Bab I

Intro to Python

Python sebagai bahasa pemrograman yang populer dan komprehensif dengan menggabungkan kapabilitas, sintaksis kode yang jelas serta dilengkapi pustaka standar yang mempunyai fungsionalitas sangat besar. Python termasuk dari jajaran bahasa pemrograman tingkat tinggi seperti bahasa pemrograman C, C++, Java, Perl dan Pascal.

A. Keywords

Keywords adalah kata yang dicadangkan oleh Python sehingga tidak boleh digunakan sebagai nama variabel, fungsi atau identifier lainnya. Keywords sendiri digunakan untuk mendefinisikan sintaks dan struktur dari bahasa Python. Keyword dalam Python merupakan case sensitive.

Ada 33 keyword dalam Python 3.7. Semua keywords selain True, False, dan None ditulis dalam lowercase dan harus ditulis apa adanya. Berikut adalah keywords dalam Python.

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

B. Identifiers

Identifiers merupakan nama yang digunakan untuk mengidentifikasi class, function, variable, dll. Identifiers berguna untuk membedakan suatu entitas dengan entitas lainnya.

Aturan dalam menulis identifiers :

1. Identifiers dapat berupa kombinasi dari huruf dalam lowercase (a-z) atau uppercase (A-Z) atau digit (0-9) atau underscore (_). Contoh : myClass, var_1, this_is_a_long_variable
2. Identifiers tidak boleh diawali dengan digit.

```
1variable # Penulisan Salah
variable1 # Penulisan Benar
```

3. Keywords tidak dapat digunakan sebagai identifiers.

```
True = 1
```

Output:

```
File "d:\Code\tes.py", line 1
  True = 1
       ^
SyntaxError: invalid syntax
```

4. Tidak diperbolehkan menggunakan symbol special seperti !, @, #, $, %, dll.

```
x$ = 1
```

Output:

```
File "d:\Code\tes.py", line 1
  x$ = 1
      ^
SyntaxError: invalid syntax
```

5. Sebuah identifier dapat memiliki panjang berapapun.
6. Identifier Class dimulai dengan huruf uppercase sedangkan semua identifiers dimulai dengan huruf lowercase.
7. Identifier yang dimulai dengan underscore (_<name>) menandakan identifier tersebut merupakan identifier private.
8. Identifier yang dimulai dengan dua buah underscore (__<name>) menandakan identifier tersebut merupakan identifier yang sangat private.
9. Jika identifier tersebut juga diakhiri dengan dua buah underscore (_<name>_), identifier tersebut merupakan nama yang telah ditetapkan Python.

Hal yang perlu diingat!

· Python merupakan bahasa pemrograman yang case-sensitive. Sehingga, Variable dan variabel tidak sama.

C. Statements

Statements adalah instruksi atau pernyataan yang diberikan untuk dieksekusi oleh mesin. Penulisan statements dalam Python tidak diakhiri dengan tanda titik koma (;). Contohnya sebagai berikut:

```
x = 1
y = x
z = x + y
```

Dalam Python, akhir dari statement ditandai dengan baris baru. Tapi, kita dapat memperpanjang sebuah statement lebih dari beberapa baris secara eksplisit dengan menggunakan karakter forward slash (\). Contohnya :

```
x = 1 + 2 + 3 + \
  4 + 5 + 6 + \
  7 + 8 + 9
```

Multi-line statements juga terdapat dalam tanda kurung (), kurung siku [], kurung kurawal {}. Sebagai contoh, kita dapat mengimplementasikan contoh di atas menjadi :

```
x = (1 + 2 + 3 +
  4 + 5 + 6 +
  7 + 8 + 9)
colors = ['red',
        'blue',
        'green']
```

Kita juga dapat menyingkat penulisan statements menjadi satu baris menggunakan tanda titik koma ;.

```
a = 1; b = 2; c = 3
```

D. Indentation

Sebagian besar Bahasa pemrograman seperti C, C++, dan Java menggunakan kurung kurawal **{}** untuk mendefinisikan sebuah blok kode sedangkan Python menggunakan indentasi. Indentation sendiri adalah penulisan yang menjorok masuk ke dalam dari sebuah kode.

Sebuah blok kode (body dari sebuah function, loop, etc) dalam Python dimulai dengan indentasi dan diakhiri dengan baris yang tidak diindentasi. Jumlah spasi dari indentasi itu bebas, tetapi jumlah spasinya harus konsisten. Biasanya, 4 spasi digunakan sebagai indentasi dan lebih dipilih daripada tab. Sebagai contoh:

- Penulisan Benar

```
for i in range(1, 20):
    if i == 3:
        print("it's three")
        Break
```

- Penulisan Salah

```
for i in range(1, 20):
if i == 3:
print("it's three")
Break
```

Penggunaan dari indentasi dalam Python membuat kode terlihat rapi dan bersih sehingga menghasilkan sebuah kode yang terlihat mirip dan konsisten. Indentasi membuat kode tersebut menjadi lebih mudah dibaca. Sebagai contoh:

```
if True:
    print('Yes')
    x = 10
```

akan lebih mudah dibaca daripada,

```
if True:print('Yes'); x = 10
```

Indentasi yang salah akan menghasilkan error **IndentationError**.

E. Comments

Comments sangatlah penting dalam penulisan program. Comments membantu mendeskripsikan isi dari kode tersebut sehingga orang lain tidak sulit dalam memahami kode yang kita tulis. Penulisan comment dalam Python terbagi menjadi:

- Single-line comment

```
#This is a single-line comment
```

- Multi-line comment

```
"""
This is
a
Multi-line comments """
```

F. Variables & Constant

Variabel merupakan representasi dari alamat memori yang digunakan untuk menyimpan nilai dari data. Sintaks dari penulisan variabel adalah **name = value**. Contohnya,

```
x = 10
```

Di sini, kita telah membuat sebuah variabel bernama **x** dan telah memberinya value **10**. Variabel dapat kita anggap sebagai tas untuk menyimpan buku di dalamnya dan buku itu dapat diganti kapan saja. Hal ini berarti sebuah value dari variabel dapat diubah-ubah. Sebagai contoh:

```
x = 10
print(x)
x = 10.5
print(x)
```

Output:

```
10
10.5
```

Kita juga dapat memberikan beberapa nilai ke beberapa variabel sekaligus. Contoh:

```
x, y, z = 1, 3.2, "System Information"
print(x)
print(y)
print(z)
```

Output:

```
1
3.2
System Information
```

Jika kita ingin menetapkan value yang sama ke banyak variabel sekaligus, kita dapat melakukannya seperti:

```
x = y = z = "System Information"
print(x)
print(y)
print(z)
```

Output:

```
System  Information
System  Information
System Information
```

Constant merupakan sebuah tipe variabel yang valuenya tidak dapat diubah. Constant dapat kita anggap sebagai sebuah tas untuk menyimpan sebuah buku yang isinya tidak dapat diubah lagi. Contoh dari constant adalah

```
PI = 3.14
GRAVITY = 9.8
```

Aturan dan Ketentuan dalam penamaan variabel dan constant:

1. Nama constant dan variabel harus memiliki kombinasi huruf kecil (a-z) atau huruf besar (A-Z) atau angka (0-9) atau garis bawah (_)

   ```
   snake_case
   MACRO_CASE
   camelCase
   CapWords
   ```

2. Buat nama yang masuk akal. Misalnya length lebih masuk akal daripada l.
3. Jika ingin menulis nama variabel yang lebih dari dua kata, gunakan garis bawah untuk memisahkannya.

   ```
   car_name
   this_is_a_variable
   ```

4. Gunakan huruf kapital untuk mendeklarasikan sebuah constant.

   ```
   PI
   G
   MASS
   SPEED_OF_LIGHT
   TEMP
   ```

5. Jangan gunakan simbol spesial seperti !, @, #, $, %, etc.
6. Jangan memulai nama variabel dengan angka.

G. Data Types

Data types adalah klasifikasi atau kategorisasi item data yang mewakili jenis nilai yang memberi tahu operasi apa yang dapat dilakukan pada data tertentu. Karena semuanya adalah objek dalam Python, tipe data sebenarnya adalah sebuah class dan variable adalah instance (object) dari class ini.

Ada berbagai macam tipe data di Python yang sering digunakan, sebagai berikut:

1. **Python Numbers**

   Dalam Python, tipe data numerik mewakili data yang memiliki value numerik. Nilai numerik dapat berupa Integer, Float, dan bilangan Complex. Nilai – nilai ini didefinisikan sebagai kelas int, float, dan complex dalam Python.

   a. Integer – Nilai ini diwakili oleh kelas int. Integer berisi bilangan bulat positif atau negative (tanpa pecahan atau decimal).
   b. Float – Nilai ini diwakili oleh kelas float. Float adalah bilangan real dengan representasi floating point atau ditentukan oleh titik decimal. Secara opsional, karakter e atau E yang diikuti dengan bilangan bulat positif atau negative dapat ditambahkan untuk menentukan notasi ilmiah. Nilai float sendiri hanya akurat hingga 15 angka decimal.
   c. Complex number. Bilangan kompleks diwakili oleh kelas complex. Ini dispesifikasikan sebagai (bagian real) _ (bagian imajiner). Misalnya 2 + 3j.

   Kita dapat menggunakan fungsi type() untuk mengetahui tipe dari tipe data tersebut.

```
x = 10
print("Type of x: ", type(x))

y = 10.2
print("Type of y: ", type(y))

z = 3 + 5j
print("Type of z: ", type(z))
```

   Output:

```
Type of x:  <class 'int'>
Type of y:  <class 'float'>
Type of z: <class 'complex'
```

2. **Python Strings**

   Dalam Python, String adalah array byte yang mewakili karakter Unicode. String adalah kumpulan dari satu atau lebih karakter yang ditulis dalam tanda kutip tunggal **‘<text>’**, tanda kutip ganda **“<text>”**, tanda kutip

tiga ‘’’**<text>**’’’ atau **“””<text>”””**. Dalam Python, tidak ada tipe data **char** sehingga char adalah string dengan panjang satu. String sendiri diwakili oleh class str.

```
s = 'This is a single quotes string'
print(s)
s = "This is a double quotes string"
print(s)
# String dengan tanda kutip tiga dapat #
membuat multi-line string
s = '''This is a triple
   quotes string'''
print(s)
```

Output:

```
This is a single quotes string
This is a double quotes string
This is a triple
quotes string
```

Karena string merupakan sebuah array, maka operator slicing **[ ]** dapat digunakan untuk mengakses karakter dalam string.

```
s = "Hello World"
print(s[6])
print(s[6:11])
```

Output:

```
W
World
```

3. **Python List**

List adalah sebuah urutan item yang berurutan. List merupakan salat satu tipe data yang paling sering digunakan dalam Python dan sangat fleksibel yang berarti semua item dalam list tidak harus bertipe sama. Untuk mendeklarasi sebuah list

```
a = [1, 2.2, 'python']
```

4. **Python Tuple**

Tuple sendiri mirip dengan dengan list tetapi item dalam tuple tidak dapat diubah. Tuple setelah dibuat tidak dapat dimodifikasi. Ini dikarenakan

tuple digunakan untuk melindungi data dan biasanya lebih cepat daripada list yang dapat berubah secara dinamis. Tuple sendiri dideklarasikan menggunakan tanda kurung **(<item>, <item>, …)** dan itemnya dipisah menggunakan tanda koma.

```
a = (1, 2.2, 'python')
```

5. **Python Set**

Set adalah sebuah koleksi dari berbagai item unik. Set didefinisikan oleh item yang dipisah oleh koma di dalam sebuah kurung kurawal **{<item>, <item>, …}**. Item dari sebuah set tidak terurut. Karena set itu memiliki value unik maka set akan mengeliminasi value yang duplikat.

```
a = {1, 2.2, 'python'}
```

6. **Python Dictionary**

Dictionary adalah sebuah koleksi tak terurut dari pasangan key-value. Dictionary sering digunakan ketika berhadapan dengan data yang besar karena adanya key-value sehingga pengambilan data lebih optimal. Setiap pasangan key-value dari dictionary dipisahkan oleh tanda titik dua **:**, di mana setiap key dipisahkan oleh koma dan setiap key tidak boleh sama (unik). Dalam Python, dictionary dibuat dengan menaruh kumpulan pasangan key-value tersebut ke dalam sebuah kurung kurawal **{key: value, key: value, …}**. Key dari dictionary ialah case sensitive, yang berarti jika sebuah key memiliki nama yang sama namun penulisannya berbeda maka akan dianggap sebagai key yang berbeda.

```
a = {'python': 1, 'Python': 1}
```

7. **Python Boolean**

Boolean adalah tipe data yang dapat menampung dua nilai, yaitu **True** dan **False**, operasi dengan *Logical Operator* juga akan menghasilkan nilai boolean, sehingga tipe data ini biasa digunakan dalam penyeleksian kondisi dan perulangan. Nilai **True** akan mengembalikan nilai **1** dan **False** akan mengembalikan nilai **0**.

```
is_raining = True
is_walking = False
x = (1 == True)
```

```
y = (1 == False)
a = True + 4
b = False + 10
```

Selain tipe data di atas, berikut adalah tipe data lengkap dari Python:

- **Numeric data types**: *int, float, complex*
- **String data types**: *str*
- **Sequence types**: *list, tuple, range*
- **Binary types**: *bytes, bytearray, memoryview*
- **Mapping data type**: *dict*
- **Boolean type**: *bool*
- **Set data types**: *set, frozenset*

H. Conversion of Data Types

Sebuah proses pengubahan nilai dari suatu tipe data ke tipe data yang lainnya disebut konversi tipe data. Python sendiri memiliki 2 jenis konversi data:

- **Implicit Type Conversion**

  Dalam konversi implisit, Python akan secara otomatis mengubah tipe data dari nilai tersebut. Proses ini tidak membutuhkan campur tangan dari user. Sebagai contoh:

```
num_int = 345
num_flo = 3.45

new = num_int + num_flo

print("datatype of num_int:",type(num_int))
print("datatype of num_flo:",type(num_flo))

print("Value of new:",new)
print("datatype of new:",type(new))
```

  Output:

```
datatype of num_int: <class 'int'>
datatype of num_flo: <class 'float'>
Value of new: 348.45
datatype of new: <class 'float'>
```

Dari program di atas dapat dilihat bahwa kita akan melakukan operasi penjumlahan dari variabel integer **num_int** dan float **num_flo**. Hasil dari penjumlahan tersebut akan menghasilkan sebuah value yang bertipe data **float**, hal ini terjadi karena Python akan selalu mengkonversi tipe data kecil ke tipe data besar untuk menghindari adanya kehilangan data.

- **Explicit Type Conversion**

  Dalam konversi eksplisit, user mengkonversi tipe data dari sebuah objek ke tipe data yang dibutuhkan dengan bantuan fungsi konversi tipe seperti **int()**, **float()**, **str()**, dll. Proses konversi tipe ini biasa disebut sebagai *typecasting*. Syntax : **<required_datatypes>(expression)**.

```
num_int = 123
num_str = "456"

print("Data type of num_int:",type(num_int))
print("Data type of num_str before Type Casting:",type(num_str))

num_str = int(num_str)
print("Data type of num_str after Type Casting:",type(num_str))

num_sum = num_int + num_str

print("Sum of num_int and num_str:",num_sum)
print("Data type of the sum:",type(num_sum))
```

Output:

```
Data type of num_int: <class 'int'>
Data type of num_str before Type Casting: <class 'str'>
Data type of num_str after Type Casting: <class 'int'>
Sum of num_int and num_str: 579
Data type of the sum: <class 'int'>
```

Dari program di atas dapat dilihat bahwa kita akan melakukan operasi penjumlahan dari variabel integer **num_int** dan string **num_str**. Karena akan terjadi error bila tipe data integer dan string dijumlahkan langsung, maka perlu dilakukan konversi tipe data string ke integer dengan menggunakan fungsi **int()**. Setelah dilakukan konversi maka kedua variabel tersebut dapat dijumlahkan dan menghasilkan sebuah value yang bertipe data **integer**.

I. Input

Untuk menambah fleksibilitas dalam program, kita mungkin mau mengambil input dari user. Python menyediakan fungsi input() untuk mengambil inputan dari user lalu menyimpannya dalam sebuah variabel. Syntax dari input() adalah:

```
input([prompt])
```

dimana prompt adalah sebuah string yang ingin ditampilkan secara opsional. Fungsi dari input() akan mengembalikan sebuah value string.

```
name = input("Input a name : ")
print(name)
```

Output:

```
Input a name : Sistem Informasi
Sistem Informasi
```

J. Operators

Operator adalah sebuah symbol special dalam Python yang berguna untuk menjalankan komputasi aritmatika atau logika. Value yang dioperasikan oleh operator disebut operand.

Dalam Python ada beberapa operator, sebagai berikut:

1. Arithmetic Operators

Operator aritmatik digunakan untuk menjalankan operasi matematika seperti penjumlahan, pengurangan, perkalian, dll.

Operator	Arti	Contoh
+	Menjumlahkan dua operand atau unary plus	x + y + 2
-	Mengurangi operand kanan dari kiri atau unary minus	x – y -- 2
*	Mengalikan operand kiri dengan kanan	x * y
/	Membagikan operand kiri dengan kanan (akan menghasilkan float)	x / y
%	Modulus – hasil bagi dari operand kiri dengan kanan	x % y

//	Floor Division – pembagian yang hasilnya menjadi bilangan bulat disesuaikan ke kiri pada garis bilangan	x // y
**	Pangkat – operand kiri dipangkatkan oleh operand kanan	x ** y

2. Comparison (Relational) Operators

Operator komparasi digunakan untuk membandingkan value dan akan mengembalikan nilai **True** atau **False** berdasarkan kondisinya.

Operator	Arti	Contoh
>	Lebih besar dari – True jika operand kiri lebih besar dari kanan	x > y
<	Kurang dari – True jika operand kiri lebih kecil dari kanan	x < y
==	Sama dengan – True jika kedua operand bernilai sama	x == y
!=	Tidak sama dengan – True jika kedua operand tidak bernilai sama	x != y
>=	Lebih besar atau sama dengan – True jika operand kiri lebih besar atau sama dengan kanan	x >= y
<=	Lebih kecil atau sama dengan – True jika operand kiri lebih kecil atau sama dengan kanan	x <= y

3. Logical (Boolean) Operators

Operator	Arti	Contoh
and	True jika kedua operand bernilai True	x and y
or	True jika kedua atau salah satu operand bernilai True	x or y
not	True jika operand bernilai False	not x

4. Bitwise Operators

Operator bitwise bertindak pada operand seolah-olah mereka adalah string digit binary.

Pada tabel di bawah : misalkan x = 10 (0000 1010) dan y = 4 (0000 0100)

Operator	Arti	Contoh
&	Bitwise AND	x & y = 0 (0000 0000)
|	Bitwise Or	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x >> 2 = 2 (0000 0010)
<<	Bitwise left shift	x << 2 = 40 (0010 1000)

5. Assignment Operators

Operator assignment digunakan dalam Python untuk menetapkan nilai ke variable.

Operator	Arti	Sama Dengan
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
**=	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
|=	x |= 5	x = x | 5
^=	x ^= 5	x = x ^ 5

>>=	x >>= 5	x = x >> 5
<<=	x <<= 5	x = x << 5

6. Identity Operator

Operator	Arti	Contoh
is	True jika kedua operand identik	x is True
is not	True jika operand tidak identik	x is not True

7. Membership Operator

Operator	Arti	Contoh
in	True jika value/variable ditemukan dalam collections	x in True
not in	True jika value/variable tidak ditemukan dalam collection	x not in True