1. Python Packages (Math, Random, ETC)

Python Packages

Kita biasanya tidak menyimpan semua file di komputer kita, di lokasi yang sama. kita menggunakan hirarki direktori yang terorganisir dengan baik untuk akses yang lebih mudah.

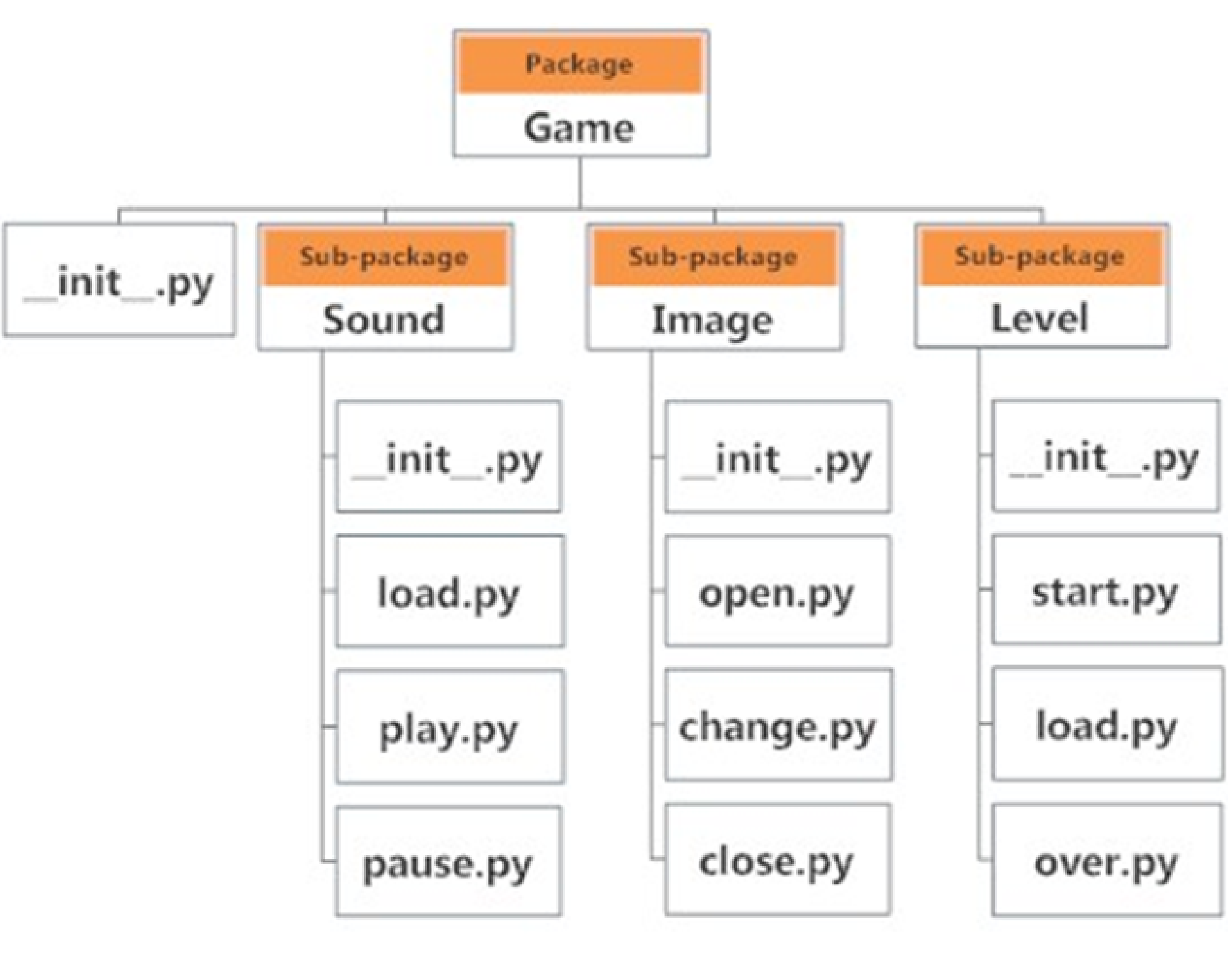
File serupa disimpan di direktori yang sama, misalnya, kita dapat menyimpan semua lagu di direktori "musik". Analog dengan ini, Python memiliki paket untuk direktori dan modul untuk file.

Saat program aplikasi kita tumbuh lebih besar ukurannya dengan banyak modul, kita menempatkan modul serupa dalam satu paket dan modul berbeda dalam paket berbeda. Ini membuat proyek (program) mudah dikelola dan jelas secara konseptual.

Demikian pula, karena direktori dapat berisi subdirektori dan file, paket Python dapat memiliki sub-paket dan modul.

Direktori harus berisi file bernama **init**.py agar Python dapat menganggapnya sebagai sebuah paket. File ini dapat dibiarkan kosong tetapi kita biasanya menempatkan kode inisialisasi untuk paket itu di file ini.

Contoh



How to use python packages

```
from Game.Level import start
```

We can now call the function simply as follows:

```
start.select_difficulty(2)
```

Another way of importing just the required function (or class or variable) from a module within a package would be as follows:

```
from Game.Level.start import select_difficulty
```

Now we can directly call this function.

```
select_difficulty(2)
```

Contoh

In [1]:
```
import numpy as np
```

In [2]:
```
import random

print(random.randint(3, 9))
```

```
9
```

In [3]:
```
import random
import string

def get_random_string (length):
    letters = string.ascii_lowercase  # biar hasilnya huruf kecil
    result_str = ''.join(random.choice(letters) for i in range(length))
    print('Random string of length', length, 'is : ', result_str)

get_random_string(8)
get_random_string(6)
get_random_string(5)
get_random_string(10)
```

```
Random string of length 8 is :  pflhkrzh
Random string of length 6 is :  gruuvg
Random string of length 5 is :  alrlo
Random string of length 10 is :  wntrfydgvn
```

2. Familiar with functions

Macam-macam dalam membuat fungsi

1. Simple Function

In [4]:
```
# Define the function shout
def shout():
    """Print a string with three exclamation marks"""
    # Concatenate the strings: shout_word
    shout_word = 'congratulations' + '!!!'

    # Print shout_word
    print(shout_word)

# Call shout
shout()
```

```
congratulations!!!
```

1. Single Parameter Functions

In [5]:
```
# Define shout with the parameter, word
def shout(word):
    """Print a string with three exclamation marks"""
    # Concatenate the strings: shout_word
    shout_word = word + '!!!'

    # Print shout_word
    print(shout_word)

# Call shout with the string 'congratulations'
shout('congratulations')
```

```
congratulations!!!
```

1. Functions that return single values

In [6]:
```
# Define shout with the parameter, word
def shout(word):
    """Return a string with three exclamation marks"""
    # Concatenate the strings: shout_word
    shout_word = word + '!!!'

    # Replace print with return
    return shout_word

# Pass 'congratulations' to shout: yell
yell = shout('congratulations')

# Print yell
print(yell)
```

```
congratulations!!!
```

1. Funtions with multiple parameters

In [7]:
```
# Define shout with parameters word1 and word2
def shout(word1, word2):
    """Concatenate strings with three exclamation marks"""
    # Concatenate word1 with '!!!': shout1
    shout1 = word1 + '!!!'

    # Concatenate word2 with '!!!': shout2
    shout2 = word2 + '!!!'

    # Concatenate shout1 with shout2: new_shout
    new_shout = shout1 + shout2

    # Return new_shout
    return new_shout

# Pass 'congratulations' and 'you' to shout(): yell
yell = shout('congratulations', 'you')

# Print yell
print(yell)
```

```
congratulations!!!you!!!
```

Familiar with Functions

In [8]:
```
def greet(name):
  print("Hello, " + name + ". Good Morning")
```

In [9]:
```
greet('Hadi')
```

```
Hello, Hadi. Good Morning
```

In [10]:
```
def iseng(umur):
  print(f"umur saya adalah {umur}")
```

In [11]:
```
iseng(21)
```

```
umur saya adalah 21
```

In [12]:
```
def nama_dosen(nama):
  print("Nama Dosen " + nama)
```

In [13]:
```
nama_dosen("Jajang")
```

```
Nama Dosen Jajang
```

In [14]:
```
# Kalo sebelumnya hanya bisa string, sekarang coba angka
def my_func():
  x = 10
  print("Value inside function:", x)

x = 20
my_func()
print("Value outside function:", x)
```

```
Value inside function: 10
Value outside function: 20
```

In [15]:
```
def tinggi():
  x = 140
  print("Tinggi kamu", x)
```

In [16]:
```
tinggi()
```

```
Tinggi kamu 140
```

In [17]:
```
def nilai():
  x = 90
  print("Nilai kamu adalah", x)

x = 98
nilai()
print("Nilai dia adalah", x)
```

```
Nilai kamu adalah 90
Nilai dia adalah 98
```

The Return Statements

```
return [expression_list]
```

Pernyataan ini dapat berisi ekspresi yang dievaluasi dan nilainya dikembalikan.

Jika tidak ada ekspresi dalam pernyataan atau pernyataan kembali itu sendiri tidak ada di dalam fungsi, maka fungsi tersebut akan mengembalikan objek None.

In [18]:
```
# contoh
def absolute_value(num):

  if num >= 0:
    return num * 2
  else:
    return -num
```

In [19]:
```
print(absolute_value(88))
```

```
176
```

In [20]:
```
print(absolute_value(-2))
```

```
2
```

In [21]:
```
def nilai_mahasiswa(nilai):

  if nilai >= 75:
    return 'kamu pinter'
  elif nilai == 50:
    return 'kamu b aja'
  else:
    return nilai
```

In [22]:
```
print(nilai_mahasiswa(89))
```

```
kamu pinter
```

In [23]:
```
nilai = 'jati rahardi'
k = list(nilai.split())
len(k)
```

Out[23]:
```
2
```

3. Create Multiple Arguments

Create Multiple Arguments

Dalam topik fungsi yang ditentukan pengguna, kita belajar tentang mendefinisikan fungsi dan memanggilnya.

Jika tidak, pemanggilan fungsi akan menghasilkan kesalahan.

In [24]:
```
def my_function(fname):
    print(fname + " LULUS")

my_function("Hadi")
```

```
Hadi LULUS
```

In [25]:
```
def my_function(nilai):
    print("nilai kamu adalah " + str(nilai))

my_function(8)
```

```
nilai kamu adalah 8
```

In [26]:
```
# mempunyai 2 argumen
def my_function(fname, lname):
    print(fname + " " + lname)

my_function('Muhamad', 'Hadiyansyah')
```

```
Muhamad Hadiyansyah
```

In [27]:
```
def nama(fname, lname):
    print(lname + " "+ fname)

nama('Muhamad', 'Hadiyansyah')
```

```
Hadiyansyah Muhamad
```

In [28]:
```
def my_function(*kids):  # satu argument bisa banyak nama
    print("The youngest child is " + kids[-1])

my_function('Emil', 'Tobias', 'Linus')
```

```
The youngest child is Linus
```

In [29]:
```
def my_function(child3, child2, child1):
    print("The youngest child is " + child3)

my_function(child3 = 'Emil', child2 = 'Tobias', child1 = 'Linus')
```

```
The youngest child is Emil
```

In [30]:
```
def my_function(**kids): # banyak argument
    print("His last name is " + kids['lname'])

my_function(fname = 'Tobias', lname = 'Linus')
```

```
His last name is Linus
```

In [31]:
```
def my_function(**kid):
    print('His last name is ' + kid['lname'] + ' his age is ' + kid['age'])

my_function(fname = 'Tobias', lname = 'Linus', age = '19')
```

```
His last name is Linus his age is 19
```

In [32]:
```
def my_function(country = 'Norway'):
    print('I am from ' + country)

my_function('Indonesia')
```

```
I am from Indonesia
```

In [33]:
```
def my_function(country = 'Norway'):
    print('I am from ' + country)

my_function()
```

```
I am from Norway
```

In [34]:
```
# Contoh
def greet(name, msg):
  print("Hello", name + ', ' + msg)

greet("Hadi", 'Congratulation')
```

```
Hello Hadi, Congratulation
```

In [35]:
```
# contoh lain
def greet(name, msg='Good Job'):
  print("Hello", name + ', ' + msg)

greet("Hadi")
greet('kiwar', 'Excellent')
```

```
Hello Hadi, Good Job
Hello kiwar, Excellent
```

```
# 2 keyword arguments
greet(name = "Bruce",msg = "How do you do?")

# 2 keyword arguments (out of order)
greet(msg = "How do you do?",name = "Bruce")

1 positional, 1 keyword argument
greet("Bruce", msg = "How do you do?")
```

Error in Creating Arguments

```
>>> greet("Monica")    # only one argument
TypeError: greet() missing 1 required positional argument: 'msg'
```

```
>>> greet()    # no arguments
TypeError: greet() missing 2 required positional arguments: 'name' and 'msg'
```

In [36]:
```
# contoh, hasil bakal error
greet(name="Bruce", "How do you do?")
```

```
  File "<ipython-input-36-c97cab45edb3>", line 2
    greet(name="Bruce", "How do you do?")
                       ^
SyntaxError: positional argument follows keyword argument
```

Python Arbitrary Arguments

In [37]:
```
# contoh
def greet(*nama):
  # nama is a tuple with arguments
  for val in nama:
    print("Hai", val)

greet("Hadi", 'Kiwar', 'Alif')
```

```
Hai Hadi
Hai Kiwar
Hai Alif
```

4. Lambda Function

```
lambda arguments: expression
```

```
# Program to show the use of lambda functions
double = lambda x: x * 2

print(double(5))
```

Output

```
10
```

In [38]:
```
coba = lambda x: x + 10

print(coba(10))
```

```
20
```

```
double = lambda x: x * 2
```

is nearly the same as:

```
def double(x):
   return x * 2
```

In [39]:
```
coba = lambda x: x * 10

print(coba(2))
```

```
20
```

In [40]:
```
def coba(x):
  return x * 10

coba(2)
```

Out[40]:
```
20
```

In [41]:
```
x = lambda a: a + 10
print(x(5))
```

```
15
```

In [42]:
```
nilai = lambda a: a * 10 #>> a itu variabel, dan a * 10 adalah perintah
print(nilai(7))
```

```
70
```

In [43]:
```
x = lambda a, b: a * b
print(x(2, 3))
```

```
6
```

In [44]:
```
x = lambda a, b, c : a + b + c
print(x(1, 2, 3))
```

```
6
```

In [45]:
```
def myfunc(n):
    return lambda a: a ** n

pangkat = myfunc(3) # ini merupakan n
# belum selesai, karena a nya belum ada
print(pangkat(3)) # ini merupakan a
```

```
27
```

In [46]:
```
def nilai(n):
    return lambda a: a * n

nilai_mahasiswa = nilai(10)

print(nilai_mahasiswa(7))
```

```
70
```

In [47]:
```
print(nilai_mahasiswa(8))
```

```
80
```

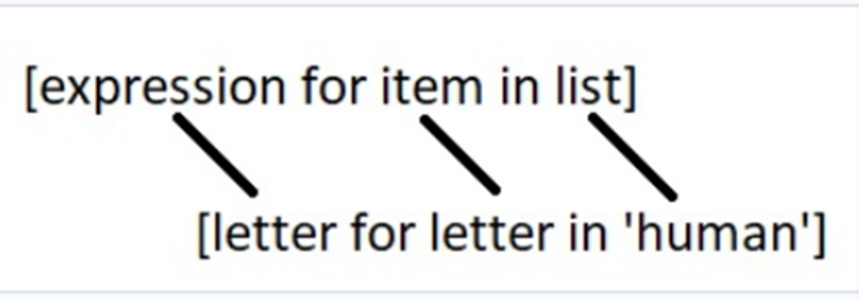
In [48]:
```
def nilai(n):
    return lambda a: a / n

nilai_mahasiswa = nilai(2)

print(nilai_mahasiswa(7))
```

```
3.5
```

tambahan: * atau bintang satu artinya argumennya satu tapi bisa diisi banyak. ** atau bintang 2 maka argumennya bisa lebih dari satu.

5. List Comprehensive

```
[expression for item in list]
```



```
h_letters = [ letter for letter in 'human' ]
print( h_letters)
```

When we run the program, the output will be:

```
['h', 'u', 'm', 'a', 'n']
```

In [49]:
```
nama_lengkap = [ letter for letter in 'Muhamad Hadiyansyah']
print(nama_lengkap)
```

```
['M', 'u', 'h', 'a', 'm', 'a', 'd', ' ', 'H', 'a', 'd', 'i', 'y', 'a', 'n', 's', 'y', 'a', 'h']
```

Mau looping secara singkat

In [50]:
```
[i for i in range(20)]
```

Out[50]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

In [51]:
```
# gimana kalo kita ingin ada kondisi
[i for i in range(20) if i % 3 > 0]
```

Out[51]:
```
[1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19]
```

In [52]:
```
[i for i in range(20) if i % 3 == 0] #pembagian yang sisanya nol
```

Out[52]:
```
[0, 3, 6, 9, 12, 15, 18]
```

In [53]:
```
[n ** 2 for n in range(12)] # maksudnya di range dulu dari 0 - 11 kemudian masing-masing dipangkatin 2
```

Out[53]:
```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121]
```

In [54]:
```
[(i, j) for i in range(2) for j in range(3)]
#
```

Out[54]:
```
[(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2)]
```