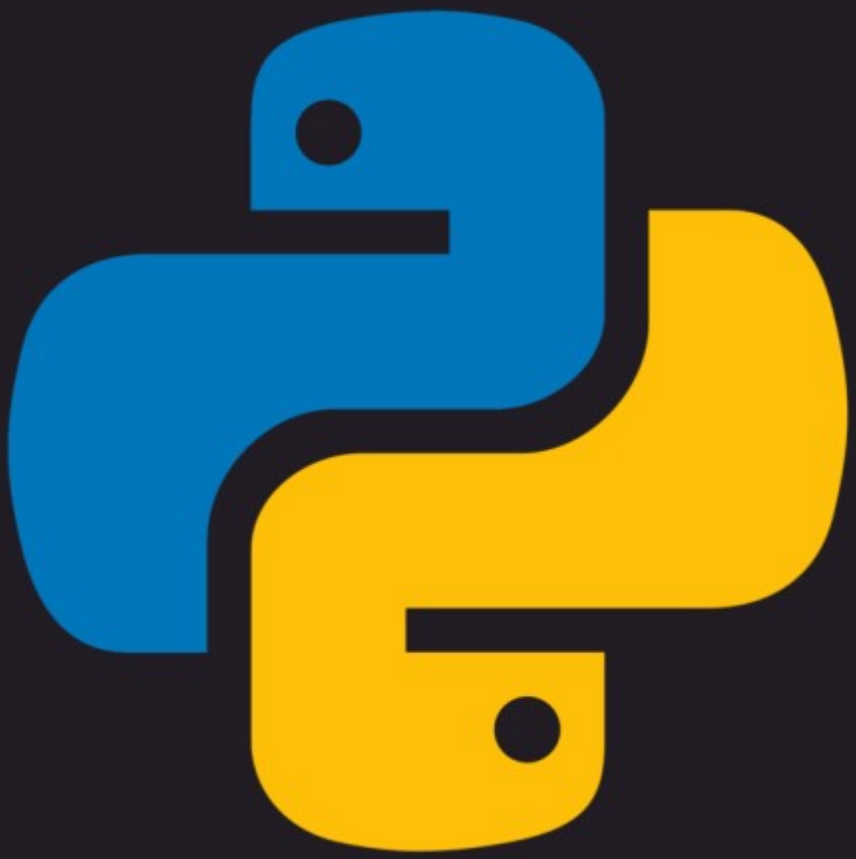
Python for Beginners



Lukas S. Tanuwijaya

Daftar Isi

Cara Menginstall Visual Studio Code pada Windows

1. Pertama kita harus download terlebih dahulu file intaler Visual Studio Code melalui situs resminya https://code.visualstudio.com/
2. Double klik pada file installer nya atau klik kanan kemudian pilih Run as Administrator.

Jika muncul peringatan Run as Administrator, silahkan klik Yes.

3. Pilih “I accept the aggrement” untuk menyetujui “License Agreement”, kemudian klik Next.
4. Untuk Select Destination Location bisa di biarkan saja jika lokasi instalasi tidak akan dirubah. Klik Next.
5. Klik Next lagi jika tidak akan merubah Start Menu Folder.
6. Di bagian Select Additional Tasks centang semua. Kemudian Next.
7. Lalu klik Install untuk memulai proses instalasi.
8. Tunggu sampai proses instalasi selesai.
9. Setelah selesai klik Finish

Cara Menginstall Visual Studio Code pada MacOS

1. Pertama download Visual Studio Code pada situs https://code.visualstudio.com/
2. Double klik pada arsip download untuk membuka filenya.
3. Tarik Visual Studio Code.app ke folder Applications agar tersedia di dalam Launchpad
4. Tambahkan VSCode ke dalam Dock-mu dengan mengklik kanan pada ikon dan pilih opsi “Keep in Doc”.

Cara Menginstall Visual Studio Code pada Ubuntu

1. Pertama, download file “.deb” pada situs resmi VSCode.
2. Bukalah paket “.deb” untuk VSCode yang sebelumnya sudah terdownload.
3. Lakukan setup instalasi seperti menginstall sebuah aplikasi di Windows pada umumnya.

Pelajaran 1: Basic (Hello World)

Program paling awal dalam mempelajari bahasa pemgrograman adalah membuat program Hello World. Dalam pelajaran ini, kita akan belajar untuk menampilkan sebuah kalimat “Hello World” yang nantinya akan tampil dalam output atau terminal.

Input:

```
print ("Hello, World!")
```

Output:

```
Hello World
```

Penjelasan:

Syntax “print” adalah syntax paling dasar yang dipelajari dalam Python. Syntax tersebut berguna untuk menampilkan kalimat atau kata dalam terminal. Syntax itu akan sangat sering dipakai dalam bahasa pemrograman.

Pelajaran 2: Make a Shape

Dalam pelajaran ini, kita masih akan menggunakan syntax print untuk menampilkan sesuatu dalam terminal. Sekarang cobalah untuk membuat sebuah bentuk ataupun tulisan 2D.

Input:

```
print ("_______   _____   _____")
print ("|    |   |   |   |   |")
print ("|    |   |   |   |   |")
print ("|    |____|   |   |   |")
print ("|     _____   |   |   |")
print ("|    |    |   |   |   |")
print ("|    |    |   |   |   |")
print ("|____|    |___|   |___|")
```

Output:

```
|    |    |   |   |   |
|    |    |   |   |   |
|    |____|   |   |   |
|     _____    |   |   |
|    |    |    |   |   |
|    |    |    |   |   |
|____|    |___|   |___|
```

Ketika program dieksekusi dalam terminal, akan nampak tulisan “HI” namun dengan bentuk balok. Walaupun pada dokumen ini, output yang ditampilkan kurang menampakkan bentuk “HI”, kalian dapat mencobanya dalam terminal atau IDE.

Pelajaran 3: Multiplication

Di dalam pelajaran ini, kita akan belajar cara untuk mengalikan kalimat atau kata. Contohnya jika kita mau mengalikan kata “Hello ” tiga kali, maka hasilnya nanti adalah “Hello Hello Hello”. Sekarang simak kode berikut….

Input:

```
print (“Hello” * 3)
```

Output:

```
HelloHelloHello
```

Pada output nampak tulisan “Hello” yang dikalikan tiga menjadi “HelloHelloHello”. Lalu bagaimana supaya satu tulisan “Hello” tidak berdempetan dengan tulisan lainnya? Kita dapat menambahkan spasi setelah kata “Hello”.

Input:

```
print (“Hello “ * 3)
```

Output:

```
Hello Hello Hello
```

Sekarang nampak output yang ditampilkan adalah tulisan “Hello” yang telah dikalikan tiga dengan tulisan yang tidak saling berdempetan.

Pelajaran 4: Variable

Dalam setiap bahasa pemgrograman selalu ada sesuatu yang mencakup sebuah nilai, yang dinamakan dengan “variabel”. Lalu bagaimana cara untuk membuat sebuah variabel dalam bahasa pemrograman Python? Seandainya kita ingin membuat sebuah toko online, maka kita akan membutuhkan beberapa info seperti nama barang, harga, nama penjual, dan lain sebagainya. Sekarang simaklah kode berikut…

Input:

```
item_name = ‘Spoon’
price = 20
rating = 4.9
seller = 'Bob'
is_published = True
```

Input di atas ini adalah cara untuk membuat beberapa variabel yang mencakup beberapa nilai yang berbeda. Variabel “item_name” berisikan nama barang yang dijual, variabel “price” berisikan harga barang yang dijual, variabel “rating” berisikan nilai ulasan untuk barang yang dijual, variabel “seller” berisikan keterangan nama si penjual, dan variabel “is_published” berisikan keterangan apakah barangnya sudah dipublikasikan ke shop atau belum.

Tipe-tipe Data:

1. String (Teks)
2. Integer (Nilai bulat)
3. Float (Nilai desimal)
4. Boolean (True/False)

Exercise

We check in a patient named John Smith. He is 20 year old and he is a new patient.

Solution:

```
patient_name = ‘John Smith’
age = 20
is_new = True
```

Pelajaran 5: Getting Input

Dalam pemrograman, juga ada yang dinamakan dengan “Input”. Input ini berfungsi untuk meminta sebuah jawaban dari user untuk dijadikan sebuah data dalam program. Sekarang simaklah kode berikut…

Input:

```
Name = input(‘What is your name? ‘)
print (“Hello, “ + Name)
```

Output:

```
What is your name? Mosh
Hello, Mosh
```

Ketika program dijalankan atau dieksekusi, program akan meminta keterangan dari user untuk dijadikan sebuah dalam dalam bentuk variabel. Dengan variabel yang telah diisi data tersebut, kita dapat melakukan sesuatu terhadap data itu.

Exercise

Ask two questions: person's name and the favorite colour. Then, print a message like "Mosh likes blue"

Solution:

```
Name = input(‘What is your name? ‘)
Fav_Colour = input(‘What is your favorite colour? ‘)
print (Name + “ likes “ + Fav_Colour
```

Output:

```
What is your name? Mosh
What is your favorite colour? Blue
Mosh likes Blue
```

Pelajaran 6: Type Conversion

Sebelumnya kita telah mengenal macam-macam tipe data dalam Python. Sekarang kita akan mengubah atau mengonversikan salah satu tipe data ke tipe data yang lain. Coba simaklah kode berikut…

Input:

```
birth_year = input(‘Birth year: ‘)
age = 2021 – int(birth_year)
print (age)
```

Output:

```
Birth year: 2004
17
```

Program tersebut bertujuan untuk mengetahui berapa umur seseorang hanya dengan memasukkan data tahun kelahiran seseorang. Namun, pada baris kedua terdapat syntax baru yang tertulis “int(birth_year)”. Syntax tersebut berguna untuk mengonversikan data dari variabel birth_year ke integer. Itu dikarenakan input memiliki tipe default string, yang berarti ketika melewati kode baris kedua, string tidak dapat dikurangi dengan angka sehingga mengakibatkan eror pada saat program dijalankan.

Macam-macam konversi nilai:

1. str(variabel) Untuk mengubah data ke nilai string
2. int(variabel) Untuk mengubah data ke nilai integer
3. float(variabel) Untuk mengubah data ke nilai float
4. bool(variabel) Untuk mengubah data ke nilai boolean

Exercise

Sekarang, cobalah untuk menghitung persenan baterai ponsel yang telah habis di ponsel kalian. Contohnya baterai kalian sekarang tersisa 40%, carilah berapa persen yang telah habis pada ponsel kalian.

Solution:

```
Now = input(‘Baterai saat ini: ‘)
Wasted = 100 – int(Now)
print (“Baterai yang telah habis: “ + Wasted + “%”
```

Pelajaran 7: Strings

Dalam menampilkan sebuah kalimat atau kata, kita dapat menggunakan syntax print. Seringkali juga kita melihat penggunaan tanda petik (“) dalam print ada yang menggunakan tanda petik satu dan tanda petik dua. Manakah yang sebaiknya kita gunakan untuk membuat string? Coba simak kode berikut ini…

Input:

```
print (‘’Renatta’s Cake’)
```

Output:

```
Invalid syntax
```

Sekarang, coba kalian amati mengapa hasil eksekusi program menjadi eror? Itu karena program kebingungan dalam membaca tanda petik dalam syntax print. Jika kita ingin menampilkan sebuah kalimat dengan tanda petik satu di dalamnya, kita harus menggunakan tanda petik dua untuk memasukkan string dalam syntax print.

Input:

```
print (“Renatta’s Cake”)
```

Output:

```
Renatta’s Cake
```

Sekarang, program dapat dijalankan dengan baik tanpa adanya eror. Itu karena tanda petik yang digunakan sudah tidak lagi menggangu tulisan string dalam syntax print. Lalu bagaimana jadinya jika ingin mengampilkan kalimat yang mencakup tanda petik dua di dalamnya?

Input:

```
print (‘Python for “Beginners”’)
```

Output:

```
Python for “Beginners”
```

Dengan begitu, program akan dapat membaca string dengan jelas dan benar. Tanda petik pada awalan dan akhiran haruslah sama karena itu yang akan menjadi tanda awalan dan akhiran kalimat atau kata.

Lalu bagaimana cara untuk membuat string dengan menggunakan beberapa baris kode? Simaklah kode berikut…

Input:

```
Print (‘”
Hi, John

This is our message to you

Thank you,
The Support Team”’)
```

Output:

```
Hi, John

This is our message to you

Thank you,
The Support Team
```

Input di atas adalah bagaimana cara untuk membuat string yang membutuhkan beberapa baris kode. Untuk membuat multi-line string, kita hanya perlu membuat tanda petik sebanyak 3 kali di awalan dan juga di akhiran. Tanda petik yang digunakan juga harus sama pada awalan dan akhiran.

Sekarang, kita akan memelajari metode lain yang dapat kita lakukan terhadap string. Bagaimana caranya untuk menampilkan huruf-huruf berdasarkan nomor indeks terhadap sebuah string? Simaklah kode berikut ini…

Input:

```
Name = ‘Jennifer’
print (Name[0] + “” + Name[1:3])
```

Output:

```
J en
```

Untuk memahaminya, kita perlu mengerti bagaimana cara atau proses memandang sebuah string dengan nomor indeks. Angka indeks dimulai dari angka 0, dan kemudian dilanjutkan dengan angka 1 dan seterusnya. Jika menggunakan string ‘Jennifer’, maka huruf “J” merupakan indeks 0, huruf “e” merupakan indeks 1 dan seterusnya. Jika kita menampilkan nilai dari “Name[0]” maka akan tampil huruf “J” yang diambil dari string “Jennifer”. Jika kita menampilkan nilai dari “Name[1:3]”, maka akan tampil “Jen”. [1:3] memiliki arti bahwa nilai tersebut diambil dari indeks ke 1 hingga ke 2, sedangkan 3 tidak termasuk dan hanya menjadi sebuah batasan.

Pelajaran 8: Formatted Strings

Apa yang dimaksud dengan formatted strings? Formatted strings adalah pembuatan strings dengan cara atau metode yang berbeda namun tetap menggunakan syntax print. Sekarang simaklah kode berikut…

Input:

```
first = 'John'
last = 'Smith'
msg = f'{first} [{last}] is a coder'
print (msg)
```

Output:

```
John [Smith] is a coder
```

Jika dibandingkan dengan cara biasanya, tentu akan lebih cepat dan nyaman jika kita menggunakan formatted strings. Namun, hal itu kembali kepada pribadi masing-masing, apakah kalian lebih nyaman menggunakan formatted strings atau menggunakan metode yang biasanya.

Pelajaran 9: String Methods

Ada banyak hal yang dapat kita lakukan terhadap string. Dalam pelajaran ini, kita akan belajar menggunakan beberapa metode terhadap sebuah string.

Method 1: Len

Input:

```
course = 'Python for Beginners'
print (len(course))
```

Output:

```
20
```

Len berfungsi untuk menghitung jumlah semua karakter yang terkadung dalam sebuah string.

Method 2: Upper

Input:

```
course = 'Python for Beginners'
print (course.upper())
```

Output:

```
PYTHON FOR BEGINNERS
```

Upper berfungsi untuk mengkapilatisasi semua karakter yang terkandung dalam sebuah string.

Method 3: Lower

Input:

```
course = 'Python for Beginners'
print (course.lower())
```

Output:

```
python for beginners
```

Lower berfungsi untuk mengubah semua karakter yang terkandung dalam sebuah string menjadi huruf non-kapital.

Method 4: Find

Input:

```
course = 'Python for Beginners'
print (course.find('P'))
```

Output:

```
0
```

Find berfungsi untuk mencari nilai index dari salah satu karakter yang terkandung dalam sebuah string.

Method 5: Replace

Input:

```
course = 'Python for Beginners
print (course.replace('Beginners', 'Professionals'))
```

Output:

```
Python for Professionals
```

Replace berfungsi untuk mengubah nilai menjadi nilang string yang lain, yang terkandung dalam sebuah string.

Method 6: Checks Boolean Value

Input:

```
course = 'Python for Beginners'
print ('Python' in course)
```

Output:

```
True
```

Metode ini berfungsi untuk mencari keberadaan sebuah nilai dalam sebuah string.

Pelajaran 10: Arithmetic Operations

Dalam pemrograman selalu ada suatu pengoperasian aritmatika. Dalam pelajaran ini kita akan mempelajari cara membuat pengoperasian aritmatika dalam Python. Sekarang simaklah kode berikut ini…

Input:

```
x = 10
y = 5
print (x + y)
```

Output:

```
15
```

Diketahui nilai variabel x adalah 10 dan variabel y adalah 5. Kemudian, program akan menampilkan nilai dari variabel x setelah ditambahkan dengan nilai dari variabel y, maka hasilnya akan menampilkan 15 (Berasal dari 10 + 5).

Contoh lainnya yaitu…

```
x = 50
y = x + 600
print (y)
```

Output:

```
650
```

Macam-macam operasi aritmatika:

1. Penjumlahan (+)
2. Pengurangan (-)
3. Perkalian (*)
4. Pembagian (/)
5. Pembagian Bernilai Integer (//)
6. Modulus (%)
7. Eksponen (**)

Augmented Operation

Contoh:

```
x = 10
x += 5
print (x)
```

Output:

```
15
```

Diketahui sebuah variabel x bernilai 10, kemudian “x += 5” berarti nilai x dijumlahkan dengan 5. Kemudian ketika program dijalankan, akan tampil nilai x setelah melalui proses “x += 5” menjadi 15. Operasi ini juga berlaku untuk operasi matematika lainnya seperti pengurangan, perkalian, pembagian, dan lain sebagainya.

Contoh lain:

```
x = 11
x -= 1
print (x)
```

Output:

```
10
```

Pelajaran 11: Operator Precendence

Dalam matematika, kita mengenal berbagai tingkatan operasi hitungan yang memiliki kasta atau tahta. Contohnya ketika terdapat operasi perkalian dan penjumlahan dalam sebuah soal, maka untuk mendapatkan hasilnya, operasi perkalian haruslah yang dijalankan terlebih dahulu sebelum menjalankan operasi pertambahan. Di pelajaran ini, kita akan mempelajari tingkatan operasi tersebut.

Contoh:

```
x = (5 + 5) * 7 - 8 ** 2
print (x)
```

Output:

```
6
```

Jika operasi tersebut dihitung secara langkah-langkah, maka akan seperti berikut…

$x = (5 + 5) \times 7 - 8^2$

$x = 10 \times 7 - 8^2$

$x = 10 \times 7 - 64$

$x = 70 - 64$

$x = 6$

Tingkatan Utama Operasi Aritmatika:

1. Brackets (Dalam kurung)
2. Exponentiation (Eksponensial atau Pemangkatan)
3. Multiplication or Division (Perkalian atau Pembagian)
4. Addition or Substraction (Penjumlahan atau Pengurangan)

Pelajaran 12: Math Functions

Dalam pelajaran ini, kita akan memelejari fungsi matematika di dalam pemrograman bahasa Python. Ada beberapa syntax yang memerlukan module dan ada juga yang tidak memerlukan module. Sekarang simaklah kode berikut…

1. Round (Pembulatan)

Input:

```
X = 2.89
print (round(X))
```

Output:

```
3
```

Terlihat pada output menghasilkan nilai 3 yang merupakan nilai atau hasil pembulatan dari variabel X.

2. Absolutely (Nilai Mutlak)

Input:

```
X = -3
print (abs(X))
```

Output:

```
3
```

Pada output menghasilkan nilai 3 yang merupakan nilai mutlak dari variabel X.

Syntax:

1. Round (Pembulatan) : round(var)
2. Absolutely (Nilai Mutlak) : abs(var)

Tidak hanya itu saja, masih banyak lagi fungsi matematika yang dapat digunakan dalam pemrograman Python ini. Namun, untuk dapat menggunakan fungsi matematika yang lainnya memerlukan module yang dibuat untuk dipakai di bidang matematika.

Pelajaran 13: If Statements

Di dalam dunia pemrograman seharusnya tidak asing dengan istilah “If statements”. If statements adalah sebuah metode yang berfungsi untuk menjalan statements sesuai dengan kondisi yang telah diatur oleh si programmer. Sebelum mengamati kode, simaklah kondisi-kondisi berikut ini…

If it's hot
  It's a hot day
  Drink plety water
otherwise if it's cold
  It's a cold day
  Wear warm clothes
otherwise
  It's a lovely day

Bagaimana jika kondisi-kondisi tersebut kita aplikasikan ke dalam pemrograman? Sekarang simaklah kode di bawah ini.

Input:

```
is_hot = True

if is_hot:
   print ("It's a hot day")
   print ("Drink plenty of water")
print ("It's a lovely day")
```

Output:

```
It's a hot day
Drink plenty drink
It's a lovely day
```

Diketahui sebuah variabel “is_hot” dengan nilai True. Kemudian, terdapat sebuah if statements yang menggunakan variabel “is_hot” sebagai kondisi yang dipilih. Di dalam fungsi tersebut terdapat beberapa statements yang akan berjalan ketika variabel “is_hot” bernilai True. Output akan menampilkan kalimat “It’s a hot day” dan juga “Drink plenty of water” karena pada kondisi “is_hot” bernilai True atau terpenuhi, sedangkan kalimat “It’s a lovely day” akan ditampilkan meskipun variabel “is_hot” bernilai False karena statement tersebut tidaklah tercakup dalam if statements “is_hot”.

Kemudian bagaimana jika kita menambahkan dua situasi yang berbeda tanpa menambahkan variabel lain yang bernilai boolean?

Simaklah kode berikut…

Input:

```
is_hot = False

if is_hot:
   print ("It's a hot day")
   print ("Drink plenty of water")
else:
   print ("It's a cold day")
   print ("Wear warm clothes")
print ("It's a lovely day")
```

Output:

```
It's a cold day
Wear warm clothes
It's a lovely day
```

Jika diketahui variabel “is_hot” bernilai False dan kita ingin menambahkan situasi yang lain, maka kita hanya perlu menggunakan fungsi “else”. Fungsi tersebut bertujuan untuk memberikan statements pada kondisi yang memiliki nilai selain pada fungsi if yang telah dibuat. Sehingga ketika variabel tersebut bernilai False, maka output atau terminal akan menampilkan semua statements pada fungsi else.

Sekarang kita akan mencoba untuk menggunakan 2 situasi dengan menggunakan 2 variabel boolean. Simaklah kode berikut…

Input:

```
is_hot = True
is_cold = False

if is_hot:
   print ("It's a hot day")
   print ("Drink plenty of water")
elif is_cold:
   print ("It's a cold day")
   print ("Wear warm clothes")
else:
   print ("It's a lovely day")
```

Output:

```
It's a hot day
Drink plenty of water
```

Terlihat pada output hanya menampilkan statements yang berada pada fungsi if. Itu dikarenakan kondisi yang diatur pada if bernilai True. Apabila variabel “is_cold” saja yang bernilai True, maka statements pada fungsi elif (else if) saja yang akan berjalan. Tetapi jika kedua variabel tersebut bernilai False, maka output akan menampilkan statements pada fungsi else.

Syntax:

1. If
2. Elif (Else If)
3. Else

Exercise

Diketahui sebuah kondisi yang berkonsepkan seperti berikut ini…

```
Price of a house is $1M
If buyer has a good credit
   They need to put down 10%
Otherwise
   They need to put down 20%
Print the down payment
```

Solution:

```
Price = 1000000
has_good_credit = True

if has_good_credit:
   down_payment = 0.1 * Price
else:
   down_payment = 0.2 * Price
print (down_payment)
```

Pelajaran 14: Logical Operator

Dalam pelajaran ini kita masih akan bersangkutan dengan pelajaran sebelumnya yaitu If Statements. Pada pelajaran ini kita akan memelajari If Statements yang sifatnya lebih kompleks dibandingkan dengan sebelumnya. Sekarang simaklah kondisi-kondisi berikut…

```
If he/she's healthy and has a good spiritual
  He/she can pass the test
If he/she's healthy but has a bad spiritual
  He/she cannot pass the test
If he/she's not healthy but has a good spiritual
  He/she still may pass the test
Else
  He cannot pass the test
```

Apabila kondisi tersebut diaplikasikan ke dalam pemrograman Python, maka akan menjadi seperti berikut ini…

Input:

```
is_healthy = True
has_good_spiritual = True

if is_healthy and has_good_spiritual:
   print ("He or she may pass the test")
elif is_healthy and not has_good_spiritual:
   print ("He or she cannot pass the test")
elif not is_healthy and has_good_spiritual:
   print ("He or she still may pass the test")
else:
   print ("He or she cannot pass the test")
```

Output:

```
He or she may pass the test
```

Pada contoh, kita mengatur nilai boolean keduanya True, maka statements yang akan berjalan adalah statements pada fungsi "if", karena "if is_healthy and has_good_spiritual:". Itu berarti statements tersebut akan berjalan ketika nilai boolean keduanya adalah True. Jika has_good_spiritual adalah False, sedangkan is_healthy tetap bernilai True, maka fungsi elif yang pertamalah yang akan berjalan. Jika has_good_spiritual adalah True, namun is_healthy adalah False, maka yang berjalan adalah syntax "elif" yang kedua. Jika kedua nilai boolean tersebut adalah False, maka fungsi yang berjalan adalah fungsi "else:" karena hanya tinggal satu kemungkinan kondisi saja yang tersisa.

Contoh di atas hanya menggunakan kondisi “and” dan juga “not”, tetapi masih ada satu kondisi lagi yang dapat kita gunakan yaitu kondisi “or”.

Simaklah konsep berikut…

Jika kita mendapat tugas membawa bahan ketrampilan, dan terdapat satu bahan yang memiliki bahan cadangan. Contohkan saja ketika kita mau membuat cetakan topeng, kita dapat menggunakan tanah liat atau juga dapat menggunakan plastisin.

Apabila diaplikasikan ke dalam Python, maka…

```
plastisin = True
tanah_liat = False

if plastisin or tanah_liat:
   print ("Anda sudah dapat membuat topeng")
else:
    print ("Anda belum dapat membuat topeng")
```

Output:

```
Anda sudah dapat membuat topeng
```

Syntax:

1. And (Ketika semua kondisi if atau elif bernilai True)
2. Not (Ketika salah satu atau kedua kondisinya bernilai False)
3. Or (Ketika salah satu kondisi dari dua kondisi yang dipilih bernilai True)

Pelajaran 15: Comparison Operator

Dalam pelajaran ini kita masih akan berkaitan dengan pelajaran If Statements. Jika sebelumnya kita sudah memahami if statements yang menggunakan nilai boolean, maka pada pelajaran ini kita akan memelajari if statements yang menggunakan nilai integer.

Perhatikan kondisi berikut…

```
Jika bilangan lebih besar daripada 0
  Maka bilangan tersebut positif
Jika bilangan lebih kecil daripada 0
  Maka bilangan tersebut negatif
Selain itu
  Maka bilang tersebut adalah 0
```

Input:

```
number = 3

if number > 0:
  print ("Merupakan bilangan positif")
elif number < 0:
  print ("Merupakan bilangan negatif")
else:
  print ("Merupakan bilangan bernilai nol")
```

Output:

```
Merupakan bilangan positif
```

Pada fungsi if menyatakan jika nilai variabel “number” bernilai lebih besar daripada nol, lalu pada fungsi elif menyatakan jika nilai variabel “number” bernilai lebih kecil daripada nol, serta pada fungsi else menyatakan jika nilai variabel “number” tidaklah memiliki kondisi yang sama dengan fungsi if dan elif.

Exercise

Jika temperatur di atas 30, maka cuacanya panas. Jika temperatur di bawah 20, cuacanya dingin. Selain itu, cuacanya sejuk. Temperatur silahkan diatur semaunya.

Solution:

```
Temperatur = 38

if temperatur > 30:
  print ("Cuacanya cukup panas")
elif temperatur < 20:
  print ("Cuacanya dingin")
else:
  print ("Cuacanya cukup sejuk")
```

Pelajaran 16: Weight Converter Program

Dalam pelajaran ini, kita akan belajar menggunakan if statements dengan menggunakan nilai strings. Pertama-tama amatilah output berikut ini…

```
Weight: 72
Lbs or Kg: Kg
You are 160 pounds
```

Ketika dicompile programnya, maka akan menjadi input seperti berikut ini…

```
weight = input ('Weight: ')
choice = input ("Lbs or Kg: ")

if choice == 'Lbs':
   Converted = int(weight) * 0.45
   print (f'You are {Converted} kilos')
else:
   Converted = int(weight) / 0.45
   print (f'You are {Converted} pounds')
```

Pada fungsi if kita menemukan cara baru dalam membuat kondisi, yaitu “if choice == ‘Lbs’”. Ini memiliki pengertian bahwa fungsi tersebut akan berjalan ketika variabel “choice” memiliki nilai string “Lbs”.

Pelajaran 17: While Loops

Dalam setiap bahasa pemrograman selalu ada sebuah fungsi yang bernama “While loops”. Untuk memahaminya silahkan simak kode berikut…

Input:

```
i = 3
while i <= 5:
   print (i)
   i = i + 1
print ("Done")
```

Output:

```
3
4
5
Done
```

Pada output hanya menampilkan angka 3 sampai 5 yang kemudian diakhiri dengan kata “Done” secara berbaris. Itu karena diketahui sebuah variabel i bernilai 3, dan kemudian terdapat sebuah fungsi while yang menyatakan selama variabel i bernilai kurang dari sama dengan 5. Di dalam while tersebut terdapat statements yang menampilkan nilai i kemudian nilai i ditambah satu setiap kali nilai i telah ditampilkan. Ketika nilai i sudah mencapai angka 5, maka fungsi tersebut akan berhenti. Itu dikarenakan pada fungsi while menyatakan i kurang dari sama dengan 5 sehingga pada output, pertambahan nilai i akan berhenti ketika sudah mencapai 5.

Berikut ini adalah contoh lainnya…

```
i = 4
while i < 10:
   print (i)
   i = i + 1
print ("Done")
```

Output:

```
4
5
6
7
8
9
Done
```

Pelajaran 18: For Loops

Pelajaran ini memiliki persamaan dengan pelajaran sebelumnya yaitu While Loops. Untuk dapat memahaminya, perhatikan kode berikut ini…

```
groceries = ['Carrot', 'Apple', 'Bread']

for items in groceries:
  print (items)
```

Output:

```
Carrot
Apple
Bread
```

Diketahui sebuah variabel yang bernilai arrays yang mengandung 3 nilai strings. Variabel tersebut kemudian ditampilkan dengan menggunakan fungsi for loops. Maka output yang ditampilkan adalah semua nilai strings pada variabel “groceries” secara berbaris.

Contoh lain…

```
names = 'Mosh'

for name in names:
  print (names)
```

Output:

```
M
o
s
h
```

Ketika diketahui sebuah variabel yang memiliki nilai strings tunggal ditampilkan dengan menggunakan fungsi for loops , maka output akan menampilkan kata yang sudah dipisahkan per hurufnya secara berbaris.

Menggunakan range pada for loops

Apa itu range? Metode range adalah sebuah metode yang sering digunakan dalam for loops untuk menampilkan integer dalam jangkauan atau besaran tertentu dari besaran yang telah kita atur di dalam program.

Input:

```
for numbers in range(0, 10, 2):
  print (numbers)
```

Output:

```
0
2
4
6
8
```

Exercise

Sekarang cobalah untuk menghitung total harga ketika diketahui harga-harga berikut…
price = [50, 10, 30]

Solution:

```
price = [50, 10, 30]
total = 0

for I in price:
   total += I
print (f'The total is {total}')
```

Pelajaran 19: Nested Loops

Apa yang dimaksud dengan nested loops? Nested loops adalah sebuah situasi atau kondisi di mana terdapat satu atau lebih for loops di dalam sebuah for loops. Sebagai contoh nya kita akan menampilkan output seperti berikut ini…

```
(0, 0)
(0, 1)
(0, 2)
(0, 3)
(1, 0)
(1, 1)
(1, 2)
(1, 3)
```

Lalu, bagaimana cara untuk membuat output seperti itu?

```
for x in range (2):
    for y in range(4):
        print (f'({x}, {y})')
```

Exercise

Untuk melatih pemahaman dalam pelajaran ini, cobalah untuk menampilkan output berikut dengan menggunakan konsep nested loops.

```
xxxxx
xx
xxxxx
xx
xx
```

Solution:

```
numbers = [5, 2, 5, 2, 2]

for x in numbers:
    output = ""
    for y in range(x):
        output += "x"
print (output)
```

Pelajaran 20: Lists

Dalam pelajaran ini, kita akan belajar lebih dalam mengenai lists atau yang kita kenal juga dengan sebutan arrays.

Sebelumnya kita sudah mengenal apa yang disebut dengan index. Pertama-tama kita ingat kembali bagaimana penggunaan index secara mendasar. Perhatikan kode berikut ini…

```
names = [‘Mosh’, ‘Hamedani’]
print (names[0])
print (names[1])
```

Output:

```
Mosh
Hamedani
```

Perlu kita ingat selalu bahwa index selalu dimulai dari angka nol, yang mana kata ‘Mosh’ menempati index ke-0 sedangkan kata ‘Hamedani’ menempati index ke-1. Sekarang coba kita mulai memahami secara lebih dalam lagi mengenai penggunaan index. Perhatikan kode berikut…

```
names = ['John', 'Bob', 'Mosh', 'Sarah', 'Mary']
print (names[2:])
print (names[:2])
print (names[1:3])
```

Output:

```
[‘Mosh’, ‘Sarah’, ‘Mary’]
[‘John’, ‘Bob’]
[‘Bob’, ‘Mosh’]
```

Ketika kita menampilkan nilai dari [2:], maka yang akan muncul adalah nilai dari index ke-2 sampai akhir index dalam arrays atau lists tersebut. Apabila kita menampilkan nilai dari [:2], maka yang akan muncul adalah nilai dari index ke-0 sampai index ke-1 sedangkan nilai dari index ke-2 sudah tidak termasuk. Apabila kita menampilkan nilai dari [1:3], maka yang akan muncul adalah nilai dari index ke-1 sampai index ke-2 sedangkan index ke-3 sudah tidak termasuk.

Kemudian, terdapat sebuah metode yang dapat kita gunakan dalam lists seperti mengganti salah satu nilai yang terdapat dalam sebuah lists. Perhatikan kode berikut…

```
names = ['John', 'Bob', 'Mosh', 'Sarah', 'Mary']
names[0] = 'Matthew'
print (names[0])
```

Output:

```
Matthew
```

Exercise

Buatlah sebuah lists yang berisikan beberapa integer yang memiliki nilai yang berbeda-beda, kemudian tampilkan nilai maksimal yang terdapat lists tersebut.

Solution:

```
numbers = [1, 4, 2, 5, 11, 10]
max = numbers[0]

for number in numbers:
    if number > max:
        max = number
print (max)
```

Output:

```
11
```

Pelajaran 21: 2D Lists

Dalam pelajaran ini, kita akan mencoba untuk membuat lists dalam pola matrix. Matrix sendiri sudah kita kenali sebagai salah satu bab di pelajaran Matematika, dan pada pelajaran ini kita akan memahami bagaimana membuat matrix tersebut. Coba simak kode berikut…

Input:

```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

print (matrix[0])
print (matrix[2])
```

Output:

```
[1, 2, 3]
[7, 8, 9]
```

Perhitungan index pada lists berbentuk matrix dihitung dengan index berdasarkan lists ke berapa. Contohnya jika kita menampilkan lists baris pertama, maka kita akan menggunakan index 0 yang nantinya menjadi "print (matrix[0])". Jika kita ingin menampilkan lists baris kedua, maka kita akan menggunakan index 1 yang nantinya menjadi "print (matrix[1])", dan seterusnya…

Kemudian, jika kita ingin menampilkan salah satu nilai dalam matriks tersebut seperti angka 2, kita dapat melakukannya seperti berikut…

Input:

```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

print (matrix[0][1])
```

Output:

```
2
```

Jika kita ingin menampilkan nilai dalam setiap matriks tersebut dalam satu deret, maka kita dapat melakukannya seperti berikut…

Input:

```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

for row in matrix:
    for item in row:
        print (item)
```

Output:

```
1
2
3
4
5
6
7
8
9
```

Pelajaran 22: List Methods

Dalam penggunaan list, kita dapat menggunakan berbagai macam metode yang sangat berguna. Berikut merupakan metode-metode yang sering digunakan dalam mengolah lists.

[Menambahkan data]

Input:

```
numbers = [1, 2, 5, 8, 2, 6]
numbers.append(4)
print (numbers)
```

Output:

```
[1, 2, 5, 8, 2, 6, 4]
```

Kita dapat menambahkan data pada list dengan menggunakan syntax "variable.append(object)". Data yang ingin kita tambahkan akan muncul pada urutan atau index terakhir dari list.

[Menambahkan data pada urutan/index tertentu]

Input:

```
numbers = [1, 2, 5, 8, 2, 6]
numbers.insert(0, 4)
print (numbers)
```

Output:

```
[4, 1, 2, 5, 8, 2, 6]
```

Kita dapat menambahkan data pada list dengan urutan tertentu dengan menggunakan syntax "variable.insert(index, object)". Data yang ingin kita tambahkan akan muncul pada index yang telah kita tentukan pada list.

[Menghapus suatu nilai pada list]

Input:

```
numbers = [1, 2, 5, 8, 2, 6]
numbers.remove(5)
print (numbers)
```

Output:

```
[1, 2, 8, 2, 6]
```

Kita dapat menghapus suatu data pada list dengan menggunakan "variable.remove(object)".

[Menghapus semua data dalam list]

Input:

```
numbers = [1, 2, 5, 8, 2, 6]
numbers.clear()
print (numbers)
```

Output:

```
[]
```

Kita dapat menghapus semua data pada list dengan menggunakan "variable.clear()".

[Menghapus data pada index terakhir dalam list]

Input:

```
numbers = [1, 2, 5, 8, 2, 6]
numbers.pop()
print (numbers)
```

Output:

```
[1, 2, 5, 8, 2]
```

Kita dapat menghapus data terakhir pada arrays dengan menggunakan "variable.pop()".

[Mencari index dari data yang dicari dalam list]

Input:

```
numbers = [1, 2, 5, 8, 2, 6]
print (numbers.index(5))
```

Output:

```
1
```

Kita dapat mencari index dari data yang kita cari pada arrays dengan menggunakan "variable.index(object)".

[Mengetahui ada atau tidaknya salah satu data dalam list]

Input:

```
numbers = [1, 2, 5, 8, 2, 6]
print (2 in numbers)
```

Output:

```
True
```

Kita dapat mengetahui apakah data yang kita cari terdapat dalam list dengan menggunakan "object in variabel". Jika True, maka data yang kita cari berada dalam arrays, jika False, maka data yang kita cari tidak terdapat dalam list.

[Menghitung jumlah data yang sama]

Input:

```
numbers = [1, 2, 5, 8, 2, 6, 2]
print (numbers.count(2))
```

Output:

```
3
```

Kita dapat menghitung jumlah data yang kita cari dalam sebuah arrays dengan menggunakan "variable.count(object)". Output akan menampilkan jumlah data tersebut dalam arrays yang kita tujukan.

[Mengurutkan nilai pada list]

Input:

```
numbers = [1, 2, 5, 8, 2, 6]
numbers.sort()
print (numbers)
```

Output:

```
[1, 2, 2, 5, 6, 8]
```

Kita dapat mengurutkan nilai dalam arrays dengan menggunakan "variable.sort()".

[Membalik urutan pada list]

Input:

```
numbers = [1, 2, 2, 5, 6, 8]
numbers.reverse()
print (numbers)
```

Output:

```
[8, 6, 5, 2, 2, 1]
```

Kita dapat membalik urutan pada arrays dengan menggunakan "variable.reverse()".

Pelajaran 23: Tuples

Tuples memiliki persamaan dengan lists. Namun, fungsi yang diperuntukkan pada tuples lebih sedikit dan minimalis atau sederhana. Coba simak kode berikut…

Input:

```
numbers = (1, 2, 3)
print (numbers.count(2))
print (numbers.index(1))
print (numbers[2])
```

Output:

```
1
0
3
```

Dalam tuples untuk tingkat pemula, kita hanya dapat menggunakan ketiga metode tersebut. "numbers.count(Object)" untuk menghitung jumlah data yang kita cari, "numbers.index(Object)" untuk mencari index dari data yang kita cari, dan "number[Index]" untuk menampilkan data yang kita cari berdasarkan nomor indexnya.

Bahkan jika kita ingin mengubah nilai salah satu data pada tuples, tuple tidak memiliki metode seperti itu, sehingga jika diexecute akan menampilkan eror.

Input:

```
numbers = (1, 2, 3)
numbers[0] = 2
print (numbers[0])
```

Output:

```
TypeError: 'tuple' object does not support item assignment
```

Pelajaran 24: Unpacking

Dalam pelajaran ini, kita masih akan berkaitan dengan tuples. Coba simak kode berikut ini terlebih dahulu…

Input:

```
coordinates = (1, 2, 3)
x = coordinates[0]
y = coordinates[1]
z = coordinates[2]
print (y)
```

Output:

```
2
```

Terlihat dalam kode tersebut kita membuat variabel x, y, dan z sebagai masing masing nilai dalam variabel "coordinates". Namun, kita tentu akan kelelahan atau mungkin menguras waktu untuk membuat variabel tersebut. Kita dapat menggunakan metode lain terhadap tuples.

Sekarang, coba amati kode berikut…

Input:

```
coordinates = (100, 200, 300)
x, y, z = coordinates
print (x)
```

Output:

```
100
```

Terlihat bahwa kode tersebut lebih sederhana dan simpel, dan bahkan untuk membuatnya tidak menguras waktu yang banyak. Metode tersebut hanya berlaku terhadap tuples saja.

Pelajaran 25: Dictionaries

Dalam Python terdapat suatu metode yang dinamakan "dictionaries". Coba kita perhatikan data berikut ini…

Name: John Smith
Email: john@gmail.com
Phone: 1234

Kita dapat membuat data tersebut dalam metode dictionaries. Coba perhatikan kode berikut…

Input:

```
customer = {
   "Name" : "John Smith",
   "Email" : "john@gmail.com",
   "Phone" : 1234
}
print (customer["Name"])
```

Output:

```
John Smith
```

Data-data yang sebelumnya kita perhatikan dapat kita kaji menjadi sebuah dictionary dengan metode seperti itu. Kita juga masih dapat melakukan beberapa metode terhadap dictionary tersebut.

Berikut ini adalah metode-metode yang dapat gunakan terhadap dictionaries…

[Mengubah data]
Input:

```
customer = {
   "Name" : "John Smith",
   "Email" : "john@gmail.com",
   "Phone" : 1234
}
customer["Name"] = "Erwin Smith"
print (customer["Name"])
```

Output:

```
Erwin Smith
```

Kita dapat mengubah nilai data dalam dictionaries berdasarkan jenis datanya.

[Menambahkan data]

Input:

```
customer = {
   "Name" : "John Smith",
   "Email" : "john@gmail.com",
   "Phone" : 1234
}
customer["Birthdate"] = "Jan, 1 1980"
print (customer["Birthdate"])
```

Output:

```
Jan, 1 1980
```

Kita dapat menambahkan data dengan membuat variabel yang sama dengan keterangan index yang ingin kita jadikan data baru.

[Menghindari eror ketika data tidak terdaftar]

Input:

```
customer = {
   "Name" : "John Smith",
   "Email" : "john@gmail.com",
   "Phone" : 1234
}
print (customer.get["Birthdate"])
```

Output:

```
None
```

Kita dapat menghindari eror ketika program menampilkan sebuah data yang tidak terdaftar atau tidak muncul dengan menggunakan syntax "variable.get[Object]".

Exercise

Untuk latihan ini, kita akan belajar untuk mengubah angka dalam bentuk integer ke kata seperti 1 menjadi "satu". Ketika program dijalankan, akan nampak seperti berikut…

```
Phone: 1234
One Two Three Four
```

Solution:

```
Phone = input('Phone: ')
Dict = {
    "1" : "One",
    "2" : "Two",
    "3" : "Three",
    "4" : "Four"
}

Output = ""
for i in Phone:
    output += Dict(i, "!") + ""
print (output)
```

Pelajaran 26: Functions

Functions adalah sebuah fungsi yang bertujuan untuk memanggil semua statements di dalam fungsi tersebut. Coba simak kode berikut ini…

Input:

```
def greet_user():
    print ('Hi there!')
    print ('Welcome aboard')

print ("Start")
greet_user()
print ("Finish")
```

Output:

```
Start
Hi there!
Welcome abord
Finish
```

Pada saat program ini dijalankan, output akan menampilkan print dan juga fungsi yang sudah kita buat pada awal awal. Dari syntax print kita tampilkan "Start" dan "Finish", dan di tengah tengah print itu kita memanggil fungsi yang kita buat yaitu fungsi "greet_user()" yang akan menampilkan sebuah statements di dalam fungsi tersebut ke dalam output.

Pelajaran 27: Parameters

Dalam pelajaran ini, kita akan memelajari cara memasukkan sebuah informasi ke dalam fungsi. Silahkan simak kode berikut…

Input:

```
def greet_user(name):
    print (f'Hi, {name}!')
    print ('Welcome aboard')

print ("Start")
greet_user("Joseph")
greet_user("Mary")
print ("Finish")
```

Output:

```
Start
Hi, Joseph!
Welcome aboard
Hi, Mary!
Welcome aboard
Finish
```

Pada output, kita menggunakan functions yang memiliki parameter. Pada function "greet_user" terdapat parameter yaitu "(name)". Dan jika kita mengisi parameter itu dengan nilai string "Joseph", maka fungsi akan memanggil statements sesuai dengan informasi yang diisi pada parameter yang daripada function. Apabila kita hanya memanggil fungsi tersebut tanpa mengisi informasi pada parameternya, maka ketika program itu dijalankan, output akan menampilkan error.

Kita juga dapat membuat parameter ganda pada fungsi yang kita buat dengan cara berikut...

Input:

```
def greet_user(first_name, last_name):
    print (f'Hi, {first_name} {last_name}!')
    print ('Welcome aboard')

print ("Start")
greet_user("Joseph", "Christian")
print ("Finish")
```

Output:

```
Start
Hi, Joseph Christian!
Welcome aboard
Finish
```

Untuk menambahkan parameter, kita dapat menggunakan pemisah dengan tanda koma (,) dan untuk memanggilnya kita juga harus mengisi semua info pada parameter dengan pemisah yaitu tanda koma(,).

Pelajaran 28: Keyword Arguments

Di dalam Python kita mempunyai sebuah metode yang namanya "Keyword Arguments". Apakah itu? Sekarang simaklah kode berikut…

Input:

```
def greet_user(first_name, last_name):
    print (f'Hi, {first_name} {last_name}!')
    print ('Welcome aboard')

print ("Start")
greet_user(last_name = "Christian", first_name = "Joseph")
print ("Finish")
```

Output:

```
Start
Hi, Joseph Christian!
Welcome aboard
Finish
```

Dalam memasukkan informasi ke dalam parameter, biasanya kita akan menggunakan posisional argumen atau memasukkan informasi berdasarkan urutan parameternya. Namun, pada input di atas kita menggunakan keyword argument atau memasukkan informasi dengan menuliskan ulang parameternya ke dalam fungsi.

Pelajaran 29: Return Statements

Di dalam pelajaran ini, kita akan belajar menggunakan syntax yang bernama return. Pertama-tama simaklah kode berikut ini…

Input:

```
def square(number):
        return number * number

print(square(5))
```

Output:

```
25
```

Penggunaan syntax return biasanya digunakan pada saat kita mendefinisikan sebuah fungsi. Untuk menampilkkan hasil dari return yang berada di dalam sebuah fungsi, kita harus menampilkannya menggunakan syntax print atau memasukkan hasilnya ke dalam sebuah variabel terlebih dahulu kemudian menampilkan nilai dari variabel tersebut. Itu dikarenakan jika kita hanya menuliskan "square(5)" saja, nantinya tidak akan ada suatu hasil yang tampil pada output. Jika dibandingkan dengan menuliskan "print (number * number)", hasil pada output akan sedikit berbeda.

Berikut adalah hasil output ketika mengganti "return number * number" dengan "print (number * number)".

Output:

```
25
None
```

Pada layar output akan ada dua baris hasil, baris pertama adalah hasil hitungan dari dalam fungsi, sedangkan baris kedua merupakan tulisan "None". Itu dikarenakan setiap fungsi akan mengembalikan kalimat "None" secara default. None itu seperti nothing atau null. Maka dari itu akan lebih baik menggunakan syntax return untuk menggantikan posisi syntax print ketika membuat sebuah fungsi.

Pelajaran 30: Creating a Reusable Function

Dalam pelajaran ini kita akan memelajari cara untuk membuat sebuah fungsi yang dapat digunakan kembali. Kita ambil contoh saja dari proyek Emoji Converter…

Input:

```
def emoji_converter(message):
    words = message.split()
    emoj = {
        ":)" : "     ",
        ":(" : "     "
    }

    output = ""
    for i in words:
     output += emoj.get(i, i) + " "
    return output

message = input('> ')
print (emoji_converter(message))
```

Output:

```
> Hello :)
Hello
```

Perlu diingat bahwa tidak diperbolehkan adanya input di dalam sebuah fungsi karena dapat menyebabkan program error pada saat dijalankan.

Pelajaran 31: Execeptions

Dalam pelajaran ini, kita akan memelajari bagaimana cara menangani berbagai macam eror yang akan seringkali muncul dalam program Python. Simak kode berikut ini…

Input:

```
age = int(input('Age: '))
print (age)
```

Output:

```
Age: 16
16
```

Program tersebut akan berjalan normal tanpa adanya eror ketika input dimasukkan dengan nilai integer, karena input tersebut sudah dikonversikan ke dalam nilai integer. Namun, apabila input dimasukkan dengan nilai string, output akan menampilkan eror. Lalu bagaimana cara supaya terminal tidak mengalami eror? Sekarang simaklah kode berikut ini…

Input:

```
try:
        age = int(input('Age: '))
        print (age)
except ValueError:
        print ("Invalid value")
```

Output:

```
Age: asd
Invalid value
```

Ketika input dimasukkan dengan nilai yang tidak sesuai dengan proses konversinya, maka output akan menampilkan statement pada syntax "except ValueError". Kata "ValueError" sendiri dapat kita temukan ketika output menampilkan jenis erornya.

Input:

```
try:
        age = int(input('Age: '))
   income = 20000
   risk = income / age
        print (age)
except ZeroDivisionError:
   print ("Age cannot be 0")
except ValueError:
        print ("Invalid value")
```

Output:

```
Age: 0
Age cannot be 0
```

Kode di atas juga sama dengan sebelumnya. Hanya saja kode tersebut ditambahkan exception yang dijalankan ketika input yang dimasukkan bernilai 0. Masih banyak exception yang dapat kita tambahkan dengan melihat eror yang terjadi pada outputnya.

Pelajaran 32: Comments

Di dalam setiap bahasa pemrograman selalu ada sebuah syntax yang dipakai untuk membuat komnetar pada kode yang kita buat. Dalam Python, kita dapat membuatnya dengan menggunakan syntax "#(objects)". Simaklah kode berikut…

Input:

```
#Data: Age
Age = input('Your age: ')

#Data: Address
Addr = input('Your address: ')
```

Output:

```
Your age: 16
Your address: Jl. Kebenaran
```

Komentar di dalam sebuah program tidak akan ditampilkan dalam output atau terminal. Komentar biasanya berfungsi hanya sebagai catatan bagi sang developer atau programmer agar tertata dengan rapi dan terstruktur.

Pelajaran 33: Classes

Classes dalam pemrograman sangat-sangatlah penting. Kita menggunakan classes untuk mendefinisikan suatu tipe baru. Contohnya dalam Python kita sudah memelajari numbers, strings, dan booleans. Tipe tipe data tersebut merupakan tipe data yang mendasar. Sedangkan kita juga sudah memelajari tipe data yang kompleks seperti lists, dan dictionaries. Kita akan belajar bagaimana cara mendefinisikan sebuah tipe baru yang dinamakan "Called point", dan tipe ini akan menggunakan metode baru untuk bekerja dengan points. Contohnya seperti berikut…

Input:

```
class Point:
        def move(self):
                print ("move")

        def draw(self):
                print ("draw")

point1 = Point()
point1.draw()
```

Output:

```
draw
```

Pertama-tama yang perlu diingat bahwa dalam penamaan classes, kita harus menggunakan penamaan Pascal (PascalNamingConvention) dan tidak diperbolehkan untuk menggunakan tanda garis bawah (_). Sedangkan saat membuat variabel kita biasanya menggunakan huruf kecil dan juga garis bawah. Kemudian, class secara sederhana dibuat untuk mengklasifikasikan sekumpulan fungsi. Kemudian, kita dapat mengolah classes tersebut melalui variabel baru yang kemudian untuk memanggil fungsi tertentu, kita dapat memanggilnya dengan format "variabel classes.fungsi()". Jangan lupa, dalam mendefinisikan sebuah fungsi dalam classes, tanda kurung yang terdapat pada setiap fungsi harus diisi dengan "self".

Tidak hanya itu saja, kita juga dapat menambahkan object pada suatu class dengan cara berikut…

Input:

```
class Point:
        def move(self):
                print ("move")

        def draw(self):
                print ("draw")
point1 = Point()
point1.x = 10
print (point1.x)
```

Output:

```
10
```

Dengan cara itu, kita dapat menambahkan suatu object baru dengan lebih cepat.

Pelajaran 34: Constructors

Apa yang dimaksud dengan Constructors? Constructors adalah sebuah metode spesial yang dapat kita lakukan terhadap classes yang fungsinya menambahkan sebuah objek pada class. Untuk dapat memahaminya, simaklah kode berikut ini…

Input:

```
class Point:
        def __init__(self, x, y):
                self.x = x
                self.y = y

        def move(self):
                print ("move")

        def draw(self):
                print ("draw")

point1 = Point(10, 20)
point1.x = 11
print (point1.x)
```

Output:

```
11
```

Pada class tersebut kita menambahkan sebuah fungsi baru, dan di dalam fungsi tersebut kita tambahkan objek baru, itulah yang disebut dengan constructors. Kita menambahkan object x dan y ke dalam fungsi "__init__" kemudian menyatakan "self.x = x" dan "self.y = y" yang akan diambil dari nilai x dan y pada "point1 = Point(10, 20)". Jika kita sudah menambahkan constructors, maka untuk dapat memanggil suatu fungsi dalam class tersebut, kita tetap harus mengisi informasi constructors walaupun fungsi yang kita panggil bukanlah si constructors seperti "point1.draw()" dan "point1.move()".

Exercise

Dalam latihan memahami pelajaran ini, cobalah untuk mendefinisikan sebuah pribadi (Class: Person), dan objects dari pribadi ini harus memiliki atribut nama dan juga metode berbicara. Di dalam metode berbicara ini, fungsi akan menampilkan tulisan "Hi, I'm (name)". Kemudian panggillah semua fungsi dalam class tersebut dengan kedua nama pribadi berikut: "John Smith" dan "Bob".

[Gambaran Soal]

Person

- name
- talk()

Solution:

```
class Person:
    def __init__(self, name):
        self.name = name

    def talk(self):
        print (f"Hi, I'm {self.name}")

john = Person("John Smith")
print (john.name)
john.talk()

bob = Person("Bob")
print (bob.name)
bob.talk()
```

Pelajaran 35: Inheritance

Di dalam pelajaran ini, kita akan belajar mengenai inheritance. Inheritance adalah sebuah mekanisme dalam menggunakan sebuah kode. Perhatikan kode berikut ini…

Input:

```
class Dog:
	def walk(self):
		print ("Walk")

class Cat:
	def walk(self):
		print ("Walk")
```

Diketahui sebuah class Dog yang mendefinisikan sebuah fungsi "Walk". Kemudian, jika beberapa saat setelah itu kita ingin membuat class Cat dengan fungsi yang sama, akan buruk jika kita membuat class Cat dengan fungsi yang sama seperti pada input di atas ini. Maka, bagaimana cara yang lebih baik? Perhatikan kode berikut ini…

Input:

```
class Mammal:
	def walk(self):
		print ("Walk")

class Dog(Mammal):
	pass

class Cat(Mammal):
	pass

dog1 = Dog()
dog1.walk()
```

Output:

```
Walk
```

Pertama-tama, kita membuat sebuah parent class yang tujuannya akan menjadi pusat yang memiliki fungsi yg sama, sehingga kita mendefinisikan fungsi yang sama tadi ke dalam parent class. Kemudian, kita buat lagi "class Dog(Mammal):" dengan memasukkan parameter dengan nama parent class. Di dalam class Dog tersebut berisi "pass" untuk menghindari eror supaya intrepeter atau penerjemah bahasa Python pada komputer tidak menganggap class tersebut adalah sebuah kesalahan. Selanjutnya seperti biasa kita buat sebuah variabel agar dapat memanggil fungsi tersebut.

Kita juga dapat menambahkan sebuah method ke dalam salah satu classnya tanpa perlu mengubah parent class yang sebelumnya. Contohnya khusus class Dog kita akan menambahkan method pada class tersebut, perhatikan kode berikut…

Input:

```
class Mammal:
	def walk(self):
		print ("Walk")

class Dog(Mammal):
	def bark(self):
		print ("Bark")

class Cat(Mammal):
	pass

dog1 = Dog()
dog1.bark()
```

Output:

```
Bark
```

Dengan begitu, sekarang kita mempunyai fungsi walk dan bark pada class Dog, sedangkan pada Cat hanya memiliki 1 fungsi yaitu walk. Jika ingin menambahkan fungsi khusus pada Cat, kita hanya perlu melakukan hal yang sama pada class Cat.

Pelajaran 36: Modules

Modules secara mendasar merupakan sebuah file yang berisikan kode Python yang telah mengalami defirisiensi atau perubahan fungsi ke fungsi khusus. Namun, kita dapat membuat module kita sendiri dengan suatu cara. Perhatikan kode berikut terlebih dahulu…

Input:

```
def lbs_to_kg(weight):
        return weight * 0.45

def kg_to_lbs(weight):
        return weight / 0.45
```

Seandainya kita ingin membuat kode di atas menjadi sebuah module. Pertama-tama sebagai contoh kita buat file dengan nama "converters.py" yang berisikan input di atas. Lalu, kita buat sebuah file baru yang akan berisikan module "converters.py", anggap saja file baru ini bermama "app.py". Di dalam file "app.py" memiliki input berikut ini…

Input:

```
import converters

print (converters.lbs_to_kg(70))
```

Output:

```
155.5555555554
```

Kode di atas merupakan cara bagaimana kita dapat menjadikan sebuah file Python kita menjadi sebuah module. Perlu diingat bahwa file yang menjadi module harus berada di dalam 1 folder yang sama dengan file utama.

Terdapat cara lain juga dalam memanggil sebuah module ke dalam sebuah program, perhatikan kode berikut ini…

Input:

```
from converters import lbs_to_kg
lbs_to_kg(70)
```

Output:

```
155.5555555554
```

Pelajaran 37: Packages

Seandainya kita ingin menggunakan beberapa module yang kita buat sendiri, namun sesuai peraturannya, file yang menjadi module haruslah berada di dalam 1 folder yang sama dengan program. Apabila kita ingin menggunakan semuanya, maka nantinya akan membuat folder yang kita pakai menjadi penuh dan tidak beraturan. Pada situasi tersebut kita akan menggunakan packages. Apa itu packages? Jika diumpamakan, packages adalah sebuah container yang mencakup beberapa modules sekaligus.

Contohnya kita ingin membuat sebuah folder yang menjadi package "ecommerce" yang isinya terdapat module "shipping.py" (Package atau folder diletakkan dalam sebuah folder). Kemudian kita buat sebuah program "app.py".

[shipping.py]

Input:

```
def calc_shipping():
    print ("Shipping calculation")
```

[app.py]

Input:

```
from ecommerce.shipping import calc_shipping
calc_shipping()
```

Output:

```
Shipping calculation
```

Input pada "app.py" merupakan cara mengimport dan menggunakan module pada package "shipping.py". Maka dapat disimpulkan bahwa format untuk mengimport sebuah module dari sebuah package ke dalam program adalah "import package.module". Namun terdapat sebuah cara lain dalam mengimport module seperti "from... import..." yang nantinya akan menjadi input seperti berikut ini…

[app.py]

Input:

```
from ecommerce.shipping import calc_shipping
calc_shipping()
```

Output:

```
Shipping calculation
```

Pelajaran 38: Generating Random Values

Di dalam Python terdapat banyak sekali module yang sudah terdaftar secara default tanpa menginstallnya terlebih dahulu. Jika ingin mengetahui apa saja module-module itu, kita dapat mencarinya di Google dengan keyword "Python Module Index". Pada situs tersebut akan ditampilkan semua daftar module yang terinstall secara default setelah kita menginstall Python pada PC kita. Pada pelajaran ini, kita akan belajar cara untuk menampilkan nilai-nilai yang digenerasi secara acak dengan menggunakan module random.

Input:

```
import random

for i in range(3):
	print(random.random())
```

Output:

```
0.562637782
0.207388274
0.926478273
```

Input di atas berfungsi untuk mengambil nilai secara acak antara 0 hingga 1.

Input:

```
import random

for i in range(3):
	print (random.randint(10, 20))
```

Output:

```
20
19
16
```

Pada input di atas berfungsi untuk mengambil nilai integer dari angka 10 hingga 20 secara acak.

Input:

```
import random

names = ["Mosh", "John", "Mary", "Matthew"]

print (random.choice(names))
```

Output:

```
Mary
```

Pada input di atas berfungsi untuk mengambil nilai secara acak dari sebuah array atau list.

Exercise

Ketika kita bermain sebuah permainan yang memerlukan dadu, kita selalu menggunakan dadu tersebut untuk mengacak nilai atau angka untuk menentukan alur dari sebuah game. Buatlah class Dice yang berisikan fungsi untuk mengacak nilai dengan dua dadu kemudian tampilkan nilai kedua dadu tersebut.

Solution:

```
import random

class Dice:
        def roll(self):
                A = random.randint(1, 6)
                B = random.randint(1, 6)
                return A, B

Single_Try = Dice()
print(Single_Try.roll())
```

Output:

```
(1,3)
```

Pelajaran 39: Working with Directories

Kita seharusnya sudah mengetahui apa yang dimaksud dengan directory. Directory adalah sebuah jalur yang menentukan isi file-file dan semua folder. Di dalam Python, kita dapat melakukan beberapa fungsi terhadap directory pada PC kita dengan menggunakan module "Pathlib". Namun, kode berikut memiliki directory seperti berikut ini…

```
file
  > app.py
  > test.py
```

[app.py]

Input:

```
from pathlib import Path

path = Path("test.py")
print (path.exists())
```

Output:

```
True
```

Fungsi "Path()* berfungsi untuk memberitahukan bahwa file atau folder yang ditulis dalam fungsi Path ada atau tidak.

[app.py]

Input:

```
from pathlib import Path

path = Path("Hello")
print (path.mkdir())
```

Output:

```
None
```

Pada output, terminal akan menampilkan tulisan "None". Namun, jika kita lihat pada directory yang sama, akan muncul folder dengan nama "Hello". Mkdir merupakan fungsi untuk membuat directory baru (Singkatan dari mkdir adalah make directory). Pada terminal, kita dapat membuat sebuah directory tanpa program Python hanya dengan mengetikkan perintah "mkdir (nama folder)" yang setelah perintah itu di-enter, nantinya akan muncul folder dengan nama yang telah kita masukkan.

Berikut adalah gambaran dari directory setelah menjalankan input di atas…

```
file
  > Hello
  > app.py
  > test.py
```

Jika fungsi sebelumnya adalah membuat directory, fungsi selanjutnya adalah menghapus directory. Perhatikan kode berikut…

[app.py]

Input:

```
from pathlib import Path

path = Path("Hello")
print (path.rmdir())
```

Output:

```
None
```

Berikut adalah gambaran directory setelah menjalankan input di atas…

```
files
  > app.py
  > test.py
```

Singkatan dari rmdir adalah remove directory. Pada termimal, kita juga dapat menghapus sebuah directory dengan sebuah perintah "rm (nama file/folder)".

Kita juga dapat mencari sebuah file dengan ekstensi tertentu seperti "file.py", "file.xls", "file.docx", dll. Perhatikan isi directory berikut terlebih dahulu…

```
files
  > app.py
  > test.py
  > rekening.xls
  > nugas.xls
  > revisi.docx
  > laporan.docx
```

Kemudian, amatilah kode berikut…

[app.py]

Input:

```
from pathlib import Path

path = Path()
for i in path.glob('*.xls'):
        print (i)
```

Output:

```
rekening.xls
nugas.xls
```

Output akan menampilkan semua file yang memiliki ekstensi "file.xls". Kita juga dapat mencari file-file dengan ekstensi "file.py" dengan fungsi "path.glob('*.py')". Namun, kita dapat menampilkan semua file dalam folder tersebut hanya dengan menggunakan fungsi "path.glob('*')".

Pelajaran 40: Pypi and Pip

Di dalam Python kita pastinya sudah mengenal apa itu module. Namun, tahukah kamu bahwa tidak semua module Python adalah buatan asli dari lembaga Python itu sendiri. Ada banyak sekali module yang dikembangkan oleh para developer-developer yang bukan dari lembaga Python. Tetapi, tentunya tidak semua module yang mereka buat terluput dari bug, ada yang masih dalam perbaikan atau bahkan ada yang masih tersimpan bug di dalamnya.

Untuk mencari module-module Python, kita dapat mencarinya di situs "https://pypi.org" kemudian mulailah untuk mencari module Python yang ingin kalian cari dan install. Lalu, bagaimanakah caranya untuk menginstall module-module yang bukan merupakan module official dari Python?

Contohnya saja kita ingin menginstall module "openpyxl" (Fungsi dari module tersebut adalah untuk membaca file excel dengan menggunakan program Python).

1. Bukalah situs "https://pypi.org"
2. Carilah module dengan memasukkan keyword "openpyxl"
3. Pada bagian di bawah judul module dan juga versi seperti tulisan "Latest version" akan nampak command yang berfungsi untuk menginstall module tersebut.
4. Kemudian, masukkan command yang nampak tadi ke dalam terminal atau command prompt.
5. Setelah itu, module akan tersimpan sebagai package dan barulah kita dapat menggunakan module itu dengan menggunakan syntax "import ..."

Berikut adalah directory letak semua module Python yang telah diinstal pada PC:

C:\Users\Administrator\AppData\Local\Programs\Python\Python38\Lib

Source

Youtube:

Python Tutorial – Python for Beginners [Full Course] by Programming with Mosh, https://youtu.be/_uQrJ0TkZlc