**Pertemuan 5 : String dan Number**

**Topik Sesi** :

1. String
2. Bilangan (Number)

**Capaian pembelajaran mata kuliah** :

1. Mahasiswa mampu menjelaskan dan mempraktikkan String menggunakan Python;
2. Mahasiswa mampu menjelaskan dan mempraktikkan Bilangan menggunakan Python.

**String**

String adalah tipe data yang paling sering digunakan di Python. Kita bisa membuat string dengan meletakkan karakter di dalam tanda kutip. Tanda kutipnya bisa kutip tunggal maupun kutip ganda. Contohnya adalah sebagai berikut:

var1 = 'Hello Python'

var2 = 'Programming with Python'

**Mengakses Nilai String**

Untuk mengakses nilai atau substring dari string, kita menggunakan indeks dalam tanda [ ].

var1 = 'Hello Python!'

var2 = "I love Python"

print("var1[0]", var1[0])

print("var2[2:6]:",var2[2:6])

Bila kode di atas dijalankan, hasilnya akan muncul seperti berikut:

var1[0]: H

var2[2:6]: love

**Mengupdate String**

String adalah tipe data immutable, artinya tidak bisa diubah. Untuk mengupdate string, kita perlu memberikan nilai variabel string lama ke string yang baru. Nilai yang baru adalah nilai string lama yang sudah diupdate.

var1 = 'Hello Python!'

var2 = var1[:6]

print("String Update: - ", var1[:6] + 'World')

Hasilnya adalah sebagai berikut:

String Update: - Hello World

**Menggabung String**

Kita bisa menggabungkan dua atau lebih string menjadi satu dengan menggunakan operator +. Selain itu kita juga bisa melipatgandakan string menggunakan operator \*.

str1 = 'Hello'

str2 = 'Python'

# menggunakan +

print('str1 + str2 =', str1 + str2)



**Menguji Keanggotaan Karakter Dalam String**

Kita bisa menguji apakah sebuah substring ada terdapat di dalam suatu string atau tidak. Caranya adalah dengan menggunakan kata kunci in

>>> 'a' in 'Programming'

True

>>> 'at' not in 'battle'

False

**Mengetahui Panjang String**

Untuk mengetahui panjang dari string, kita bisa menggunakan fungsi len().

>>> string = 'I love Python'

>>> len(string)

18

**Karakter Escape**

Kalau kita hendak mencetak string: He said, "What's there?" kita tidak bisa menggunakan tanda kutip tunggal maupun ganda. Bila kita melakukannya, akan muncul pesan error SyntaxError karena teks berisi kutip tunggal dan ganda.

>>> print("He said, "What's there?"")

...

SyntaxError: invalid syntax

>>> print('He said, "What's there?"')

...

SyntaxError: invalid syntax

Untuk hal seperti ini kita bisa menggunakan tanda kutip tiga atau menggunakan karakter escape.

Karakter escape dimulai dengan tanda backslash \. Interpreter akan menerjemahkannya dengan cara berbeda dengan string biasa. Solusi untuk error di atas adalah sebagai berikut:



Berikut adalah daftar karakter escape yang didukung oleh Python.

|  |  |
| --- | --- |
| **Karakter Escape** | **Deskripsi** |
| \newline | Backslash dan newline diabaikan |
| \\ | Backslash |
| \’ | Kutip tunggal |
| \” | Kutip ganda |
| \a | ASCII bel |
| \b | ASCII backscape |
| \f | ASCII formfeed |
| \n | ASCII linefeed |
| \r | ASCII carriage return |
| \t | ASCII tab horizontal |
| \v | ASCII tab horizontal |
| \ooo | karakter dengan nilai oktal oo |
| \xHH | karakter dengan nilai heksadesimal HH |

Berikut ini adalah beberapa contohnya:

>>> print("C:\\Python34\\Lib")

C:\Python34\Lib

>>> print("Ini adalah baris pertama\nDan ini baris dua")

Ini adalah baris pertama

Dan ini baris dua

>>> print("Ini adalah \x48\x45\x58")

Ini adalah HEX

**Raw String Untuk Mengabaikan Karakter Escape**

Kadang kala kita perlu untuk mengabaikan karakter escape yang ada dalam string. Kita bisa melakukannya dengan meletakkan huruf r atau R sebelum tanda kutip string.

>>> print("This is \x61 \ngood example")

This is a

good example

>>> print(r"This is \x61 \ngood example")

This is \x61 \ngood example

**Mengatur Format String**

Ada dua cara melakukan format pada string. Pertama dengan menggunakan fungsi format(), dan kedua dengan menggunakan cara lama (menggunakan %).

**Metode format()**

Memformat string dengan fungsi format() dibuat dengan menggunakan tanda {} sebagai placeholder atau posisi substring yang akan digantikan. Kita biasa menggunakan argumen posisi atau kata kunci untuk menunjukkan urutan dari substring.



Metode format() dapat memiliki spesifikasi format opsional. Misalnya, kita bisa menggunakan tanda < untuk rata kiri, > untuk rata kanan, ^ untuk rata tengah, dan sebagainya.

>>> # format integer

>>> "{0} bila diubah jadi biner menjadi {0:b}".format(12)

'12 bila diubah jadi biner menjadi 1100'

>>> # format float

>>> "Format eksponensial: {0:e}".format(1566.345)

'Format eksponensial: 1566345e+03'

>>> # pembulatan

>>> "Sepertiga sama dengan: {0:.3f}".format(1/3)

'Sepertiga sama dengan: 0.333'

>>> # Meratakan string

>>> "|{:<10}|{:^10}|{:>10}|".format('beras', 'gula', 'garam')

'|beras | gula | garam|'

**Format Cara Lama Dengan %**

Kita bisa menggunakan operator % untuk melakukan format string.

>>> nama = 'Budi'

>>> print('Nama saya %s' %s)

Nama saya Budi

>>> x = 12.3456789

>>> print('Nilai x = %3.2f' %x)

Nilai x = 12.35

>>> print('Nilai x = %3.4f' %x)

Nilai x = 12.3456

# Bilangan (Number)

Bilangan (number) adalah salah satu tipe data dasar di Python. Python mendukung bilangan bulat (integer), bilangan pecahan (float), dan bilangan kompleks (complex). Masing – masing diwakili oleh kelas int, float, dan complex. Integer adalah bilangan bulat, yaitu bilangan yang tidak mempunyai koma. Contohnya 1, 2, 100, -30, -5, 99999, dan lain sebagainya. Panjang integer di python tidak dibatasi jumlah digitnya. Selama memori masih cukup, maka sepanjang itulah jumlah digit yang akan ditampilkan. Float adalah bilangan pecahan atau bilangan berkoma. Contohnya adalah 5.5, 3.9, 72.8, -1.5, -0.7878999, dan lain sebagainya. Panjang angka di belakang koma untuk float ini adalah 15 digit.

Bilangan kompleks (complex) adalah bilangan yang terdiri dari dua bagian, yaitu bagian yang real dan bagian yang imajiner. Contohnya adalah 3 + 2j, 9 – 5j, dan lain sebagainya.

**Konversi Jenis Bilangan**

Kita bisa mengubah jenis bilangan dari int ke float, atau sebaliknya. Mengubah bilangan integer ke float bisa menggunakan fungsi int(num) dimana num adalah bilangan float.

>>> int(2.5)

2

>>> int(3.8)

3

>>> float(5)

5.0

Pada saat kita mengubah float ke integer, bilangan dibulatkan ke bawah. Sebaliknya saat kita mengubah integer ke float, maka bilangan bulat akan menjadi bilangan berkoma.

**Python Decimal**

Ada kalanya perhitungan menggunakan float di Python membuat kita terkejut. Kita tahu bahwa 1.1 ditambah 2.2 hasilnya adalah 3.3. Tapi pada saat kita lakukan dengan Python, maka hasilnya berbeda.

>>> (1.1 + 2.2 ) == 3.3

False

>>> 1.1 + 2.2

3.3000000000000003

Hal ini terjadi karena bilangan dalam komputer disimpan  dalam bentuk digit 0 atau 1. Bila padanan digitnya tidak sesuai, maka bilangan float seperti 0.1  dalam bilangan biner akan menjadi pecahan yang sangat panjang yaitu 0.000110011001100110011… dan komputer kita hanya akan menyimpan panjang yang terbatas. Hal inilah yang menyebabkan terjadinya masalah seperti pada contoh di atas.

Untuk menangani hal seperti itu, kita bisa menggunakan modul bawaan Python yaitu modul decimal. Float hanya memiliki presisi sampai 15 digit di belakang koma, sementara dengan modul decimal kita bisa mengatur presisi jumlah digit di belakang koma.



**Kapan Saatnya Menggunakan Decimal Dibanding Float?**

Kita lebih baik menggunakan Decimal dalam kasus:

1. Saat kita ingin membuat aplikasi keuangan yang membutuhkan presisi desimal yang pasti
2. Saat kita ingin mengontrol tingkat presisi yang diperlukan
3. Saat kita ingin menerapkan perkiraan berapa digit decimal yang signifikan
4. Saat kita ingin melakukan operasi perhitungan sama persis dengan yang kita lakukan di sekolah

**Bilangan Pecahan**

Python menyediakan modul fractions untuk mengoperasikan bilangan pecahan. Pecahan adalah bilangan yang memiliki pembilang dan penyebut, misalnya 3/2. Perhatikan contoh berikut:



**Matematika Dengan Python**

Python menyediakan modul math melakukan hal yang berbau matematis seperti trigonometri, logaritma, probabilitas, statistik, dan lain – lain.



**Sumber referensi :**

* 1. Zelle, J. M. (2002). Programming : An Introduction to Computer Science. Wartburg College.
	2. Knuth, DE. The Art of Computer Programming: Fundamental Algorithms, Addison-Wesley, 1997.
	3. Abdul Kadir (2018) Dasar Pemrograman Python.
	4. [www.python.org](http://www.python.org)
	5. [www.pythonindo.com](http://www.pythonindo.com)