(1) Syntax dasar, Case Sensitivity, Komentar pada Python

A. Syntax dasar

print() merupakan salah satu fungsi dari python untuk mencetak, dengan meletakkan kurung buka dan kurung tutup

```
print("Hello World") #menggunakan tanda petik dua
print('Hello World') #menggunakan tanda petik tunggal

Hello World
Hello World
```

diatas menggunakan tanda `""` (dibaca: tanda petik dua) atau `''` (dibaca: tanda petik tunggal), diikuti dengan string ataupun variable, dari script yang dijalankan dapat dilihat output berupa text `Hello World`

B. Case Sensitivity

bahasa pemrograman python bersifat case sensitive, yang artinya huruf besar dan huruf kecil memiliki perbedaan. sebagai contoh seperti pada contoh program di atas,menggunakan `print()` akan langsung menampilkan output nya, selanjutnya jika menggunakan `Print()`, `PRINT()`,`PrInT()` atau fungsi tidak lengkap seperti `prnt()` akan muncul pesan error seperti eksekusi program dibawah

```
Print("Hello World") #menggunakan Print()
---------------------------------------------------------------------------

NameError: name 'Print' is not defined


PRINT("Hello World") #menggunakan PRINT()
---------------------------------------------------------------------------

NameError: name 'PRINT' is not defined


PrInT("Hello World") #menggunakan PrInT()
---------------------------------------------------------------------------

NameError: name 'PrInT' is not defined

prnt("Hello World") #menggunakan prnt()
---------------------------------------------------------------------------

NameError: name 'prnt' is not defined
```

NOTE: perlu diperhatikan, case sensitive juga berlaku untuk function lainnya.

C. Komentar pada Python

komentar pada python, di tandai menggunakan # yang artinya kode tersebut tidak dieksekusi atau tidak dijalankan oleh mesin. Komentar hanya digunakan untuk menandai atau memberikan keterangan tertulis pada script.

Manfaat dari komentar tersebut, dapat memberikan keterangan mengenai script, code agar orang lain dapat memahami isi dari program anda.

Berikut contoh script yang menggunakan komentar pada python

```
# ini komentar menggunakan tanda '#' yang tidak dapat dieksekusi
#Baris satu (1)
#Baris dua  (2)

'''
Ini adalah komentar yang berisikan penjelasan lebih
satu baru yaitu dengan menggunakan tanda petik satu ''
'''

"""
Ini contoh komentar menggunakan
tanda kutip dua ""
"""

print("Hello World") #output text/string
#print('Hello World')

# Menggunakan Spesial karakter !@#$%^&*(),./;'[]\ pada komentar

#mencetak nama anda
print("Luffy")

#mencetak angka, nilai
print(12)
print("ini adalah nilai 12") #sebagai string


Hello World
Luffy
12
ini adalah nilai 12
```

ketika menjalankan script di atas, dapat dilihat output dari program `Hello World Luffy 12 ini adalah nilai 12` dan komentar tidak dieksekusi

(2). Tipe Data

Tipe data adalah suatu media atau memori pada komputer yang digunakan untuk menampung informasi. Python sendiri mempunyai tipe data yang cukup unik bila kita bandingkan dengan bahasa pemrograman yang lain. Berikut adalah tipe data dari bahasa pemrograman Python :

Tipe Data	Contoh	Penjelasan
Boolean	`True` atau `False`	Menyatakan benar `True` yang bernilai `1`, atau salah `False` yang bernilai `0`
String	`"Ayo belajar Python"`	Menyatakan karakter/kalimat bisa berupa huruf angka, dll (diapit tanda " atau ')
Integer	`25` atau `1209`	Menyatakan bilangan bulat
Float	`3.14;.4e7;.2;4.2e-4`	Menyatakan bilangan yang mempunyai koma
Binary	`0b10`	Menyatakan bilangan dalam format binary / biner (bilangan berbasis 2)
Octal	`0o10`	Menyatakan bilangan dalam format oktal (bilangan berbasis 8)
Hexadecimal	`0x10`	Menyatakan bilangan dalam format heksa (bilangan berbasis 16)
Complex	`1 + 5j`	Menyatakan pasangan angka real dan imajiner
List	`['xyz', 786, 2.23]`	Data untaian yang menyimpan berbagai tipe data dan isinya bisa diubah-ubah
Tuple	`('xyz', 768, 2.23)`	Data untaian yang menyimpan berbagai tipe data tapi isinya tidak bisa diubah
Dictionary	`{'nama': 'budi','id':2}`	Data untaian yang menyimpan berbagai tipe data berupa pasangan penunjuk dan nilai

dalam hal menjelaskan beberapa karakter pada string, dijelaskan sebagai berikut:

Escape Sequence	Penjelasan
`\'`	Literal single quote (') character
`\"`	Literal double quote (") character
`\n`	ASCII Linefeed (LF) character
`\\`	Literal backslash () character
`\b`	ASCII Backspace (BS) character
`\t`	ASCII Horizontal Tab (TAB) character
`\r`	ASCII Carriage Return (CR) character

Untuk mencoba berbagai macam tipe data, silahkan coba script Python dibawah ini.

```
#tipe data Boolean
print(True)

#tipe data String
print("string dengan menggunakan tanda kutip dua")
print('ini string menggunakan tanda kutip satu')
#tipe data string dengan menjelaskan spesial karakter atau escape sequences
print('ini adalah tanda single quote (\')')
print("ini adalah tanda double quote (\")")
print("ini adalah tanda slash (\\)")
print("Algoritma\nPemrograman") #menggunakan \n
print("Algoritma\bPemrograman") #menggunakan \b
print("Algoritma\tPemrograman") #menggunakan \t
print("Algoritma\rPemrograman") #menggunakan \b

#tipe data Integer
print(20)

#tipe data Float
print(3.14)
print(.2)
print(4.2e-3)

#tipe data Binary
print(0b10)

#tipe data octal
print(0o10)

#tipe data Hexadecimal
```

```
print('tipe data heksa desimal',0x10)

#tipe data Complex
print(5j)

#tipe data List
print([1,2,3,4,5])
print(["satu", "dua", "tiga"])

#tipe data Tuple
print((1,2,3,4,5))
print(("satu", "dua", "tiga"))

#tipe data Dictionary
print({"nama":"Budi", 'umur':20})
#tipe data Dictionary dimasukan ke dalam variabel biodata
biodata = {"nama":"Budi", 'umur':21} #proses inisialisasi variabel biodata
print(biodata) #proses pencetakan variabel biodata yang berisi tipe data Dictionary
type(biodata) #fungsi untuk mengecek jenis tipe data. akan tampil <class 'dict'> yang berarti
dict adalah tipe data dictionary
```

```
True
string dengan menggunakan tanda kutip dua
ini string menggunakan tanda kutip satu
ini adalah tanda single quote (')
ini adalah tanda double quote (")
ini adalah tanda slash (\)
Algoritma
Pemrograman
AlgoritmaPemrograman
Algoritma Pemrograman
Pemrograman
20
3.14
0.2
0.0042
2
8
tipe data heksa desimal 16
5j
[1, 2, 3, 4, 5]
['satu', 'dua', 'tiga']
(1, 2, 3, 4, 5)
('satu', 'dua', 'tiga')
{'nama': 'Budi', 'umur': 20}
{'nama': 'Budi', 'umur': 21}
```

(3). Variabel

Variabel adalah suatu pengenal (identifier) yang digunakan untuk mewakili suatu nilai tertentu di dalam proses program. Berbeda dengan konstanta yang nilainya selalu tetap, nilai dari suatu variable bisa diubah-ubah sesuai kebutuhan.

Variabel secara umumnya dapat menyimpan berbagai macam tipe data. di dalam Python, variabel bersifat dinamis, yang artinya python tidak perlu dideklarasikan tipe data tertentu dan variabel python dapat diuaah saat program dijalankan.

nama dari suatu variabel dapat ditentukan sendiri oleh pemrogram dengan aturan sebagai berikut:

1. Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf. Bahasa python bersifat case-sensitive artinya huruf besar dan kecil dianggap berbeda. Jadi antara nama,Nama dan NaMa dianggap berbeda.
2. Karakter pertama harus berupa huruf atau garis bawah atau underscore _ karakter selanjutnya dapat berupa huruf, garis bawah atau angka

dalam penulisan variabel pada python, cukup menuliskan variabel lalu mengisikannya dengan nilai ditambahkan dengan tanda sama dengan = diikuti dengan nilai untuk variabel tersebut.

Berikut penggunaan variabel dalam bahasa pemrograman python

```
#memasukkan data dalam sebuah variabel
name = "Budi Bae" #isi variabel berupa string
print(name) #mencetak variabel

#nilai dan tipe data dalam variabel
age = 21 #tipe data angka / numeric
print(age) #mencetak nilai age
print(type(age)) #melihat tipe data dari age
age = "dua puluh satu" #tipe data string
print(age) #mencetak string dari age
print(type(age)) #melihat tipe data

first_name = "Monkey"
middle_name = "D."
last_name = "Luffy"
name = first_name+" "+middle_name+" "+last_name
age = 19
hobby = "Makan"
print("Profil\n",name,"\n",age,"\n",hobby)

#contoh variabel lainnya
age = 1
Age = 2
aGe = 3
AGE = 4
a_g_e = 5
_age = 6
age_ = 7
_AGE_ = 8
print(age, Age, aGe, AGE, a_g_e, _age, age_, _AGE_) #mencetak variabel

numberofcollegegraduates = 2500
NUMBEROFCOLLEGEGRADUATES = 2500
numberOfCollegeGraduates = 2500
NumberOfCollegeGraduates = 2500
number_of_college_graduates = 2500
#mencetak variabel
```

```
print(numberofcollegegraduates, NUMBEROFCOLLEGEGRADUATES,numberOfCollegeGraduates,
NumberOfCollegeGraduates, number_of_college_graduates)
```

```
Budi Bae
21
<class 'int'>
dua puluh satu
<class 'str'>
Profil
 Monkey D. Luffy
 19
 Makan
1 2 3 4 5 6 7 8
2500 2500 2500 2500 2500
```

(4). Operator

Operator adalah symbol yang biasa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi, sebagai contoh yang dapat diimplementasikan menggunakan operator, 5 + 10 = 15, Dimana `5` dan `10` adalah operan dan `+` adalah operator.

Bahasa pemrograman Python mendukung berbagai macam operator, diantaranya :

- Operator Aritmatika (Arithmetic Operators)
- Operator Perbandingan (Comparison (Relational) Operators)
- Operator Penugasan (Assignment Operators)
- Operator Logika (Logical Operators)
- Operator Bitwise (Bitwise Operators)
- Operator Keanggotaan (Membership Operators)
- Operator Identitas (Identity Operators)

Operator Aritmatika

Operator	Contoh	Penjelasan
Penjumlahan +	`1 + 3 = 4`	Menjumlahkan nilai dari masing-masing operan atau bilangan
Pengurangan –	`4 - 1 = 3`	Mengurangi nilai operan di sebelah kiri menggunakan operan di sebelah kanan
Perkalian *	`2 * 4 = 8`	Mengalikan operan/bilangan
Pembagian /	`10 / 5 = 2`	Untuk membagi operan di sebelah kiri menggunakan operan di sebelah kanan
Sisa Bagi %	`11 % 2 = 1`	Mendapatkan sisa pembagian dari operan di sebelah kiri operator ketika dibagi oleh operan di sebelah kanan
Pangkat **	`8 ** 2 = 64`	Memangkatkan operan disebelah kiri operator dengan operan di sebelah kanan operator
Pembagian Bulat //	`10 // 3 = 3`	Sama seperti pembagian. Hanya saja angka dibelakang koma dihilangkan

Operator Perbandingan

Operator perbandingan (comparison operators) digunakan untuk membandingkan suatu nilai dari masing-masing operan.

Operator	Contoh	Penjelasan
Sama dengan ==	`1 == 1`	bernilai True Jika masing-masing operan memiliki nilai yang sama, maka kondisi bernilai benar atau True.
Tidak sama dengan !=	`2 != 2`	bernilai False Akan menghasilkan nilai kebalikan dari kondisi sebenarnya.
Tidak sama dengan <>	`2 <> 2`	bernilai False Akan menghasilkan nilai kebalikan dari kondisi sebenarnya.
Lebih besar dari >	`5 > 3`	bernilai True Jika nilai operan kiri lebih besar dari nilai operan kanan, maka kondisi menjadi benar.
Lebih kecil dari <	`5 < 3`	bernilai True Jika nilai operan kiri lebih kecil dari nilai operan kanan, maka kondisi menjadi benar.
Lebih besar atau sama dengan >=	`5 >= 3`	bernilai True Jika nilai operan kiri lebih besar dari nilai operan kanan, atau sama, maka kondisi menjadi benar.
Lebih kecil atau sama dengan <=	`5 <= 3`	bernilai True Jika nilai operan kiri lebih kecil dari nilai operan kanan, atau sama, maka kondisi menjadi benar.

Operator Penugasan

Operator penugasan digunakan untuk memberikan atau memodifikasi nilai ke dalam sebuah variabel.

Operator	Contoh	Penjelasan
Sama dengan =	`a = 1`	Memberikan nilai di kanan ke dalam variabel yang berada di sebelah kiri.
Tambah sama dengan +=	`a += 2`	Memberikan nilai variabel dengan nilai variabel itu sendiri ditambah dengan nilai di sebelah kanan.
Kurang sama dengan -=	`a -= 2`	Memberikan nilai variabel dengan nilai variabel itu sendiri dikurangi dengan nilai di sebelah kanan.
Kali sama dengan *=	`a *= 2`	Memberikan nilai variabel dengan nilai variabel itu sendiri dikali dengan nilai di sebelah kanan.
Bagi sama dengan /=	`a /= 4`	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan.
Sisa bagi sama dengan %=	`a %= 3`	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan. Yang diambil nantinya adalah sisa baginya.
Pangkat sama dengan **=	`a **= 3`	Memberikan nilai variabel dengan nilai variabel itu sendiri dipangkatkan dengan nilai di sebelah kanan.
Pembagian bulat sama dengan //=	`a //= 3`	Membagi bulat operan sebelah kiri operator dengan operan sebelah kanan operator kemudian hasilnya diisikan ke operan sebelah kiri.

Operator Logika

Berikut adalah penjelasan mengenai operator logika (Logical Operator)

Operator	Contoh	Penjelasan
and	a, b = True, True `# hasil akan True` print(a and b)	Memberikan nilai di kanan ke dalam variabel yang berada di sebelah kiri.

or	a, b = True, False
	`# hasil akan True`
	print(a or b)
	print(b or a)
	print(a or a)
	`# hasil akan False`
	print(b or b)
		Jika salah satu atau kedua operan bernilai True maka kondisi akan bernilai True. Jika keduanya False maka kondisi akan bernilai False.

not	a, b = True, False `# hasil akan True` print(not b)	Membalikkan nilai kebeneran pada operan misal jika asalnya True akan menjadi False dan begitupun sebaliknya.

Operator Bitwise (Bitwise Operators)

Berikut adalah penjelasan mengenai Operator Bitwise (Bitwise Operators)

Operator	Contoh	Penjelasan

&	a, b = 13, 37
	`# a akan bernilai '0000 1101'`
	`# b akan bernilai '0010 0101'`
	c = a & b
	`# c akan bernilai 5 = '0000 0101'`
	print(c)
		Operator biner AND, memeriksa apakah operan di sebelah kiri dan operan sebelah kanan mempunyai angka biner 1 di setiap bit. Jika keduanya bernilai 1 maka bit hasil operasi akan bernilai 1.

|	a, b = 13, 37 `# a akan bernilai '0000 1101'`	Operator biner OR, memeriksa apakah operan

	# b akan bernilai '0010 0101'
	c = a | b
	# c akan bernilai 45 = '0010 1101'
	print(c)
		di sebelah kiri dan operan sebelah kanan mempunyai angka biner 1 di setiap bit. Jika salah satunya bernilai 1 maka bit hasil operasi akan bernilai 1.

^	a, b = 13, 37
	# a akan bernilai '0000 1101'
	# b akan bernilai '0010 0101'
	c = a ^ b
	# c akan bernilai 40 = '0010 1000'
	print(c)
		Operator biner Negative, membalik nilai bit. Misal dari 1 menjadi 0, dari 0 menjadi 1.

Kali sama dengan *=	a *= 2	Operator biner OR, memeriksa apakah operan di sebelah kiri dan operan sebelah kanan mempunyai angka biner 1 di setiap bit. Jika salah .
~	a, b = 13, 37 # a akan bernilai '0000 1101' # b akan bernilai '0010 0101'	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan.

<<	a, b = 13, 37
	# a akan bernilai '0000 1101'
	# b akan bernilai '0010 0101'
	# hasil bernilai 52 = ''0011 0100'
	print(a << 2)
	# hasil bernilai 148 = '1001 0100'
	print(b << 2)
		Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan. Yang diambil nantinya adalah sisa baginya .

>>	a, b = 13, 37
	# a akan bernilai '0000 1101'
	# b akan bernilai '0010 0101'
	# hasil bernilai 3 = '0000 0011'
	print(a >> 2)
	# hasil bernilai 9 = '0000 1001'
	print(b >> 2)
		Operator penggeser biner ke kiri, deret bit akan digeser ke kiri sebanyak n kali.

Contoh penjelasan :

Bitwise AND operator

```
a = 10 = 1010 (Binary)
b = 4 =  0100 (Binary

a & b = 1010
          &
        0100
      = 0000
      = 0 (Decimal)
```

Bitwise or operator

```
a = 10 = 1010 (Binary)
b = 4 =  0100 (Binary

a | b = 1010
          |
        0100
      = 1110
      = 14 (Decimal)
```

Bitwise not operator

```
a = 10 = 1010 (Binary)

~a = ~1010
   = -(1010 + 1)
   = -(1011)
   = -11 (Decimal)
```

Bitwise xor operator

```
a = 10 = 1010 (Binary)
b = 4 =  0100 (Binary

a ^ b = 1010
         ^
        0100
      = 1110
      = 14 (Decimal)
```

Operator Keanggotaan (Membership Operators)

Berikut adalah penjelasan mengenai operator logika (Logical Operator)

Operator	Contoh	Penjelasan
in	sebuah_list = [1, 2, 3,4 ,5] print(5 in sebuah_list)	Memeriksa apakah nilai yang dicari berada pada list atau struktur data python lainnya. Jika nilai tersebut ada maka kondisi akan bernilai True.
not in	sebuah_list = [1, 2, 3,4 ,5] print(10 not in sebuah_list)	Memeriksa apakah nilai yang dicari tidak ada pada list atau struktur data python lainnya. Jika nilai tersebut tidak ada maka kondisi akan bernilai True.

Operator Identitas (Identity Operators)

Berikut adalah penjelasan mengenai Operator Identitas (Identity Operators)

Operator	Contoh	Penjelasan

is	a, b = 10, 10
	# hasil akan True
	print(a is b)
		Memeriksa apakah nilai di sebelah kiri operan memiliki identitas memori yang sama dengan nilai di sebelah kanan operan. Jika sama maka kondisi bernilai True.

is not	a, b = 10, 5
	# hasil akan True
	print(a is not b)
		Memeriksa apakah nilai di sebelah kiri operan memiliki identitas memori yang berbeda dengan nilai di sebelah kanan operan. Jika berbeda maka kondisi bernilai True.

Prioritas Eksekusi Operator di Python

Dari semua operator diatas, masing-masing mempunyai urutan prioritas yang nantinya prioritas pertama akan dilakukan paling pertama, begitu seterusnya sampai dengan prioritas terakhir.

Operator	Keterangan
**	Aritmatika
~, +, -	Bitwise
*, /, %, //	Aritmatika
+, -	Aritmatika
>>, <<	Bitwise

&	Bitwise
^,|	Bitwise
<=, <, >, >=	Perbandingan
<> , ==, !=	Perbandingan
=, %=, /=, //=, -=, +=, *=, **=	Penugasan
is, is not	Identitas
in, not in	Membership (Keanggotaan)
not, or, and	Logika

```
#OPERATOR ARITMATIKA
#Penjumlahan
print(13 + 2)
mangga = 7
pisang = 9
buah = mangga + pisang
print(buah)

#Pengurangan
hutang = 10000
bayar = 5000
sisaHutang = hutang - bayar
print("Sisa hutang Anda adalah ", sisaHutang)

#Perkalian
panjang = 15
lebar = 8
luas = panjang * lebar
print(luas)

#Pembagian
kue = 16
anak = 4
kuePerAnak = kue / anak
print("Setiap anak akan mendapatkan bagian kue sebanyak ", kuePerAnak)

#Sisa Bagi / Modulus
bilangan1 = 14
bilangan2 = 5
hasil = bilangan1 % bilangan2
print("Sisa bagi dari bilangan ", bilangan1, " dan ", bilangan2, " adalah
", hasil)

#Pangkat
bilangan3 = 8
bilangan4 = 2
hasilPangkat = bilangan3 ** bilangan4
print(hasilPangkat)

#Pembagian Bulat
print(10//3)
#10 dibagi 3 adalah 3.3333. Karena dibulatkan maka akan menghasilkan nilai
3
15
16
Sisa hutang Anda adalah  5000
120
Setiap anak akan mendapatkan bagian kue sebanyak  4.0
Sisa bagi dari bilangan  14  dan  5  adalah  4
64
3
```

(5). Kondisi

A. Kondisi If

Pengambilan keputusan (kondisi if) digunakan untuk mengantisipasi kondisi yang terjadi saat jalanya program dan menentukan tindakan apa yang akan diambil sesuai dengan kondisi.

Pada python ada beberapa statement/kondisi diantaranya adalah `if`, `else` dan `elif` Kondisi `if` digunakan untuk mengeksekusi kode jika kondisi bernilai benar `True`.

Jika kondisi bernilai salah `False` maka statement/kondisi `if` tidak akan di-eksekusi.

Dibawah ini adalah contoh penggunaan kondisi if pada Python

```
#Kondisi if adalah kondisi yang akan dieksekusi oleh program jika bernilai
benar atau TRUE

nilai = 9

#jika kondisi benar/TRUE maka program akan mengeksekusi perintah dibawahnya
if(nilai > 7):
    print("Selamat Anda Lulus")

#jika kondisi salah/FALSE maka program tidak akan mengeksekusi perintah
dibawahnya
if(nilai > 10):
    print("Selamat Anda Lulus")
Selamat Anda Lulus
```

Dari contoh diatas, jika program dijalankan maka akan mencetak string `"Selamat Anda Lulus Ujian"` sebanyak 1 kali yaitu pada if pertama. Di if kedua statement bernilai salah, jadi perintah `print("Selamat Anda Lulus")` tidak akan dieksekusi.

B. Kondisi If Else

Pengambilan keputusan (kondisi if else) tidak hanya digunakan untuk menentukan tindakan apa yang akan diambil sesuai dengan kondisi, tetapi juga digunakan untuk menentukan tindakan apa yang akan diambil/dijalankan jika kondisi tidak sesuai.

Pada python ada beberapa statement/kondisi diantaranya adalah if, else dan elif Kondisi if digunakan untuk mengeksekusi kode jika kondisi bernilai benar.

Kondisi if else adalah kondisi dimana jika pernyataan benar `True` maka kode dalam if akan dieksekusi, tetapi jika bernilai salah `False` maka akan mengeksekusi kode di dalam else.

Dibawah ini adalah contoh penggunaan kondisi if else pada Python

```
#Kondisi if else adalah jika kondisi bernilai TRUE maka akan dieksekusi
pada if, tetapi jika bernilai FALSE maka akan dieksekusi kode pada else

nilai = 3
#Jika pernyataan pada if bernilai TRUE maka if akan dieksekusi, tetapi jika
FALSE kode pada else yang akan dieksekusi.
if(nilai > 7):
    print("Selamat Anda Lulus")
else:
    print("Maaf Anda Tidak Lulus")
Maaf Anda Tidak Lulus
```

Pada contoh diatas, jika program dijalankan maka akan mencetak string `"Maaf Anda Tidak Lulus"` karena pernyataan pada if bernilai `False`

C. Kondisi Elif

Pengambilan keputusan (kondisi if elif) merupakan lanjutan/percabangan logika dari "kondisi if". Dengan elif kita bisa membuat kode program yang akan menyeleksi beberapa kemungkinan yang bisa terjadi. Hampir sama dengan kondisi "else", bedanya kondisi "elif" bisa banyak dan tidak hanya satu.

Dibawah ini adalah contoh penggunaan kondisi elif pada Python

```
#Contoh penggunaan kondisi elif

hari_ini = "Minggu"

if(hari_ini == "Senin"):
    print("Saya akan kuliah")
elif(hari_ini == "Selasa"):
    print("Saya akan kuliah")
elif(hari_ini == "Rabu"):
    print("Saya akan kuliah")
elif(hari_ini == "Kamis"):
    print("Saya akan kuliah")
elif(hari_ini == "Jumat"):
    print("Saya akan kuliah")
elif(hari_ini == "Sabtu"):
    print("Saya akan kuliah")
elif(hari_ini == "Minggu"):
    print("Saya akan libur")
Saya akan libur
```

Pada contoh diatas, jika program dijalankan maka akan mencetak string `"Saya akan libur"`.

(6). Pengulangan

A. While Loop

Pengulangan While Loop di dalam bahasa pemrograman Python dieksesusi statement berkali-kali selama kondisi bernilai benar atau `True`.

Dibawah ini adalah contoh penggunaan pengulangan `While Loop`

```
#Contoh penggunaan While Loop

count = 0
while (count < 9):
    print ("The count is: ", count)
    count = count + 1

print ("Good bye!")
The count is:  0
The count is:  1
The count is:  2
The count is:  3
The count is:  4
The count is:  5
The count is:  6
The count is:  7
The count is:  8
Good bye!
x = "FASILKOM"
while x:
    print(x, " ")
    x = x[1:]
FASILKOM
ASILKOM
SILKOM
ILKOM
LKOM
KOM
OM
M
```

B. For Loop

Pengulangan `for` pada Python memiliki kemampuan untuk mengulangi item dari urutan apapun, seperti `list` atau `string`.

Dibawah ini adalah contoh penggunaan pengulangan For Loop.

```
#Contoh pengulangan for sederhana
angka = [1,2,3,4,5]
for x in angka:
    print(x)

#Contoh pengulangan for
buah = ["nanas", "apel", "jeruk"]
for makanan in buah:
    print ("Saya suka makan", makanan)
```

```
1
2
3
4
5
Saya suka makan nanas
Saya suka makan apel
Saya suka makan jeruk
nama = ['budi', 'andi', 'rudi', 'sandi']
usia = [20, 18, 22, 19]
for i in range(len(nama)) :
    print(nama[i], ' berusia ', usia[i], ' tahun')
budi  berusia  20  tahun
andi  berusia  18  tahun
rudi  berusia  22  tahun
sandi  berusia  19  tahun
```

C. Nested Loop

Bahasa pemrograman Python memungkinkan penggunaan satu lingkaran di dalam loop lain. Bagian berikut menunjukkan beberapa contoh untuk menggambarkan konsep tersebut.

Dibawah ini adalah contoh penggunaan Nested Loop.

```
#Contoh penggunaan Nested Loop

i = 2
while(i < 100):
    j = 2
    while(j <= (i/j)):
        if not(i%j): break
        j = j + 1
    if (j > i/j) : print(i, " is prime")
    i = i + 1

print("Good bye!")
2  is prime
3  is prime
5  is prime
7  is prime
11  is prime
13  is prime
17  is prime
19  is prime
23  is prime
29  is prime
31  is prime
37  is prime
41  is prime
43  is prime
47  is prime
53  is prime
59  is prime
61  is prime
67  is prime
71  is prime
73  is prime
79  is prime
83  is prime
```

```
89  is prime
97  is prime
Good bye!
```