Week 3 (File Operations, Strings, Dictionary)

File Operations

File operations dibagi menjadi 2:

1. opening / closing
2. reading / writing

Opening and Closing File

Cara membuka file di Python adalah dengan kata kunci `open()`.

Contoh:

In [1]:
```
# f = open(path) <- untuk membuka file

# read data from f
# OR
# write data from f

# f.close() <- selalu berada di akhir untuk menutup file
```

`path` bisa berupa `nama file` atau `directory`. Cara membedakan nama file dan directory:

- nama file: "text.txt", "data.csv", "main.py", "index.html"
- directory: "U:\programmingFundamentals", "data/movies.csv"

Reading File(s)

Misalkan kita telah mempunyai file yang diberi nama `text.txt` dan di dalam file `text.txt` terdapat kata yang berisi: `Hello\nWorld`. `\n` disini berarti baris baru. Fungsi read file adalah untuk membaca apa yang berada di dalam file tersebut. Yang memiliki tampilan sebagai berikut:

```
Hello
World
```

Maka, cara mengimplimentasi kode tersebut untuk membaca nilai dalam file tersebut adalah dengan kode dibawah.

In [2]:
```
f = open("text.txt")
r = f.read()
f.close()

print(r)
```

```
Hello
World
```

Writing File(s)

Menulis sebuah file sedikit berbeda dengan membaca sebuah file. Untuk menulis sebuah file, kita harus menggunakan kata kunci `"w"` di parameter `open()`. Contohnya seperti kode dibawah.

In [3]:
```
f = open("file.txt", "w")
```

Setelah mengetikkan kode tersebut, maka kita bisa menulis isi di dalam file tersebut dengan menggunakan kata kunci `write()`. Pada kasus ini, kita akan menuliskan beberapa kata / kalimat di dalam file `file.txt` karena kita telah membuat sebuah file baru di atas dengan nama `file.txt`.

In [4]:
```
f.write("Hello\n My name is\n Winsten")
# Abaikan output 26. Ouput ini hanya berlaku di Jupyter Notebook.
# 26 disini menunjukkan jumlah karakter yang terdapat di dalam file
# tersebut
```

Out[4]:
```
26
```

Setelah menulis file, ingat untuk selalu menutup file dengan kata kunci `close()`.

In [5]:
```
f.close()
```

Setelah selesai, maka kalian akan melihat bahwa file baru yang bernama `file.txt` telah berhasil dibuat.

String

String Method

String di Python memiliki beberapa jenis method. Diantaranya:

String Method	Purpose
s.upper()	change string to all upper case
s.lower()	opposite of upper()
s.strip()	remove whitespace (space, tab, etc) before and after
s.lstrip()	remove whitespace from left side
s.rstrip()	remove whitespace from right side
s.format(args…)	replace instances of "{}" in string with args
s.find(needle)	find index of needle in s
s.startswith(prefix)	does s begin with the given prefix?
s.endswith(suffix)	does s end with the given suffix?
s.replace(a, b)	replace all instances of a in s with b

Contoh:

In [6]:
```
s = "hello"
```

In [7]:
```
print(s.upper())
```

```
HELLO
```

Kita juga bisa memeriksa panjang sebuah string dengan kata kunci `len()`

In [8]:
```
print(len(s))
```

```
5
```

String Sequence

Perlu diingat bahwa string juga merupakan sebuah list. Oleh karena itu, cara indexing, slicing, cara mengakses list, dan cara iterasi string tersebut sama.

In [9]:
```
s = "Japan"
```

In [10]:
```
print(s[0])
```

```
J
```

In [11]:
```
print(s[:3])
```

```
Jap
```

In [12]:
```
print(s[-1])
```

```
n
```

In [13]:
```
# Cara while loop
i = 0
while i < len(s):
    print(s[i])
    i += 1
```

```
J
a
p
a
n
```

In [14]:
```
# Cara for loop
for j in s:
    print(j)
```

```
J
a
p
a
n
```

Dictionary (Data Structure)

Anggap saja dictionary seperti sebuah kamus. Misalkan jika kalian membuka kamus dan ingin mencari kata apel, maka yang akan kalian lakukan adalah mencari kata kunci "A" di dalam kamus lalu mencari kata "apel". "A" disini merupakan sebuah kunci atau disebut sebagai `key` dalam bahasa pemrograman, dan "apel" disini biasa disebut sebagai `value`.

Cara mendeklarasikan sebuah dictionary adalah dengan `nama_dictionary = {"key": "value"}`.

In [15]:
```
d = {} # <- empty dictionary
print(type(d)) # dict disini berarti tipe data dictionary
```

```
<class 'dict'>
```

Cara penulisan dictionary

In [16]:
```
d = {"A": "apel", "B": "bola", "C": "cacing"}
```

Accessing Value in Dictionary

Cara mengakses nilai pada sebuah dictionary sedikit berbeda dengan list. Pada list, kita mengakses nilai dengan menuliskan `nama_list[indeks]`. Sedangkan pada dictionary, kita menggunakan `key`. Contohnya: `nama_dictionary[key]`.

In [17]:
```
# Contohnya untuk mengakses nilai apel
print(d["A"])
```

```
apel
```

Mutations: Update, Delete and Insert

Untuk **mengupdate** nilai pada dictionary, kita bisa menuliskannya dengan `nama_dictionary[key] = value`. Sebagai contoh, kita ingin mengubah nilai `bola` menjadi `balon`.

In [18]:
```
print("Sebelum diganti")
print(d)

d["B"] = "balon"
print('-' *25)
print("Sesudah diganti")
print(d)
```

```
Sebelum diganti
{'A': 'apel', 'B': 'bola', 'C': 'cacing'}
-------------------------
Sesudah diganti
{'A': 'apel', 'B': 'balon', 'C': 'cacing'}
```

Untuk **menghapus** nilai pada dictionary, kita bisa menggunakan kata kunci `pop()` sama halnya dengan list (jika kalian membaca catatan dokumentasi yang diberikan). Bedanya, yang kita hapus adalah `key` dari `value` tersebut.

In [19]:
```
# Sebelum dihapus
print(d)

d.pop("C")

# Sesudah dihapus
print(d)
```

```
{'A': 'apel', 'B': 'balon', 'C': 'cacing'}
{'A': 'apel', 'B': 'balon'}
```

Untuk **menginsert** nilai pada dictionary, kita bisa menggunakan `nama_dictionary[new_key] = new_value`. Kita ambil contoh dictionary diatas, kita ingin mengembalikan `key` C dan `value` cacing ke dalam dictionary. Caranya:

In [20]:
```
# Sebelum
print(d)

d["C"] = "cacing"

# Sesudah
print(d)
```

```
{'A': 'apel', 'B': 'balon'}
{'A': 'apel', 'B': 'balon', 'C': 'cacing'}
```