

## Modul 1

# Tinjauan Ulang Python

Di bab ini akan kita tinjau ulang bahasa Python yang sudah dipelajari di kuliah semester lalu. **Pembahasan di bab ini tidak akan komprehensif dan tidak bisa menggantikan buku Python yang bagus ataupun kuliah pemrograman satu semester.** Dengan demikian mahasiswa diminta mereview materi kuliah semester lalu secara menyeluruh dan menyiapkan buku acuan yang memadai untuk mendampingi pemakaian Python pada beberapa bulan ke depan.

**Penting:** mahasiswa diminta mengerjakan soal-soal latihan di modul praktikum ini dengan sungguh-sungguh. Ingat! Apa yang akan kamu panen besok bergantung dari apa yang kamu tanam hari ini.

Pastikan bahwa program Python versi 3.7.2 sudah terinstal di komputermu. Kalau belum, silakan unduh dari [python.org](https://www.python.org). Disarankan pula untuk mengunduh IDLEX atau Thonny sebagai IDE-nya. Kamu dapat mengunduhnya dengan mengetik `pip install idlex` atau `pip install thonny` di *command prompt*.

### 1.1 Memulai Python

Nyalakan Python. Kamu akan dihadapkan pada jendela Python dengan prompt yang siap diisi seperti ini:

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52)
[MSC v.1916 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

Kita mulai dengan menggunakan Python sebagai kalkulator. Pada prompt, ketikkan yang berikut ini diikuti menekan tombol <Enter>:

```
>>> 2 + 3
```

Apa yang kamu dapatkan? Lanjutkan:

```
>>> 2+3*5-6/2
```

Perhatikan bahwa perkalian dan pembagian lebih didahului ketimbang penjumlahan dan pengurangan. Kamu dapat mengubah *precedence* ini dengan memakai tanda kurung. Python juga dapat dipakai untuk menghitung sampai bilangan yang besar. Berapakah  $2^{1000}$ ? Ketik ini:

```
>>> # Menghitung dua pangkat seribu
>>> 2**1000
```

Simbol `#` menandakan komentar dan baris itu tidak akan dieksekusi oleh Python interpreter.

### Variabel dengan tipe data int dan float

Variabel adalah memberi nama pada suatu objek. Coba yang berikut ini:

```
>>> radius = 4
>>> pi = 3.14159
>>> area = pi * radius * radius
>>> print(area)
```

Jika kita memberi nilai pada variabel yang sudah ada, nilainya akan berganti:

```
>>> x = 4
>>> print(x)
4
>>> x = 5
>>> print(x)
5
```

Pada kode-kode di atas kita telah berkenalan dengan tipe data `int` dan `float`. Kamu bisa memeriksa tipe data suatu variable dengan fungsi `type()`. Contoh:

```
>>> a = 5
>>> b = 6.2
>>> type(a)
<class 'int'>
>>> type(b)
<class 'float'>
```

### String

String adalah serangkaian karakter. Beri nilai ‘Apa kabar’ pada string `s` seperti berikut<sup>1</sup>:

```
>>> s = 'Apa kabar'
>>> s
'Apa kabar'
>>> type(s)
<class 'str'>
```

Tipe data variable `s` adalah `str` (maksudnya ‘string’). Karakter juga dapat ‘dijumlahkan’.

Coba ini:

```
>>> a = 'Halo'
>>> b = ' mas'
>>> c = ' Data!'
```

<sup>1</sup>Gunakan tanda petik tunggal atau petik ganda, yang letak kuncinya di keyboard-mu adalah di sebelah kiri tombol <Enter>. JANGAN pakai tanda petik yang terletak di bagian kiri atas keyboard sebelah angka 1.

```
>>> d = a + b + c  
>>> d  
'Halo mas Data!'
```

Kamu tidak bisa menjumlahkan variabel dengan tipe data 'str' dengan variable dengan tipe data 'int'. Namun kamu dapat merubah, jika keadaannya tepat, tipe data suatu variabel. Seperti ini:

```
>>> g = '34'  
>>> h = 23  
>>> g + h  
Traceback (most recent call last):  
File "<pyshell#7>", line 1, in <module>  
g+h  
TypeError: can only concatenate str (not "int") to str  
>>> int(g) + h  
57
```

Variable `g` adalah bertipe data `str` (karena ada tanda kutipnya). Jadi tidak bisa dijumlahkan dengan `h`, yang bertipe data `int`. Tapi perintah `int(g)` membuat sebuah objek baru yang bertipe data integer (bilangan bulat).

## 1.2 List dan Tuple (dan String lagi)

List adalah sekumpulan objek yang berurutan. Tuple sama seperti itu, hanya saja dia tidak bisa diubah nilai elemennya. Perhatikan yang berikut ini:

```
>>> f = 2.5  
>>> g = 7  
>>> d = [f, g, 3.9, 8, 'Apa kabar']  
>>> d  
[2.5, 7, 3.9, 8, 'Apa kabar']  
>>> type(d)  
<class 'list'>
```

Variabel `d` adalah sebuah list dengan elemen-elemen seperti ditampilkan di atas. Kita bisa mengakses dan mengubah tiap-tiap elemennya seperti ini

```
>>> d[0] = 55  
>>> d  
[55, 7, 3.9, 8, 'Apa kabar']
```

Tampak bahwa elemen ke-0 sudah berubah nilainya. Kamu juga dapat mengakses elemen-elemen yang lain dengan index-nya. Ingat pelajaran tentang slicing dan index negatif.

Suatu list dapat di-*iterate* (“ditunjuk satu-per-satu secara urut”) seperti ini:

```
>>> for i in d:
    print(i)    ## di sini pencet '<Enter>' dua kali

55
7
3.9
8
Apa kabar
>>>
```

Sesungguhnya semua objek yang bersifat *iterable* dapat diiterasi seperti di atas, dapat ‘diiris’ (*sliced*, lihat di bawah), dan ‘dapat dikenai pencarian’.

### Slicing

Slicing adalah mengiris elemen-elemen di suatu list atau tuple dengan pola tertentu. Ketik seperti berikut

```
>>> a = 'Wacana keilmuan dan keislaman'
>>> b = [43,44,45,46,47,48,49,50]
```

Sekarang cobalah yang berikut ini dan amati hasilnya

```
a[0:6]
a[7:15]
a[::-1]
a[-7:-2]
a[-7:100]
len(a)
a[0:29]
a[0:100]
a[0:29:2]
a[0:200:2]
```

Lakukan hal yang serupa untuk variable **b** di atas.

### 1.3 Dictionary

Dictionary (atau `dict`) adalah tipe data di Python yang sangat berguna. Akan kita bahas secara ringkas.

Kalau list atau tuple mempunyai index bilangan bulat (saja), maka dict mempunyai index (disebut ‘key’) yang bisa berupa tipe data lainnya. Dictionary menyimpan data dalam bentuk **pasangan kunci-nilai**. Berikut ini contohnya

```
>>> dd = {'nama': 'joko', 'umur': 21, 'asal': 'surakarta'}
>>> dd['nama']
'joko'
>>>
```

Untuk membuat sebuah dictionary yang kosong, gunakan perintah `d = dict()` atau `d = {}`.

## 1.4 Set

Set atau himpunan adalah struktur data yang mirip sebuah list, tapi elemennya tidak berulang. Untuk membuat sebuah set, bisa kita gunakan kurung kurawal ataupun fungsi `set()`. Untuk membuat set kosong, pakailah perintah `set()`, bukan `{}`. Yang terakhir ini adalah untuk dictionary kosong. Perlu diperhatikan bahwa anggota dalam set *tidak* mengenal urutan (beda dengan `list`, yang mengenal urutan). Jadi ketika suatu object set di-print, bisa jadi urutannya berubah.

Berikut ini adalah demonstrasi set. Silakan dijalankan!

```
>>> keranjang = {'apel', 'jeruk', 'apel', 'manggis', 'jeruk', 'pisang'}
>>> print(keranjang)      # tunjukkan bahwa duplikat telah dibuang
{'manggis', 'pisang', 'jeruk', 'apel'}
>>> 'jeruk' in keranjang
True
>>> 'rumput' in keranjang
False
>>> ## Akan didemokan operasi set pada huruf unik di dua kata
>>> a = set('surakarta')
>>> b = set('yogyakarta')
>>> a                                # huruf di a, tanpa diulang
{'s', 'u', 'r', 'a', 'k', 't'}
>>> b                                # huruf di b, tanpa diulang
{'y', 'o', 'g', 'y', 'j', 'a', 'k', 't'}
>>> a - b                            # huruf di a tapi tidak di b
{'s', 'u'}
>>> a | b                            # huruf di a atau di b atau di keduanya
{'o', 'r', 'g', 'y', 's', 'u', 'k', 'a', 't'}
>>> a & b                            # huruf di a yang sekaligus juga di b
{'r', 'k', 't', 'a'}
>>> a ^ b                            # huruf di a atau b tapi tidak sekaligus di keduanya
{'o', 'y', 'g', 's', 'u'}
```

## 1.5 Operator relasional dan tipe data Boolean

Tipe data boolean adalah tipe data dengan dua kemungkinan nilai: `True` (“benar”) atau `False` (“salah”). Operator relasional adalah operator yang membandingkan dua variabel. Ketik contoh berikut:

```
p = 3
q = 7
p > q
p < q
p == q    # perhatikan bahwa ada dua tanda '='.
```

Kita juga dapat secara langsung mengetikkan langsung objeknya. Coba yang berikut:

```
4 < 8
4 > 8
4 == 4    # ada dua tanda '='.
4 < 4
4 <= 4
```

String juga dapat dibandingkan. Silakan ketik dan amati ini:

```
'UMS' > 'UGM'    ## True atau False? Jangan lupa memberi tanda kutipnya.
'UMS' > 'ITB'
'Emas' < 'Sayur'
'a' > 'b'
'a' < 'z'
'A' > 'a'
```

Tebak: bagaimanakah kira-kira antar string dibandingkan satu sama lain? Tipe data boolean ini dapat disimpan juga di suatu variabel. Silakan coba yang berikut:

```
v = 5 < 7      # v berisi True
ff = 'UMS' > 'UGM'
type(ff)
ff           # ketik nama variabelnya lalu tekan <Enter> .
g = 3 == 3
g
```

## 1.6 File .py

Perintah python dapat disimpan dalam suatu file .py dan dijalankan dengan menekan tombol **F5** atau mengklik Run –> Run Module.

**Latihan 1.1** Buatlah suatu file baru lewat IDLE<sup>2</sup>. Simpan sebagai *LatReview1.py*. Ketikkan kode python berikut<sup>3</sup>

```
1 a = 4
2 b = 5
3 c = a + b
4 print('Nilai a =', a)
5 print('Nilai b =', b)
6 print('Sekarang, c = a + b =', a, '+', b, '=', c)
7 print('')
8 print('Sudah selesai.')
```

□

Larikan program di atas dengan memencet tombol **F5**. Kamu akan mendapati ini di jendela Python Shell<sup>4</sup>:

```
Nilai a = 4
Nilai b = 5
Sekarang, c = a + b = 4 + 5 = 9

Sudah selesai.
```

<sup>2</sup>Klik 'File' lalu pilih 'New Window'. Atau pencet tombol **[Ctrl] + N**

<sup>3</sup>Aku ingatkan dirimu lagi, bahwa gunakanlah tanda petik tunggal atau petik ganda, yang letak kuncinya di keyboard-mu adalah di sebelah kiri tombol <Enter>. JANGAN pakai tanda petik yang terletak di bagian kiri atas keyboard sebelah angka 1.

<sup>4</sup>Jendela yang ada >>>-nya

### Mengambil nilai dari keyboard pengguna

Kita juga dapat mengambil nilai dari pengguna melalui keyboard. Perintah yang digunakan adalah

```
var = input('Silakan mengetik lalu tekan enter:> ')
```

Ketika ini dieksekusi, pengguna diminta memasukkan string lewat keyboard. Hasilnya (selalu bertipe string), ditampung dalam variable var.

**Latihan 1.2** Buatlah file berikut:

```
LatReview2.py
1 print('Kita perlu bicara sebentar...')
2 nm = input('Siapa namamu? (ketik di sini)> ')
3 print('Selamat belajar,', nm)
4 angkaStr = input('Masukkan sebuah angka antara 1 sampai 100> ')
5 a = int(angkaStr)
6 kuadratnya = a*a
7 print(nm + ', tahukah kamu bahwa', a, 'kuadrat adalah', kuadratnya, '?')
```

Larikan program di atas dengan memencet tombol **[F5]**. Jangan lupa untuk menjawab pertanyaan yang muncul!

□

## 1.7 Fungsi

Fungsi adalah semacam prosedur yang bisa kita panggil sewaktu-waktu.

**Latihan 1.3** Kita mulai dengan membuat tiga buah fungsi dalam satu file.

```
LatReview3.py
1 def ucapanSalam():
2     print("Assalamu 'alaikum!")
3
4 def sapa(nama):
5     ucapanSalam() # Ini memanggil fungsi ucapanSalam() di atas.
6     print('Halo',nama)
7     print('Selamat belajar!')
8
9 def kuadratkan(b):
10    h = b*b
11    return h
```

Larikan program itu dengan menekan tombol **[F5]**. Tidak terjadi apa-apa sepertinya... Tapi di memori sudah termuat fungsi-fungsi itu. Sekarang di Python Shell ketikkan perintah-perintah berikut<sup>5</sup>

```
ucapanSalam() # JANGAN TINGGALKAN tanda kurungnya!
sapa('Mas Wowok') # Atau, lebih baik lagi, cantumkan namamu.
b = kuadratkan(5)
b
k = 9
print('Bilangannya',k, ', kalau dipangkatkat dua jadinya', kuadratkan(k))
```

<sup>5</sup>Tentu saja, kalau pas di Python Shell komentarnya tidak perlu ditulis. Capek deh.

□

Perhatikan bahwa suatu fungsi bisa mempunyai ataupun tidak menerima parameter, bisa mengembalikan atau tidak mengembalikan sesuatu. Contoh fungsi yang menerima sesuatu dan mengembalikan sesuatu adalah fungsi `kuadratkan()` di atas.

Perhatikan juga bahwa Python memakai *indentation* (spasi ke kanan, biasanya sebanyak 4) untuk menandai blok eksekusi.

Beberapa bahasa lain –seperti C, C++, Java, C#, PHP, JavaScript– memakai kurung kurawal (simbol '{' dan '}') sebagai pembatas blok eksekusi. Bahasa yang lain lagi –seperti Pascal, Delphi, Lazarus, VHDL, L<sup>A</sup>T<sub>E</sub>X– memakai bentuk kata-kata `begin` — `end` untuk membuat blok eksekusi.

**Latihan 1.4** Buatlah sebuah fungsi `selesaikanABC()` yang menyelesaikan persamaan kuadrat

$$ax^2 + bx + c = 0. \quad (1.1)$$

Maksudnya, dengan diberi nilai  $a, b, c$ , cari (kalau ada) dua nilai  $x_1$  dan  $x_2$  yang memenuhi persamaan di atas. Ingat pelajaran SMA dulu bahwa solusi persamaan di atas diberikan oleh

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \quad (1.2)$$

Ketik kode berikut

```
1 from math import sqrt as akar
2 def selesaikanABC(a,b,c):
3     a = float(a) # mengubah jenis integer menjadi float
4     b = float(b)
5     c = float(c)
6     D = b**2 - 4*a*c
7     x1 = (-b + akar(D))/(2*a)
8     x2 = (-b - akar(D))/(2*a)
9     hasil = (x1,x2) # tuple yang terdiri dari dua elemen
10    return hasil
```

Larikan program ini, lalu panggil di Python Shell seperti berikut

```
>>> k = selesaikanABC(1,-5,6)
>>> k
(3.0, 2.0)
>>> k[0]
3.0
>>> k[1]
2.0
```

Perhatikan bahwa kita memanggil fungsi dan mengembalikan hasilnya ke variabel `k`, yang bertipe data tuple.

Baris pertama file di atas: dari sebuah modul (`math`), meng-import sebuah fungsi (`sqrt`), lalu melekatkan ke sebuah nama '`akar`'.

Program di atas masih belum menangkap semua kemungkinan input. Jika dipanggil dengan `selesaikanABC(1,2,3)` akan menghasilkan sebuah error, karena determinan  $D = b^2 - 4ac$  bernilai negatif.

Ubahlah program itu agar bisa menangkap kasus ini. □

## 1.8 Pengambilan keputusan

Kita mulai dengan beberapa contoh.

**Latihan 1.5** Buat suatu fungsi `apakahGenap()` yang mengembalikan objek boolean.

Jika input ke fungsi itu genap, kembalikan `True`. Jika ganjil, kembalikan `False`.

File berikut bisa menjadi jawabannya (ketiklah!)

```
LatReview5.py
1 def apakahGenap(x):
2     if (x%2 == 0):
3         return True
4     else:
5         return False
```

Larikan kode di atas dengan memencet `F5`. Di Python Shell, panggil fungsi itu:

```
apakahGenap(48)
apakahGenap(37)
```

Apakah hasilnya? □

**Latihan 1.6** Buat suatu fungsi yang apabila diberi suatu bilangan bulat positif akan mencetak suatu kalimat dengan aturan seperti berikut:

- Jika bilangan itu kelipatan 3, akan mencetak “Bilangan itu adalah kelipatan 3”
- Jika bilangan itu kelipatan 5, akan mencetak “Bilangan itu adalah kelipatan 5”
- Jika bilangan itu kelipatan 3 dan 5 sekaligus, akan mencetak “Bilangan itu adalah kelipatan 3 dan 5 sekaligus”
- Jika bilangan bukan kelipatan 3 ataupun 5, akan mencetak “Bilangan itu bukan kelipatan 3 maupun 5”

Yang di bawah ini bisa menjadi jawabannya (ketiklah):

*LatReview6.py*

```

1 def tigaAtauLima(x):
2     if (x%3==0 and x%5==0):
3         print('Bilangan itu adalah kelipatan 3 dan 5 sekaligus')
4     elif x%3==0:
5         print('Bilangan itu adalah kelipatan 3')
6     elif x%5==0:
7         print('Bilangan itu adalah kelipatan 5')
8     else:
9         print('Bilangan itu bukan kelipatan 3 maupun 5')

```

Larikan program itu dengan memencet tombol **F5**, lalu panggil di Python Shell. Berikut ini contohnya.

```

>>> tigaAtauLima(9)
Bilangan itu adalah kelipatan 3
>>> tigaAtauLima(10)
Bilangan itu adalah kelipatan 5
>>> tigaAtauLima(15)
Bilangan itu adalah kelipatan 3 dan 5 sekaligus
>>> tigaAtauLima(17)
Bilangan itu bukan kelipatan 3 maupun 5

```

Renungkanlah jalan logika program di atas. □

**Latihan 1.7** Mencari seseorang di dictionary.

*LatReview7.py*

```

1 staff = { 'Santi' : 'santi@ums.ac.id', \
2           'Jokowi' : 'jokowi@solokab.go.id', \
3           'Endang' : 'Endang@yahoo.com', \
4           'Sulastri': 'Sulastri3@gmail.com' }
5
6 yangDicari = 'Santi'
7 if yangDicari in staff:
8     print('emailnya', yangDicari, 'adalah', staff[yangDicari])
9 else :
10    print('Tidak ada yang namanya', yangDicari)

```

□

Bentuk pengambilan keputusan lain yang harus kamu ketahui adalah memakai **case** (tidak dibahas di sini).

## 1.9 Loop

Loop adalah perulangan. Meskipun semua perintah perulangan bisa dibuat dengan konstruk **if-elif-else**, banyak kemudahan yang bisa dicapai dengan memakai konstruk perulangan yang sudah tersedia. Ada dua konstruk yang akan kita bahas di sini: **for** dan **while**.

**Konstruk For**

**Latihan 1.8** “Cetaklah bilangan dari 0 sampai 9.” Ini dapat dicapai dengan program berikut

```
LatReview8.py  
1 for i in range(0,10):  
2     print(i)
```

yang menghasilkan, setelah menjalankan program di atas,

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

□

Seperti dijelaskan di halaman 4, semua objek yang *iterable* (*list*, *tuple*, *str*,...) dapat menjadi argumen untuk konstruk **for** ini.

**Latihan 1.9** Mencetak tiap-tiap huruf di sebuah kalimat. Misal kita punya string *s*="Ini Budi", maka program berikut akan mencetak huruf-huruf di *s* satu persatu.

```
LatReview9.py  
1 for i in s:  
2     print(i)
```

Yang akan menghasilkan

```
I  
n  
i  
  
B  
u  
d  
i
```

□

Cobalah mengubah *s* di atas menjadi suatu list: *s*=[4,3,2,5,6]. Bagaimanakah hasilnya?

**Latihan 1.10** Mencetak isi suatu dictionary. Ingat bahwa dictionary, berbeda dari list dan tuple, mempunyai pasangan **kunci–nilai**. Perhatikan program berikut:

```
LatReview10.py
1 dd = {'nama': 'joko', 'umur': 21, 'asal': 'surakarta'}
2 for kunci in dd:
3     print(kunci, '<---->', dd[kunci])
```

### Konstruk while

Konstruk **while** dapat diilustrasikan lewat contoh berikut

**Latihan 1.11** Cetaklah bilangan 0 sampai suatu bilangan tertentu beserta kuadratnya, asalkan kuadratnya itu kurang dari 200. Program di bawah bisa menjadi penyelesaiannya (ketiklah).

```
LatReview11.py
1 bil = 0
2 while(bil*bil<200):
3     print(bil, bil*bil)
4     bil = bil + 1
```

Larikan program di atas. Hasilnya akan seperti berikut

```
0 0
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
11 121
12 144
13 169
14 196
```

Tepat sebelum  $\text{bil} * \text{bil} \geq 200$ , programnya keluar dari perulangan itu. Jadi bisa kamu lihat bahwa program akan mengeksekusi blok di bawah **while** kalau kondisinya terpenuhi. Kalau sudah tidak terpenuhi, dia akan keluar dari perulangan itu.

□

## 1.10 Kata-kata kunci di Python

Katakunci adalah kata yang dipakai untuk kepentingan operasional suatu bahasa, dalam hal ini Python. Kamu sudah menemui banyak katakunci ini. Misalnya **for**, **in**, **if**,...

Seperti sudah kamu duga, katakunci-katakunci ini tidak boleh (dan bahkan tidak bisa) dipakai sebagai nama variabel. Daftar katakunci ini dapat dilihat dengan mengeksekusi ini:

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await',
'break', 'class', 'continue', 'def', 'del', 'elif', 'else',
'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in',
'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return',
'try', 'while', 'with', 'yield']
```

## 1.11 Soal-soal untuk Mahasiswa

Sebelum mengerjakan soal-soal di bawah, kerjakan dulu latihan-latihan di atas.

- Buatlah suatu fungsi `cetakSiku(x)` yang akan mencetak yang berikut:

```
*  
**  
***  
****  
*****
```

Nilai `x` menunjukkan tinggi segitiga itu (gambar di atas berarti bisa didapatkan dari menjalankan `cetakSiku(5)`) Gunakan perulangan dua kali (*double loop*)!

- Buatlah sebuah fungsi yang menerima dua integer positif, yang akan menggambar bentuk persegi empat. Contoh pemanggilan:

```
>>> gambarlahPersegiEmpat(4,5) #tombol <enter> dipencet
@ @ @ @
@   @
@   @
@ @ @ @
```

- Berikut ini adalah dua soal yang saling berkaitan

- Buatlah sebuah fungsi yang menerima sebuah string dan mengembalikan sebuah list yang terdiri dari dua integer. Dua integer kembalian ini adalah: jumlah huruf di string itu dan jumlah huruf vokal (huruf vokal adalah huruf hidup) di string itu. Contoh pemanggilan:

```
>>> k = jumlahHurufVokal('Surakarta')
>>> k
(9, 4) # Sembilan huruf, dan empat di antaranya huruf vokal
```

- Sama dengan soal (a) di atas, tapi sekarang yang dihitung adalah huruf konsonan. Hanya ada satu baris yang berbeda di dalam kodennya! Contoh pemanggilan:

```
>>> k = jumlahHurufKonsonan('Surakarta')
>>> k
(9, 5) # Sembilan huruf, dan lima di antaranya huruf konsonan
```

- Buatlah sebuah fungsi yang menghitung rerata sebuah array yang berisi bilangan. Rerata mempunyai rumus

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}. \quad (1.3)$$

Namun ingatlah bahwa Python memulai index dari 0. Fungsi itu harus mempunyai bentuk `rerata(x)`, dengan `x` adalah list berisi bilangan yang ingin dihitung reratanya. Jadi, pekerjaanmu akan mempunyai bentuk:

- Buat suatu file dengan isi seperti ini

```

1 def rerata(b):
2     # Di sini letak
3     # programmu
4     # ...
5     return hasil

```

- Larikan program itu dengan memencet `[F5]`, lalu panggil program itu seperti ini

```

rerata([1,2,3,4,5]) # hasilnya 3
g = [3,4,5,4,3,4,5,2,2,10,11,23]
rerata(g)

```

*Extra credit:* Buat pula fungsi untuk menghitung *variance* dan *standard deviation*-nya dengan prototype, secara berurutan, `variance(x)` dan `stdev(x)`.

5. Buatlah suatu fungsi untuk menentukan apakah suatu bilangan bulat adalah bilangan prima atau bukan. Untuk mudahnya, lengkapilah program di bawah ini

```

1 from math import sqrt as sq
2 def apakahPrima(n):
3     n = int(n) # Kalau pecahan, dibuang pecahannya.
4     assert n>=0 # Hanya menerima bilangan non-negatif.
5     primaKecil = [2,3,5,7,11] # Kalau angkanya kecil, akan
6     bukanPrKecil = [0,1,4,6,8,9,10]# tertangkap di sini.
7     if n in primaKecil:
8         return True
9     elif n in bukanPrKecil:
10        return False
11    else:
12        for i in range(2,int(sq(n))+1): #Cukup sampai akar nya.
13            .... # Tugasmu
14            .... # mengisi
15            .... # titik-titik ini.

```

Setelah selesai, larikan program di atas dan lalu tes di Python Shell:

```

apakahPrima(17)
apakahPrima(97)
apakahPrima(123)

```

6. Buatlah suatu program yang mencetak semua bilangan prima dari 2 sampai 1000. Kamu tidak harus memanfaatkan fungsi di atas<sup>6</sup>.
7. Buatlah suatu program yang menerima bilangan bulat positif dan memberikan *faktorisasi-prima*-nya. Faktorisasi prima adalah pemfaktoran suatu bilangan bulat ke dalam bilangan-bilangan prima yang menjadi konstituenya. Contoh:

<sup>6</sup>Salah satu algoritma yang bisa dipakai adalah *Sieve of Eratosthenes*. Silakan lihat di Wikipedia.

```
>>> faktorPrima(10)
(2, 5)
>>> faktorPrima(120)
(2, 2, 2, 3, 5)
>>> faktorPrima(19)
(19,)
```

8. Buat suatu fungsi `apakahTerkandung(a,b)` yang menerima dua string `a` dan `b`, lalu menentukan apakah string `a` terkandung dalam string `b`. Eksekusinya seperti ini:

```
>>> h = 'do'
>>> k = 'Indonesia tanah air beta'
>>> apakahTerkandung(h,k)
True
>>> apakahTerkandung('pusaka',k)
False
```

9. Buat program untuk mencetak angka dari 1 sampai 100. Kalau angkanya pas kelipatan 3, cetak 'Python'. Kalau pas kelipatan 5, cetak 'UMS'. Kalau pas kelipatan 3 sekaligus kelipatan 5, cetak 'Python UMS'. Jadi hasilnya:

```
1
2
Python
4
UMS
Python
7
8
Python
UMS
11
Python
13
14
Python UMS
16
17
...
...
```

10. Buat modifikasi pada Contoh 1.4, agar bisa menangkap kasus di mana determinannya kurang dari nol. Jika ini terjadi, tampilkan peringatan di layar seperti ini:

```
>>> selesaikanABC(1,2,3)
Determinannya negatif. Persamaan tidak mempunyai akar real.
>>>
```

11. Buat suatu fungsi `apakahKabisat()` yang menerima suatu angka (tahun). Jika tahun itu kabisat, kembalikan True. Jika bukan kabisat, kembalikan False.

Tahun kabisat – tahun yang memiliki tanggal 29 Februari – adalah tahun yang habis dibagi 4, kecuali dia habis dibagi 100 (maka dia bukan tahun kabisat). Tapi kalau dia habis dibagi 400, dia adalah tahun kabisat (meski habis dibagi 100).

Berikut ini adalah beberapa contoh:

- 1896 tahun kabisat (habis dibagi 4)
  - 1897 bukan tahun kabisat (sudah jelas)
  - 1900 bukan tahun kabisat (meski habis dibagi 4, tapi habis dibagi 100, dan tidak habis dibagi 400)
  - 2000 tahun kabisat (habis dibagi 400)
  - 2004, 2008, 2012, 2016, ..., 2096 tahun kabisat
  - 2100, 2200, 2300 bukan tahun kabisat
  - 2400 tahun kabisat
12. Program permainan tebak angka. Buat program yang alurnya secara global seperti ini:
- Komputer membangkitkan bilangan bulat random antara 1 sampai 100. Nilainya disimpan di suatu variabel dan tidak ditampilkan ke pengguna.
  - Pengguna diminta menebak angka itu, diinputkan lewat keyboard.
  - Jika angka yang diinputkan terlalu kecil atau terlalu besar, pengguna mendapatkan umpan balik dari komputer (“Angka itu terlalu kecil. Coba lagi”)
  - Proses diulangi sampai angka itu tertebak atau sampai sekian tebakan meleset<sup>7</sup>.

Ketika programnya dilarikan, prosesnya kurang lebih seperti di bawah ini

```
Permainan tebak angka.
Saya menyimpan sebuah angka bulat antara 1 sampai 100. Coba tebak.
Masukkan tebakan ke-1:> 50
Itu terlalu kecil. Coba lagi.
Masukkan tebakan ke-2:> 75
Itu terlalu besar. Coba lagi.
Masukkan tebakan ke-3:> 58
Ya. Anda benar
```

13. Buat suatu fungsi `katakan()` yang menerima bilangan bulat positif dan mengembalikan suatu string yang merupakan pengucapan angka itu dalam Bahasa Indonesia. Contoh:

```
>>> katakan(3125750)
'Tiga juta seratus dua puluh lima ribu tujuh ratus lima puluh'
```

Batasi inputnya agar lebih kecil dari satu miliar. *Extra credit:* gunakan rekursi.

14. Buat suatu fungsi `formatRupiah()` yang menerima suatu bilangan bulat positif dan mengembalikan suatu string yang merupakan bilangan itu dengan ‘format rupiah’. Contoh:

```
>>> formatRupiah(1500)
'Rp 1.500'
>>> formatRupiah(2560000)
'Rp 2.560.000'
```

---

<sup>7</sup>Kalau angka yang harus ditebak adalah antara 1 dan 100, seharusnya maksimal jumlah tebakan adalah 7. Dapatkah kamu melihat kenapa? Kalau antara 1 dan 1000, maksimal jumlah tebakan adalah 10. Apa polanya?