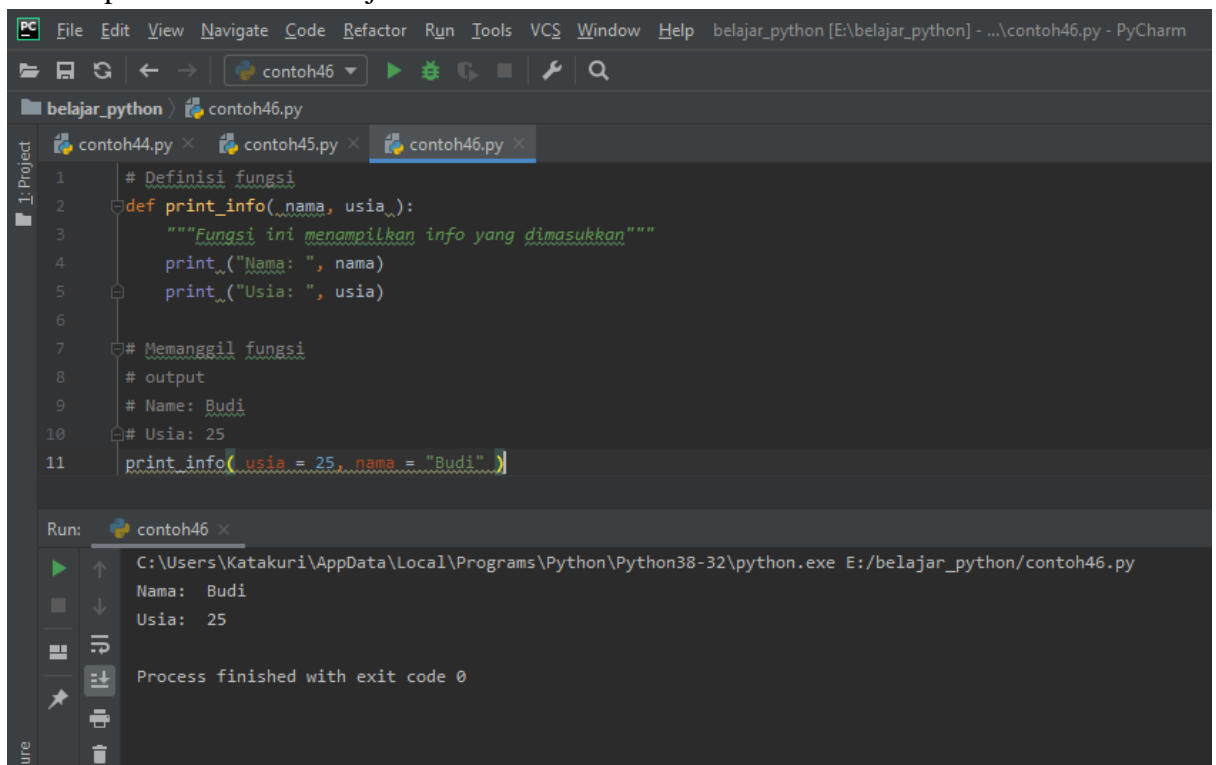


The screenshot shows the PyCharm IDE with a project named 'belajar\_python'. The file 'contoh45.py' is open and contains the following code:

```
1 # definisi fungsi print_string
2 def print_string(_str):
3     """Menampilkan argumen string str ke layar"""
4     print(_str)
5
6 # Kita memanggil fungsi dengan kata kunci
7 print_string(_str = "Hello Python")
```

The Run window shows the execution of 'contoh45.py' using the Python 3.8 interpreter. The output is 'Hello Python' and the process finished with exit code 0.

Urutan parameter tidak menjadi masalah. Perhatikan contoh berikut:

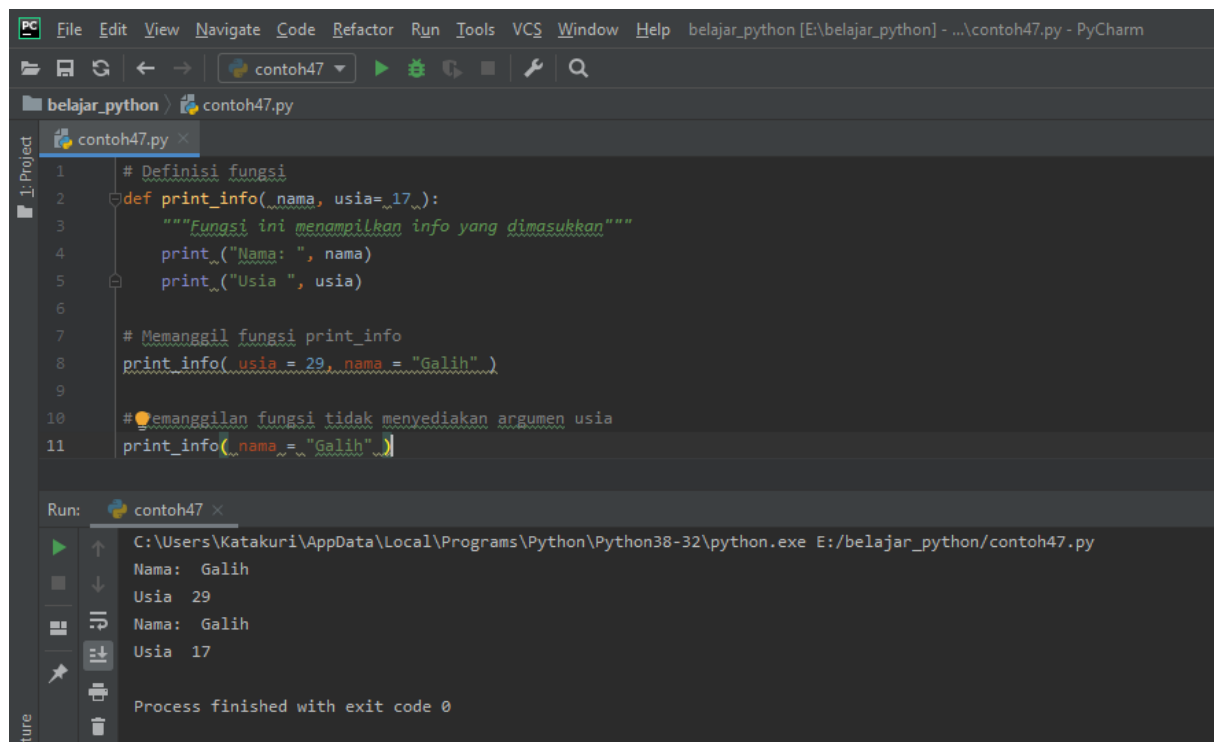


The screenshot shows the PyCharm IDE with a project named 'belajar\_python'. The file 'contoh46.py' is open and contains the following code:

```
1 # Definisi fungsi
2 def print_info(_nama, _usia):
3     """Fungsi ini menampilkan info yang dimasukkan"""
4     print("Nama: ", _nama)
5     print("Usia: ", _usia)
6
7 # Memanggil fungsi
8 # output
9 # Name: Budi
10 # Usia: 25
11 print_info(_usia = 25, _nama = "Budi")
```

The Run window shows the execution of 'contoh46.py' using the Python 3.8 interpreter. The output is 'Nama: Budi' and 'Usia: 25' on separate lines, and the process finished with exit code 0.

- Argumen default  
Fungsi dengan argumen default menggunakan nilai default untuk argumen yang tidak diberikan nilainya pada saat pemanggilan fungsi. Pada contoh berikut, fungsi akan menampilkan usia default bila argumen usia tidak diberikan:



The screenshot shows the PyCharm IDE with a project named 'belajar\_python'. The file 'contoh47.py' is open, showing the following code:

```
1 # Definisi fungsi
2 def print_info(nama, usia=17):
3     """Fungsi ini menampilkan info yang dimasukkan"""
4     print("Nama: ", nama)
5     print("Usia ", usia)
6
7 # Memanggil fungsi print_info
8 print_info(usia = 29, nama = "Galih")
9
10 # Memanggilan fungsi tidak menyediakan argumen usia
11 print_info(nama = "Galih")
```

The Run window shows the output of the script:

```
Run: contoh47 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh47.py
Nama: Galih
Usia 29
Nama: Galih
Usia 17
Process finished with exit code 0
```

Pada contoh di atas, pemanggilan fungsi kedua tidak menyediakan nilai untuk parameter usia, sehingga yang digunakan adalah nilai default yaitu 17.

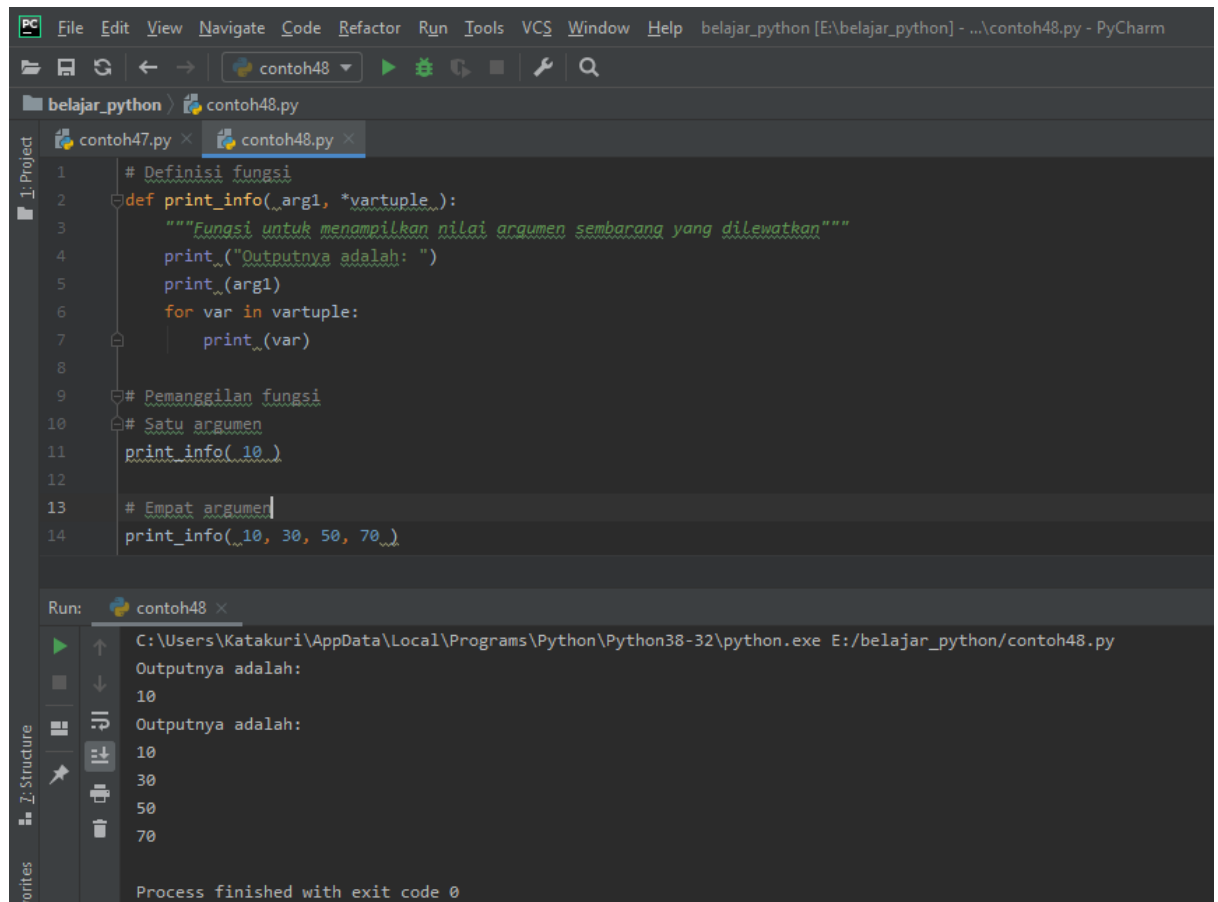
- Argumen dengan panjang sembarang

Terkadang kita butuh untuk memproses fungsi yang memiliki banyak argumen. Nama – nama argumennya tidak disebutkan saat pendefinisian fungsi, beda halnya dengan fungsi dengan argumen wajib dan argumen default. Sintaksnya fungsi dengan argumen panjang sembarang adalah seperti berikut:

```
def function_name([formal_args,] *var_args_tuple):
    """function_docstring"""
    statement(s)
    return [expression]
```

Tanda asterisk (\*) ditempatkan sebelum nama variabel yang menyimpan nilai dari semua argumen yang tidak didefinisikan. Tuple ini akan kosong bila tidak ada argumen tambahan pada saat pemanggilan fungsi. Berikut adalah contohnya:





The screenshot shows the PyCharm IDE interface. The top toolbar includes menus like File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. Below the toolbar, the project structure on the left shows a folder named 'belajar\_python' containing two files: 'contoh47.py' and 'contoh48.py'. The main editor window displays the code for 'contoh48.py'.

```
1 # Definisi fungsi
2 def print_info(arg1, *vartuple):
3     """Fungsi untuk menampilkan nilai argumen sembarang yang dilewatkan"""
4     print("Outputnya adalah: ")
5     print(arg1)
6     for var in vartuple:
7         print(var)
8
9 # Pemanggilan fungsi
10 # Satu argumen
11 print_info(10)
12
13 # Empat argumen
14 print_info(10, 30, 50, 70)
```

Below the editor, the 'Run' panel shows the execution command: `C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh48.py`. The output of the program is displayed in the console:

```
Outputnya adalah:
10
Outputnya adalah:
10
30
50
70

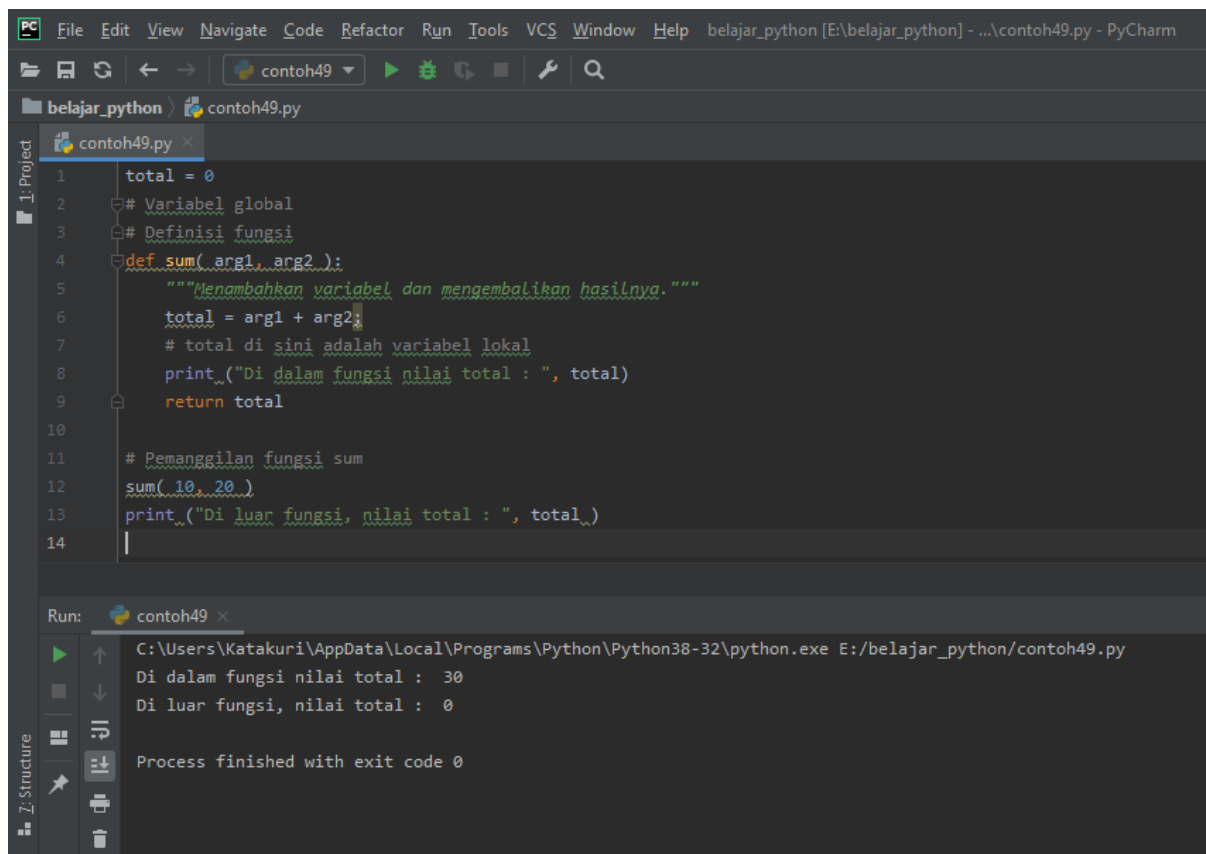
Process finished with exit code 0
```

### 3.6. Ruang Lingkup (Scope) Variabel

Di Python, tidak semua variabel bisa diakses dari semua tempat. Ini tergantung dari tempat dimana kita mendefinisikan variabel. Ruang lingkup variabel ada dua, yaitu:

- Global
- Local

Variabel yang didefinisikan di dalam fungsi memiliki scope lokal, sedangkan variabel yang didefinisikan di luar fungsi memiliki scope global. Ini berarti, variabel lokal hanya bisa diakses dari dalam fungsi di mana ia didefinisikan, sedangkan variabel global bisa diakses dari seluruh tempat dimanapun di dalam program. Berikut adalah contohnya:



The screenshot displays the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The toolbar shows icons for saving, running, and other development actions. The main editor window is titled 'contoh49.py' and contains the following Python code:

```
1 total = 0
2 # Variabel global
3 # Definisi fungsi
4 def sum( arg1, arg2 ):
5     """Menambahkan variabel dan mengembalikan hasilnya."""
6     total = arg1 + arg2
7     # total di sini adalah variabel lokal
8     print("Di dalam fungsi nilai total : ", total)
9     return total
10
11 # Pemanggilan fungsi sum
12 sum( 10, 20 )
13 print("Di luar fungsi, nilai total : ", total)
14
```

Below the editor, the 'Run' console shows the execution results:

```
Run: contoh49 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh49.py
Di dalam fungsi nilai total : 30
Di luar fungsi, nilai total : 0
Process finished with exit code 0
```

Perhatikan bagaimana variabel total di dalam dan di luar fungsi adalah dua variabel yang berbeda.

## Pertemuan 10

### Penganan Eksepsi

Pada saat menulis dan menjalankan program, kita sering dihadapkan pada munculnya kesalahan atau error. Seringkali error menyebabkan program berhenti sendiri.

Error dapat terjadi akibat kesalahan struktur (sintaks) program. Hal ini disebut syntax error. Contohnya adalah seperti berikut:

```
>>> if x < 5

File "<stdin>", line 1

    if x < 5

SyntaxError: invalid syntax
```

Kita bisa melihat bahwa penyebabnya adalah lupa titik dua pada pernyataan if.

Error juga dapat terjadi pada saat runtime (saat program berjalan). Error seperti ini disebut eksepsi. Misalnya, bila kita membuka file yang tidak ada, maka akan muncul pesan kesalahan FileNotFoundError. Bila kita membagi bilangan dengan nol akan muncul ZeroDivisionError, dan lain sebagainya.

Pada saat terjadi eksepsi, Python akan menampilkan traceback dan detail dimana kesalahan terjadi.

```
>>> 1/0

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

ZeroDivisionError: division by zero
```

Tabel 10.1 Daftar Eksepsi Built-in Python

Eksepsi	Penyebab Error
AssertionError	Muncul pada saat pernyataan <code>assert</code> gagal
AttributeError	Muncul pada saat penugasan terhadap attribute atau referensi gagal

EOFError	Muncul saat fungsi <code>input()</code> mendapatkan kondisi akhir file (end-of-file)
FloatingPointError	Muncul saat operasi terhadap bilangan float gagal
GeneratorExit	Muncul saat metode <code>close()</code> generator dipanggil
ImportError	Muncul saat modul yang hendak diimpor tidak ditemukan
IndexError	Muncul saat indeks dari sequence berada di luar range
KeyError	Muncul saat suatu key tidak ditemukan di dalam dictionary
KeyboardInterrupt	Muncul saat user menekan tombol interupsi (Ctrl + C)
MemoryError	Muncul saat operasi kehabisan memori
NameError	Muncul saat variabel tidak ditemukan
NotImplementedError	Muncul oleh metode abstrak
OSError	Muncul saat sistem operasi bersangkutan mengalami error
OverflowError	Muncul saat hasil operasi perhitungan terlalu besar untuk direpresentasikan
ReferenceError	Muncul saat <i>weak reference</i> digunakan untuk mengakses referensi sampah program

RuntimeError	Muncul saat error yang terjadi di luar semua kategori eksepsi lain
StopIteration	Muncul oleh fungsi <code>next()</code> untuk menunjukkan bahwa tidak ada lagi item yang tersisa pada iterator
SyntaxError	Muncul oleh parser saat terjadi kesalahan sintaks
IndentationError	Muncul saat ada indentasi yang salah
TabError	Muncul saat indentasi memiliki jumlah spasi atau tab yang tidak konsisten
SystemError	Muncul saat interpreter mendeteksi kesalahan internal
SystemExit	Muncul oleh fungsi <code>sys.exit()</code>
TypeError	Muncul saat melakukan operasi pada tipe data yang tidak sesuai
UnboundLocalError	Muncul saat referensi dibuat untuk variabel lokal dari fungsi, tapi tidak ada nilainya.
UnicodeError	Muncul saat terjadi kesalahan berkenaan dengan encoding dan decoding unicode
UnicodeEncodeError	Muncul saat terjadi kesalahan pada proses encoding
UnicodeDecodeError	Muncul saat terjadi kesalahan pada proses decoding

UnicodeTranslateError	Muncul saat terjadi kesalahan berkenaan dengan penerjemahan unicode
ValueError	Muncul saat fungsi menerima argumen yang tipe datanya salah
ZeroDivisionError	Muncul saat terjadi operasi pembagian bilangan dengan nol

### 3.1. Menangani Eksepsi Dengan Try, Except, dan Finally

Terjadinya eksepsi pada program dapat menyebabkan program terhenti. Untuk mencegah hal tersebut, kita harus mengantisipasi hal tersebut. Python menyediakan metode penanganan eksepsi dengan menggunakan pernyataan try dan except.

Di dalam blok try kita meletakkan baris program yang kemungkinan akan terjadi error. Bila terjadi error, maka penanganannya diserahkan kepada blok except. Berikut adalah contoh penanganan eksepsi pada operasi pembagian bilangan.

```

1  # import modul sys untuk memperoleh jenis eksepsi
2  import sys
3
4  lists = ['a', 0, 4]
5  for each in lists:
6      try:
7          print("Masukan:", each)
8          r = 1/int(each)
9          break
10     except:
11         print("Upps!", sys.exc_info()[0], " terjadi.")
12         print("Masukan berikutnya.")
13         print()
14     print("Kebalikan dari ", each, " = ", r)
15
16 for each in lists:
17     try:
18         print("Masukan:", each)
19         r = 1/int(each)
20         print("Kebalikan dari ", each, " = ", r)
21     except:
22         print("Upps!", sys.exc_info()[0], " terjadi.")
23         print("Masukan berikutnya.")
24         print()

```

Run: contoh51 x

```

C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh51.py
Masukan: a
Upps! <class 'ValueError'> terjadi.
Masukan berikutnya.

Masukan: 0
Upps! <class 'ZeroDivisionError'> terjadi.
Masukan berikutnya.

Masukan: 4
Kebalikan dari 4 = 0.25

```

Pada program di atas kita mencari kebalikan dari bilangan, misalnya 4, maka kebalikannya adalah  $1/4 = 0.25$ .

Pembagian dengan huruf 'a', dan juga dengan 0 tidak bisa dilakukan, sehingga muncul error. Bila tidak dilakukan penanganan eksepsi, maka program akan langsung terhenti pada saat terjadi error.

### 3.2. Menangani Eksepsi Tertentu

Pada contoh di atas kita hanya menangani error secara umum. Tidak dikelompokkan, apakah dia adalah `TypeError`, `ValueError`, `SyntaxError`, dan lain sebagainya. Sebuah pernyataan `try`, bisa memiliki sejumlah pernyataan `except` untuk menangani jenis – jenis eksepsi secara lebih spesifik. Kita juga bisa mendefinisikan beberapa error sekaligus menggunakan tuple. Contohnya adalah seperti berikut:

```
try:
    # lakukan sesuatu
    pass

except ValueError:
    # tangani eksepsi ValueError
    pass

except (TypeError, ZeroDivisionError):
    # menangani multi eksepsi
    # TypeError dan ZeroDivisionError
    pass

except:
    # menangani eksepsi lainnya
    pass
```

Pernyataan `pass` adalah pernyataan yang tidak melakukan apa-apa. Istilahnya adalah statemen kosong. `pass` sering digunakan untuk mengisi blok fungsi atau kelas yang masih kosong.

### 3.3. Memunculkan Eksepsi

Eksepsi muncul bila terjadi error pada saat runtime atau saat program berjalan. Akan tetapi, kita juga bisa memunculkan eksepsi dengan sengaja untuk maksud tertentu dengan menggunakan kata kunci `raise`. Contohnya adalah seperti berikut:

```
>>> raise KeyboardInterrupt
Traceback (most recent call last):
...
KeyboardInterrupt

>>> try:
    a = int(input("Masukkan sebuah bilangan positif: "))
    if a <= 0:
        raise ValueError("Itu bukan bilangan positif!")
```

```
except ValueError as ve:  
    print(ve)
```

Masukkan sebuah bilangan positif: -3  
Itu bukan bilangan positif!



## Pertemuan 11

### Object-Oriented Programming (OOP)

**OOP** (Object Oriented Programming) adalah suatu metode pemrograman yang berorientasi kepada objek. **Tujuan dari OOP diciptakan** adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Jadi setiap bagian dari suatu permasalahan adalah objek, nah objek itu sendiri merupakan gabungan dari beberapa objek yang lebih kecil lagi.

Sebagai salah satu contoh Pesawat, Pesawat adalah sebuah objek. Pesawat itu sendiri terbentuk dari beberapa objek yang lebih kecil lagi seperti mesin, roda, baling-baling, kursi, dll. Pesawat sebagai objek yang terbentuk dari objek-objek yang lebih kecil saling berhubungan, berinteraksi, berkomunikasi dan saling mengirim pesan kepada objek-objek yang lainnya. Begitu juga dengan program, sebuah objek yang besar dibentuk dari beberapa objek yang lebih kecil, objek-objek itu saling berkomunikasi, dan saling berkirim pesan kepada objek yang lain.

Tabel 11.1 Istilah Dalam OOP

Istilah	Penjelasan
Class	Prototipe yang ditentukan pengguna untuk objek yang mendefinisikan seperangkat atribut yang menjadi ciri objek kelas apa pun. Atribut adalah data anggota (variabel kelas dan variabel contoh) dan metode, diakses melalui notasi titik.
Object	Contoh unik dari struktur data yang didefinisikan oleh kelasnya. Objek terdiri dari kedua anggota data (variabel kelas dan variabel contoh) dan metode.
Method	Jenis fungsi khusus yang didefinisikan dalam definisi kelas.
Instance	Objek individu dari kelas tertentu. Obyek obj yang termasuk dalam Lingkaran kelas, misalnya, adalah turunan dari Lingkaran kelas.
Instantiation	Penciptaan sebuah instance dari sebuah kelas.
Class variable	Sebuah variabel yang dibagi oleh semua contoh kelas. Variabel kelas didefinisikan dalam kelas tapi di luar metode kelas manapun. Variabel kelas tidak digunakan sesering variabel contoh.
Data member	Variabel kelas atau variabel contoh yang menyimpan data yang terkait dengan kelas dan objeknya.
Function overloading	Penugasan lebih dari satu perilaku ke fungsi tertentu. Operasi yang dilakukan bervariasi menurut jenis objek atau argumen yang terlibat.
Operator overloading	Penugasan lebih dari satu fungsi ke operator tertentu.
Inheritance	Pengalihan karakteristik kelas ke kelas lain yang berasal darinya.
Instantiation	Penciptaan sebuah instance dari sebuah kelas.

#### 3.1. Konsep OOP

- **Kelas** merupakan deskripsi abstrak informasi dan tingkah laku dari sekumpulan data.
- **Kelas** dapat diilustrasikan sebagai suatu cetak biru(blueprint) atau prototipe yang digunakan untuk menciptakan objek.
- **Kelas** merupakan tipe data bagi objek yang mengenkapsulasi data dan operasi pada data dalam suatu unit tunggal.
- **Kelas** mendefinisikan suatu struktur yang terdiri atas data kelas (data field), prosedur atau fungsi (method), dan sifat kelas (property).

### 3.2. Class

Pada konsep pemrograman berbasis object, anda tidak akan asing lagi mendengar istilah class, object, attribute, behaviour, inheritance, dll. Semua itu pasti akan anda temui di semua bahasa pemrograman yang support OOP. Jika dianalogikan, class merupakan suatu tubuh dari OOP. Class merupakan abstraksi atau *blueprint* yang mendefinisikan suatu object tertentu. Class akan menampung semua attribute dan perilaku dari object itu. Berikut contoh implementasi class pada Python:

```
class Car:

    color = 'black'
    transmission = 'manual'

    def __init__(self, transmission):
        self.transmission = transmission
        print('Engine is ready!')

    def drive(self):
        print('Drive')

    def reverse(self):
        print('Reverse. Please check your behind.')
```

Jika diperhatikan, dalam class **Car** terdapat 2 attribute yaitu **color = 'black'**, **transmission = 'manual'** dan method yaitu **drive()**, **reverse()**. Method dalam konsep OOP mewakili suatu 'behaviour' dari class atau object itu sendiri. Kita akan bahas lebih detail mengenai method.

### 3.3. Function/Method

Fungsi method dalam konsep OOP adalah untuk merepresentasikan suatu behaviour. Dalam contoh di atas suatu object 'mobil' memiliki behaviour antara lain adalah bergerak dan mundur. Suatu method bisa juga memiliki satu atau beberapa parameter, sebagai contoh:

```
gear_position = 'N'

def change_gear(self, gear):
    self.gear_position = gear
    print('Gear position on: ' + self.gear_position)
```

Pada method **change\_gear()** terdapat 1 parameter yaitu **gear**. Ketika method tersebut dipanggil dan anda tidak memberikan value pada parameter tersebut, maka program akan melempar error. Bagaimanapun juga parameter yang sudah didefinisikan pada suatu method harus memiliki value meskipun value tersebut `None`. Cara lainnya adalah dengan mendefinisikan default value pada parameter tersebut sejak awal method tersebut dibuat:

```
gear_position = 'N'

def change_gear(self, gear='N'):
    self.gear_position = gear
    print('Gear position on: ' + self.gear_position)

self.change_gear()
```

```
>>> 'Gear position on: N'
self.change_gear('R')
>>> 'Gear position on: R'
```

Jika diperhatikan, terdapat keyword **self** pada salah satu parameter method di atas. Keyword **self** mengacu pada Class Instance untuk mengakses attribute atau method dari class itu sendiri. Dalam bahasa pemrograman Java, terdapat keyword **this** yang memiliki fungsi yang mirip dengan keyword **self** pada Python. Pemberian keyword **self** pada parameter awal suatu method menjadi wajib jika anda mendefinisikan method tersebut di dalam block suatu class.

Suatu method juga bisa mengembalikan suatu value ketika method tersebut dipanggil. Berikut contoh implementasinya:

```
def get_gear_position(self):
    return self.gear_position

gear_position = self.get_gear_position()
```

### 3.4. Constructor

Pada contoh awal tentang penjelasan class, terdapat sebuah method bernama **\_\_init\_\_()**. Method itulah yang disebut dengan constructor. Suatu constructor berbeda dengan method lainnya, karena constructor akan otomatis dieksekusi ketika membuat object dari class itu sendiri.

```
class Car:
    color = 'black'
    transmission = 'manual'

    def __init__(self, transmission):
        self.transmission = transmission
        print('Engine is ready!')

    ...

honda = Car('automatic')
>>> 'Engine is ready!'
```

Ketika object honda dibuat dari class **Car**, constructor langsung dieksekusi. Hal ini berguna jika anda membutuhkan proses inisialisasi ketika suatu object dibuat. Suatu constructor juga bisa memiliki satu atau beberapa parameter, sama seperti method pada umumnya namun constructor tidak bisa mengembalikan value.

### 3.5. Object

Object merupakan produk hasil dari suatu class. Jika class merupakan blueprint dari suatu rancangan bangunan, maka object adalah bangunan itu sendiri. Begitulah contoh analogi yang bisa saya gambarkan mengenai relasi antara class dan object. Berikut contoh implementasi dalam bentuk code program:

```
class Car:

    color = 'black'
```

```

transmission = 'manual'
gear_position = 'N'

def __init__(self, transmission):
    self.transmission = transmission
    print('Engine is ready!')

def drive(self):
    self.gear_position = 'D'
    print('Drive')

def reverse(self):
    self.gear_position = 'N'
    print('Reverse. Please check your behind.')

def change_gear(self, gear='N'):
    self.gear_position = gear
    print('Gear position on: ' + self.gear_position)

def get_gear_position(self):
    return self.gear_position

car1 = Car('manual')
car1.change_gear('D-1')

car2 = Car('automatic')
gear_position = car2.get_gear_position()
print(gear_position)
>>> 'N'

```

Dari contoh di atas, terdapat 2 buah object **car1** dan **car2** yang dibuat dari class yang sama. Masing-masing dari object tersebut berdiri sendiri, artinya jika terjadi perubahan attribute dari object **car1** tidak akan mempengaruhi object **car2** meskipun dari class yang sama.

### 3.6. Inheritance

Salah satu keuntungan dari konsep OOP ialah *reusable codes* yang bisa mengoptimalkan penggunaan code program agar lebih efisien dan meminimalisir redundansi.

- **Kita dapat mendefinisikan** suatu kelas baru dengan mewarisi sifat dari kelas lain yang sudah ada.
- **Penurunan sifat** ini bisa dilakukan secara bertingkat tingkat, sehingga semakin ke bawah kelas tersebut menjadi semakin spesifik.
- **Sub kelas** memungkinkan kita untuk melakukan spesifikasi detail dan perilaku khusus dari kelas supernya.
- **Dengan konsep pewarisan**, seorang programmer dapat menggunakan kode yang telah ditulisnya pada kelas super berulang kali pada kelas-kelas turunannya tanpa harus menulis ulang semua kodekode itu.

Semua itu berkat adanya fitur inheritance yang memungkinkan suatu class (parent) menurunkan semua attribute dan behaviour nya ke class (child) lain. Berikut contoh penerapannya:

```

class Tesla(Car):
    pass    # use 'pass' keyword to define class only

tesla = Tesla()
tesla.drive()

```

```
>>> 'Drive'
```

Pada potongan code di atas, class **Tesla** merupakan turunan dari class **Car**. Jika diperhatikan pada class **Tesla** tidak didefinisikan method **drive()** namun class tersebut bisa memanggil method **drive()**. Method tersebut berasal dari class parentnya yaitu class **Car**, sehingga tidak perlu lagi didefinisikan ulang pada class childnya. Dengan cara seperti ini anda bisa melakukan reusable codes sehingga source code menjadi lebih *clean*.

### 3.7. Overriding

Ada suatu kondisi dimana suatu method yang berasal dari parent ingin anda modifikasi atau ditambahkan beberapa fitur sesuai kebutuhan pada class child, disinilah peran dari 'overriding method'. Dengan menggunakan fungsi **super()**, anda bisa memanggil instance dari class parent di dalam suatu method untuk memanggil fungsi dari parent tersebut. Perhatikan contoh di bawah ini:

```
class Tesla(Car):  
  
    def drive(self):  
        super().drive()  
        print('LOL Gas')
```

### 3.8. Private Attribute/Function

Tidak semua attribute maupun method bisa diturunkan pada class child. Anda bisa menentukan mana attribute atau method yang ingin diproteksi agar tidak bisa digunakan pada class turunannya. Berikut caranya:

```
__factory_number = '0123456789'  
  
def __get_factory_number(self):  
    return self.__factory_number
```

### 3.9. Polymorphism

polimorfisme yang memungkinkan anda untuk membuat banyak bentuk dari satu object. Berikut contoh implementasinya:

```
class Car:  
    def fuel(self):  
        return 'gas'  
  
class Honda(Car):  
    pass  
  
class Tesla(Car):  
    def fuel(self):  
        return 'electricity'  
  
def get_fuel(car):  
    print(car.fuel())  
  
get_fuel(Tesla())  
get_fuel(Honda())  
>>> 'electricity'  
>>> 'gas'
```

- **Polimorfisme** merupakan kemampuan objek objek yang berbeda kelas namun terkait dalam pewarisan untuk merespon secara berbeda terhadap suatu pesan yang sama.
- **Polimorfisme** juga dapat dikatakan kemampuan sebuah objek untuk memutuskan method mana yang akan diterapkan padanya, tergantung letak objek tersebut pada jenjang pewarisan.