Print

```
print("Hello World")
print("10") # print as character
print(10) # print as number
```

```
    Hello World
    10
    10
```

end

```
# space between comma
print("Hello", "World")

print("Hello", end="")
print("World")
print("Hello", end=" ")
print("World")
print("Hello", end=",")
print("World")
```

```
    Hello World
    HelloWorld
    Hello World
    Hello,World
```

```
# number
print(10+5)
print(10-5)
print(10*5)
print(10/5)
print("10+5")
```

```
    15
    5
    50
    2.0
    10+5
```

%d, %f, %s

- %d : nteger
- %f : real number
- %s : character

```
print("number : 100")
print("number : %d" %100) # number 100 goes into %d
print("number : %d" %100.5) # since %d is integer, 100.5 wll give 100

print("real number : %f" %100.5)

print("character : %s" %"python")
print("character : %s" %100) # since %s is character, 100 is rather a character than a number
```

```
    number : 100
    number : 100
    number : 100
    real number : 100.500000
    character : python
    character : 100
```

```
# example
# Q: print 5 + 3 = 8, but you should use addition for the result

print("%d + %d = %d" %(5, 3, 5+3))
```

```
    5 + 3 = 8
```

```
# %num d : there are num spaces in printing

print("%5d" %100)
print("%5d" %100000)
print("%5d" %1000000) # just push back
print("%05d" %100) # fill rest spaces with zero
```

```
  100
100000
1000000
00100
```

```
print("%6.2f" %123.45) # there are total of 6 spaces and 2 decimal spaces
print("%6.2f" %123.456) # round up to 2 decimal spaces
print("%6.2f" %123.4) # fill the rest empty with zero
```

```
123.45
123.46
123.40
```

```
# example
# Q: 5 / 3 = ?; one with integer and one with real number with 3 decimal spaces

print("%d / %d = %d" %(5, 3, 5/3))
print("%d / %d = %.3f" %(5, 3, 5/3))
```

```
5 / 3 = 1
5 / 3 = 1.667
```

format function

```
print("{0} {1}".format("apple", "banana"))
print("{1} {0}".format("apple", "banana"))
print("{1} {2} {0}".format("apple", " ", "banana"))
```

```
apple banana
banana apple
  banana apple
```

```
# example
print("The age of {0:5s} is {1:03d}.".format("Apple", 20))
print("The height of {0} is {1:6.2f}.".format("Banana", 160.5))
```

```
The age of Apple is 020.
The height of Banana is 160.50.
```

Variable

```
name = "Apple"
age = 20

print(name)
print(age)

name = "Banana"

print(name)
print(age)
```

```
Apple
20
Banana
20
```

```
num1 = 10
num2 = 3
print("%d / %d = %.3f" %(num1, num2, num1 / num2))
```

```
10 / 3 = 3.333
```

```
name = "Apple"
age = 20
weight = 50.5
isLover = False

print(type(name))
print(type(age))
print(type(weight))
print(type(isLover))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
```

```
print("Her name is {0}. She is {1} year's old. Her wegiht is {2}. Is she a lover? Answer is {3}".format(name, age, weight, isLover))
```

```
Her name is Apple. She is 20 year's old. Her wegiht is 50.5. Is she a lover? Answer is False
```

```
# delete the variable
del isLover

# print(isLover)
# NameError : name 'isLover' is not defined
```

Input

- gain data from the user
- all data are treated as character (although the user type numbers)

```
name = input()
print(name)
```

```
Apple
Apple
```

```
name = input("Your name : ")
print("Hello %s!" %name)
```

```
Your name : Apple
Hello Apple!
```

```
# split() : split a given string or a line by specifying one of the substrings of the given string as the delimiter
name, age, height = input("Write your name, age, and height : ").split(" ")
print("{} is {} years old and {} ft.".format(name, age, height))
```

```
Write your name, age, and height : Apple 20 6'1
Apple is 20 years old and 6'1 ft.
```

```
name, age, height = input("Write your name, age, and height : ").split(",")
print("{} is {} years old and {} ft.".format(name, age, height))
```

```
Write your name, age, and height : Apple,20,6'1
Apple is 20 years old and 6'1 ft.
```

```
# example
# Get grades for English, Math, and Science
# Then calculate the total and average scores

print("This is average calculator")
eng, math, sci = input("Please put down your English, Math, and Science scores : ").split(" ")

total = int(eng) + int(math) + int(sci)
avg = float(total / 3)

print("The total score is %.2f" %total)
print("The average socre is %.2f" %avg)
```

```
This is average calculator
Please put down your English, Math, and Science scores : 100 72 88
The total score is 260.00
The average socre is 86.67
```

String

```
str1 = "Hello World"
str2 = "Hello World Hello Python"
str3 = '''Hello
World
Hello
Python
'''

print(str1)
print(str2)
print(str3)
```

```
Hello World
Hello World Hello Python
Hello
World
Hello
Python
```

Index

```
# python index starts from 0
print(str1[0])
print(str1[1])
print(str3[6])
print(str3[7])
```

```
H
e


W
```

```
print(str2[0:8])
print(str2[7:11])
print(str2[6:-1]) # negative indicates the end
print(str2[:15])
print(str2[12:])
print(str2[::3]) # return index with the multiple of 3
```

```
Hello Wo
orld
World Hello Pytho
Hello World Hel
Hello Python
HlWlHlPh
```

len()

- returns the length of the string

```
len(str2)
```

```
24
```

```
len(str3)
```

```
27
```

count()

- returns the frequency of appearance of parameter

```
print(str2.count("Hello"))
print(str2.count("World"))
```

```
2
1
```

find()

- returns the index of character that first appear

```
print(str2.find("e"))
print(str2.find("q")) # if does not exist, returns -1
```

```
1
-1
```

index()

returns the index of character that first appears

```
print(str2.index("e"))
# print(str2.index("q")) # if not exist, then error
# ValueError: substring not found
```

```
1
```

replace()

```
print(str2.replace("World", "Python"))
```

```
Hello Python Hello Python
```

upper(), lower()

```
print(str2.upper())
print(str2.lower())
```

```
HELLO WORLD HELLO PYTHON
hello world hello python
```

strip()

- remove the spaces at the end or the start
- lstrip() : remove the starting space
- rstrip() : remove the end space

```
str4 = "      Hello World       "
print(str4)
print(str4.lstrip())
print(str4.rstrip())
print(str4.strip())
```

```
      Hello World
Hello World
      Hello World
Hello World
```

join()

- add specific character into the string

```
str5 = "hello"
print("❤️".join(str5))
```

```
h❤️e❤️l❤️l❤️o
```

List

- separate the data by , and save them into []
- different type of data can be stored in the same list

```
li1 = [1, 3, 5, 7]
print(li1)

li2 = ["Apple", "Banana", "Cookie"]
print(li2)

li3 = [1, "1", "Apple"]
print(li3)

li4 = [1, 2, "3", ["Apple", "Banana"], True]
print(li4)
```

```
[1, 3, 5, 7]
['Apple', 'Banana', 'Cookie']
[1, '1', 'Apple']
[1, 2, '3', ['Apple', 'Banana'], True]
```

```
# list indexing
print(li1[0])
print(li4[:4])
print(li1[0] + li1[1])
print(li4[3][-1])
```

```
1
[1, 2, '3', ['Apple', 'Banana']]
4
Banana
```

```
li5 = [10, 20, 30]
li6 = [40, 50, 60]

print(li5 + li6)
print(li5 * 3)
li6[1] = '🖤'
# print(li6[0] + li6[1]) TypeError: unsupported operand type(s) for +: 'int' and 'str'
print(li6[0] + li6[2])
```

```
[10, 20, 30, 40, 50, 60]
[10, 20, 30, 10, 20, 30, 10, 20, 30]
100
```

```
# modification in the list

li7 = [10, 20, 30]
li7[1] = 100
print(li7)

li7[1:2] = ['🥝', '🍇', '🍓']
print(li7)

li7[1] = ['🥝', '🍇', '🍓']
print(li7)

print(li7[0:3])

li7[1:3] = [] # delete the elements
print(li7)

del li7[0]
print(li7)
```

```
[10, 100, 30]
[10, '🥝', '🍇', '🍓', 30]
[10, ['🥝', '🍇', '🍓'], '🍇', '🍓', 30]
[10, ['🥝', '🍇', '🍓'], '🍇']
[10, '🍓', 30]
['🍓', 30]
```

list functions

```
# append() : add the element in last

li8 = [10, 20, 30]
print(li8)

li8.append(40)
print(li8)
```

```
[10, 20, 30]
[10, 20, 30, 40]
```

```
# pop() : return the last element and delete it

li9 = [10, 20, 30, 40]
print(li9)

print(li9.pop())
print(li9)
```

```
[10, 20, 30, 40]
40
[10, 20, 30]
```

```
# remove() : delete the data that first appeared

li10 = [10, 20, 30, 40, 20, 10]
li10.remove(10)
print(li10)
```

```
[20, 30, 40, 20, 10]
```

```
# insert () : add data in specific place

li11 = [10, 20, 30]
li11.insert(1, 100)
print(li11)

li11.insert(2,(500, 400))
print(li11)
```

```
[10, 100, 20, 30]
[10, 100, (500, 400), 20, 30]
```

```
# index() : return the index if exists
print(li11.index(10))
# print(li11.index(80)) #ValueError: 80 is not in list
```

```
0
```

```
# reverse() : reverse the order of the list
li12 = [10, 20, 30, 40, 50]
li12.reverse()
print(li12)
```

```
[50, 40, 30, 20, 10]
```

```
# sort() : order the list in alphabetically or increasing

li13 = ["cookie", "egg", "donut", "apple", "banana"]
li13.sort()
print(li13)

li13.sort(reverse=True)
```

```
print(li13)

li14 = ['4', '6', '9', "apple", "cookie", "&", "%", "egg", '100']
li14.sort()
print(li14)
```

```
['apple', 'banana', 'cookie', 'donut', 'egg']
['egg', 'donut', 'cookie', 'banana', 'apple']
['%', '&', '100', '4', '6', '9', 'apple', 'cookie', 'egg']
```

```
# count() : return the number of appearance

li15 = [1, 1, 2, 2, 2, 2, 5]
print(li15.count(1))
print(li15.count(2))
print(li15.count(5))
```

```
2
4
1
```

Tuple

- share similar characteristics to list but use ()
- cannot eidt or delete

```
tu1 = ()
print(tu1)

tu2 = (1, 3, 5, 7)
print(tu2)
print(tu2[1])

tu3 = 1, 3, 5, 6
print(tu3)

tu4 = ('apple', 'banana', ('🍔', '🍟'))
print(tu4)

tu5 = ('apple', 'banana', ['🍔', '🍟'])  # can insert the list inside of the tuple
print(tu5)
print(type(tu5))
```

```
()
(1, 3, 5, 7)
3
(1, 3, 5, 6)
('apple', 'banana', ('🍔', '🍟'))
('apple', 'banana', ['🍔', '🍟'])
<class 'tuple'>
```

```
tu6 = (1, 2, "apple", "banana")

# indexing & slicing
print(tu6[0])
print(tu6[:-1])
print(tu6[::2])

print(tu6 + (30, 40))

print(tu6 * 3)

print(len(tu6))
```

```
1
(1, 2, 'apple')
(1, 'apple')
(1, 2, 'apple', 'banana', 30, 40)
(1, 2, 'apple', 'banana', 1, 2, 'apple', 'banana', 1, 2, 'apple', 'banana')
4
```

If-Else

```
print(10 > 5)
print(10 < 5)
print (10>= 5)
print(10 <= 5)
print(10 != 5)
print(10 == 5)
```

```
    True
    False
    True
    False
    True
    False
```

```
# If

age = int(input("What is your age? "))

print("Your age is ", age)
if age >= 21:
  print("You are eligible to buy alcohol.")
```

```
    What is your age? 20
    Your age is  20
```

```
# if-else
age = int(input("What is your age? "))

print("Your age is ", age)
if age >= 21:
  print("You are eligible to buy alcohol.")
else: # age < 21
  print("You are NOT eligible to buy alcohol.")
```

```
    What is your age? 20
    Your age is  20
    You are NOT eligible to buy alcohol.
```

```
#if-elif-else

score = int(input("What is your score on exam? "))

print("Your score is ", score)
if score >= 90:
  print("Your grade is A")
elif score >= 80:
  print("Your grade is B")
elif score >= 70:
  print("Your grade is C")
elif score >= 60:
  print("Your grade is D")
else: # score < 60
  print("You failed the exam.")
```

```
    What is your score on exam? 89
    Your score is  89
    Your grade is B
```

```
# AND
userid = input("Type your ID : ")
userpw = input("Type your Password :")

if userid == "admin" and userpw == "1234":
  print("You successfully logged in")
else:
  print("Please check your ID or Password")
```

```
    Type your ID : kim
    Type your Password :1234
    Please check your ID or Password
```

```
# OR
position = input("What is your position at UW? ")

if position == "student" or position == "faculty":
  print("Welcome Huskies")
elif position == "alumni":
  print("Welcome Huskies alumni")
else:
  print("You are ineligible to access UW resources")
```

```
What is your position at UW? none
You are ineligible to access UW resources
```

While loop

- if condition is true, then repeat

```
num = 10

num += 1
print(num)

num -= 1
print(num)

num *= 10
print(num)

num /= 10
print(num)

num %= 2 # residual
print(num)
```

```
11
10
100
10.0
0.0
```

```
i = 1

while i <= 5:
  print("Hello Python")
  i += 1

print("current i is : {}".format(i))
```

```
Hello Python
Hello Python
Hello Python
Hello Python
Hello Python
current i is : 6
```

```
# example
# add 1 to 10

i = 1
sum = 0

while i <= 10:
  sum += i
  i += 1

print(sum)
```

```
55
```

```
# example
# generate multiplication table with the number user asks for
```

```
num = int(input("Which multiplication table do you need? "))
num2 = 0
multiple = 0

while num2 <= 20:
  multiple = num * num2
  print("%d x %d = %d" %(num, num2, multiple))
  num2 += 1
```

```
Which multiplication table do you need? 5
5 x 0 = 0
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
5 x 12 = 60
5 x 13 = 65
5 x 14 = 70
5 x 15 = 75
5 x 16 = 80
5 x 17 = 85
5 x 18 = 90
5 x 19 = 95
5 x 20 = 100
```

for loop

```
for i in range(10):
  print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

```
for i in range(2,11,2):
  print(i)
```

```
2
4
6
8
10
```

```
for i in range(10):
  print(i, end=' ')
```

```
0 1 2 3 4 5 6 7 8 9
```

```
# example
# Q: find the sum of even number from 1 to 100

sum = 0

for i in range(2, 101, 2):
  sum += i

print(sum)
```

```
2550
```

```
# example
# Q: find the sum of even number from 1 to 100 using if else instead of range(2, 101, 2)

sum = 0

for i in range(1, 101):
  if (i%2) == 0:
    sum += i

print(sum)
```

```
2550
```

```
# example
# Q: find the sum of even number form 1 to 100 using while loop

i = 0
sum = 0

while i <= 100:
  if (i%2) == 0:
    sum += i
  i += 1

print(sum)
```

```
2550
```

```
for i in range(1, 6):
  for j in range(1,6):
    print('🖤', end=' ')
  print()
```

```
🖤 🖤 🖤 🖤 🖤
🖤 🖤 🖤 🖤 🖤
🖤 🖤 🖤 🖤 🖤
🖤 🖤 🖤 🖤 🖤
🖤 🖤 🖤 🖤 🖤
```

```
for i in range(5):
  for j in range(i, 5):
    print('🖤', end=' ')
  print()
```

```
🖤 🖤 🖤 🖤 🖤
🖤 🖤 🖤 🖤
🖤 🖤 🖤
🖤 🖤
🖤
```

```
for i in range(5):
  for j in range(i+1):
    print('🖤', end=' ')
  print()
```

```
🖤
🖤 🖤
🖤 🖤 🖤
🖤 🖤 🖤 🖤
🖤 🖤 🖤 🖤 🖤
```

Set

- the order is random when store the data in the set but not allow the repeition of the same element

```
s1 = {1, 3, 5, 7}
print(s1)

s2 = set([1, 3, 5, 7])
print(s2)
```

```
s3 = {1, 3, 5, 7, 9, 9, 3, 7}
print(s3)

     {1, 3, 5, 7}
     {1, 3, 5, 7}
     {1, 3, 5, 7, 9}
```

```
print(2 in s2)
print(7 in s3)
print(3 not in s1)

     False
     True
     False
```

set functions

```
# add() : add a single data in the set
s1 = {100, 200}
s1.add(150)
print(s1)
s1.add(20)
print(s1)

# update() : add multiple data in the set
s2 = {10, 20, 30}
s2.update([40, 50, 60, 20])
print(s2)

# remove() : remove the data in the set, if not exist, return error
s2.remove(50)
print(s2)
#s2.remove(50)  KeyError: 50

# discard() : remove the data in the set, if not exist, does not return error
s2.discard(30)
print(s2)
s2.discard(30)  #데이터가 없어도 에러가 나지 않고 결과값이 그대로 나옴
print(s2)

# copy() : copt the set
s3 = {10, 20, 30}
s4 = s3.copy()
print(s3)
print(s4)

#s5[0] = 100  #TypeError: 'set' object does not support item assignment
# since the order is random, there is no indexing

     {200, 100, 150}
     {200, 100, 20, 150}
     {40, 10, 50, 20, 60, 30}
     {40, 10, 20, 60, 30}
     {40, 10, 20, 60}
     {40, 10, 20, 60}
     {10, 20, 30}
     {10, 20, 30}
```

```
s1 = {10, 20, 30}
s2 = {30, 40, 50, 60, 70}

# union : |
result = s1 | s2
print(result)
result = s1.union(s2)
print(result)

# intersecton : &
result = s1 & s2
print(s2)
result = s1.intersection(s2)
print(result)

# defference : -
result = s1 - s2
```

```
print(result)
result = s1.difference(s2)
print(result)

# symmetric_difference : ^  (2개 합집합에서 교집합 뺀거)
result = s1 ^ s2
print(result)
result = s1.symmetric_difference(s2)
print(result)
```

```
{70, 40, 10, 50, 20, 60, 30}
{70, 40, 10, 50, 20, 60, 30}
{50, 70, 40, 60, 30}
{30}
{10, 20}
{10, 20}
{50, 20, 70, 40, 10, 60}
{50, 20, 70, 40, 10, 60}
```

Dictionary

- combination of key & value
- key cannot be repeated
- value can be repeated

```
#{key:value}

dic1 = {}
print(dic1)

dic2 = {'no':1, 'userid':'apple', 'name':'Apple', 'hp':'010-1111-1111'}
print(dic2)
```

```
{}
{'no': 1, 'userid': 'apple', 'name': 'Apple', 'hp': '010-1111-1111'}
```

```
print(dic2['userid'])
print(dic2['name'])
```

```
apple
Apple
```

```
# add data
dic3 = {1:"apple"}
print(dic3)

dic3[100] = "orange"
print(dic3)
```

```
{1: 'apple'}
{1: 'apple', 100: 'orange'}
```

Dictionary function

```
dic2 = {'no':1, 'userid':'banna', 'name':'Apple', 'number':'206-111-1111'}

# keys() : only converts key to list
print(dic2.keys())

# values() : only converts value to list
print(dic2.values())

# get() : get value by using key
print(dic2["userid"])
print(dic2.get("userid"))
print(dic2.get('age', 'unknown'))

# in : check if key is in dictionary
print('name' in dic2)
print('age' in dic2)
```

```
    dict_keys(['no', 'userid', 'name', 'number'])
    dict_values([1, 'banna', 'Apple', '206-111-1111'])
    banna
    banna
    unknown
    True
    False
```

Function

- user define function to repetitively use

```
def func1():
  print("This is first function I create!")

func1()
```

```
    This is first function I create!
```

```
# contain parameter
def func2(num):
  print("Input number : %d" %num)

func2(10)
func2(20)
```

```
    Input number : 10
    Input number : 20
```

```
# contain return value
def func3():
  return('🎁')

func3()
```

```
    '🎁'
```

```
present = func3()
present
```

```
    '🎁'
```

```
def func4(num1, num2):
  return num1 + num2

func4(1, 2)
```

```
    3
```

```
# default parameter
def func5(num1 = 1, num2 = 2):
  return num1 + num2

# func4(1) # TypeError
```

```
print(func5(num1 = 2, num2 = 2))
print(func5(2,2))
print(func5(2))
```

```
    4
    4
    4
```

```
# parameter variability

def func6(*args):
  result = 0
  for i in args:
    result += i
  return result

print(func6(10))
print(func6(1,2,3,4,5))
```

```
    10
    15
```

```
# parameter as dictionary
def func7(**args):
  return args

user = func7(userid="apple", name="Apple", age=20)
print(user)
```

```
    {'userid': 'apple', 'name': 'Apple', 'age': 20}
```

Class & Object-Oriented Programming (OOP)

create class

```
class name:
  def __init__(self): # _init_ is the first thing to proceed no matter what
    self.field1 = value1
    self.feild2 = value2
    ...
  def method1 (param1, param2, ...):
```

```
# class with no function

class Dog:
  pass # nothing saved
```

```
def func1():
  pass
```

```
# create object through class
Rucy = Dog() #Rucy is indicating the address of Dog class
print(Rucy, type(Rucy))

Choco = Dog()
print(Choco, type(Choco))
```

```
    <__main__.Dog object at 0x7fb0849166e0> <class '__main__.Dog'>
    <__main__.Dog object at 0x7fb084915690> <class '__main__.Dog'>
```

```
# __init__
class Dog:
  def __init__(self):
    print(self, "call init")
```

```
Rucy = Dog()
print(Rucy)
```

```
Choco = Dog()
print(Choco)
```

```
<__main__.Dog object at 0x7fb084914430> call init
<__main__.Dog object at 0x7fb084914430>
<__main__.Dog object at 0x7fb0849166e0> call init
<__main__.Dog object at 0x7fb0849166e0>
```

```
class Dog:
  def __init__(self):
    print(self, "call init")
    self.name = "no name"
    self.age = 0
```

```
Rucy = Dog()
print(Rucy)
print(Rucy.name)
print(Rucy.age)
```

```
<__main__.Dog object at 0x7fb0849160b0> call init
<__main__.Dog object at 0x7fb0849160b0>
no name
0
```

```
Rucy.name = "Rucy"
Rucy.age = 12
print(Rucy.name)
print(Rucy.age)
```

```
Rucy
12
```

```
Choco = Dog()
print(Choco.name)
print(Choco.age)
```

```
<__main__.Dog object at 0x7fb084917520> call init
no name
0
```

```
class Dog:
  def __init__(self, name, age, family="retriever"):
    self.name = name
    self.age = age
    self.family = family # retriever is the default setting
```

```
Rucy = Dog("Rucy", 12)
print(Rucy.name)
print(Rucy.age)
print(Rucy.family)

Choco = Dog("Choco", 6, "pomeranian")
print(Choco.name)
print(Choco.age)
print(Choco.family)
```

```
Rucy
12
retriever
Choco
6
pomeranian
```

```
# define method
class Counter:
  def __init__(self): # the first method to proceed
    self.num = 0
  def increment(self):
    self.num += 1
  def current_value(self):
    return self.num
  def reset(self):
    self.num = 0



Chase = Counter()
print(Chase.num)

Chase.increment()
Chase.increment()
print(Chase.num)

print(Chase.current_value())

print("Waiting line : %d" %Chase.current_value())

Chase.reset()
print("New Waiting line : %d" %Chase.current_value())
```

```
0
2
2
Waiting line : 2
New Waiting line : 0
```

```
# staticmethod : do not need to create object

class Math:
  @staticmethod
  def add(x,y):
    return x+y
  def multiply(x,y):
    return x*y



result1 = Math.add(10, 5)
result2 = Math.multiply(10, 5)
result1, result2
```

```
(15, 50)
```

Inherit the class

- inherit the class
- can modify inherited class
- call inheriting class - parent, super, base
- call the inherited class - child, sub

```
class Animal:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def eat(self, food):
    print("{} is eating {}".format(self.name, food))

  def sleep(self, hour):
    print("{} is sleep {} hours".format(self.name, hour))


cat = Animal("Bella", 3)
cat.eat('tuna')
cat.sleep('10')
```

```
Bella is eating tuna
Bella is sleep 10 hours
```

```
# inherit - class child_name (parent_name)

class Cat(Animal):
  pass


Bella = Cat("Bella", 6)
Bella.eat("tuna")
Bella.sleep(10)
```

```
    Bella is eating tuna
    Bella is sleep 10 hours
```

```
# overwrite

class Cat(Animal):
  def eat(self, food, count):
    print('{} is eatng {} of {}.'.format(self.name, food, count))
```

```
Bella = Cat("Bella", 6)
Bella.eat("tuna", 3)
Bella.sleep(10)
```

```
    Bella is eatng tuna of 3.
    Bella is sleep 10 hours
```

```
# multiple inheritance - ** parent class must have same __init__ paramenter

class Human:
  def __init__ (self, name, age):
    self.name = name
    self.age = age

  def study(self, hour):
    print("{} is studying {} hours".format(self.name, hour))
```

```
class Kim(Animal, Human):
   pass
```

```
kim = Kim("Apple", 20)
kim.eat("rice")
kim.study(4)
```

```
    Apple is eating rice
    Apple is studying 4 hours
```

Module

- bring a module with all functions

```
import module_name
# or
from module_name import *
```

- bring a module with a specific function

```
from module_name  import function_name
```

- give nickname to module

```
import module_name as module_nickname
```

- how to use

```
module_name.function_name()
# or
module_nickname.function_name()
```

Try-Except

```
try:
  sentence expected to be error
  ...
  ...
except typeerror1:
  if this error happens, sentence trying to proceed
  ...
  ...
except typeeroor2:
  if this error happens, sentence trying to proceed
  ...
  ...
else:
  when no error occurs
  ...
finally:
  no matter what, proceeds
```

```
print(10/2)
print(5/0)  # ZeroDivisionError: division by zero
print(4/2)
```

```
5.0
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-158-32f30cfde485> in <cell line: 2>()
      1 print(10/2)
----> 2 print(5/0)  # ZeroDivisionError: division by zero
      3 print(4/2)

ZeroDivisionError: division by zero
```

SEARCH STACK OVERFLOW

```
try:
  print(10/2)
  print(5/0)  # ZeroDivisionError: division by zero
  print(4/2)
except ZeroDivisionError:
  print('Cannot divide it by zero')

print('program ends')
```

```
5.0
Cannot divide it by zero
program ends
```

```
data = [10, 20, 30]
print(data[3])  # IndexError: list index out of range
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-160-327616b9023c> in <cell line: 2>()
      1 data = [10, 20, 30]
----> 2 print(data[3])  # IndexError: list index out of range

IndexError: list index out of range
```

SEARCH STACK OVERFLOW

```
data = [10, 20, 30]
try:
  print(data[2])
  print(data[3])
  print(data[0])
```

```
except ZeroDivisionError:
  print('Cannot divide it by zero')
except:  # all kinds of errors
  print('Error occurred')

print('Program ends')
```

```
    30
    Error occurred
    Program ends
```

```
data = [10, 20, 30]
try:
  print(data[2])
  print(data[3])
  print(data[0])
except ZeroDivisionError:
  print('Cannot divide iy by zero')
#except:
#  print('Error occurred')   >> Order mismatch
except IndexError:
  print('Index is not correct')
except:
  print('Error occurred')
finally:
  print('Program ends')
```

✓ 0s completed at 11:52 PM