

Praktikum Komputasi Statistika 2
Pertemuan 1
Pengenalan Bahasa Pemrograman R dan Python

A. Pengenalan Bahasa R

1. Tipe Data

Dalam R, terdapat beberapa struktur data yang sering dijumpai, yaitu vektor, matriks, data frame, list, dan factors. Struktur data ini akan terdiri dari beberapa jenis tipe data. Untuk memeriksa tipe data dari suatu variabel, kita dapat menggunakan fungsi `class()`.

i. Numeric

Tipe data numerik di R dibagi menjadi dua, yaitu double (memuat desimal) dan integer (bilangan bulat). Contoh:

Double → 5.1, 289.09

Integer → 1, 2e5, 100

ii. Complex

Contoh → 2+5i

iii. Logical/Boolean

Tipe data yang memiliki 2 nilai, yaitu True atau False.

iv. Character/String

Merupakan tipe data yang memuat data dalam bentuk karakter. Contoh:

“A”, “B”, “Budi”, “Zaki”

Lebih detailnya, bentuk-bentuk tipe data dan pendahuluan di R sudah dipelajari di mata kuliah sebelumnya (Komputasi Statistika I), Silahkan untuk mempelajari kembali di modul-modul yang telah diberikan di praktikum Komputasi Statistika I. Modul pengenalan tipe data di R diberikan di github kelas.

2. Control Flow

Barisan perintah di R dieksekusi per baris. Dalam R, barisan perintah yang diletakkan di dalam tanda “{ }” sebagai satu group dipandang sebagai satu ekspresi tunggal dalam R, tetapi barisan ekspresi yang dinyatakan dalam bentuk grup ini juga dijalankan baris perbaris. Untuk mengatur proses eksekusi, diperlukan perintah-perintah yakni *control flow*. Bentuk *control flow* ini akan banyak digunakan dalam menulis suatu fungsi yang dibentuk oleh pengguna. Beberapa perintah *control flow* yang sering dijumpai di R adalah sebagai berikut:

i. Decision Making-if

Terdapat dua bentuk syntax, yaitu

```
if(cond){expression}
```

Evaluasi nilai *cond*, jika benar maka dijalankan *expression* yang ditulis,

```
if(cond){expression}else(condexpression)
```

Evaluasi nilai *cond*, jika benar maka dijalankan *expression* yang ditulis, jika salah maka eksekusi *condexpression*.

Contoh:

#Bilangan Positif Negatif

x=0

```
if(x>0){cat(x, " adalah bilangan positif","\n")
      }else if (x<0){cat(x, "adalah bilangan
negatif", "\n")
      }else{cat(x, " adalah bilangan nol",
"\n")
}
```

Output:

```
0 adalah bilangan nol
>
```

#Bilangan Ganjil Genap

y=2

```
if(y%%2==0){cat(y, "adalah bilangan genap", "\n")
            }else{cat(y, "adalah bilangan ganjil", "\n")
            }
}
```

Output:

```
2 adalah bilangan genap
>
```

ii. Decision Making-ifelse

Syntax dasar fungsi:

```
ifelse(condition,expression1,expression2)
```

Fungsi ifelse ini merupakan bentuk vektor dari statement if. Pertama-tama dievaluasi nilai condition, jika TRUE maka di evaluasi di nilai expression1, jika FALSE dievaluasi nilai expression2.

Contoh:

```
fungsi.tanda=function(x){
  ifelse(x>0,1,ifelse(x<0,-1,0))
}
fungsi.tanda(-5:5)
```

```
> fungsi.tanda(-5:5)
[1] -1 -1 -1 -1 -1 0 1 1 1 1 1
```

iii. Decision Making-Switch

Syntax dasar fungsi:

```
switch(expression)
```

evaluasi dari nilai expression, yang bernilai karakter atau numerik

Nilai dari expression dibandingkan dengan nilai dari argumen

setelahnya, jika cocok dengan salah satu, nilai dari argumen tersebut kemudian dimunculkan sebagai nilai dari fungsi switch. Fungsi ini dapat menggantikan nested if untuk melakukan multiple conditional.

Contoh:

```
my.switch = function(n){  
  switch(n,  
    "satu" = c(1,2,3),  
    "dua" = c(4,5,6),  
    "tiga" = c(6,7,8),  
    "empat" = c(1:9))  
}  
my.switch("satu")
```

Output:

```
> my.switch("satu")  
[1] 1 2 3
```

```
my.switch("dua")
```

Output:

```
> my.switch("dua")  
[1] 4 5 6
```

iv. Looping-For

Syntax dasar fungsi:

```
for(name in expression1){expression2}
```

evaluasi nilai dari expression2 untuk setiap nilai di suatu barisan index di expression1

Contoh:

```
fungsi.tanda=function(x){  
  signx=NULL  
  for(i in 1:length(x)){  
    if(x[i]>0){signx[i]=1  
  
    }else if(x[i]<0){signx[i]=-1  
    }else{signx[i]=0}
```

```
}  
signx  
}  
fungsi.tanda(-5:5)
```

Output:

```
> fungsi.tanda(-5:5)  
[1] -1 -1 -1 -1 -1 0 1 1 1 1 1
```

v. **Looping-While**

Syntax dasar fungsi:

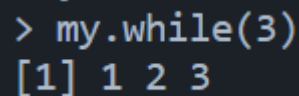
```
while(condition){expression}
```

Evaluasi nilai boolean dari condition, jika TRUE maka evaluasi nilai expression, kemudian kembali ke awal untuk mengevaluasi expression lagi sampai nilai boolean dari condition bernilai FALSE.

Contoh:

```
my.while=function(n){  
  temp=NULL  
  if(n>0){  
    while(n>0){  
      temp[n]=n  
      n=n-1  
    }  
    temp  
  }  
  else cat("Pesan kesalahan : n harus bernilai positif!\n")  
}  
my.while(3)
```

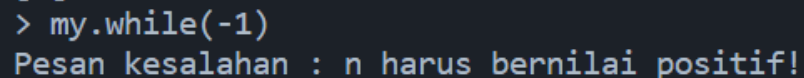
Output:



```
> my.while(3)  
[1] 1 2 3
```

```
my.while(-1)
```

Output:



```
> my.while(-1)  
Pesan kesalahan : n harus bernilai positif!
```

vi. **Looping-Repeat**

Syntax dasar fungsi:

```
repeat{expression}
```

mengulangi eksekusi expression secara terus menerus

Dalam menggunakan repeat, diperlukan statement untuk menghentikan perulangan eksekusi (misalnya menggunakan break atau next).

Contoh:

```
my.repeat=function(n){  
  temp=NULL  
  if(n>0){  
    repeat{  
      temp[n]=n  
      n=n-1  
      if(n==0) break  
    }  
    temp  
  }
```

```

        else cat("Pesan kesalahan : n harus bernilai
positif!\n")
    }

```

```
my.repeat(3)
```

Output:

```

> my.repeat(3)
[1] 1 2 3

```

```
my.repeat(-2)
```

Output:

```

> my.repeat(-2)
Pesan kesalahan : n harus bernilai positif!

```

vii. Statement-Return & Stop

Kedua statement tersebut digunakan untuk menghentikan eksekusi suatu fungsi yang sedang diakses dan kembali ke R prompt.

Syntax:

```
return(expression)
```

stop fungsi yang sedang di akses/evaluate dan munculkan output dari expression di prompt

```
stop(message)
```

digunakan untuk memberikan tanda adanya kesalahan dengan menghentikan evaluasi dari fungsi yang sedang diakses dan menampilkan message di prompt sebagai pesan kesalahan dan kembali ke R prompt.

Contoh:

```

kalibagi<-function(a,b){
  c = a*2
  d = b/2
  z = c(c,d)
  return(z)
}

```

```
kalibagi(2,3)
```

Output:

```

> kalibagi(2,3)
[1] 4.0 1.5

```

```
#STOP
```

```
x=c(T,F,F,T)
```

```

if(!is.numeric(x)){
  stop("Data harus dalam mode numeric!")
}

```

Output:

```
> if(!is.numeric(x)){  
+ stop("Data harus dalam mode numeric!")  
+ }  
Error: Data harus dalam mode numeric!
```

B. Pengenalan Bahasa Python

1. Tipe data

Secara umum tipe data dalam bahasa pemrograman python ada beberapa macam diantaranya yaitu :

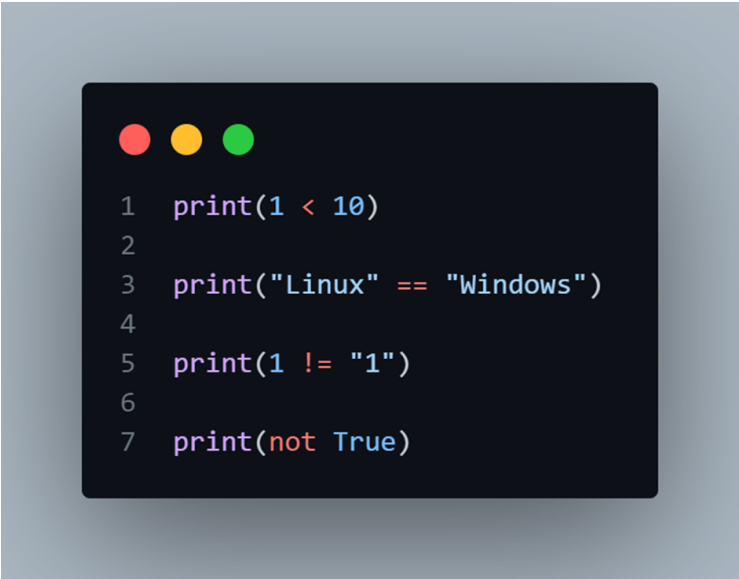
a. Boolean

Boolean adalah tipe data yang hanya mempunyai 2 nilai yaitu True atau False. Tipe data ini sering kali digunakan dalam *comparison* dan *logical operators*. Untuk *comparison operator* contohnya adalah sebagai berikut :

- `a == b` : a is equal to b
- `a != b` : a is different than b
- `a < b` : a is smaller than b
- `a <= b` : a is smaller or equal to b
- `a > b` : a is bigger than b
- `a >= b` : a is bigger or equal to b

Untuk logical operators dapat beberapa contohnya adalah sebagai berikut :

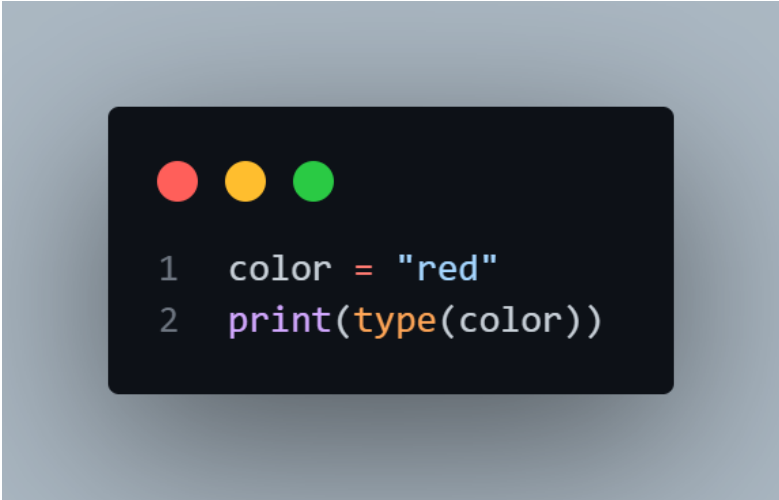
- `a and b`: True if both a and b are True. False otherwise.
- `a or b`: True if either a or b or both are True. False if both are False.
- `not a`: True if a is False, False if a is True.



```
1 print(1 < 10)  
2  
3 print("Linux" == "Windows")  
4  
5 print(1 != "1")  
6  
7 print(not True)
```

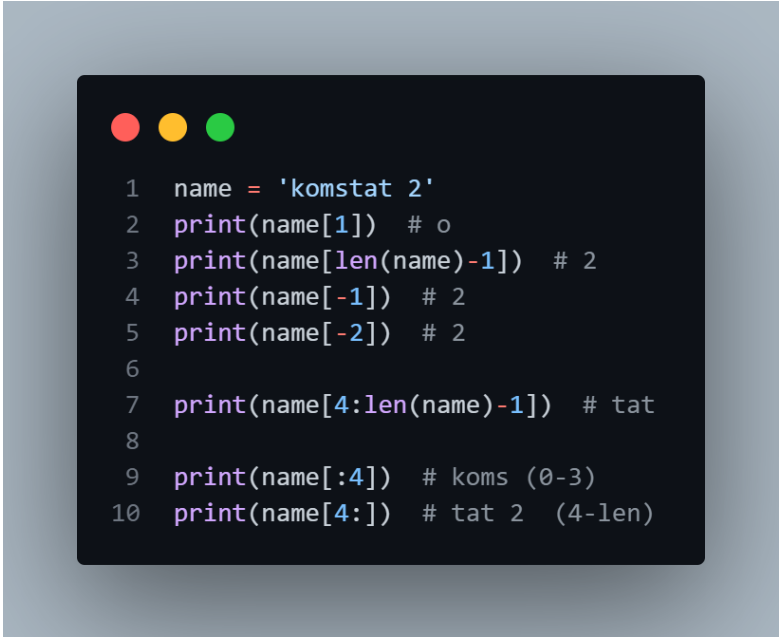
b. String

String adalah tipe data yang memuat karakter/kalimat bisa berupa huruf, angka, dan sebagainya yang diapit tanda " atau '. Dalam python tipe data ini dinyatakan sebagai berikut :



```
1 color = "red"
2 print(type(color))
```

Untuk mengakses substring, gunakan index atau slicing. Python mulai menghitung indeks dari 0 bukan 1. Indeks akses lebih besar dari panjangnya - 1, memicu indeks di luar jangkauan. Indeks negatif dimulai dari belakang. Untuk mengakses substring, gunakan slicing, mirip dengan index, dengan range menggunakan tanda titik dua sebagai pemisah, dimulai dari angka pertama, sampai 1 kurang dari yang terakhir. Slicing dengan salah satu dari dua indeks berarti indeks lainnya adalah 0 untuk nilai pertama atau panjangnya untuk nilai kedua.



```
1 name = 'komstat 2'
2 print(name[1]) # o
3 print(name[len(name)-1]) # 2
4 print(name[-1]) # 2
5 print(name[-2]) # 2
6
7 print(name[4:len(name)-1]) # tat
8
9 print(name[:4]) # koms (0-3)
10 print(name[4:]) # tat 2 (4-len)
```

String dalam Python tidak dapat diubah, artinya tidak dapat dimodifikasi, tidak dapat mengubah karakter individu. Itu akan memicu objek TypeError tidak mendukung item assignment. Untuk

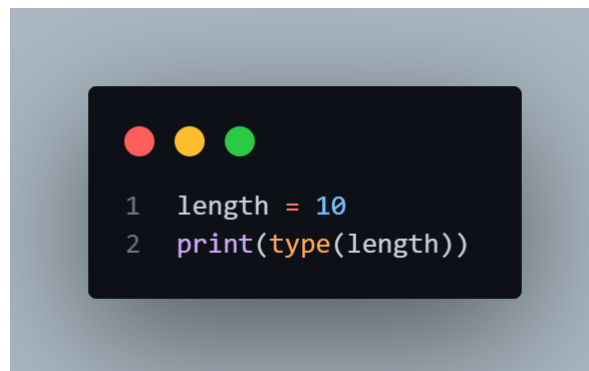
mengubah string, ganti dengan string baru. Gunakan kata kunci in untuk memeriksa apakah substring merupakan bagian dari string.



```
1 year = "it's 2021"
2 year[-1] = "0" # TypeError
```

c. Integer

Tipe data integer adalah tipe data yang menyatakan bilangan bulat. Dalam python dapat dituliskan sebagai berikut :



```
1 length = 10
2 print(type(length))
```

d. Float

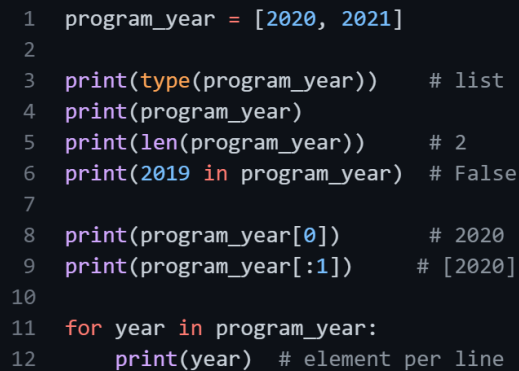
Tipe data float adalah tipe data yang mempunyai desimal. Dalam python dapat dituliskan sebagai berikut :



```
1 width = 2.5
2 print(type(width))
```

e. List

list wadah dengan ruang di dalamnya dibagi menjadi slot yang berbeda. Setiap slot dapat berisi nilai yang berbeda. Python menggunakan tanda kurung siku [] untuk menunjukkan di mana indeks dimulai dan diakhiri. List indeks dimulai dari 0, seperti string, juga dapat melakukan slicing untuk mengembalikan list lain.

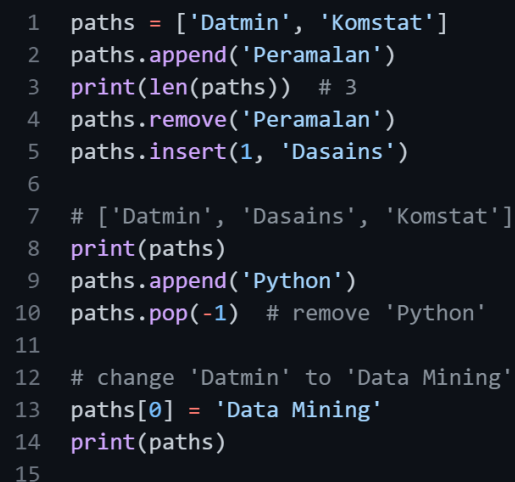


```

1  program_year = [2020, 2021]
2
3  print(type(program_year))    # list
4  print(program_year)
5  print(len(program_year))    # 2
6  print(2019 in program_year) # False
7
8  print(program_year[0])      # 2020
9  print(program_year[:1])    # [2020]
10
11 for year in program_year:
12     print(year) # element per line

```

Jika string tidak dapat diubah, list dapat diubah, artinya dapat menambah, menghapus, atau memodifikasi elemen dalam list. Gunakan `append` untuk menambah elemen terakhir. Untuk menambahkan indeks tertentu, gunakan `insert`. Untuk menghapus elemen, gunakan hapus dengan elemen atau `pop` dengan indeks. Untuk modifikasi elemen, ubah langsung ke indeks tertentu.



```

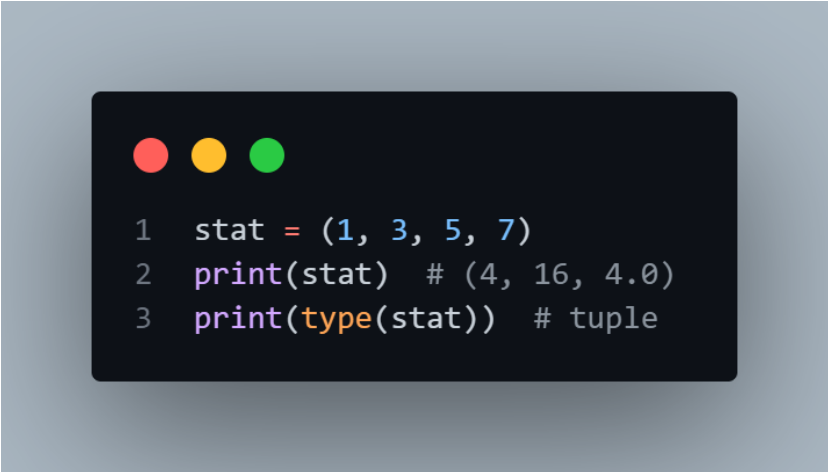
1  paths = ['Datmin', 'Komstat']
2  paths.append('Peramalan')
3  print(len(paths)) # 3
4  paths.remove('Peramalan')
5  paths.insert(1, 'Dasains')
6
7  # ['Datmin', 'Dasains', 'Komstat']
8  print(paths)
9  paths.append('Python')
10 paths.pop(-1) # remove 'Python'
11
12 # change 'Datmin' to 'Data Mining'
13 paths[0] = 'Data Mining'
14 print(paths)
15

```

f. Tuple

Tuple seperti list. Mereka dapat berisi elemen dari tipe data apa pun. Tapi, tidak seperti list, tuple tidak dapat diubah. Python menggunakan tanda kurung `()` untuk menunjukkan di mana tuple


dimulai dan diakhiri. Contoh tuple yang bagus adalah ketika suatu fungsi mengembalikan banyak nilai. String, list, dan Tuple disertakan sebagai tipe urutan. Dalam python dapat dituliskan sebagai berikut :



```
1 stat = (1, 3, 5, 7)
2 print(stat) # (4, 16, 4.0)
3 print(type(stat)) # tuple
```

g. Dictionary

Seperti list, dictionary digunakan untuk mengatur elemen ke dalam koleksi. Tidak seperti list, dictionary tidak mengakses elemen di dalam menggunakan posisi (indeks). Data di dalam dictionary berbentuk pasangan key dan value. Untuk mendapatkan value dari dictionary, gunakan key yang sesuai. Tidak seperti indeks list harus berupa angka, tipe key dalam dictionary menggunakan string, bilangan bulat, tupel & lainnya. dictionary menggunakan kurung kurawal {}. Dalam python dapat dituliskan sebagai berikut :



```
1 students = {
2     'komstat 2': 60,
3     'data mining': 90,
4     'peramalan': 30
5 }
6 print(type(students)) # dict
7 print(students['peramalan']) # 30
8
9 # keys: ['komstat', 'data mining', 'peramalan']
10 for key in students.keys():
11     # eg: komstat 2:60
12     print(key + ': ' + str(students[key]))
```

2. Fungsi

Fungsi adalah blok kode yang hanya berjalan saat dipanggil. kita dapat meneruskan data, yang dikenal sebagai parameter, ke dalam suatu fungsi. Suatu fungsi dapat mengembalikan data sebagai hasilnya. Dalam *Python* fungsi memiliki beberapa aturan yaitu :

- Tentukan fungsi dengan kata kunci def.

- Fungsi memiliki badan, ditulis sebagai blok setelah titik dua dalam definisi fungsi. Blok telah menjorok ke kanan.
- Untuk mendapatkan nilai dari suatu fungsi gunakan kata kunci return.

Dalam python dapat dituliskan sebagai berikut :



```

1  def greeting(name):
2      return 'Hello, ' + name
3
4  print(greeting("Komstat 2 2023"))

```

3. Control Flow

Barisan perintah dalam Python biasanya di eksekusi baris per baris. Dalam Python, barisan perintah dipisahkan dengan : diikuti dengan spasi dan indentasi sebagai satu group dipandang sebagai satu ekspresi tunggal dalam Python, tetapi barisan ekspresi yang dinyatakan dalam bentuk grup ini juga di execute baris perbaris. Untuk mengatur proses eksekusi, diperlukan perintah - perintah control flow. Bentuk control flow ini akan banyak digunakan dalam menulis suatu fungsi yang dibentuk oleh user. Beberapa perintah control flow yang dikenal di dalam Python sebagai berikut

a. Statement if

Ada tiga bentuk dasar statement if :

1) If cond:

 expression

Pertama akan dievaluasi nilai cond, jika benar eksekusi barisan tersebut.

2) If cond:

 expression

elif cond:

 condexpression

Pertama akan dievaluasi nilai cond, jika benar eksekusi expression, jika salah eksekusi condexpression.

3) If cond:

 expression

elif cond:


 condexpression

else:

 condexpression2

Pertama akan dievaluasi nilai cond, jika benar eksekusi expression, jika salah cek cond ke 2 jika benar eksekusi condexpression dan jika salah eksekusi condexpression2.

Dalam python dapat dituliskan sebagai berikut :



```
1 def check(number):
2     if number > 0:
3         return "Positive"
4     elif number == 0:
5         return "Zero"
6     else:
7         return "Negative"
8
9 print(check(10)) # Positive
```

b. Statement match

Syntax dasar fungsi berikut:

match expression:

Akan dievaluasi dari nilai expression, yang bernilai karakter atau numerik Nilai dari expression dibandingkan dengan nilai dari argumen setelahnya, jika cocok dengan salah satu, nilai dari argumen tersebut kemudian dimunculkan sebagai nilai dari fungsi switch. Fungsi ini dapat menggantikan nested if untuk melakukan multiple conditional. Dalam python dapat dituliskan sebagai berikut :



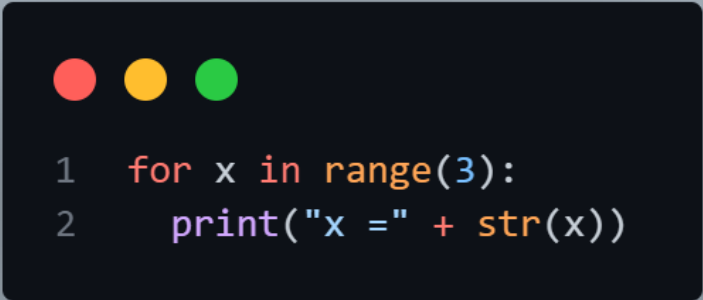
```
1 lang = input("What's the programming language you want to learn? ")
2
3 match lang:
4     case "JavaScript":
5         print("You can become a web developer.")
6
7     case "Python":
8         print("You can become a Data Scientist")
9
10    case "PHP":
11        print("You can become a backend developer")
12
13    case "Solidity":
14        print("You can become a Blockchain developer")
15
16    case "Java":
17        print("You can become a mobile app developer")
18    case _:
19        print("The language doesn't matter, what matters is solving problems.")
```

c. Statement for

Syntax dasar fungsi sebagai berikut:

```
for name in expression1:  
    expression2
```

Pertama akan dievaluasi nilai dari expression2 untuk setiap nilai di suatu barisan index di expression1. Dalam python dapat dituliskan sebagai berikut :




```
1  for x in range(3):  
2      print("x =" + str(x))
```

d. Statement while

Syntax dasar statement adalah sebagai berikut:

```
while <condition>:  
    # Jika kondisi benar  
    run_this_statement
```

Evaluasi nilai boolean dari condition, jika TRUE maka evaluasi nilai run_this_statement, kemudian kembali ke awal untuk mengevaluasi run_this_statement lagi. Dalam python dapat dituliskan sebagai berikut :



```
1  x = 7 # also try with x = 0  
2  
3  while x > 0:  
4      print("positive x=" + str(x))  
5      x = x - 1  
6      print("now x=" + str(x))
```

4. Data Frame

Sama halnya seperti data frame (atau dataframe) dalam R, Python juga mendukung penggunaan struktur data frame. Untuk menggunakannya diperlukan library bernama pandas.



Data frame dapat diilustrasikan seperti halnya *spreadsheet* atau tabel, dimana suatu data disimpan dalam baris dan kolom tertentu. Tipe data yang disimpan dalam data frame dapat berbeda-beda, akan tetapi satu kolom harus memiliki tipe data yang sama.

	Nama	NIM	Nilai	Lulus	Tinggi_badan
0	Mas Iqbal	19/PA/007	90	True	180.3
1	Riifat	20/PI/008	70	True	186.7
2	Zayn	21/PU/009	50	True	189.3
3	Zuko	22/PE/010	30	False	184.6

a. Membentuk Data Frame

Membentuk Data Frame

Sebelum membentuk dataframe, jalankan kode berikut terlebih dahulu untuk menggunakan library Pandas.

```
import pandas as pd
```

Apabila ditemukan pesan error, pastikan bahwa library tersebut sudah terunduh dan terpasang dengan menggunakan salah satu dari kode berikut.

```
pip install pandas
!pip install pandas #google colab
%pip install pandas #visual code
```

Penggunaan 'as' adalah untuk mempersingkat pandas menjadi pd. Singkatan ini dapat diubah sesuai preferensi masing-masing (atau tidak disingkat sama sekali).

Untuk membentuk dataframe, digunakan fungsi / methods bernama [`pd.DataFrame\(...\)`](#). Fungsi tersebut menerima beberapa argumen, akan tetapi argumen utama yang harus diberikan adalah data yang kita inginkan.

`pd.DataFrame(...)` dapat menerima berbagai jenis data seperti numpy array atau Python dictionary. Misal kita ingin membentuk dataframe yang berisi data mahasiswa Komputasi Statistika II. Setiap mahasiswa memiliki variabel berupa nama, NIM, dan nilai. Berikut adalah kode yang dapat digunakan.

```
pd.DataFrame(  
    {  
        'Nama' : ['Mas Iqbal', 'Riifat', 'Zayn',  
        'Zuko'],  
        'NIM' : ['19/PA/007', '20/PI/008', '21/PU/009',  
        '22/PE/010'],  
        'Nilai' : [90, 70, 50, 30]  
    }  
)
```

	Nama	NIM	Nilai
0	Mas Iqbal	19/PA/007	90
1	Riifat	20/PI/008	70
2	Zayn	21/PU/009	50
3	Zuko	22/PE/010	30

Kita juga dapat membentuk dictionary terlebih dahulu sebelum mengirimkannya sebagai argumen.

```
data_mahasiswa = {
    'Nama' : ['Mas Iqbal', 'Riifat', 'Zayn',
    'Zuko'],
    'NIM' : ['19/PA/007', '20/PI/008', '21/PU/009',
    '22/PE/010'],
    'Nilai' : [90, 70, 50, 30]
}
pd.DataFrame(data_mahasiswa)
```

Kita dapat menyimpan dataframe tersebut ke dalam sebuah objek/variabel. Untuk menampilkannya, cukup ketik saja nama objek tersebut (atau gunakan `print(objek)`).

```
dataku = pd.DataFrame(data_mahasiswa)
print(dataku)
```

b. Manipulasi Data Frame

1) Mengakses dan mengubah isi sebuah sel

Untuk mengakses isi sebuah sel, kita dapat menggunakan salah satu dari banyak cara berikut. Perlu diingat bahwa setiap cara memiliki kelebihan dan kekurangannya masing-masing.

```
dataku.loc[1, 'Nilai']
dataku.iloc[2,1]
dataku.iat[2,2]
```

Untuk merubah nilai sel tersebut, cukup lakukan variable assignment seperti biasanya.

```
#merubah nilai sel
dataku.loc[1, 'Nilai'] = 79
dataku.iloc[2,1] = '16/PK/923'
dataku.iat[2,2] = 120
dataku
```


	Nama	NIM	Nilai
0	Mas Iqbal	19/PA/007	90
1	Riifat	20/PI/008	79
2	Zayn	16/PK/923	120
3	Zuko	22/PE/010	30

2) Mengubah nama kolom/baris

Untuk memberi nama (atau label) ke kolom/baris, kita dapat menggunakan method [pd.rename\(...\)](#). Perhatikan bahwa `pd` diubah menjadi nama objek data frame yang dimiliki. Kita dapat menggunakan salah satu dari kode berikut.

```
dataku = dataku.rename(columns = {'Nama' :
'Nama Panggilan'}, index = {0 : 'Mhsw', 1 :
'Mhsw jg'})
```

```
dataku = dataku.rename({'Nama Panggilan' :
'Nama aja deh'}, axis = 1)
```

```
dataku.columns = ['Nama Panggilan', 'NIU',
'Nilai Ujian']
dataku.index = ['A', 'B', 'C', 'D']
```

	Nama Panggilan	NIU	Nilai Ujian
A	Mas Iqbal	19/PA/007	90
B	Riifat	20/PI/008	79
C	Zayn	16/PK/923	120
D	Zuko	22/PE/010	30

Tidak disarankan untuk menggunakan spasi (whitespace) dalam penamaan kolom/baris. Hal ini dikarenakan beberapa fungsionalitas dalam pandas tidak akan berfungsi apabila terdapat whitespace.

3) Mengakses dan mengubah isi sebuah baris/kolom

Untuk mengakses seluruh sel dalam satu kolom, gunakan salah satu dari kode berikut.

```
dataku.NIU
dataku['Nilai Ujian']
dataku.loc[:, 'NIU'] #semua baris di kolom NIU
```

Untuk mengubahnya sama seperti sebelumnya. Apabila kolom yang disebut belum ada dalam dataframe, maka secara otomatis akan dibentuk kolom baru.

```
dataku.NIU = ['22/PE/010', '21/PE/010',
              '22/PE/001', '10/HK/233']
```

```
dataku['Nilai Ujian'] = [(i+1)*10 for i in
                          range(4)]
```

```
dataku['Nilai Tugas'] = [(i+7)*10 for i in
                          range(4)] # kolom baru
```

dataku

	Nama Panggilan	NIU	Nilai Ujian	Nilai Tugas
A	Mas Iqbal	22/PE/010	10	70
B	Riifat	21/PE/010	20	80
C	Zayn	22/PE/001	30	90
D	Zuko	10/HK/233	40	100

Untuk [mengakses seluruh sel dalam satu baris](#), gunakan salah satu dari kode berikut.

```
dataku.loc['A',:] #semua kolom di baris 'A'
dataku.iloc[2,:] #semua kolom di baris ke-3
```

Untuk mengubahnya sama seperti sebelumnya.

```
dataku.loc['A',:] = ['Iqbal', '21/PE/010',
                     90, 100]
```

	Nama Panggilan	NIU	Nilai Ujian	Nilai Tugas
A	Iqbal	21/PE/010	90	100
B	Riifat	21/PE/010	20	80
C	Zayn	22/PE/001	30	90
D	Zuko	10/HK/233	40	100

4) Duplikasi Data Frame

Untuk membentuk duplikat, terdapat hal yang harus diperhatikan.

Case 1: Apabila kita mengubah data dalam `dataku_2`, maka data dalam `dataku` tidak akan berubah (dan sebaliknya).

```
dataku_2 = dataku.copy()
```

Case 2: Apabila diinginkan kedua dataframe yang ikut berubah, kita dapat menambahkan argumen berupa `deep = False`.

```
dataku_3 = dataku.copy(deep = False)
```

Sangat tidak disarankan untuk menggunakan

```
dataku_2 = dataku
```

5) Menghapus kolom/baris

Untuk menghapus kolom/baris, gunakan `pd.drop(...)`. Apabila ingin menghapus beberapa kolom/baris sekaligus, tuliskan nama kolom/baris ke dalam sebuah list.

Nilai `axis = 0` akan menghapus baris, sedangkan `axis = 1` menghapus kolom.

```
dataku2 = dataku2.drop('Nilai Tugas',axis=1)
#axis: 0 untuk baris, 1 untuk kolom
```

```
dataku2 = dataku2.drop('A', axis = 0)
```

```
dataku2 = dataku2.drop(['NIU', 'Nilai Ujian'], axis = 1)
```

dataku2

6) Mengubah tipe data sebuah kolom

Untuk mengubah tipe data sebuah kolom, gunakan method [pd.astype\(...\)](#) seperti pada contoh berikut. Untuk melihat tipe data setiap kolom, gunakan [pd.dtypes](#). Perhatikan bahwa `pd.dtypes` adalah sebuah attribute (bukan method) sehingga tidak perlu menambahkan tanda kurung ().

dataku.dtypes

```
Nama Panggilan    object
NIU               object
Nilai Ujian       int64
Nilai Tugas       int64
dtype: object
```

```
dataku=dataku.astype({'Nilai Ujian':'float'})
```

dataku.dtypes

```
Nama Panggilan    object
NIU               object
Nilai Ujian       float64
Nilai Tugas       int64
dtype: object
```

c. Impor dan Ekspor Data Frame

1) CSV (Comma Separated Values)

Kita bisa mengimpor file CSV dengan menggunakan `pd.read_csv(...)` dan mengekspor file CSV dengan menggunakan `pd.to_csv(...)`

```
mydata3 = pd.read_csv(
    'https://people.sc.fsu.edu/~jburkardt/data/csv/
    v/news_decline.csv', sep = ',')
#CSV dengan separator (pemisah) berupa tanda
koma
```

```
mydata3.tail()
```

	Show	"2009"	"2010"	"2011"
1	48 Hours Mystery	4.1	3.9	3.6
2	20/20	4.1	3.7	3.3
3	Nightline	2.7	2.6	2.7
4	Dateline Friday	4.1	4.1	3.9
5	Dateline Sunday	3.5	3.2	3.1

```
mydata3.to_csv('/Users/t-muhammad.zaki/Desktop/komstat_/local/namafile.csv', index = False)
```

Argumen yang penting untuk diketahui adalah:

- **filepath** : lokasi file CSV, baik lokasi di komputer maupun URL.
Apabila file berada di Google Colab, pilih tombol opsi di sebelah kanan file, lalu *copy path*.
- **sep** : separator atau tanda pemisah antara data dalam CSV, umumnya adalah ',' atau ';'.
- **header** : 0 jika ada nama kolom, None jika tidak ada
- **index** : nama/label baris akan ikut di export jika True

2) XLSX/XLS (Microsoft Excel)

Gunakan `pd.read_excel(...)` untuk mengimpor data dan `pd.to_excel(...)` untuk mengekspor data.

```
mydata =
pd.read_excel('https://www.cmu.edu/blackboard/files/evaluate/tests-example.xls', header = None, sheet_name = 'Example Test')
```

```
mydata.head()
```

	0	1	2	3	4	5	6	7	8
0	MC	What is 2+2?	4	correct	3	incorrect	NaN	NaN	NaN
1	MA	What C datatypes are 8 bits? (assume i386)	int	NaN	float	NaN	double	NaN	char
2	TF	Bagpipes are awesome.	true	NaN	NaN	NaN	NaN	NaN	NaN
3	ESS	How have the original Henry Hornbostel buildin...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	ORD	Rank the following in their order of operation.	Parentheses	Exponents	Division	Addition	NaN	NaN	NaN

Argumen yang digunakan mirip seperti CSV, dengan tambahan argumen penting yaitu **sheet_name** (nama *sheet* yang ingin diambil). Jika tidak dituliskan, maka sheet pertama akan diambil.

3) Table dalam Clipboard

Gunakan `pd.read_clipboard(...)` untuk mengimpor data atau `pd.to_clipboard(...)` untuk mengekspor data.

```
mydata7 = pd.read_clipboard(header = None)
mydata7.head()
```

Argumen yang digunakan mirip seperti CSV.

4) File lainnya (JSON, XML, SAV, dsb)

Silakan untuk mencoba!

```
pd.read_xml(...)
pd.read_spss(...)
pd.read_sql(...)
pd.read_html(...)
pd.read_json(...)
```

Dst :D

d. Common Methods / Attribute

1) [pd.describe\(\)](#)

Ringkasan dataframe untuk setiap kolom

```
dataku.describe()
```

	Nilai Ujian	Nilai Tugas
count	4.000000	4.000000
mean	45.000000	92.500000
std	31.091264	9.574271
min	20.000000	80.000000
25%	27.500000	87.500000
50%	35.000000	95.000000
75%	52.500000	100.000000
max	90.000000	100.000000

2) [pd.sort_values\(\)](#)

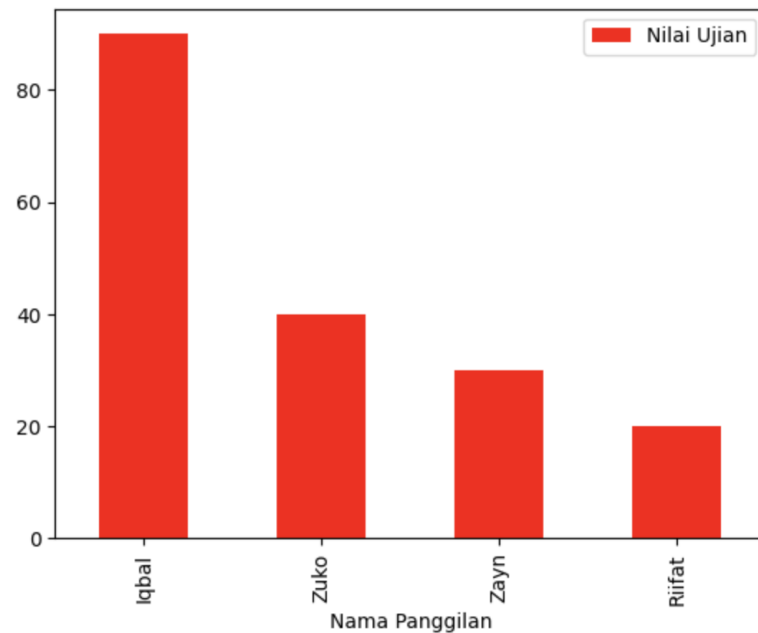
Mengurutkan data berdasarkan kolom tertentu, dengan argumen berupa nama kolom dan ascending (True/False).

3) [pd.plot\(\)](#)

Plot sederhana, tipe yang umum adalah histogram ('hist'), boxplot ('box'), barplot ('bar' atau 'barh'). Terdapat banyak

argumen yang dapat ditambahkan untuk memodifikasi plot yang ditampilkan, seperti title (menambah judul), color (warna), dsb.

```
dataku.sort_values('Nilai Ujian', ascending =  
False).plot(y = 'Nilai Ujian', kind = 'bar',  
color = 'red', x = 'Nama Panggilan')
```



4) [pd.dropna\(\)](#)

Buang missing value. Terdapat beberapa argumen yang dapat digunakan.

- axis : 0 hapus baris yang mengandung missing value, 1 hapus kolom yang mengandung missing value (default: 0)
- thresh : batas minimum banyaknya missing value sebelum baris/kolom dihapus (default = 1)

```
mydata.dropna(axis = 0, thresh = 2)
```

5) [pd.fillna\(\)](#)

Ubah missing value menjadi nilai tertentu. Terdapat beberapa argumen.

- value : nilai pengganti missing value, bisa satu nilai saja atau menggunakan dictionary
- method : 'bfill' mengambil data non-missing selanjutnya sebagai pengganti, 'ffill' mengambil data non-missing sebelumnya sebagai pengganti

```
mydata = mydata.fillna('replacer')
```

	0	1	2	3	4	5	6	7	8
0	MC	What is 2+2??	4	correct	3	incorrect	replacer	replacer	replacer
1	MA	What C datatypes are 8 bits? (assume i386)	int	replacer	float	replacer	double	replacer	char
2	TF	Bagpipes are awesome.	true	replacer	replacer	replacer	replacer	replacer	replacer
3	ESS	How have the original Henry Hornbostel buildin...	replacer	replacer	replacer	replacer	replacer	replacer	replacer
4	ORD	Rank the following in their order of operation.	Parentheses	Exponents	Division	Addition	replacer	replacer	replacer
5	FIB	The student activities fee is	95	dollars for students enrolled in	19	units or more,	replacer	replacer	replacer
6	MAT	Match the lower-case greek letter with its cap...	λ	Λ	α	γ	Γ	φ	Φ

6) [pd.merge\(\)](#) dan [pd.join\(\)](#)

Digunakan untuk menggabungkan dua dataframe menjadi satu kesatuan yang utuh, adil, dan beradab. [pd.merge\(\)](#) menggunakan teknik yang mirip seperti SQL (INNER JOIN, LEFT JOIN, dsb) dengan menambahkan argumen.

```
dataku.merge(dataku2, how = 'outer')
```

```
dataku.join(dataku3)
```

7) [pd.shape](#)

Melihat banyaknya (baris, kolom) dari sebuah dataframe.

```
mydata.shape
```

8) [pd.head\(\)](#) dan [pd.tail\(\)](#)

Melihat beberapa baris pertama dan terakhir dari sebuah dataframe. Kita bisa mengirimkan argumen berupa angka yang menunjukkan banyaknya baris yang ingin ditampilkan.

```
dataku.head(3)
```

```
mydata.tail(5)
```