Lecture 2
Python Basic

Jaeyun Kang

Python Basic - Function

```
def (function_name) ((parameter_name)):
        statement1
        statement2
        ...


def sum(a, b):
        return a + b

print (sum(3, 4)) // 7
```

Python Basic - Function

```
def sum(a, b):
        result = a + b
        return result

a = sum(3, 4)
print (a) // 7
```

Python Basic - Function

```
def sum(a, b):
        print (a + b)

sum(3, 4) // 7

a = sum(3, 4)
print (a) // none
```

Python Basic - Function

```
def say():
        return 'Hi'

a = say()
print (a) // Hi

def say():
        print ('Hi')

say() // Hi
```

Python Basic - Function

```
def sum_and_mul(a, b):
        return a+b, a*b

a = sum_and_mul(3, 4) // a = (7, 12)
sum, mul = sum_and_mul(3, 4) // sum=7, mul=12

def sum_and_mul(a, b):
        return a+b
        return a*b

a = sum_and_mul(3, 4) // a = 7
```

Python Basic - Function

```
def say_nick(nick):
        if nick == "바보":
                return
        print("나의 별명은 %s입니다." %nick)

say_nick('야호') // 나의 별명은 야호입니다.
say_nick('바보') // nothing
```

Python Basic - Function

```
def say_myself(name, old, man = True):
        print("나의 이름은 %s 입니다." % name)
        print("나의 이름은 %d살입니다." % old)
        if man:
                print("남자입니다")
        else:
                print("여자입니다")

say_myself("박응용", 27)
say_myself("박응용", 27, True)
// 나의 이름은 박응용입니다.
// 나이는 27살입니다.
// 남자입니다.
```

Python Basic - Function

```
def say_myself(name, old, man = True):
        print("나의 이름은 %s 입니다." % name)
        print("나의 이름은 %d살입니다." % old)
        if man:
                print("남자입니다")
        else:
                print("여자입니다")

say_myself("박응선", 27, False)
// 나의 이름은 박응선입니다.
// 나이는 27살입니다.
// 여자입니다.
```

Python Basic - Function

```
def say_myself(name, man = True, old):
        print("나의 이름은 %s 입니다." % name)
        print("나의 이름은 %d살입니다." % old)
        if man:
                print("남자입니다")
        else:
                print("여자입니다")

say_myself("박응용", 27) // error
```

Python Basic - Function

```
a = 1
def vartest(a):
        a = a + 1

vartest(a)
print(a) // 1

def vartest(hello):
        hello = hello + 1
```

Python Basic - Function

```
a = 1
def vartest(a):
        a = a + 1
        return a

a = vartest(a)
print(a) // 2
```

Exercise

1. Make function which calculates and returns an average of three numbers

avg(1, 2, 3) should return 2

```
print (avg(1, 2, 3)) // 2
```

2. Make function which calculates n factorial

fact(4) should return 24

```
print (fact(4)) // 24
```

Python Basic - Read and Write

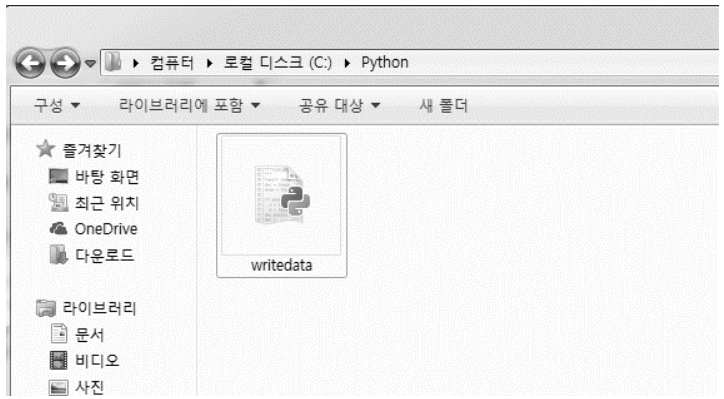
```
# writedata.py
f = open("새파일.txt", 'w')
f.close()
```

'r': reading mode
'w': writing mode
'a': appending mode

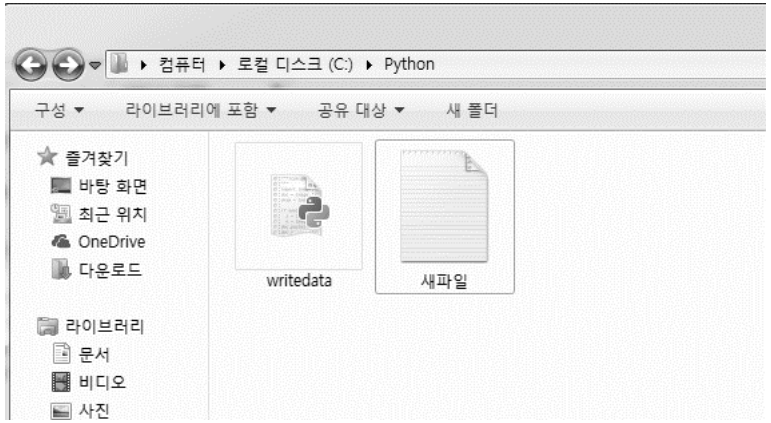
Python Basic - Read and Write

```
f = open("C:/Python/새파일.txt", 'w')
for i in range(1, 11):
        data = "%d번째 줄입니다.₩n" % i
        f.write(data)
f.close()
```

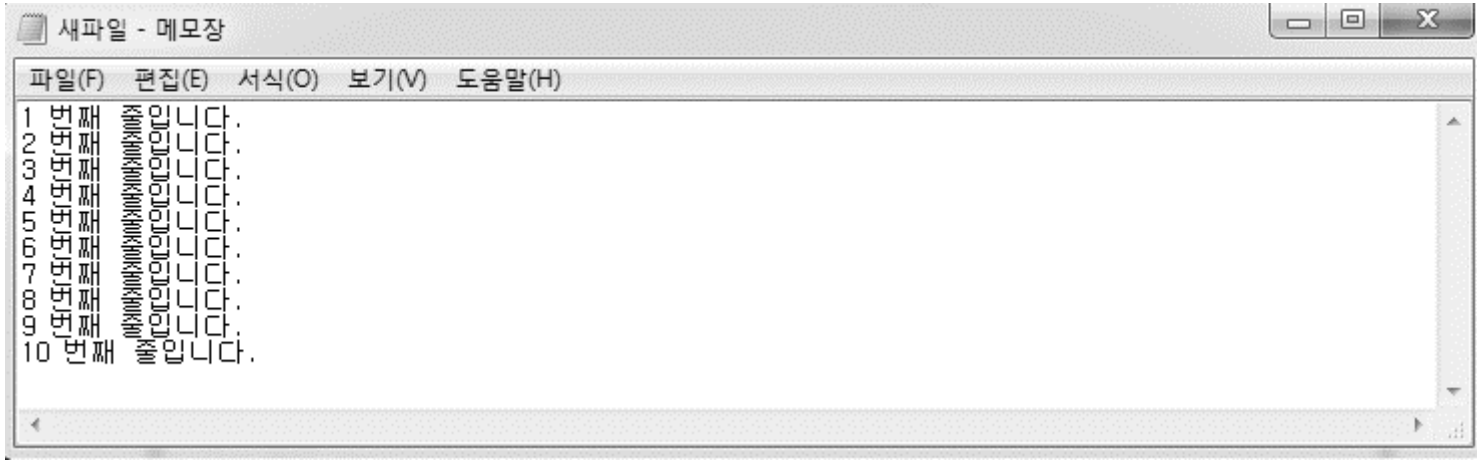
Python Basic - Read and Write



writedata.py
실행



Python Basic - Read and Write



Python Basic - Read and Write

```
f = open("C:/Python/새파일.txt", 'r')
while True:
        line = f.readline()
        if not line: break
        print(line)
f.close()
```

Python Basic - Read and Write

```
f = open("C:/Python/새파일.txt", 'r')
lines = f.readlines()
for line in lines:
        print(line)
f.close()

f = open("C:/Python/새파일.txt", 'r')
data = f.read()
print(data)
f.close()
```

Python Basic - Read and Write

```
f = open("C:/Python/새파일.txt",'a')
for i in range(11, 20):
        data = "%d번째 줄입니다.₩n" % i
        f.write(data)
f.close()
```

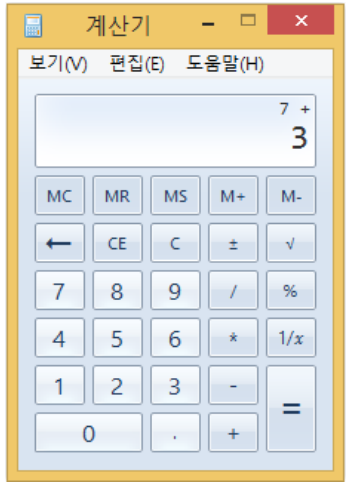
Python Basic - Read and Write

```
f = open("foo.txt", 'w')
f.write("Life is too short, you need python")
f.close()

with open("foo.txt", "w") as f:
    f.write("Life is too short, you need python")
```

Python Basic - Class

계산기

1. 사칙연산 함수가 정의되어 있어야 한다.
2. 이전에 계산된 결과값을 기억하고 있어야 한다.



Python Basic - Class

```
result = 0
def adder(num):
        global result
        result += num
        return result
print(adder(3)) // 3
print(adder(4)) // 7
```

Python Basic - Class

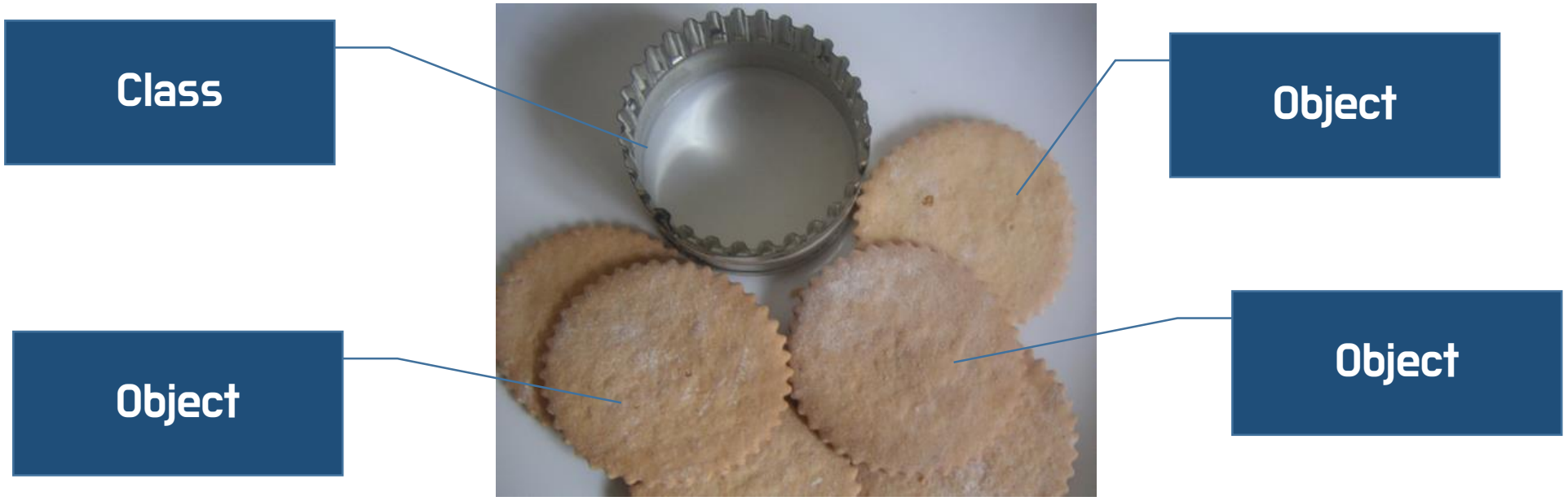
```
result1 = 0
result2 = 0
def adder1(num):
    global result1
    result1 += num
    return result1
def adder2(num):
    global result2
    result2 += num
    return result2

print(adder1(3)) // 3
print(adder1(4)) // 7
print(adder2(3)) // 3
print(adder2(7)) // 10
```

Python Basic - Class

```
class Calculator:
    def __init__(self):
        self.result = 0

    def adder(self, num):
        self.result += num
        return self.result

cal1 = Calculator()
cal2 = Calculator()

print(cal1.adder(3)) // 3
print(cal1.adder(4)) // 7
print(cal2.adder(3)) // 3
print(cal2.adder(7)) // 10
```

Python Basic - Class

Class & Object (클래스와 객체)



Python Basic - Class

```
class Programmer:
        pass

kim = Programmer()
park = Programmer()
```

*kim 은 Programmer Class의 Instance
*kim 은 Object

Python Basic - Class

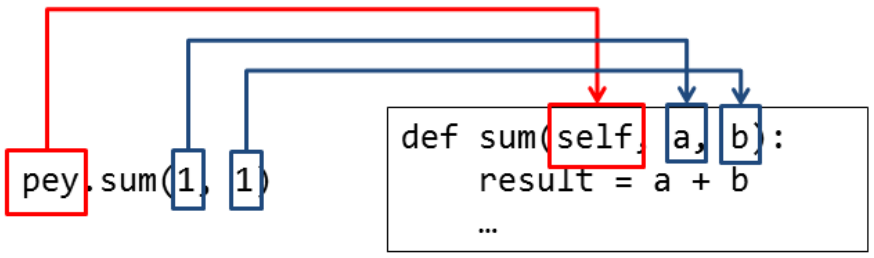
```
class Service:
        secret = "영구는 외계인이다."

pey = Service()
print(pey.secret) // "영구는 외계인이다."
```

Python Basic - Class

```
class Service:
        secret = "영구는 외계인이다."
        def sum(self, a, b):
                result = a + b
                print("%s + %s = %s입니다." % (a, b, result))

pey = Service()
pey.sum(1,1) // 1 + 1 = 2입니다.
```

Python Basic - Class

```
class Service:
        secret = "영구는 외계인이다."
        def sum(self, a, b):
                result = a + b
                print("%s + %s = %s입니다." % (a, b, result))

pey = Service()
pey.sum(1,1) // 1 + 1 = 2입니다.
```



Python Basic - Class

```
class Service:
        secret = "영구는 외계인이다."
        def setname(self, name):
                self.name = name
        def sum(self, a, b):
                result = a + b
                print("%s님 %s + %s = %s입니다." % (self.name, a, b, result))

pey = Service()
pey.setname("홍길동") // pey.name = name, pey.name = "홍길동"
pey.sum(1, 1) // 홍길동님 1 + 1 = 2입니다.
```

Python Basic - Class

```
babo = Service()
babo.sum(1, 1)   ???

class Service:
        secret = "영구는 외계인이다."
        def __init__(self, name):
                self.name = name
        def sum(self, a, b):
                result = a + b
                print("%s님 %s + %s = %s입니다." % (self.name, a, b, result))

pey = Service("홍길동")
pey.sum(1, 1) // 홍길동님 1 + 1 = 2입니다.
```

Python Basic - Class

```
class ClassName[(Inherit ClassName)]:
        〈Class Variable 1〉
        〈Class Variable 2〉
        …
        def Method1(self[, arg1, arg2…]):
                statement1
                statement2
        …
        def Method2(self[, arg1, arg2…]):
                statement1
                statement2
                …
        …
```

Python Basic - Class

#사칙연산 클래스 TODO

```
a = FourCal()
a.setdata(4, 2)
print(a.sum()) // 6
print(a.mul()) // 8
print(a.sub()) // 2
print(a.div()) // 2
```

Python Basic - Class

```
class FourCal:
        pass


a = FourCal()
print(type(a)) // <class '__main__.FourCal'>
```

Python Basic - Class

```
#TODO
a.setdata(4, 2)



class FourCal:
        def setdata(self, first, second):
                self.first = first
                self.second = second
```

Python Basic - Class

```
a = FourCal()
a.setdata(4, 2)
print(a.first) // 4
print(a.second) // 2

b = FourCal()
b.setdata(3, 7)
print(b.first) // 3
print(a.first) // 4
```

Python Basic - Class

```
#TODO
a = FourCal()
a.setdata(4, 2)
print(a.sum()) // 6

class FourCal:
        def setdata(self, first, second):
                self.first = first
                self.second = second
        def sum(self):
                result = self.first + self.second
                return result
```

Python Basic - Class

```
class FourCal:
        def setdata(self, first, second):
                self.first = first
                self.second = second
        def sum(self):
                result = self.first + self.second
                return result
        def mul(self):
                result = self.first * self.second
                return result
        def sub(self):
                result = self.first - self.second
                return result
        def div(self):
                result = self.first / self.second
                return result
```

Python Basic - Class

```
a = FourCal()
b = FourCal()
a.setdata(4, 2)
b.setdata(3, 7)

a.sum() // 6
a.mul() // 8
a.sub() // 2
a.div() // 2

b.sum() // 10
b.mul() // 21
b.sub() // -4
b.div() // 0
```

Python Basic - module

```
#mod1.py
def sum(a, b):
        return a + b
def mul(a, b):
        return a * b

import mod1
print (mod1.sum(3, 4)) // 7
print (mod1.mul(3, 4)) // 12
```

Python Basic - module

```
#mod1.py
def sum(a, b):
        return a + b
def mul(a, b):
        return a * b

import mod1 as m
print (m.sum(3, 4)) // 7
print (m.mul(3, 4)) // 12
```

Python Basic - module

```
#mod1.py
def sum(a, b):
        return a + b
def mul(a, b):
        return a * b

from mod1 import sum
print (sum(3, 4)) // 7
print (mul(3, 4)) // error
```

Python Basic - module

```
#mod1.py
def sum(a, b):
        return a + b
def mul(a, b):
        return a * b

from mod1 import sum, mul
print (sum(3, 4)) // 7
print (mul(3, 4)) // 12
```

Python Basic - module

```
#mod1.py
def sum(a, b):
        return a + b
def mul(a, b):
        return a * b

from mod1 import *
print (sum(3, 4)) // 7
print (mul(3, 4)) // 12
```

Python Basic - module

```
#mod1.py
def sum(a, b):
        return a + b
def mul(a, b):
        return a * b
print (sum(3, 4)) // 7
```

mod1.py 실행시 7 출력

Python Basic - module

```
#mod1.py
def sum(a, b):
        return a + b
def mul(a, b):
        return a * b
print (sum(3, 4)) // 7

#test.py
import mod1
```

test.py 실행시에도 7 출력 ?

Python Basic - module

```
#mod1.py
def sum(a, b):
        return a + b
def mul(a, b):
        return a * b
if __name__ == "__main__":
        print (sum(3, 4)) // 7
```

mod1.py 실행시 7 출력 (__name__ = "main")

Python Basic - module

```
#mod1.py
def sum(a, b):
        return a + b
def mul(a, b):
        return a * b
if __name__ == "__main__":
        print (sum(3, 4)) // 7

#test.py
import mod1
```

test.py 실행시에 7 출력 x (__name__ = "mod1.py")

Python Basic - module

```
# mod2.py
PI = 3.141592
class Math:
        def solv(self, r):
                return PI * (r ** 2)
def sum(a, b):
        return a+b

if __name__ == "__main__":
        a = Math()
        print(sum(PI , 4.4))
```

Python Basic - module

```
import mod2

print(mod2.PI) // 3.141592

a = mod2.Math()
print(a.solv(2)) // 12.566368

print(mod2.sum(mod2.PI, 4.4)) // 7.541592
```

Question