01. Python Tutorial

LOAD PACKAGES!

In [1]:

```
import numpy as np
print("Loading package(s)")
print(np.__name__, np.__version__)
```

```
Loading package(s)
numpy 1.14.2
```

PRINT function usages

In [2]:

```
print ("Hello, world")

# THERE ARE THREE POPULAR TYPES
# 1. INTEGER
x = 3;
print ("Integer: %01d, %02d, %03d, %04d, %05d" % (x, x, x, x, x))
# 2. FLOAT
x = 123.456;
print ("Float: %.0f, %.1f, %.2f, %1.2f, %2.2f" % (x, x, x, x, x))
# 3. STRING
x = "Hello, world"
print ("String: [%s], [%3s], [%20s]" % (x, x, x))
```

```
Hello, world
Integer: 3, 03, 003, 0003, 00003
Float: 123, 123.5, 123.46, 123.46, 123.46
String: [Hello, world], [Hello, world], [        Hello, world]
```

FOR + IF/ELSE

In [3]:

```
lectures = ["Search", "CBR", "ML", "NN", "DL"]

for lec in lectures:
    if lec in ["ML", "DL"]:
        print ("We have seen %s" % (lec))
```

```
We have seen ML
We have seen DL
```

In [4]:

```
lectures = ["Search", "CBR", "ML", "NN", "DL"]
for lec in lectures:
    if lec in ["Search", "CBR", "ML"]:
        print ("%s is a traditional method." % (lec))
    elif lec in ["NN"]:
        print ("%s is a Neural Networks." % (lec))
    else:
        print ("%s is a Deep Learning." % (lec))

# Little more advanced?
print("\nFOR loop with index.")
for lec, i in zip(lectures, range(len(lectures))):
    if lec in ["Search", "CBR", "ML"]:
```

```
        print ("[%d/%d] %s is a traditional method." % (i, len(lectures), lec))
    elif lec in ["NN"]:
        print ("[%d/%d] %s is a Neural Networks." % (i, len(lectures), lec))
    else:
        print ("[%d/%d] %s is a Deep Learning." % (i, len(lectures), lec))
```

```
Search is a traditional method.
CBR is a traditional method.
ML is a traditional method.
NN is a Neural Networks.
DL is a Deep Learning.

FOR loop with index.
[0/5] Search is a traditional method.
[1/5] CBR is a traditional method.
[2/5] ML is a traditional method.
[3/5] NN is a Neural Networks.
[4/5] DL is a Deep Learning.
```

Note that, index starts with 0 !

Function

In [5]:

```
# Function definition looks like this
def sum(a, b):
    return a+b
X = 10.
Y = 20.
# Usage
print ("%.1f + %.1f = %.1f" % (X, Y, sum(X, Y)))
```

```
10.0 + 20.0 = 30.0
```

LIST

append

In [6]:

```
a = []
b = [1, 2, 3]
c = ["Hello", ",", "world"]
d = [1, 2, 3, "x", "y", "z"]
x = []
print(x)
x.append('a')
print(x)
x.append(123)
print(x)
x.append(["a", "b"])
print(x)
print("Length of x is %d " % (len(x)))


for i in range(len(x)):
    print("[%02d/%02d] %s" % (i, len(x), x[i]))

z = []
z.append(1)
z.append(2)
z.append(3)
z.append('Hello')
for i in range(len(z)):
    print (z[i])

print(z)
```

```
[]
['a']
['a', 123]
['a', 123, ['a', 'b']]
Length of x is 3
[00/03] a
[01/03] 123
[02/03] ['a', 'b']
1
2
3
Hello
[1, 2, 3, 'Hello']
```

extend

In [7]:

```
a = []
b = [1, 2, 3]
c = ["Hello", ",", "world"]
d = [1, 2, 3, "x", "y", "z"]
x = []
print(x)
x.extend('a')
print(x)
#x.extend(123) # It has a problem
#print(x)
x.extend(["a", "b"])
print(x)
print("Length of x is %d " % (len(x)))

print("\nusing index")
for i in range(len(x)):
    print("[%02d/%02d] %s" % (i, len(x), x[i]))

z = []
#z.extend(1) # It has a problem
#z.extend(2)
#z.extend(3)
z.extend('Hello')
for i in range(len(z)):
    print (z[i])

print(z)
```

```
[]
['a']
['a', 'a', 'b']
Length of x is 3

using index
[00/03] a
[01/03] a
[02/03] b
H
e
l
l
o
['H', 'e', 'l', 'l', 'o']
```

DICTIONARY

In [8]:

```
dic1 = dict()
dic1["name"] = "Sanghyun"
dic1["heights"] = 171
```

```
dic1["research area"] = "Artificial Intelligence"
print("case1: ", dic1)

dic2 = {}
dic2['heights'] = 171
dic2 = {'heights': 186}
dic2['research area'] = "Artificial Intelligence"
dic2 = {"name": "Sanghyun2"}
print("case2: ", dic2)

dic3 = {"name": "Sanghyun3"}
dic3['heights'] = 171
dic3['heights'] = 186
dic3['research area'] = "Artificial Intelligence"
print("case3: ", dic3)
```

```
case1:  {'name': 'Sanghyun', 'heights': 171, 'research area': 'Artificial Intelligence'}
case2:  {'name': 'Sanghyun2'}
case3:  {'name': 'Sanghyun3', 'heights': 186, 'research area': 'Artificial Intelligence'}
```

Class

In [9]:

```
class Greeter:
    # Constructor
    def __init__(self, name):
        self.name = name  # Create an instance variable

    # Instance method
    def greet(self, loud=False):
        if loud:
            print ('HELLO, %s!' % self.name.upper())
        else:
            print ('Hello, %s' % self.name)

g = Greeter('world')  # Construct an instance of the Greeter class
g.greet()             # Call an instance method; prints "Hello, world"
g.greet(loud=True)    # Call an instance method; prints "HELLO, WORLD!"


#lower?
```

```
Hello, world
HELLO, WORLD!
```

In [10]:

```
class Greeter:
    # Constructor
    def __init__(self, name):
        self.name = name  # Create an instance variable

    # Instance method
    def greet(self, quiet=False):
        if quiet:
            print ('Hello, %s'
                   % self.name.lower())
        else:
            print ('HELLO, %s!'
                   % self.name)

g = Greeter('WORLD')  # Construct an instance of the Greeter class
g.greet()             # Call an instance method; prints "HELLO, WORLD!"
g.greet(quiet=True)   # Call an instance method; prints "Hello, world!"
```

```
HELLO, WORLD!
Hello, world
```