

HMGI Long Course

Basic Python for Seismology 1

- Tujuan: Memperkenalkan dasar pemrograman Python
- Keluaran: Peserta dapat menjalankan Python dan memahami penggunaannya
- Sesi: Materi
- Waktu/Tempat: Sabtu, 18 September 2021/ Zoom Meeting

1 Variabel dan Operasi

1.1 Variabel

Variabel adalah suatu nilai yang kita simpan di memori dan bisa kita panggil saat akan digunakan. Contoh kasus variabel adalah seperti saat kita menyimpan memori di otak kita bahwa nilai π (variabel) adalah 3,14. Saat mengingat atau memanggil π di memori otak kita maka akan muncul angka 3,14. Cara kerja komputer juga demikian sehingga Python sebagai salah satu bahasa pemrograman juga bertindak seperti itu. Dalam implementasinya cara 'menyimpan ingatan' ini dilakukan dengan cara yang mudah:

In [47]:

```
pi=3.14
```

Pada contoh di atas kita menyimpan di ingatan komputer bahwa nilai `pi` adalah 3.14, dengan begitu kita dapat memanggilnya dengan cara:

In [48]:

```
pi
```

Out[48]:

```
3.14
```

Variabel dalam Python sendiri dibedakan menjadi beberapa jenis seperti:

- Integer : bilangan bulat
- Float : *floating point*, bilangan dengan desimal
- String : teks
- Complex numbers : bilangan kompleks
- Boolean : benar (True) atau salah (False)

Python akan menginterpretasikan jenis variabel dari **cara kita menulis** variabel tersebut:

In [49]:



```
angka=1 #integer
Angka=1.0 #float
ANGKA="1" #string
ANGKA2='1' #string
bil_kompleks=5-1j #complex
bools=True #boolean
```

Sama seperti ingatan manusia, jika suatu menyimpan ke memori dengan kata kunci yang sama maka nilai sebelumnya akan dilupakan, contoh:

In [50]:



```
angka=2
```

In [51]:



```
angka
```

Out[51]:

2

Saat kita panggil `angka` maka 2 yang akan muncul, bagaimana kalau nilai `angka` kita ganti:

In [52]:



```
angka=3
```

In [53]:



```
angka
```

Out[53]:

3

1.2 Operasi

Jika kita sudah bisa menyimpan suatu nilai ke dalam ingatan komputer dengan mendefinisikan suatu `variable`, selanjutnya kita akan mengoperasikan variabel-variabel tersebut. Beberapa operasi standar yang dapat kita lakukan adalah:

Simbol Operasi	Kegunaan
+	penjumlahan
-	pengurangan
*	perkalian
/	pembagian
%	mengembalikan sisa pembagian (<i>modulo</i>).
**	pangkat

Simbol Operasi	Kegunaan
<code>+=</code>	menambah dan mengganti
<code>-=</code>	mengurangkan dan mengganti
<code>==</code>	mengetes kesamaan
<code>!=</code>	mengetes ketidaksamaan

Contoh implementasi operasi-operasi di atas:

In [54]:

```
angka1=3
angka2=5
angka3=angka1+angka2
print(angka3)
```

8

In [55]:

```
angka4=angka1-angka3
print(angka4)
```

-5

In [56]:

```
angka5=angka1**angka2
print(angka5)
```

243

Jenis variabel yang dioperasikan akan berpengaruh terhadap hasil operasinya, misalkan operasi perkalian antar `integer` :

In [57]:

```
10*10
```

Out[57]:

100

Hasil operasi di atas berupa `integer` karena variabel yang dikalikan sama-sama `integer` . Jika salah satu atau kedua data merupakan `float` maka hasil dari operasi akan berjenis `float` :

In [58]:

```
10*10.0
```

Out[58]:

100.0

Bagaimana dengan operasi string ?

In [59]:

```
"1"+"3"
```

Out[59]:

```
'13'
```

In [60]:

```
"13"-"3"
```

```
-----
-----
TypeError                                 Traceback (most recent call
  last)
/tmp/ipykernel_4387/2527148311.py in <module>
----> 1 "13"-"3"
```

TypeError: unsupported operand type(s) for -: 'str' and 'str'

Operasi penjumlahan pada string akan 'menumpuk' string tersebut sedangkan operasi pengurangan tidak bisa dijalankan. Operasi perkalian antar string juga akan menghasilkan error:

In []:

```
"1"*"3"
```

string bisa dikalikan dengan integer, hasil perkalian ini berupa penumpukan string sebanyak angka yang dikalikan:

In []:

```
"1"*3
```

operasi dengan float akan menghasilkan error:

In []:

```
"1"*3.
```

2 Struktur Data

Data dalam geofisika biasanya merupakan kumpulan nilai atau angka yang erat kaitannya dengan pengukuran. Kita akan menggunakan studi kasus pengukuran suhu suatu ruangan setiap 4 jam dalam satu hari, dengan kata lain kita akan memiliki 6 angka hasil pengukuran. Bagaimana mewakili 6 angka tersebut dalam Python?

Python menyediakan beberapa jenis struktur data yang bisa digunakan untuk menyimpan data kita yang masing-masing memiliki fungsinya sendiri. Beberapa jenis struktur data yang umum di dalam Python adalah:

- List
- Tuple
- Dictionary
- Sets

2.1 List

List ditulis dalam Python dengan `[]` yang bisa diisi berbagai macam variable yang sama ataupun berbeda dan masing-masing variabel tersebut dipisahkan dengan tanda koma (,):

In [61]:

```
suhu_24h=[34.0, 35.0, 34.3, 32.0, 31.1, 29.0]
print(suhu_24h)
```

```
[34.0, 35.0, 34.3, 32.0, 31.1, 29.0]
```

dalam kasus pengukuran suhu ruangan tersebut kita mendapatkan 6 angka pengukuran yang kemudian dituangkan dalam `list` di Python seperti di atas. Setiap element `list` berasosiasi dengan `index` yang dimulai dengan angka 0 pada element pertama dan seterusnya. Contoh apabila kita ingin mengambil pengukuran ketiga maka `index` nya adalah 2:

In [62]:

```
pengukuran_3=suhu_24h[2]
print(pengukuran_3)
```

```
34.3
```

Kita juga dapat memanggil dari belakang dengan `index` -1 untuk element terakhir, -2 untuk element kedua dari akhir, dan seterusnya, seperti contoh:

In [63]:

```
pengukuran_terakhir=suhu_24h[-1]
print(pengukuran_terakhir)
```

```
29.0
```

Dengan `index` kita juga bisa mengambil lebih dari 1 elemen dengan tambahan simbol `:` :

In [64]:

```
pengukuran_1dan2=suhu_24h[0:2]
print(pengukuran_1dan2)
```

```
[34.0, 35.0]
```

Element dalam `list` dapat diganti/*mutable* seperti pada contoh dibawah ini:

In [65]:



```
suhu_24h[0]=37.0
print(suhu_24h)
```

```
[37.0, 35.0, 34.3, 32.0, 31.1, 29.0]
```

list memiliki banyak sekali method yang digunakan untuk mengolah list tersebut, opsi-opsi atau method tersebut dapat kita lihat dengan mengetikkan help :

In [66]:



```
help(suhu_24h)

The sort is in-place (i.e. the list itself is modified) and
stable (i.e. the
order of two equal elements is maintained).

If a key function is given, apply it once to each list item
and sort them,
ascending or descending, according to their function values.

The reverse flag can be set to sort in descending order.

-----
Class methods defined here:
__class_getitem__(...) from builtins.type
    See PEP 585
-----
Static methods defined here:
```

Berdasarkan petunjuk di atas kita akan mencoba 1 method yaitu pop yang petunjuknya adalah:

```
pop(self, index=-1, /)
|    Remove and return item at index (default last).
```

berdasarkan petunjuk tersebut maka pop akan mengambil elemen list sekaligus menghapusnya dari list kita:

In [67]:



```
print("list sebelum pop", suhu_24h)
elemen_terhapus=suhu_24h.pop()
print("elemen yang dihapus",elemen_terhapus)
print("list setelah pop", suhu_24h)
```

```
list sebelum pop [37.0, 35.0, 34.3, 32.0, 31.1, 29.0]
elemen yang dihapus 29.0
list setelah dihapus [37.0, 35.0, 34.3, 32.0, 31.1]
```

2.2 Tuple

Tuple sangat mirip dengan List hanya saja Tuple merupakan *sequence* yang elemennya tidak bisa diganti (*immutable*). Penulisan Tuple dalam Python menggunakan tanda `()` :

In [68]:

```
suhu_24h_lock=(37.0, 35.0, 34.3, 32.0, 31.1, 29.0)
```

Pemanggilan element sama dengan list yaitu menggunakan index :

In [69]:

```
pengukuran_1=suhu_24h_lock[0]
print(pengukuran_1)
```

37.0

Jika kita coba mengedit element dalam tuple maka akan muncul error:

In [70]:

```
suhu_24h_lock[0]=29.0
```

```
-----
-----
TypeError                                Traceback (most recent call
last)
/tmp/ipykernel_4387/2770928713.py in <module>
----> 1 suhu_24h_lock[0]=29.0
```

TypeError: 'tuple' object does not support item assignment

2.3 Sets

Sets adalah data dalam bentuk koleksi unik yang tak berurutan dan sifatnya *immutable* seperti tuple . Secara sederhana sets bisa kita artikan sebagai himpunan dalam bahasa Indonesia. Kegunaannya juga mirip seperti himpunan:

In [71]:

```
Sayur={"seledri", "tomat", "bayam"}
Sayur
```

Out[71]:

```
{'bayam', 'seledri', 'tomat'}
```

elemen sets harus unik dan tidak ada yang duplikat:

In [72]:



```
Sayur={"seledri", "tomat", "bayam", "tomat"}  
Sayur
```

Out[72]:

```
{'bayam', 'seledri', 'tomat'}
```

Walaupun bersifat *immutable* tapi `sets` dapat ditambah elemennya:

In [73]:



```
Sayur.add("kol")  
Sayur
```

Out[73]:

```
{'bayam', 'kol', 'seledri', 'tomat'}
```

`sets` juga dapat dioperasikan dengan `sets` yang lain:

In [74]:



```
Buah={"tomat", "apel", "jeruk"}  
Buah
```

Out[74]:

```
{'apel', 'jeruk', 'tomat'}
```

dalam contoh pertama kita akan melihat elemen dari dua `sets` yang sama dengan method `intersection` :

In [75]:



```
Sayur.intersection(Buah)
```

Out[75]:

```
{'tomat'}
```

Dari contoh di atas kita seperti mencari irisan dari dua buah himpunan yaitu Sayur dan Buah, karena tomat masuk keduanya maka tomat adalah irisannya. Contoh selanjutnya adalah dengan metode `difference` dan `union` kira-kira apa yang terjadi?

In [76]:



```
Sayur.difference(Buah)
```

Out[76]:

```
{'bayam', 'kol', 'seledri'}
```


In [77]:



```
Sayur.union(Buah)
```

Out[77]:

```
{'apel', 'bayam', 'jeruk', 'kol', 'seledri', 'tomat'}
```

2.4 Dictionary

Dictionary menyimpan data di dalam keys bersifat alfanumerik, keys bisa dianalogikan sebagai sebuah judul kolom yang masing-masing kolom tersebut terdapat data. Penulisan dictionary mirip dengan sets yaitu menggunakan {}. Dalam contoh di bawah ini masih menggunakan studi kasus pengukuran suhu, dengan dictionary kita dapat mendefinisikan data lain yaitu waktu pengukuran dalam menit:

In [79]:



```
pengukuran_suhu={"waktu_menit": [10, 20, 30], "suhu": [29, 30, 27]}  
print(pengukuran_suhu)
```

```
{'waktu_menit': [10, 20, 30], 'suhu': [29, 30, 27]}
```

Jika dalam list dan dalam tuple kita bisa mengambil elemen dengan menggunakan indeks, dalam dictionary kita menggunakan keys dari data yang akan kita ambil. Apabila kita ingin mengambil data waktu pengukuran maka kita menggunakan keys waktu_menit:

In [82]:



```
waktu_pengukuran=pengukuran_suhu['waktu_menit']  
waktu_pengukuran
```

Out[82]:

```
[10, 20, 30]
```

Dengan memanggil waktu_pengukuran kita mendapatkan isi dari key tersebut yang berupa list, seperti yang kita definisikan di atas.

3 Conditional (Pembuatan Keputusan)

Pada pembuatan suatu aplikasi pasti akan sampai dalam sebuah titik dimana kita harus mengambil keputusan dan harus memilih satu dari beberapa pilihan tergantung kondisi. Dalam implementasinya kita akan mengenal if, elif, dan else:

In [83]:



```
waktu_menit=70

if waktu_menit > 60:
    print("pengukuran lebih dari 1 jam")
```

pengukuran lebih dari 1 jam

Pada contoh di atas kita membuat variabel `waktu_menit`, apabila nilai dalam variabel tersebut melebihi 60 menit (`if waktu_menit > 60`) maka akan dicetak tulisan: `pengukuran lebih dari 1 jam`. Jika waktu kurang dari 60 maka tidak akan terjadi apa-apa:

In [84]:



```
waktu_menit=10

if waktu_menit > 60:
    print("pengukuran lebih dari 1 jam")
```

Kita bisa menambah percabangan lagi dengan `else` apabila `waktu_menit` tidak lebih besar dari 60:

In [85]:



```
waktu_menit=10

if waktu_menit > 60:
    print("pengukuran lebih dari 1 jam")
else:
    print("pengukuran kurang dari 1 jam")
```

pengukuran kurang dari 1 jam

Untuk percabangan yang lebih dari 2 maka kita perlu menggunakan `elif` atau `else-if`:

In [86]:



```
waktu_menit=60

if waktu_menit > 60:
    print("pengukuran lebih dari 1 jam")
elif waktu_menit == 60:
    print("pengukuran tepat 1 jam")
else:
    print("pengukuran kurang dari 1 jam")
```

pengukuran tepat 1 jam

Conditional ini dapat bertingkat (`nested`), contoh apabila pengukuran selesai setelah 120 menit atau 2 jam:

In [87]:

```
waktu_menit=130
timer_off=120

if waktu_menit > 60:
    if waktu_menit < timer_off:
        print("pengukuran lebih dari 1 jam")
    else:
        print("pengukuran selesai karena sudah 2 jam")
elif waktu_menit == 60:
    print("pengukuran tepat 1 jam")
else:
    print("pengukuran kurang dari 1 jam")
```

pengukuran selesai karena sudah 2 jam

4 Loop (perulangan)

Loop digunakan untuk melakukan perulangan proses pada setiap elemen pada data *sequence* seperti *list* dan *tuple*. Ada dua macam perulangan di Python yaitu *for* dan *while*. Misalkan kita akan mencetak hasil pengukuran suhu dengan *for*:

In [88]:

```
for elemen in suhu_24h:
    print("Suhu sekarang adalah:", elemen)
```

Suhu sekarang adalah: 37.0
Suhu sekarang adalah: 35.0
Suhu sekarang adalah: 34.3
Suhu sekarang adalah: 32.0
Suhu sekarang adalah: 31.1

Setiap elemen akan dari *suhu_24h* akan dicetak secara bergantian dan berurutan. Untuk mempermudah intuisi kita bisa berikan delay 1 detik untuk setiap perintah *print* agar terlihat bahwa fungsi *print* ini dilakukan satu per satu:

In [136]:

```
import time
for elemen in suhu_24h:
    print("Suhu sekarang adalah:", elemen)
    time.sleep(1)
```

Suhu sekarang adalah: 37.0
Suhu sekarang adalah: 35.0
Suhu sekarang adalah: 34.3
Suhu sekarang adalah: 32.0
Suhu sekarang adalah: 31.1

In [138]:



```
for elemen in suhu_24h:
    suhu_kelvin=elemen+273
    print("Suhu sekarang adalah:", suhu_kelvin)
    time.sleep(1)
```

```
Suhu sekarang adalah: 310.0
Suhu sekarang adalah: 308.0
Suhu sekarang adalah: 307.3
Suhu sekarang adalah: 305.0
Suhu sekarang adalah: 304.1
```

Sama seperti pada `conditional`, `loop` juga dapat kita buat bertingkat (*nested*), contoh apabila ingin mencetak waktu dengan perulangan jam dan menit:

In [92]:



```
menit=[0,10,20,30,40,50]
jam=[1,2,3]

for j in jam:
    for m in menit:
        print("Pukul: ",j,"lebih: ",m," menit")
```

```
Pukul: 1 lebih: 0 menit
Pukul: 1 lebih: 10 menit
Pukul: 1 lebih: 20 menit
Pukul: 1 lebih: 30 menit
Pukul: 1 lebih: 40 menit
Pukul: 1 lebih: 50 menit
Pukul: 2 lebih: 0 menit
Pukul: 2 lebih: 10 menit
Pukul: 2 lebih: 20 menit
Pukul: 2 lebih: 30 menit
Pukul: 2 lebih: 40 menit
Pukul: 2 lebih: 50 menit
Pukul: 3 lebih: 0 menit
Pukul: 3 lebih: 10 menit
Pukul: 3 lebih: 20 menit
Pukul: 3 lebih: 30 menit
Pukul: 3 lebih: 40 menit
Pukul: 3 lebih: 50 menit
```

5 Fungsi dan Modul

Sebelum melangkah ke penjelasan fungsi kita akan coba mengonversi pengukuran suhu kita dari celcius ke dalam derajat Fahrenheit dengan rumus:

$$T_F = \left(\frac{9}{5} * T_c\right) + 32$$

dengan pengetahuan kita tentang operasi kita bisa menerapkannya dalam Python:

In [93]:

```
suhu_pertama_c=suhu_24h[0]

suhu_pertama_F=(9/5*suhu_pertama_c)+32

print("Suhu pertama dalam C: ", suhu_pertama_c)
print("Suhu pertama dalam F: ", suhu_pertama_F)
```

```
Suhu pertama dalam C:  37.0
Suhu pertama dalam F:  98.60000000000001
```

Kita juga bisa melakukan hal yang sama pada pengukuran yang kedua:

In [94]:

```
suhu_kedua_c=suhu_24h[0]

suhu_kedua_F=(9/5*suhu_kedua_c)+32

print("Suhu kedua dalam C: ", suhu_kedua_c)
print("Suhu kedua dalam F: ", suhu_kedua_F)
```

```
Suhu kedua dalam C:  37.0
Suhu kedua dalam F:  98.60000000000001
```

5.1 Fungsi

Dalam konversi suhu seperti kasus di atas kita perlu menuliskan rumus lengkap dari konversi setiap akan melakukannya. Hal seperti demikian tidak efektif, rawan terjadi error, dan susah saat akan *maintenance*. Suatu operasi yang akan dilakukan berkali-kali dapat kita definisikan sebagai sebuah fungsi agar bisa kita panggil:

In [95]:

```
def celcius_to_fahrenheit(suhu_celcius):
    suhu_fahrenheit=(9/5*suhu_celcius)+32
    return suhu_fahrenheit
```

Dalam kode di atas dapat kita lihat bahwa sebuah fungsi diawali dengan `def` diikuti nama fungsi, nama fungsi kita adalah `celcius_to_fahrenheit`. Setelah nama fungsi diikuti parameter dalam tanda kurung (`suhu_celcius`), baris selanjutnya adalah *body* dari fungsi tersebut. Di bagian akhir kita mengembalikan variabel yang menjadi hasil dari fungsi kita.

Setelah mendefinisikan fungsi `celcius_to_fahrenheit` seperti di atas kita bisa memanggilnya:

In [96]:

```
suhu_celcius=10
suhu_fahrenheit=celcius_to_fahrenheit(suhu_celcius)
print(suhu_fahrenheit)
```

```
50.0
```

In [97]:

```
celcius_to_fahrenheit(15)
```

Out[97]:

59.0

Dalam penulisan sebuah fungsi disarankan kita memberikan petunjuk tentang bagaimana cara menggunakan fungsi tersebut atau sering disebut sebagai `docstring`. Petunjuk ini biasanya ditulis di bawah baris pendefinisian fungsi (yang diawali `def`):

In [98]:

```
def celcius_to_fahrenheit(suhu_celcius):
    """Fungsi konversi suhu Celcius ke Fahrenheit
    Argument:
        suhu_celcius (float/int): Suhu dalam celcius yang akan dikonversi
    Return:
        suhu_fahrenheit (float): Hasil konversi berupa suhu dalam Fahrenheit
    """
    suhu_fahrenheit=(9/5*suhu_celcius)+32
    return suhu_fahrenheit
```

Petunjuk tersebut dapat kita akses dengan `?` atau dengan fungsi `help`:

In [99]:

```
celcius_to_fahrenheit?
```

In [100]:


```
help(celcius_to_fahrenheit)
```

Help on function celcius_to_fahrenheit in module `__main__`:

```
celcius_to_fahrenheit(suhu_celcius)
    Fungsi konversi suhu Celcius ke Fahrenheit
    Argument:
        suhu_celcius (float/int): Suhu dalam celcius yang akan dikonve
rsi
    Return:
        suhu_fahrenheit (float): Hasil konversi berupa suhu dalam Fahr
enheit
```

5.2 Modul

Modul secara sederhana adalah kumpulan dari fungsi-fungsi yang bisa kita panggil. Fungsi `celcius_to_fahrenheit` di atas bisa kita masukkan ke dalam modul `temperature_conversion`. Cara membuat modul adalah dengan membuat file dengan ekstensi `.py` dengan diberi nama seperti nama modul kita. Jika modul akan kita namai `temperature_conversion` maka nama file kita adalah `temperature_conversion.py` dan akan kita simpan pada folder `utilities`, buat file `__init__.py` juga dalam folder tersebut dengan isi kosong. Selanjutnya masukkan fungsi kita ke dalam modul tersebut:

 **jupyter** temperature_conversion.py ✓ a few seconds ago Logout

File Edit View Language Python

```
1 def celcius_to_fahrenheit(suhu_celcius):
2     """Fungsi konversi suhu Celcius ke Fahrenheit
3     Argument:
4         suhu_celcius (float/int): Suhu dalam celcius yang akan dikonversi
5     Return:
6         suhu_fahrenheit (float): Hasil konversi berupa suhu dalam Fahrenheit
7     """
8     suhu_fahrenheit=(9/5*suhu_celcius)+32
9     return suhu_fahrenheit
```

Kita bisa panggil modul tersebut dengan perintah `import` :

In [140]:

```
import utilities.temperature_conversion as temperature_conversion
```

Selanjutnya kita bisa mencoba memanggil fungsi `celcius_to_fahrenheit` :

In [141]:

```
temperature_conversion.celcius_to_fahrenheit(10)
```

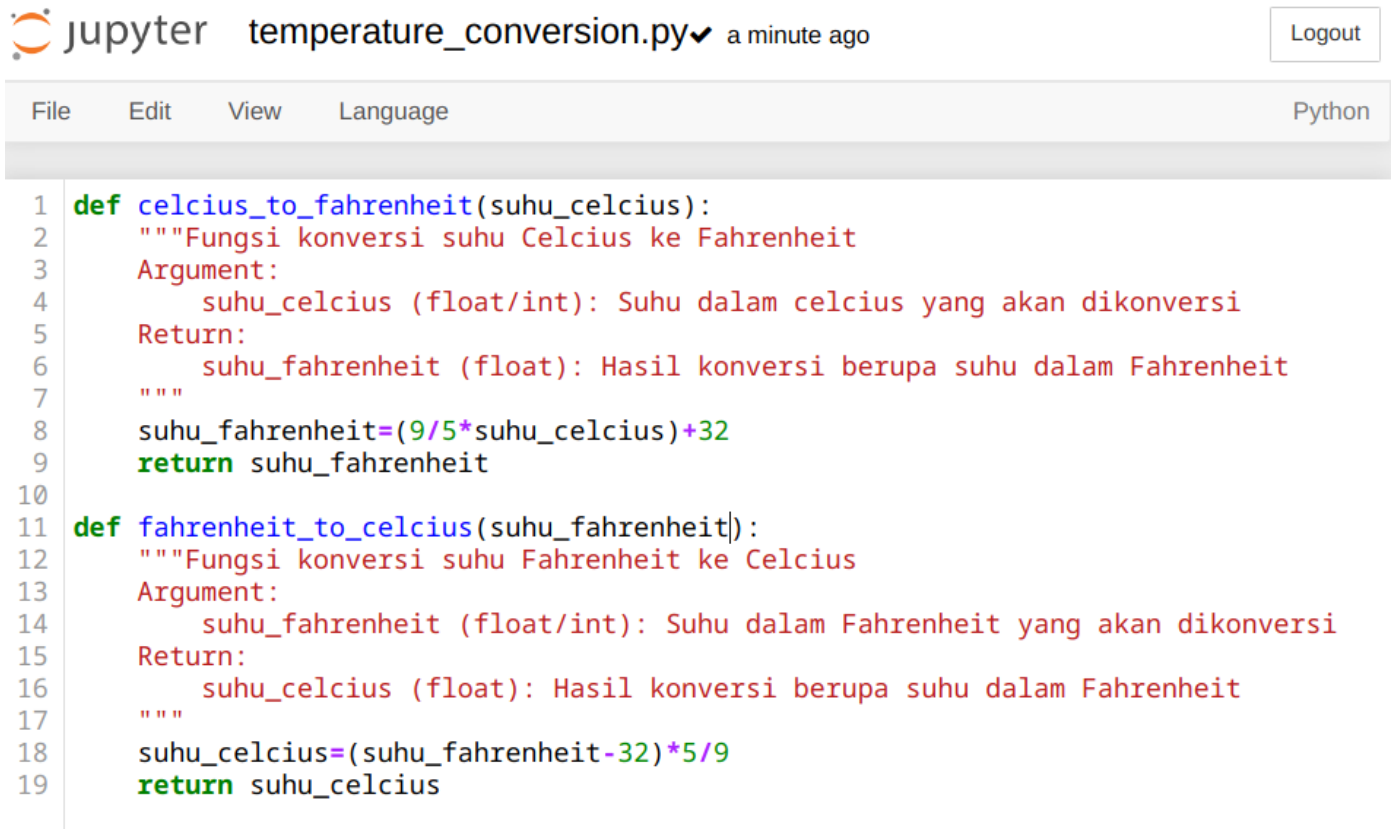
Out[141]:

50.0

Modul `temperature_conversion` bisa kita periksa lagi misalkan kita tambah fungsi untuk mengonversi sebaliknya yaitu dari suhu derajat Celcius ke derajat Fahrenheit:

In [142]:

```
def fahrenheit_to_celcius(suhu_fahrenheit):
    """Fungsi konversi suhu Fahrenheit ke Celcius
    Argument:
        suhu_fahrenheit (float/int): Suhu dalam Fahrenheit yang akan dikonversi
    Return:
        suhu_celcius (float): Hasil konversi berupa suhu dalam Fahrenheit
    """
    suhu_celcius=(suhu_fahrenheit-32)*5/9
    return suhu_celcius
```



```

1 def celcius_to_fahrenheit(suhu_celcius):
2     """Fungsi konversi suhu Celcius ke Fahrenheit
3     Argument:
4         suhu_celcius (float/int): Suhu dalam celcius yang akan dikonversi
5     Return:
6         suhu_fahrenheit (float): Hasil konversi berupa suhu dalam Fahrenheit
7     """
8     suhu_fahrenheit=(9/5*suhu_celcius)+32
9     return suhu_fahrenheit
10
11 def fahrenheit_to_celcius(suhu_fahrenheit):
12     """Fungsi konversi suhu Fahrenheit ke Celcius
13     Argument:
14         suhu_fahrenheit (float/int): Suhu dalam Fahrenheit yang akan dikonversi
15     Return:
16         suhu_celcius (float): Hasil konversi berupa suhu dalam Fahrenheit
17     """
18     suhu_celcius=(suhu_fahrenheit-32)*5/9
19     return suhu_celcius

```

Modul sebelumnya sudah kita impor dengan `import temperature_conversion`, jika isi modul tersebut berubah maka kita harus reload dengan memanfaatkan fungsi `reload` dari modul `importlib`:

In [144]:

```

from importlib import reload

reload(temperature_conversion)

```

Out[144]:

```

<module 'utilities.temperature_conversion' from '/media/anangsahroni/1
ocaldiskD/job/hmgi_course/material/utilities/temperature_conversion.p
y'>

```

Selanjutnya kita bisa mencoba fungsi yang baru saja ditambahkan:

In [122]:

```

temperature_conversion.fahrenheit_to_celcius(50)

```

Out[122]:

```

10.0

```

Modul Built-in (bawaan) Python

Python memiliki beberapa modul bawaan seperti `math`, `cmath`, `os`, atau secara lebih lengkap: [Built-in modules \(https://docs.python.org/3/py-modindex.html\)](https://docs.python.org/3/py-modindex.html). Seperti modul buatan kita sendiri tadi, kita bisa panggil modul bawaan Python dengan perintah `import`:

In [125]:

```
import math  
help(math)
```

When the iterable is empty, return the start value. This function is intended specifically for use with numeric values and may reject non-numeric types.

```
radians(x, /)  
Convert angle x from degrees to radians.
```

```
remainder(x, y, /)  
Difference between x and the closest integer multiple of y.
```

Return $x - n*y$ where $n*y$ is the closest integer multiple of y . In the case where x is exactly halfway between two multiples of y , the nearest even value of n is used. The result is always exact.

In [126]:

```
math.pi
```

Out[126]:

3.141592653589793

Modul dapat diberikan nama alias, misalkan kita ganti `math` dengan `matematika` :

In [127]:

```
import math as matematika  
matematika.pi
```

Out[127]:

3.141592653589793

In [128]:

```
matematika.log10(1000)
```

Out[128]:

3.0

Jika sebelumnya kita langsung mengimpor modul, pada contoh dibawah kita hanya mengimpor fungsi-fungsi yang kita butuhkan saja:

In [130]:



```
from math import radians, sin, cos, tan

sudut=60
sudut_radian=radians(sudut)
sin_sudut=sin(sudut_radian)
print(sin_sudut)
```

0.8660254037844386

Modul External

Modul eksternal adalah modul yang dikembangkan diluar pengembang asli Python. Modul eksternal memiliki kegunaan yang spesifik dan berisi kumpulan fungsi-fungsi yang mendukung kegunaan tersebut. Modul-modul eksternal umum yang akan sering kita gunakan seperti `numpy` untuk data-data numerik, `scipy` untuk pengolahan data saintifik, `matplotlib` untuk plotting data, dan `pandas` untuk pengelolaan data. Dalam bidang yang lebih spesifik seperti pada seismologi kita mengenal modul `obspy`.

Modul eksternal dapat diinstall menggunakan bantuan *package manager*, jika kita menggunakan Anaconda/Miniconda kita dapat menginstall modul dengan perintah standar di dalam Anaconda Prompt:

```
conda install [nama modul]
```

seperti:

```
conda install numpy
```

Beberapa modul perlu parameter lain seperti lokasi kanal pengunduhan (`-c`), seperti saat teman-teman menginstall Jupyter Notebook ini kemarin melalui kanal `conda-forge` :

```
conda install -c conda-forge notebook
```

Contoh penggunaan modul Numpy dan Matplotlib untuk mengolah data array dan mengeplot:

In [131]:



```
import numpy as np

sudut=np.arange(0,360,10)
sudut
```

Out[131]:

```
array([ 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120,
        130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250,
        260, 270, 280, 290, 300, 310, 320, 330, 340, 350])
```

In [133]:



```
sudut_radian=np.radians(sudut)
sudut_radian
```

Out[133]:

```
array([0.          , 0.17453293, 0.34906585, 0.52359878, 0.6981317 ,
        0.87266463, 1.04719755, 1.22173048, 1.3962634 , 1.57079633,
        1.74532925, 1.91986218, 2.0943951 , 2.26892803, 2.44346095,
        2.61799388, 2.7925268 , 2.96705973, 3.14159265, 3.31612558,
        3.4906585 , 3.66519143, 3.83972435, 4.01425728, 4.1887902 ,
        4.36332313, 4.53785606, 4.71238898, 4.88692191, 5.06145483,
        5.23598776, 5.41052068, 5.58505361, 5.75958653, 5.93411946,
        6.10865238])
```

In [134]:



```
sinus=np.sin(sudut_radian)
sinus
```

Out[134]:

```
array([ 0.00000000e+00,  1.73648178e-01,  3.42020143e-01,  5.00000000e
-01,
        6.42787610e-01,  7.66044443e-01,  8.66025404e-01,  9.39692621e
-01,
        9.84807753e-01,  1.00000000e+00,  9.84807753e-01,  9.39692621e
-01,
        8.66025404e-01,  7.66044443e-01,  6.42787610e-01,  5.00000000e
-01,
        3.42020143e-01,  1.73648178e-01,  1.22464680e-16, -1.73648178e
-01,
       -3.42020143e-01, -5.00000000e-01, -6.42787610e-01, -7.66044443e
-01,
       -8.66025404e-01, -9.39692621e-01, -9.84807753e-01, -1.00000000e
+00,
       -9.84807753e-01, -9.39692621e-01, -8.66025404e-01, -7.66044443e
-01,
       -6.42787610e-01, -5.00000000e-01, -3.42020143e-01, -1.73648178e
-01])
```

In [135]:

```
import matplotlib.pyplot as plt  
plt.plot(sudut, sinus)
```

Out[135]:

```
[<matplotlib.lines.Line2D at 0x7fcc62532220>]
```

