MI3103
Praktikum Antar Muka Komputer
Pengantar Pemrograman Python

Fadjar Fathurrahman

2019

1 Tujuan

Tujuan dari praktikum ini adalah sebagai berikut.

- dapat membuat dan mengeksekusi program Python sederhana
- mampu mengenal dan menggunakan tipe data dasar pada Python
- mampu mengenal dan menggunakan kontrol program sederhana pada Python (percabangan dan perulangan)
- mampu membuat fungsi sederhana (subrutin) pada Python

Mampu menjawab pertanyaan-pertanyaan berikut.

2 Perangkat lunak yang diperlukan

Berikut adalah beberapa perangkat lunak yang diperlukan.

- Linux OS
- Distribusi Anaconda untuk Python 3 `https://www.anaconda.com/distribution/`
- Internet browser seperti
- Editor teks seperti:
  - gedit (biasanya sudah tersedia pada banyak distro Linux)
  - VSCode (`https://code.visualstudio.com/`)
  - Atom (`https://atom.io/`)

  atau Python IDE seperti Spyder (sudah terinstall pada Anaconda).

3 Cek versi Python

Ada dua versi utama (*major*) Python yang sekarang digunakan, yaitu versi 2 dan versi 3. Versi Python yang akan kita gunakan adalah versi 3. Untuk mengecek versi Python, Anda dapat mengetikkan perintah berikut.

```
python --version
```

Pada Ubuntu, biasanya terpasang Python versi 2 dan versi 3. Untuk Python 2 biasanya digunakan executable `python` dan untuk Python 3 digunakan executable `python3`.

4 Mengaktifkan Python dari Anaconda

Python biasanya sudah terpasang secara default pada banyak distribusi Linux, termasuk pada Ubuntu di LabKom TF. Anda dapat mengecek apakah dengan menggunakan perintah berikut.

```
which python3
```

Jika keluaran yang Anda peroleh adalah

```
/usr/bin/python3
```

berarti Python yang Anda gunakan adalah Python dari sistem (Ubuntu).

Distribusi Python Anaconda telah terpasang di LabKom TF pada `/opt/anaconda3`. Anda harus mengaktifkannya dengan menggunakan perintah berikut.

```
source /opt/anaconda3/bin/activate
```

5 Mengenal interpreter Python (konsol Python)

Terdapat banyak pilihan untuk berinteraksi dengan interpreter atau konsol Python:

- Konsol default Python, dapat dijalankan dengan mengetikkan `python` pada terminal.
- IPython, merupakan interpreter Python dengan berbagai fitur tambahan seperti *tab-completion* dan *syntax highlighting*. IPython
- Jupyter qtconsole
- Jupyter notebook

Tampilan awal konsol Python default:

```
Python 3.6.6 |Anaconda custom (64-bit)| (default, Jun 28 2018, 17:14:51)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Tampilan awal IPython

```
Python 3.6.6 |Anaconda custom (64-bit)| (default, Jun 28 2018, 17:14:51)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.4.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

Dalam praktikum ini Anda sebaiknya menggunakan IPython karena lebih mudah digunakan. Jika IPython belum terpasang, Anda dapat menggunakan konsol Python biasa.

Anda dapat menggunakan interpreter ini untuk menjalankan kode Python. Kita akan mulai dengan contoh sederhana yaitu operasi aritmatika sederhana.

```
>>> 2 + 3
5
>>> 2 / 3
0.6666666666666666
```

Anda juga dapat mendefinisikan variabel dengan menggunakan operator penugasan. Operator penugasan (*assigment*) pada Python adalah tanda `=`.

```
>>> a = 3
>>> b = 4.1
>>> a*b
12.299999999999999
>>> a - b
-1.0999999999999996
>>> a/b
0.7317073170731708
```

Fungsi `print` dapat digunakan menuliskan nilai suatu ekspresi dan variabe

```
>>> x = 1.2
>>> print(x)
1.2
>>> print(x + 2)
3.2
```

Pada interpreter, kita dapat langsung mengetikkan variabel untuk mengetahui nilai dari variabel atau suatu ekspresi (tanpa menggunakan fungsi `print`):

```
>>> x
1.2
>>> x + 2
3.2
```

Python juga mendukung bilangan kompleks. Contoh:

```
>>> a = 1.5 + 0.5j # tidak ada spasi sebelum j
>>> a.real
1.5
>>> a.imag
0.5
```

Contoh variabel boolean

```
>>> 3 > 4
False
>>> test = (3 > 4)
>>> test
False
```

Fungsi `type` dapat digunakan untuk mengetahui tipe data dari suatu nilai dari variabel atau eskpresi.

```
>>> type(1)
<type 'int'>
>>> type(1.)
<type 'float'>
>>> type(1. + 0j )
<type 'complex'>
>>> a = 3
>>> type(a)
<type 'int'>
```

Berbeda dengan C/C++, suatu variabel tidak perlu dideklarasikan atau tipenya sebelum digunakan. Python akan mendeduksi tipe variabel berdasarkan nilai yang diberikan. Berbeda dengan C, variabel pada Python dapat merujuk pada suatu nilai dengan tipe data yang berbeda.

```
>>> z = 1.2
>>> type(z)
<class 'float'>
>>> z = 2
>>> type(z)
<class 'int'>
>>> z = 1 + 2j
>>> type(z)
<class 'complex'>
>>> z = 2 > 3
>>> type(z)
<class 'bool'>
```

Tugas

Carilah operator matematika yang didukung oleh Python. Beberapa sumber berikut dapat Anda gunakan:

- `https://www.tutorialspoint.com/python/python_basic_operators.htm`
- `https://en.wikibooks.org/wiki/Python_Programming/Basic_Math`

6 Kode sumber Python

Selain menggunakan konsol Python, kita dapat menuliskan program yang kita buat dalam suatu file teks dengan ekstensi `.py`. Anda dapat membuat file ini dengan editor yang mendukung Python, beberapa diantaranya adalah:

- Gedit yang biasanya sudah terpasang di Ubuntu dan banyak distribusi Linux
- Atom yang merupakan editor teks multifungsi. Editor ini dapat dijalankan baik di Windows, OSX, maupun Linux.
- Visual Studio Code, mirip dengan Atom. Meskipun dikembangkan oleh Microsoft, editor ini juga dapat dijalankan di Linux dan OSX.
- Spyder yang merupakan IDE (*integrated development environment*) untuk Python.

Misalkan file program Python Anda bernama `hitung.py`, maka Anda dapat mejalankan program ini dari terminal dengan perintah berikut.

```
python hitung.py
```

Komentar pada Python dimulai dengan tanda `#`.

7 Menggunakan modul Python

Banyak fungsionalitas dari Python yang dikumpulkan dalam sebuat modul. Pada praktikum ini dan selanjutnya, kita akan banyak menggunakan modul-modul tersebut. Modul Python ini ada yang sifatnya standard atau bawaan dan yang lainnya harus dipasang terlebih dahulu sebagai suatu paket

atau pustaka. Modul pertama yang akan kita temui adalah modul `math`. Modul ini adalah modul standard bawaan Python.

Modul ini dapat diakses dengan menggunakan perintah `import math`.

```
>>> import math
>>> dir(math)
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
↪  'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh',
↪  'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod',
↪  'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf',
↪  'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow',
↪  'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
```

Perintah `dir(math)` digunakan untuk mengetahui simbol (fungsi dan/atau konstanta) apa saja yang didefinisikan dalam module `math`. Anda dapat menggunakan perintah `help(math.log2)` misalnya, untuk membaca dokumentasi mengenai fungsi ini.

Contoh penggunaan modul `math`

```
>>> math.sin(math.pi)
1.2246467991473532e-16
>>> math.sin(0.5*math.pi)
1.0
>>> math.sin(0.3*math.pi)
0.8090169943749475
>>> math.sin(math.pi/3)
0.8660254037844386
```

Sintaks alternatif berikut ini juga sering digunakan.

```
from math import *
```

Dengan sintaks di atas, semua fungsi pada modul `math` akan diekspor ke konteks global Python. Jika kita hanya ingin mengimpor beberapa simbol saja dapat digunakan sintaks seperti berikut ini.

```
from math import pi, factorial
```

Tugas

Buatlah suatu program sederhana untuk menghitung akar-akar dari persamaan kuadrat dengan input dari pengguna. Anda dapat menggunakan fungsi input untuk membaca input dari pengguna. Gunakan fungsi `cmath.sqrt` yang dapat menghitung akar kuadrat dari bilangan negatif sebagai bilangan kompleks.

8 Tipe data numerik

Kita telah menggunakan tipe-tipe dasar numerik sebelumnya. Tipe-tipe tersebut adalah:

- integer
- float
- complex
- boolean

Beberapa tipe data dapat dikonversi antara satu dengan yang lainnya.

```
>>> a = 2
>>> type(a)
<class 'int'>
>>> b = float(a)
>>> b
2.0
>>> type(b)
<class 'float'>
>>> c = 2.3
>>> d = int(2.3)
>>> d
2
```

9 Tipe data kontainer

Python memiliki beberapa tipe data kontainer yang merupakan kumpulan dari beberapa tipe objek. Tipe data kontainer pada Python diantaranya adalah:

- list
- string
- dictionary
- tup=el

9.1 Lists

List merupakan kumpulan objek yang memiliki urutan. Objek-objek ini dapat memiliki tipe yang berbeda. Contoh

```
>>> l = [1, 2, 3, 4, 5]
>>> type(l)
<type 'list'>
```

Kita dapat mengakses suatu objek dengan menggunakan indeksnya.

```
>>> l[2]
3
```

Indeks negatif dapat digunakan untuk mengakses dari akhir.

```
>>> l[-1]
5
>>> l[-2]
4
```

Mirip dengan bahasa C/C++, indeks pada Python dimulai dari 0.

Operasi slicing dapat digunakan untuk mendapatkan sublist dari suatu list (mirip dengan Matlab/Octave/Scilab).

```
>>> l
[1, 2, 3, 4, 5]
>>> l[2:4]
[3, 4]
```

Perhatikan bawah `l[start:stop]` akan memberikan elemen dengan indeks `i` di mana `start<= i < stop`. Artinya, `l[start:stop]` akan memiliki `(stop-start)` buah element.

Sintaks untuk slicing: `l[start:stop:stride]`. Semua parameternya bersifat opsional.

```
>>> l[3:]
[4, 5]
>>> l[:3]
[1, 2, 3]
>>> l[::2]
[1, 3, 5]
```

List bersifat *mutable*, artinya nilainya dapat diubah.

```
>>> l[0] = 28
>>> l
[28, 2, 3, 4, 5]
>>> l[2:4] = [3, 8]
>>> l
[28, 2, 3, 8, 5]
```

Elemen-element pada suatu list dapat memiliki tipe yang berbeda.

```
>>> l = [3, 2, 'hello']
>>> l
[3, 2, 'hello']
>>> l[1], l[2]
(2, 'hello')
```

Menambahkan dan mengurangi elemen pada suatu list.

```
>>> l = [1, 2, 3, 4, 5]
>>> l.append(6)
>>> l
[1, 2, 3, 4, 5, 6]
>>> l.pop()
6
>>> l
[1, 2, 3, 4, 5]
>>> l.extend([6, 7])
>>> l
[1, 2, 3, 4, 5, 6, 7]
>>> l = l[:-2]
>>> l
[1, 2, 3, 4, 5]
```

Membalikkan urutan pada list.

```
>>> r = l[::-1]
>>> r
[5, 4, 3, 2, 1]
```

Menyambung (`list`) dan mengurangi element pada list.

```
>>> r + l
[5, 4, 3, 2, 1, 1, 2, 3, 4, 5]
>>> 2 * r
[5, 4, 3, 2, 1, 5, 4, 3, 2, 1]
```

Mengurutkan elemen-elemen pada list.

```
>>> r.sort()
>>> r
[1, 2, 3, 4, 5]
```

Untuk perhitungan numerik dan komputasi performa tinggi, paket Numpy lebih sering digunakan. Informasi mengenai Numpy dapat ditemukan di `www.numpy.org`.

9.2 Strings

Tipe data string pada Python ditandai dengan menggunakan tanda kutip tunggal (') atau ganda (").
Tiga tanda kutip (baik tunggal maupun ganda) dapat digunakan untuk string yang panjang.

```
s = 'Halo, nama saya Jojo'
s = "Halo, nama saya Joko"
s = '''Halo,
       nama saya Jono'''
s = """Halo,
   Nama saya Joyo"""
```

Karakter baris baru adalah `\n` dan karakter tab adalah `\t`.

String dapat dianggap sebagai list dari karakter (huruf), artinya string juga memiliki operasi yang sama dengan list.

Mengakses elemen lewat indeks.

```
>>> a = "hello"
>>> a[0]
'h'
>>> a[1]
'e'
>>> a[-1]
'o'
```

Slicing.

```
>>> a = "hello, world!"
>>> a[3:6]
'lo,'
>>> a[2:10:2]
'lo o'
>>> a[::3]
'hl r!'
```

Berbeda dengan list, string bersifat *immutable*. Kita tidak bisa mengubah karakter pada string, akan tetapi kita dapat membuat string baru dari string awal. String memiliki banyak metode yang dapat digunakan keperluan manipulasi teks.

Tugas

Baca dokumentasi Python mengenai string: `https://docs.python.org/3/library/string.html`

9.3 Dictionary/Kamus

Tipe data kamus adalah tabel yang yang memetakan kunci ke suatu nilai.

```
>>> tel = {'emmanuelle': 5752, 'sebastian': 5578}
>>> tel['francis'] = 5915
>>> tel
{'sebastian': 5578, 'francis': 5915, 'emmanuelle': 5752}
>>> tel['sebastian']
5578
>>> tel.keys()
['sebastian', 'francis', 'emmanuelle']
>>> tel.values()
[5578, 5915, 5752]
>>> 'francis' in tel
True
```

Suatu kamus dapat memiliki kunci dan nilai yang memiliki tipe berbeda

```
>>> d = {'a':1, 'b':2, 3:'hello'}
>>> d
{'a': 1, 3: 'hello', 'b': 2}
```

9.4 Tupel

Tupel dapat dianggap sebagai list yang tidak dapat dimodifikasi `immutable`. Elemen-elemen tupel ditulis dalam tanda kurung dan dipisahkan dengan tanda koma.

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> u = (0, 2)
```

10 Kontrol alur program

10.1 Percabangan: `if/elif/else`

```
>>> a = 2
>>> if a < 0:
...     print("a bernilai negatif")
... else:
...     print("a bernilai positif")
...
```

Indentasi sangat penting pada Python.

Jika Anda kesulitan atau tidak nyaman menuliskan di konsol, Anda dapat menuliskannya di file dengan ekstensi `*.py`. Anda tidak perlu menuliskan `>>>` atau `....`

Contoh lain

```
>>> a = 10
>>> if a == 1:
...     print(1)
... elif a == 2:
...     print(2)
... else:
...     print('A lot')
...
A lot
```

10.2 Perulangan: `for/range`

Perulangan dengan indeks

```
>>> for i in range(4):
...     print(i)
...
0
1
2
3
```

Iterasi terhadap nilai:

```
>>> for word in ('cool', 'powerful', 'readable'):
...     print('Python is %s' % word)
...
Python is cool
Python is powerful
Python is readable
```

10.3 Perulangan: `while/break/continue`

```
>>> a = 3;
>>> while a < 10:
...     a = a + 1
...
>>> a
10
```

`break` dapat digunakan untuk keluar dari loop `for` atau `while`

```
>>> a = 10;
>>> while a > 0:
...     a = a - 2
...     if a < 3: break
...
>>> a
2
```

`continue` dapat digunakan untuk meneruskan ke iterasi selanjutnya (biasanya untuk melangkaui suatu iterasi)

```
>>> a = [1, 0, 2, 4]
>>> for element in a:
...     if element == 0:
...         continue
...     print(1.0/element)
...
1.0
0.5
0.25
```

10.4 Iterasi pada string, list, dan dictionary

Kita dapat melakukan iterasi terhadap string, list dan dictionary

```
>>> vowels = 'aeiouy'
>>> for i in 'powerful':
...     if i in vowels:
...         print(i),
...
...
o e u
```

Contoh

```
>>> message = "Hello how are you?"
>>> message.split() # mengembalikan list
['Hello', 'how', 'are', 'you?']
>>> for word in message.split():
...     print word
...
Hello
how
are
you?
```

Contoh:

```
>>> words = ('cool', 'powerful', 'readable')
>>> for i in range(0, len(words)):
....     print(i, words[i])
....
....
0 cool
1 powerful
2 readable
```

Fungsi `enumerate` juga dapat digunakan

```
>>> words = ('cool', 'powerful', 'readable')
>>> for index, item in enumerate(words):
...     print index, item
...
0 cool
1 powerful
2 readable
```

Loop pada dictionary dengan menggunakan `items()`

```
>>> d = {"a" : 1, "b" : 1.2, "c" : 1j}
>>> for key, val in d.items():
        print('Key: %s has value: %s' % (key, val)
Key: a has value: 1
Key: c has value: 1j
Key: b has value: 1.2
```

11 Fungsi (subsprogram)

Sintaks dasar:

```
>>> def test():
...     print("in test function")
```

```
...
>>> test()
in test function
```

Ingat bahwa indentasi signifikan pada Python

Fungsi dapat mengembalikan nilai

```
>>> def disk_area(radius):
...     return 3.14 * radius * radius
...
>>> disk_area(1.5)
>>> 7.0649999999999995
```

Secara default, fungsi mengembalikan `None`.

11.1 Parameter fungsi

Parameter fungsi pada Python dapat berupa parameter wajib yang sesuai dengan posisinya pada definisi fungsi.

```
>>> def double_it(x):
...     return x*2
...
>>> double_it(3)
6
>>> double_it(3.2)
6.4
>>> double_it()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: double_it() missing 1 required positional argument: 'x'
>>> double_it(3,1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: double_it() takes 1 positional argument but 2 were given
```

Argumen fungsi pada Python juga dapat bersifat opsional, dengan menggunakan kata kunci atau nama argumen. Dengan menggunakan argument kata kunci, kita dapat memberikan nilai default pada fungsi.

```
def double_it(x=2):
    return x * 2
double_it()
double_it(3)
```

11.2 Modifikasi parameter fungsi

Sebagian besar bahasa pemrograman seperti C dan Java membedakan antara *pass-by-value* dan *pass-by-reference*. Pada fungsi dengan *pass-by-value*, argumen pada fungsi tidak dapat diubah karena hanya nilainya saja yang diberikan, sedangkan untuk *pass-by-reference* argumen pada fungsi dapat diubah nilainya. Pada Python, tidak ada aturan khusus mengenai apakah suatu argumen pada suatu fungsi bersifat *pass-by-value* atau *pass-by-reference*.

Jika nilai yang diberikan pada suatu fungsi bersifat *immutable*, fungsi tidak akan memodifikasi variabel tersebut. Jika nilai tersebut bersifat *mutable*, maka fungsi dapat mengubah nilai tersebut.

Tipe data sepertin `int`, `float`, `bool`, `str`, `tuple` bersifat *immutable*. Sedangkan `list`, `set`, `dict` bersifat *mutable*.

```
def try_to_modify(x, y, z):
    x = 23
    y.append(42)
    z = [99] # ubah z
    print(x)
    print(y)
    print(z)

a = 77    # variabel immutable
b = [99]  # variabel mutable
c = [28]
try_to_modify(a, b, c)
print(a)
print(b)
print(c)
```

Fungsi memiliki lingkup variabel sendiri. Dalam kode sebelumnya, variabel `x` hanya ada di dalam fungsi `try_to_modify`.

11.3 Variabel global

Variable yang dideklarasikan di luar sebuh fungsi dapat direferensi (digunakan) di dalam sebuah fungsi.

```
x = 5
def addx(y):
    return x + y
z = addx(10)
print(z)
```

Akan tetapi variable global tersebut tidak dapat dimodifikasi di dalam sebuah fungsi, kecuali jika dideklarasikan sebagai `global` di dalam fungsi tersebut.

Bandingkan antara keluaran kode berikut:

```
x = 5
def setx(y):
    x = y
    print("x is %d" % x)
setx(10)
print(x)
```

dengan kode berikut ini.

```
x = 5
def setx(y):
    global x
    x = y
    print("x is %d" % x)
setx(10)
print(x)
```

11.4 Metode

Metode adalah fungsi yang dapat diakses dari suatu objek. Kita telah menggunakan metode pada contoh-contoh sebelumnya. Misalnya pada objek string:

```
>>> a = "My name is Jojo"
>>> a.upper()
'MY NAME IS JOJO'
>>> a.replace("Jojo", "Koko")
'My name is Koko
```

Metode atau fungsi `upper()` dan `replace()` diakses melalui objek `a` yang merupakan sebuah string.

12 Modul

Seperti yang telah dijelaskan pada bagian awal, selain menggunakan interpreter (secara interaktif), kita juga dapat menuliskan instruksi atau program Python pada sebuah file dengan ekstensi `.py`. File ini sering juga disebut dengan *script* yang biasanya berisikan instruksi-instruksi yang relatif sederhana dan dapat dijalankan langsung oleh pengguna.

12.1 *Scripts*

Sebagai contoh sederhana mari kita coba buat file dengan nama `halo.py` dengan isi sebagai berikut.

```
message = "Hello how are you?"
for word in message.split():
    print(word)
```

Untuk mengeksekusi *script* ini, kita dapat menggunakan perintah:

```
python3 hello.py
```

Kita juga dapat menjalankan *script* ini pada saat kita menggunakan IPython dengan menggunakan sintaks `%run nama_script.py`. Contoh:

```
In[1]: %run halo.py
```

Ketika perintah di atas dipanggil, kita script telah dieksekusi dan variabel dan simbol yang ada pada script dapat kita akses pada IPython.

Scripts juga dapat menerima argumen baris perintah. Contoh pada file berikut ini, misalkan dengan nama `print_args.py`

```
import sys
print(sys.argv)
```

Kemudian pada terminal:

```
$ python3 print_args.py test arguments
['print_args.py', 'test', 'arguments']
```

12.2 Membuat modul

Jika program kita sudah semakin besar dan kompleks, biasanya kita perlu membagi program menjadi beberapa subprogram atau fungsi-fungsi, variabel, dan kelas yang ditulis pada suatu file yang mirip dengan *script* dan disebut dengan modul.

Misalkan buat file dengan nama `modulku.py`

```
def print_a():
    print("Ini adalah fungsi print_a dari modulku")

def print_b():
    print("Ini adalah fungsi print_b dari modulku")

c = 2
d = 2
```

Pada file ini kita mendefinisikan dua fungsi `print_a` dan `print_b`. Misalkan kita ingin menggunakan fungsi `print_a` dari interpreter. Kita dapat mengeksekusi file/*script* atau meng-importnya sebagai modul.

```
import modulku
modulku.print_a()
modulku.print_b()
```

Jika kita mengubah modul dan ingin mengimportnya pada suatu sesi yang sedang berjalan kita hanya akan mendapatkan modul yang lama. Untuk menggunakan module yang sudah terupdate kita harus melakukan *reload*:

```
import importlib
importlib.reload(modulku)
```