RINGKASAN PYTHON

1. Variabel
2. Cara menamakan variabe
3. Karakter. Cth: x**=**10
4. Kata. Cth: variable**=**10
5. Menggunakan underscore di antara kata nama variabel. Cth: nilai\_x **=** 15
6. Tidak boleh ada spasi antara nama variable. Cth: nilai x **=** 15
7. Tidak boleh di dahului angka. Cth: 10nilai **=** 15
8. Cara memanggil nilai variabel
9. Memanggil langsug variabel. Cth: **print**(x)
10. Dengan kata. Cth: **print**("nilai x = ",x)
11. Nilai variable
12. Kata atau angka. Cth: x = "kata" / 15
13. Variabel. Cth: x = "kata" / 15

y = x

1. Tipe Data
2. Integer -> Data yang berisi bilangan bulat. Cth: data\_integer **=** 10
3. Float -> Data yang berisi bilangan pecahan atau koma. Cth: data\_float **=** 10.5
4. String -> Data yang berada di dalam tanda petik satu (‘ ‘) atau dua (“ “). Cth: data\_string = "kata"
5. Boolean -> Data yang berisi 2 perbandingan, yaitu True dan False saja. Cth: data\_bool **=** True/False
6. Khusus
7. Bilangan kompleks. Cth: data\_complex **=** complex (5,6)
8. Tipe data dari bahasa C -> Tipe data yang berasal dari bahasa C. Cth: data\_c\_double **=** c\_double (10.5) berasal

**Catatan:**

Proses ini dilakukan ketika memori dari tipe data yang dibutuhkan dalam program yang dibuat tidak cukup. Cara menggunakannya adalah dengan cara mengimport tipe data yang mau kita gunakan dari bahasa C, dengan cara: **from** ctypes **import** namatipedata seperti c\_double, c\_long dll.

**Catatan:**

Untuk mengetahui tipe data dari suatu variabel, kita menggunakan sintaks sbbb.

**print**(type(nama\_variabel))

1. Casting

Casting adalah proses merubah satu tipe data menjadi tipe data yang lain.

1. Casting tipe data ke Integer -> casting\_integer **=** int(nama\_variabel)
2. Casting tipe data ke Float -> casting\_float **=** float(nama\_variabel)
3. Casting tipe data ke String -> casting\_string **=** str(nama\_variabel)
4. Casting tipe data ke Boolean -> casting\_boolean **=** bool(nama\_variabel)

**Catatan:**

Nilai Boolean akan menjadi True jika nilai dari variabel yang bertipe data Integer dan Float buka nol, dan variabel yang bertipe data String tidak kosong(walaupun spasi). Nilai Boolean akan manjadi False jika kondisi dari yang di atas terbalik.

1. Mengambil Input Dari User
2. Input biasa -> input\_biasa**=** input ("kata-kata")
3. Input dengan casting -> input\_casting **=** int/float/bool(input("kata-kata"))

**Catatan:**

Tipe data dari default input adalah String, ketika kita membutuhkan input daru user dengan tipe lain, maka kita harus mengcasting terlebih dahulu inputnya. Ketika ingin memunculkan nilai False pada input bertipe Boolean, maka input harus di casting ke boolean lalu di casting ke integer. Cth: data\_boolean **=** bool(int(input("masukkan data boolean :")))

1. Operator Aritmatika

x **=** 10

y **=** 5

1. Penjumlahan ( + ) -> tambah **=** x **+** y
2. Pengurangan ( - ) -> kurang **=** x **-** y
3. Perkalian ( \* ) -> kali **=** x **\*** y
4. Pembagian ( / ) -> bagi **=** x **/** y
5. Pemangkatan ( \*\* ) -> pangkat **=** x **\*\*** y
6. Modulus (sisa bagi) ( % ) -> modulus **=** x **%** y
7. Floor devision (pembulatan hasil bagi) ( // ) -> floor **=** x **//** y

**Catatan Prioritas Operasi:**

1. ()
2. \*\*
3. \* , / , % , //
4. + , -
5. Operator Komparasi (Perbandingan)

Operator komparasi adalah operasi perbandingan antar dua nilai atau lebih dan hanya memiliki dua nilai yaitu True dan False

x **=** 10

y **=** 5

1. Lebih dari ( > ) -> perbandingan **=** x **>** y
2. Kurang kecil dari ( < ) -> perbandingan **=** x **<** y
3. Lebih dari sama dengan ( >= ) -> perbandingan **=** x **>=** y
4. Kurang dari sama dengan ( <= ) -> perbandingan **=** x **<=** y
5. Sama denga ( == ) -> perbandingan **=** x **==** y
6. Tidak sama dengan ( != ) -> perbandingan **=** x **!=** y
7. Object identity
8. Is -> perbandingan **=** x **is** y
9. Is not -> perbandingan **=** x **is not** y komparasi

**Catatan:**

Object identity hampir sama dengan operator komparasi == dan != , namun object identity melakukan perbandingan terhadap memori dari data yang di bandingkan. Adapun sintaks untuk mengetahui memori suatu data atau variabel sbb.  **print**(hex(id(nama\_variabel))) atau **print**(id(nama\_variabel))

1. Operator Logika

Operator logika adalah operasi perbandingan khusus terhadap nilai boolean yaitu True dan False, dan juga memiliki dua nilai yaitu True dan False.

x **=** True

y **=** False

1. NOT ( not ) -> logika\_not **=** **not** y atau **not** x

Membalikkan nilai True menjadi False dan sebaliknya.

1. OR ( or ) -> logika\_or = x **not** y

Operasi OR akan bernilai True jika salah satu atau keduanya True

1. AND ( and ) -> logika\_and = x **and** y

Operasi AND akan bernilai True jika keduanya True

1. XOR( ^ ) -> logika\_xor = x **^** y

Operasi XOR akan bernilai True jika salah satu True

1. Operator Bitwise

Operatot bitwise adalah operasi yang melakukan operasi logika terhadap binary dari angka, huruf atau kata. Angka, huruf, atau kata akan di rubah dulu menjadi bilangan binary, lalu dilakukan operasi logika terhadap bilangan binary tersebut, yang dimana binary 1 sama dengan True dan binary 0 sama dengan False

x **=** 10

y **=** 5

1. Bitwise OR ( | ) -> bitwise\_or **=** x **|** y
2. Bitwise AND ( & ) -> bitwise\_and **=** x **&** y
3. Bitwise XOR ( ^ ) -> bitwise\_xor **=** x **^** y
4. Bitwise NOT ( ~ ) -> bitwise\_not = **~** y atau **~** x
5. Shifting
6. Shifting Right ( >> ) -> bitwise\_shifting\_right **=** x **>>** 2 atau y **>>** 2

Beroprasi dengan menggeser bilangan binary yang di shifting ke kanan dengan cara menambahkan bilangan binary 0 di awal

1. Shifting Left ( << ) -> bitwise\_shifting\_left **=** x **<<** 1 atau y **<<** 1

Beroprasi dengan menggeser bilangan binary yang di shifting ke kiri dengan cara menambahkan bilangan binary 0 di akhir

**Catatan:**

Cara kerja dari operator bitwise adalah degan merubah nilai variable menjadi bilangan binary terlebih dahulu, adapun sintaks untuk melihat bilangan binary suatu nilai sbb. **print**(format(nama\_variabel,'08b'))

1. Operator Assigment

Operator Assigment adalah operasi yang menyingkat sintaks operator lainnya. Adapun operator yang dapat melakukan operasi assigment sbb.

1. Assigment pada operator aritmatika

x **=** 5

x **+=** 2

x **-=** 2

x **\*=** 2

x **/=** 2

x **%=** 2

x **//=** 2

x **\*\*=** 2

1. Assigment pada operator logika, biswise dan shifting

x **=** True atau False

x **|=** True atau False

x **&=** True atau False

x **^=** True atau False

x **=** 0b0100 bisa 4 atau 8 karakter binary.

x **>>=** 2

x **<<=** 1

1. Pengenalan String
2. Cara membuat
3. Single qoute (Petik satu) -> single **=** 'kata-kata'
4. Double quote (Petik dua) -> double **=** ''kata-kata''
5. Multyline (Banyak baris) ->

multy **=** '''kata-kata baris 1

kata-kata baris 2

kata-kata baris n'''

Contoh pada print -> **print**('kata-kata' atau 'kata-kata' atau '''kata-kata''')

1. Backslash

Tanda Backslash mempunyai beberapa fungsi pada String sbb.

1. Membuat tanda ( ‘ ) jadi String -> **print**('mari shalat jum\'at')
2. Membuat tanda backslash ( \ ) jadi String -> **print**("C:\\user\\python")
3. Membuat tab -> **print**("ini adalah \t tab")
4. Membuat backspace -> **print**("ini adalah \bbackspace")
5. Membuat new line -> **print**("baris pertama. \nbaris kedua.")
6. Raw

Raw adalah sintaks yang hanya bisa membuat tanda ( \ ) menjadi String tanpa adanya Backslash dan membuat fungsi Backslash tidak berfungsi. Cth: **print**(r'C:\newfolder\folder\_saya')

1. Operasi dan Manipulasi String
2. Menyambung String (Concatenate)

Cth:

var\_pertama **=** 'kata-kata'

var\_akhir **=** 'kata-kata'

var\_gabungan **=** var\_pertama **+**' '**+** var\_akhir

**print**(var\_gabungan)

1. Menghitung panjang String (len). Cth: panjang **=** len(nama\_variabel)
2. Operator String
3. Operator biasa
4. Mengecek apakah ada komponen char atau string dalam String (in, not in)

Nilai dari operasi ini adalah True atau False Cth:

nama\_var **=** "kata-kata"

x **=** "carakter atau kata yanga dicari"

status **=** x **in** atau **not in** nama\_var

**print**(status)

1. Mengulang String (angka\*String atau sebaliknya). Cth: **print**(angka**\***'kata-kata')
2. Indexing atau mengambil data dari Sting ( [ n ] ) dimulai dari 0 dan di akhiri n-1

Cth: **print**(nama\_var [0])

1. Mengambil item terkecil/terbesar (min/max).

Cth: **print**(min atau max (nama\_var))

1. Asci code (ord)

Asci code berfungsi untuk mengetahui nilai dari suatu karakter atau sebaliknya

1. Nilai dari suatu karakter. Cth:

ascii\_code **=** ord('s')

**print**(str(ascii\_code))

1. Karakter dari suatu nilai. Cth:

karakter **=** 110

**print**(chr(karakter))

1. Operator dalam bentuk Method
2. Megetahui jumlah karakter dalam data ( .count ). Cth:

data **=** 'kata-kata'

jumlah **=** data.count('a')

**print**(str(jumlah))

1. Mengubah String menjadi Case

nama\_var **=** 'Hello Dunia'

1. Mengubah semua huruf dalam variabel menjadi besar ( . upper ).

Cth: var\_upper **=** nama\_var.upper()

1. Mengubah semua hurur dalam variabel menjadi kecil ( .lower ).

Cth: var\_lower **=** nama\_var.lower()

1. Pengecekan menggunakan is method
2. Mengecek apakah semua huruf dalam variabe huruf besar ( .isupper ).

Cth: var\_isUpper **=** nama\_var.isupper()

1. Mengecek apakah semua huruf dalam variabe huruf kecil ( .islower ).

Cth: var\_isLower **=** nama\_var.islower()

1. Mengecek apakah semua huruf dalam variabe huruf besar ( .istitle ).

Cth: var\_isTitle **=** nama\_var.istitle()

1. Mengecek apakah semua data dalam variabel huruf ( .isalpha )

Cth: var\_isAlpha **=** nama\_var.isalpha()

1. Mengecek apakah semua data dalam variabel angka ( .isdecimal )

Cth: var\_isDecimal **=** nama\_var.isdecimal()

1. Mengecek apakah semua data dalam variabel huruf dan angka ( .isalnum )

Cth: var\_isAlnum **=** nama\_var.isalnum()

1. Mengecek apakah semua data dalam variabel spasi, tab, newline “ \n ” ( .isspace )

Cth: var\_isSpace **=** nama\_var.isspace()

1. Mengecek awalan ( .startswith )

Cth: nama\_var **=** 'ini adalah kata-kata'.startswith('perbandingan')

1. Mengecek akhiran ( .endswith )

Cth: nama\_var **=** 'ini adalah kata-kata'.endswith('perbandingan')

1. Menggabungkan komponen ( .join).

Cth: data **=** ['aku','mau','makan']

nama-var **=** ' '.join(data)

**print**(pisah)

1. Memisahkan komponen ( .split )

Cth: data **=** 'ini adalah kata-kata'

**print**(data.split())

1. Alokasi karakter
2. Rata kanan ( .rjust ).

Cth: nama\_var **=** 'kanan'.rjust(angka,'alokasi')

1. Rata kiri ( .ljust ).

Cth: nama\_var **=** 'kiri'.ljust(angka, 'alokasi')

1. Rata tengah ( .center ).

Cth: nama\_var **=** 'tengah'.center(angka, 'alokasi')

1. Pembatalan alokasi ( .strip ).

Cth: nama\_var **=** 'kanan'.strip('alokasi')

1. Format String
2. Biasa

nama\_var **=** 10

format\_str **=** f"biasa = *{nama\_var}*"

**print**(format\_str)

1. Pada bilangan bulat ( :d )

nama\_var **=** 15

format\_str **=** f"bulat = *{nama\_var:d}*"

**print**(format\_str)

1. Pada bilangan ribuan dan selanjutnya ( :,)

nama\_var **=** 10

format\_str **=** f"ribu = *{nama\_var:,}*"

**print**(format\_str)

1. Memberikan karakter kosong, dengan jumlah yang di tentukan ( :angka)

**print**(f"*{nama\_var:4}* | *{nama\_var:40}* | *{nama\_var:40}* | *{nama\_var:5}*")

1. Membatasi jumlah karakter yang di tampilkan ( :.angka )

**print**(f"*{nama\_var:.4}* | *{nama\_var:.40}* | *{nama\_var:.40}* | *{nama\_var:.5}*")

1. Pada bilangan desimal ( :.angkaf)

nama\_var **=** 2005.54321

format\_str **=** f"desimal = *{nama\_var:.2f}*" *# :.2f untuk menampilkan berapa angka setelah koma yang akan di tampilkan*

**print**(format\_str)

1. Menampilkan leading zero

nama\_var **=** 2005.54321

format\_str **=** f"leading zero = *{nama\_var:7.2f}*" *# :7.2f untuk menampilkan berapa jumlah karakter yang akan di tampilkan, dan berapa angka setelah koma yang akan di tampilkan*

format\_str **=** f"leading zero = *{nama\_var:8.2f}*" *# jika ia melebihi jumlah karakter setelah berapa angka setelah koma di tentukan, maka ia akan menambahkan output kosong di awal*

format\_str **=** f"leading zero = *{nama\_var:08.2f}*" *# jika di tambahkan angka 0 di awalnya maka ia akan menampilkan 0 pada output yang kosong*

**print**(format\_str)

1. Menampilkan tanda “ + “

nama\_var **=** 10

format\_plus **=** f"plus1 = {nama\_var:+d}"

**print**(format\_plus)

1. Memformat persen “ % “

nama\_var **=** 0.045

format\_persen **=** f"persen1 = {nama\_var:%}"

format\_persen **=** f"persen2 = {nama\_var:.2%}" *# :2% untuk menampilkan persen dan berapa banyak angka di belakang persen*

**print**(format\_persen)

1. Melakukan operasi aritmatika dalam placeholder

data1 **=** 10000

data2 **=** 5

format\_string **=** f"harga total = Rp.{data1\*data2:,}"

**print**(format\_string)

1. Format angka binary, octal, dan hexadecimal

nama\_var **=** 225

format\_binary **=** f"binary = {bin(nama\_var)}"

format\_octal **=** f"octal = {oct(nama\_var)}"

format\_hex **=** f"hexadecimal = {hex(nama\_var)}"

**print**(format\_binary)

**print**(format\_octal)

**print**(format\_hex)

1. String Width dan Alignment ( :algiment width)
2. Rata kanan ( > )

nama\_var **=** 225

format\_str **=** f"{nama\_var:>5}"

**print**(format\_str)

1. Rata kiri ( < )

nama\_var **=** 225

format\_str **=** f"{nama\_var:<5}"

**print**(format\_str)

1. Rata tengah ( ^ )

nama\_var **=** 225

format\_str **=** f"{nama\_var:^5}"

**print**(format\_str)

1. If, Else, Elif
2. If -> jika satu kondisi

nama\_var **=** input ('masukkan kondisi :')

**if** nama\_var **==** 'kondisi':

**print**(f'kondisi *{nama\_var}* adalah benar')

1. Else -> jika dua kondisi

nama\_var **=** input ('masukkan kondisi :')

**if** nama\_var **==** 'kondisi':

**print**(f'kondisi *{nama\_var}* adalah benar')

**else:**

**print**(f'kondisi *{nama\_var}* adalah salah')

1. Elif -> jika banyak kondisi

nama\_var **=** input ('masukkan kondisi :')

**if** nama\_var **==** 'kondisi1':

**print**(f'kondisi *{nama\_var}* adalah benar')

**elif** nama\_var **==** 'kondisi2':

**print**(f'kondisi *{nama\_var}* adalah kurang tepat)

**elif** nama\_var **==** 'kondisi n':

**print**(f'kondisi *{nama\_var}* adalah mendekati)

**else:**

**print**(f'kondisi *{nama\_var}* adalah salah')

**Catatan:**

Dalam pembahasan if, else, dan elif ada namanya inline dan indentation. Dalam kasus di atas menggunakan indentation, adapun contoh yang menggunakan inline sbb.

**if** nama\_var **==**'kondisi' : **print**(f'kondisi *{nama\_var}* adalah benar')

1. For Loop
2. Looping dengan List

var\_list **=** [0,1,2,3,4,5]

**print**(var\_list,'\n')

**for** i **in** var\_list:

**print**(f'i sekarang adalah => *{i}*')

**print**('Akhir looping dengan list','\n')

1. Looping dengan Range

var\_range **=** range(5)

var\_range **=** range(1,5) *# 1,5 ==> dimulai dengan angka 1, semua angka sebelum 5*

**for** i **in** var\_range:

**print**(f'i sekarang adalah => *{i}*')

**print**('Akhir looping dengan range ','\n')

1. Looping dengan String

var\_string **=** 'kata-kata'

**for** huruf **in** var\_string:

**print**(huruf)

**print**('Akhir looping dengan string','\n')

1. While Loop

nama\_var **=** 0

**print**(f'Angka sekarang --> *{nama\_var}*')

**while** angka **<** 5:

    nama\_var **+=**1 *# perubahan nilai, Program akan berhenti jika hasilnya False*

**print**(f'Angka sekarang --> *{nama\_var}*')

**print**('Berhenti')

1. Cotinue dan Pass
2. Continue

Cotinue adalah sintaks yang akan membuat loop ke step selanjutnya, semua looping (kondisi) yang berada setelahnya akan di loncat

angka **=** 0

**print**(f'Angka sekarang ==> *{angka}*\n')

**while** angka **<**5:

    angka **+=** 1

**print**(f'Angka sekarang ==> *{angka}*') *# aksi 1*

**if** angka **==**3:

**print**(f'Sekarang angka *{angka}*')

**continue** *# akan membuat loop ke step selanjutnya*

**print**(f'===> *{angka}*') *# aksi 2*

**print**(f'Finish\n')

1. Pass

Pass adalah sintaks berfungsi sebagai dummy, tidak akan dieksekusi

angka **=** 0

**while** angka **<** 5 :

    angka **+=** 1

**if** angka **==** 3:

**pass** *# ini tidak akan di eksekusi*

**print**(angka)

**print**(f'Finish\n')

1. Break

Break adalah sintaks yanag berfungsi untuk menghentikan semua program pada looping.

angka **=** 0

**while** angka **<**5:

    angka **+=** 1

**print**(f'Angka sekarang --> *{angka}*')

**if** angka **==** 3:

**print**(f'Angka yang anda cari adalah: *{angka}*\n')

**break** *# break akan memberhentikan looping*

**print**(f'===> *{angka}*')

**print**('Finish','\n')

1. List

List adalah variabel yang menampung banyak data. Data yang ditampung tidak harus sama.

1. Membuat List biasa

var\_list **=** ['kata-kata',2,True]

**print**(var\_list,'\n')

1. Membuat List dengan Range
2. Membuat List dengan Alternatif List

var\_range **=** range(0,10,2) *# range (start. Stop, step)*

**print**(var\_range,'\n')

var\_list **=** list(var\_range)

**print**(var\_list,'\n')

1. Membuat List dengan For Loop, List comperhention

var\_list\_pakai\_for **=** [i**\*\***2 **for** i **in** range(0,10)]

**print**(var\_list\_pakai\_for,'\n'

1. Membuat List dengan For IfManipulasi List

*# contoh pertama*

**print**(10**\***'=',' List data for if ',10**\***'=')

var\_list\_pakai\_for\_if **=** [i **for** i **in** range(0,10) **if** i **!=**5]

**print**(varlist\_pakai\_for\_if,'\n')

*# contoh kedua*

var\_list\_pakai\_for\_if **=** [i **for** i **in** range(0,10) **if** i**%**2 **==**0]

**print**(var\_list\_pakai\_for\_if,'\n')

1. Manipulasi List
2. Mengambil data List ( index )

data **=** ['value1','value2','value3','value n']

*# data pertama*

data\_0 **=** data [0]

**print**(f"Data pertama (index 0) = *{data\_0}*\n")

*# data terakhir*

data\_terakhir **=** data [**-**1]

**print**(f"Data terakhir (index -1) = *{data\_terakhir}*\n")

1. Mengambil info jumlah data dalam List ( len )

panjang\_data **=** len(data)

**print**(f"Panjang data = *{panjang\_data}*\n")

1. Manipulasi data List
2. Menambah item sesuai posisi ( .insert(posisi, ‘item’) )

**print**(f"Data sebelum di tambah = *{data}*")

data.insert(2,'value')

**print**(f"Data sesudah di tambah = *{data}*\n")

1. Menambah item di akhir ( .append (‘item’))

**print**(f"Data sebelum di tambah = *{data}*")

data.append('value')

**print**(f"Data sesudah di tambah = *{data}*\n")

1. Menambah List dengan List ( .extend (var))

**print**(f"Data sebelum di tambah = *{data}*")

data\_baru **=** ['Rofik','Thoriq','Taufik']

data.extend(data\_baru)

**print**(f"Data sesudah di tambah = *{data}*\n")

1. Merubah data

data[2] **=** 'value'

**print**(f"Data yang sudah di rubah = *{data}*\n")

1. Menghapus data
2. Sesuai keinginan ( .remove(‘item’) )

data.remove('value')

**print**(f"Data yang sudah di hapus = *{data}*\n")

1. Data paling belakang ( .pop )

data.pop()

**print**(f"Data yang sudah di hapus akhirnya = *{data}*\n")

1. Operasi List

data **=** ['value1','value2','value3','value n']

1. Menghitung data ( .count(item) )

var\_jumlah **=** data.count('value2')

**print**(f"Jumlah = *{var\_jumlah}*")

1. Mengambil posisi data ( .index(item) )

var\_index **=** data.index('value2')

**print**(f"Index = *{var\_index}*")

1. Mengurutkan List
2. Dari a – z atau 1 – n ( .sort )

var\_sort **=** data.sort()

**print**(f"Data sort = *{var\_sort}*")

1. Dari z – a atau n – 1 ( .reverse )

var\_reverse **=** data.reverse()

**print**(f"Data reverse = *{var\_reverse}*")

1. Copy List

Copy List adalah salah satu cara menduplikat data pada List. Ketika data asli dirubah nilainya maka data yang sudah di copy tidak akan ikut berubah nilainya

data **=** ['value1','value2','value3','value n']

copy **=** data *# salah*

copy **=** data.copy() *# benar --> full duplikat / data baru*

1. Nested List

Nested List adalah pembahasan menganai List dalam List

data\_0 **=** [1,2]

data\_1 **=** [3,4]

nested\_list\_ **=** [data\_0,data\_1]

**print**(f"Nested list = *{nested\_list}*\n")

1. Contoh penggunaan

peserta\_0 **=** ['Sofiyan',20,'laik-laki']

peserta\_1 **=** ['Thony',19,'laik-laki']

peserta\_2 **=** ['Ramli',19,'laik-laki']

list\_peserta **=** [peserta\_0,peserta\_1,peserta\_2]

**print**(f"Peserta = *{list\_peserta}*\n")

**for** peserta **in** list\_peserta:

**print**(f"Nama \t: *{peserta[0]}*")

**print**(f"Umur \t: *{peserta[1]}*")

**print**(f"Gender \t: *{peserta[2]}*\n")

1. Deep Copy

Deep copy memiki fungsi yang sama dengan copy pada list, namun Deep Copy digunakan untuk menduplikat Nested List. Berikut adalah sintaksnya.

data\_0 **=** [1,2]

data\_1 **=** [3,4]

nested\_list\_ **=** [data\_0,data\_1,25]

copy **=** nested\_list.copy() *# salah --> ketika menduplikat nested list*

**from** copy **import** deepcopy *# harus di import*

deepcopy **=** deepcopy(nested\_list) *# benar*

1. Looping List

data **=** [3,4,7,2,8,4]

1. For Loop

**for** angka **in** data:

**print**(f"Angka = *{angka}*")

1. For dan Range

panjang **=** len(data)

**for** i **in** range(panjang):

**print**(f"Angka = *{data[i]}*")

1. While

panjang **=** len(data)

i **=** 0

**while** i **<** panjang:

**print**(f"Angka = *{data[i]}*")

    i **+=** 1

1. List Comperhention

*# contoh 1 --> biasa*

[**print**(i) **for** i **in** data]

*# contoh 2 --> loop + operasi*

angka\_kuadrat **=** [i**\*\***2 **for** i **in** data]

**print**(f"Angka kuadrat = *{angka\_kuadrat}*")

1. Enumerate

**for** index,value **in** enumerate(data):

**print**(f"index = *{index}* , data = *{value}*")

1. Tuples dan Set
2. Tuples

Tuples adalah variabel yang sama dengan List, yaitu menampung banyak data, data dalam Tuples hanya bisa di print, namun tidak bisa melakukan operasi seperti pada List, seperti menambah, menghapus dll.

data\_tuples **=** (1,4.5,7,8,9)

**print**(data\_tuples)

**print**(data\_tuples[3])

1. Set

Set adalah variabel yang sama dengan List, yaitu menampung banyak data, data dalam Set hanya bisa melakukan operasi seperti pada List, seperti menambah, menghapus dll, namun data dalam Set tidak bisa di print hanya bisa diprint secara keseluruhan.

data\_set **=** {2,3,5,9,7,8}

**print**(data\_set)

data\_set.add(10)

**print**(data\_set)

data\_set.pop()

**print**(data\_set)

1. Dictionary

Dictionary adalah variabel yang mempunya key dan value. Dictionary sama dengan List, namun List di akses berdasarkan indexnya sedangkan Dictionary diakses berdasarkan key. Contoh sintaksnya sbb.

data\_list **=** ["Anggur",'Apel','Strawbery']

data\_dict **=** {

    'key':'value',

    'spt':'Sepatu',

    'th':'Thony',

    'rm':'Ramli',

    'nmbr':100,

    'list':data\_list,

}

**print**(data\_dict)

**print**(data\_dict['nmbr'])

1. Operasi Dictionary

data\_dict **=** {

    'key':'value',

    'spt':'Sepatu',

    'th':'Thony',

    'ky':'Value',

    'nmbr':100,

}

1. Mencari panjang Dictionary ( len(nama\_var )

Panjang\_dict **=** len(data\_dict)

**print**(f"Panjang dari dict : *{panjang\_dict}*")

1. Mengecek key ada atau tidak ( in )

key **=** "spt"

cek\_key **=** key **in** data\_dict

**print**(f"Apakah *{key}* ada di data\_dict : *{cek\_key}*")

1. Mengakses data/nilai
2. Biasa

**print**(data\_dict['spt'])

1. Dengan method ( .get )

**print**(data\_dict.get('spt')) *# untuk mengetahui apakah dia dict atau tidak*

**print**(data\_dict.get('so')) *# apabila key tidak ada, maka akan keluar tampilan "None"*

**print**(data\_dict.get('sl','Key tidak ada di dalam program')) *#menentukan masssage sesuai keinginan apabila key tidak ada*

1. Mengubah data Dictionary

data\_dict['ky'] **=** "Kholis"

**print**(data\_dict)

1. Menambah data Dictionary

data\_dict['rm'] **=** 'Ramli'

**print**(data\_dict)

1. Mengupdate data dictionary

data\_dict.update({'spt':'Sepatu'}) *# kalau sudah ada di dalam datanya,maka akan di hiraukan*

data\_dict.update({'kh':'Kholik'}) *# kalau belum ada di dalam datanya, maka akan langsung di tambahkan*

**print**(data\_dict)

1. Mendelete data Dictionary

**del** data\_dict['nmbr']

**print**(data\_dict)

1. Looping Dictionary

teman\_teman **=** {

    'sf':'Sofiyan',

    'th':'Thony',

    'rm':'Ramli',

    'kh':'Kholis',

}

1. Biasa

**for** teman **in** teman\_teman:

**print**(teman)

1. Menggunakan keys

keys **=** teman\_teman.keys()

**print**(keys)

**for** key **in** teman\_teman.keys():

**print**(teman\_teman.get(key))

1. Menggunakan value

values **=** teman\_teman.values()

**print**(values)

**for** value **in** teman\_teman.values():

**print**(value)

1. Menggunakan item

item **=** teman\_teman.items()

**print**(item)

*# cara pertama*

**for** item **in** teman\_teman.items():

**print**(item)

*# cara kedua*

**for** key,value **in** teman\_teman.items():

**print**(f"Key = *{key}*   Value = *{value}*")

1. Copy dan Pop Dictionary

teman\_teman **=** {

    'sf':'Sofiyan',

    'th':'Thony',

    'rm':'Ramli',

    'kh':'Kholis',

}

1. Copy ( .copy )

Cara menduplikat data yang ada dalam variabel Dictionary sama dengan cara menduplikat data pada List

friends **=** teman\_teman.copy()

**print**('Teman-teman = ',teman\_teman)

**print**('Friends = ',friends)

1. Pop
2. Berdasarkan item ( .pop(value) )

Data\_pop **=** friends.pop('sf') *# datanya akan di terasnfer ke sini*

**print**('DataSofiyan = ',data\_pop)

**print**('Friends = ',friends)

1. Data terakhir ( .popitem )

dataterakhir **=** friends.popitem()

**print**('Dataterakhir = ',dataterakhir)

**print**('Friends = ',friends)

1. Multy keys, Nested Dictionary dan Perulangan Nested Dictionary
2. Multy keys dan Nested Dictionary

**import** datetime

mahasiswa1 **=** {

    'nama':'Muh. Sofiyan',

    'nim':140,

    'beasiswa':False,

    'lahir':datetime.datetime(2002, 1, 15),

}

mahasiswa1 **=** {

    'nama':'M. Taufik Hidayat Ramdan',

    'nim':135,

    'beasiswa':True,

    'lahir':datetime.datetime(2002, 3, 20),

}

data\_mahasiswa **=** {

    'key1':mahasiswa1,

    'key2':mahasiswa2,

}

**print**(data\_mahasiswa,'\n')

1. Perulangan pada Nested Dictionary

**for** mahasiswa **in** data\_mahasiswa:

    KEY **=** mahasiswa

    NAMA **=** data\_mahasiswa[KEY] ['nama']

    NIM **=** data\_mahasiswa [KEY] ['nim']

    BEASISWA **=** data\_mahasiswa [KEY] ['beasiswa']

    LAHIR **=** data\_mahasiswa[KEY] ['lahir'].strftime('*%x*')

**print**(f"{KEY:<6} {NAMA:<17} {SKS:<3} {BEASISWA:^8} {LAHIR:<10}")

1. Pengenalan Fungsi

**def** hello\_world():

    '''fungsi menampilkan hello world'''

**print**("Hello world")

hello\_world()

hello\_world()

1. Fungsi Dengan Argument
2. Argument biasa

**def** hello\_world(**nama**):

    ''' fungsi hello world, menerima input dengan variable nama'''

**print**(f"Selamat datang di dunia wahai *{nama}*")

hello\_world("Faqih")

1. Multiple argument

**def** tambah(**a**,**b**):

    '''Fungsi tambah '''

    hasil **=** a **+** b

**print**(f"*{a}* + *{b}* = *{hasil}*")

tambah(1,5)

tambah(1000,1)

1. Argument dengan List

**def** say\_hi(**list\_peserta**):

    data\_peserta **=** list\_peserta.copy()

**for** peserta **in** data\_peserta:

**print**(f"Yang terhormat *{peserta}*")

*# di sini tempat listnya*

anggota **=** ['Sofiyan', 'Thony', 'Ramli']

say\_hi(anggota)

1. Fungsi Dengan Return
2. Return biasa

**def** kuadrat(**input\_angka**):

    '''fungsi kuadrat'''

    hasil\_kuadrat **=** input\_angka **\*\*** 2

**return** hasil\_kuadrat

**print**(kuadrat (6))

1. Return banyak

**def** operasi\_matematika(**angka\_1**, **angka\_2**):

    kali **=** angka\_1 **\*** angka\_2

    bagi **=** angka\_1 **/** angka\_2

    tambah **=** angka\_1 **+** angka\_2

    kurang **=** angka\_1 **-** angka\_2

**return** kali, bagi, tambah, kurang

**print**(operasi\_matematika (9,5))

k,l,m,n **=** operasi\_matematika (9,5)

**print**(f"Hasil kali   = *{k}*")

**print**(f"Hasil bagi   = *{l}*")

**print**(f"Hasil tambah   = *{m}*")

**print**(f"Hasil kurang   = *{n}*")

1. Default Argument

*# contoh 1*

**def** say\_hello(**nama** **=** "Sofiyan"):

**print**(f"Hallo *{nama}*")

say\_hello() *# ketika isi argument tidak di isi, maka akan menghasilkan default argumennya (jika dia default argument)*

say\_hello ("Thony") *# ketika argument nya di isi, maka akan menghasilkan apa yang di isi*

*# contoh 2*

**def** hitung\_pangkat(**angka**, **pangkat**):

    hasil **=** angka **\*\*** pangkat

**return** hasil

**print**(hitung\_pangkat (2,4))

**print**(hitung\_pangkat (**pangkat** **=** 5, **angka** **=** 4)) *# kita bisa akses nilai berdasarkan nama argumentnya, dan tidak harus berurutan. cara ini berguna jika kita punya banyak argument. liaht contoh 3*

1. Type Hints Pada Fungsi

Type Hints berfungsi untuk memberitahu tipe dari argumen maupun keluaran pada Fungsi

**import** string

**def** sepuluhPangkat(argument:**int**) -> int:

    '''FUNGSI DENGAN HINTS'''

**output** **=** 10 **\*\*** argument

    return **output**

**hasil** **=** sepuluhPangkat (2)

**print**(**hasil**)

1. \*Args Pada Fungsi

\*Args pada fungsi adalah argument yang bisa menampung banyak nilai, dengan \*Args kita bisa mempersimple sintax pada fungsi saat membutuhkan banyak argumen, atau saat kita membutuhkan argumen yang tidak terbatas.

**def** fungsi3(\***args**):*# \*args tidak harus namanya "args", namun bisa juga dengan kata yang lain*

    nama **=** args[0]

    tinggi **=** args[1]

    berat **=** args[2]

**print**(f"*{nama}* punya tinggi *{tinggi}* dan berat *{berat}*")

fungsi3("Dudung", 156, 50)

*# studi kasus*

**def** tambah(\***data**):

*# data tipenya adalah tuple, dia bisa diiterasi*

    output **=** 0

**for** angka **in** data:

        output **+=** angka

**return** output

hasil **=** tambah(1,2,3,4,5,6,7,8,9)

**print**(f"Hasil *{hasil}*")

hasil **=** tambah(10,5,15)

**print**(f"Hasil *{hasil}*")

1. \*\*Kwargs Pada Fungsi
2. Anonimous dan Lamda Function
3. Global dan Local Scope
4. Import Statement
5. Membuat Module
6. Membuat Package
7. \_\_init.py Pada Package
8. Menggunakan Standar Library
9. Tkinter Standar Library Python GUI
10. Mengenal PIP
11. Package Numpy
12. Explore Pygame
13. \_\_main Sebagai Gerbang Program.py
14. Read External File – Open dan With
15. Write External File
16. Exception, Error, Try dan Except
17. ajfgajsdgfajs