함수

함수

- 재사용 가능한 코드 구역
  - 동일한 코드, 동일한 작업을 반복해서 작성해야 하는 문제 해결
- 이름을 가질 수 있음
- 실행되기 전에 정의되어 있어야 함

```
def 함수명(입력 인수):
    <수행할 문장1>
    <수행할 문장2>
    ...
```

```
def printMessage():
    print("Hello World!")
    return
```

```
printMessage()
def printMessage():
    print("Hello World!")
    return
```

```
def main():
    printMessage()
    return

def printMessage():
    print("Hello World!")
    return

main()
```

함수

- 입력 매개변수

```
def main():
    names = ["greenjoa1","greenjoa2", "greenjoa3"]
    newName = input("Enter last guest : ")
    names.append(newName)
    printNames(names)
    return

def printNames(names):
    for name in names:
        print(name)
    return

main()
```

함수

- 입력의 개수를 모를 때
  - 입력 값을 튜플 형태로 만들어 줌

```
def 함수명(*args):
    <수행할 문장1>
    <수행할 문장2>
    ...
```

```
def sum_mul(choice, *args):
    if choice == "sum":
        result = 0
        for i in args:
            result = result + i
    elif choice == "mul":
        result = 1
        for i in args:
            result = result * i
    return result
```

```
result = sum_mul('sum', 1,2,3,4,5)
```

함수

- 값 반환하기
  - 값은 하나만 반환함

```
def main():
    names=getNames()
    printNames(names)
    return

def getNames():
    names = ["greenjoa1","greenjoa2", "greenjoa3"]
    newName = input("Enter last guest : ")
    names.append(newName)
    return names

def printNames(names):
    for name in names:
        print(name)
    return

main()
```

```
def main():
    #names=getNames()
    printNames(getNames())
    return
```

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

* 튜플로 반환함
data = sum_and_mul(3, 4)
sum, mul = sum_and_mul(3, 4)
```

함수

- 매개변수 초기화
  - 매개변수에 초기값을 설정해 줌
  - 항상 가장 뒤쪽의 매개변수부터 초기화 시켜야 함

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

say_myself("greenjoa", 23)
say_myself("greenjoa", 23, False)
```

함수

- 매개변수는 지역변수임

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

vartest(a)
print(a)
```

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

a = vartest(a)
print(a)
```

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

vartest()
print(a)
```

배포 패키지 만들기

- 단계
  - 임의의 폴더를 생성하고, 폴더내에 .py 파일과 setup.py 파일을 위치시킴
  - cmd 창에서 해당 폴더로 이동후 아래 명령 수행

```
python setup.py sdist
```

  - 배포 패키지 설치

```
python setup.py install
```

  - 설치된 패키지 확인

```
C:₩Python34₩Lib₩site-packages
```

setup.py

```
from distutils.core import setup

setup(
    name = "PythonTestModule",
    version = "1.1.0",
    py_modules = ["PythonTestModule"],
    author ="greenjoa",
    author_email ="greenjoa@gmail.com",
    url = "http://www.greenjoa.com",
    description="Test Module",
    )
```

배포 패키지 사용하기

- import 하기

```
import PythonTestModule

sum = PythonTestModule.sum_and_mul(2,2)

print(sum)
```

- Namespace 지정하기

```
from PythonTestModule import sum_and_mul

sum = sum_and_mul(2,2)

print(sum)
```

파일 다루기

파일 다루기

- accessmode
  - 파일을 오픈한 후 파일을 가지고 하고자 하는 것을 명시
    - r : Read the file
    - w : Write to the file
    - a : Append to the existing file content
    - b : Open a binary file
    - w+ : Read/Write

```
fileName = "greenjoa.txt"
myFile = open(fileName, "w")
myFile.close()
```

* 기존 파일 없어지고 새로 만들어짐

파일 다루기

- 파일 출력하기

```
fileName = "greenjoa.txt"
WRITE ="w"

myFile = open(fileName, mode = WRITE)

myFile.write("Hi there!₩n")
myFile.write("How are you?₩n")

myFile.close()
```

파일 다루기

- 파일 읽기 : 전체 / 라인 단위

```
fileName = "test.txt"
READ = "r"
myFile = open(fileName, mode = READ)
fileContents = myFile.read()
print(fileContents)
myFile.close()
```

```
with open(fileName, READ) as myFile :
    fileContents = myFile.read()
    print(fileContents)
```

* 자동해제됨

```
fileName = "test.txt"
READ = "r"
myFile = open(fileName, mode = READ)
while True:
    content = myFile.readline()
    if not content : break
    print(content)
myFile.close()
```

파일의 끝일 경우 None을 리턴함

파일 다루기

- 파일 읽어오기
  - 파일의 모든 내용을 리스트에 저장하기

```
fileName = "test.txt"
READ = "r"
myFile = open(fileName, mode = READ)
content = myFile.readlines()
for line in content:
    print(line)
myFile.close()
```

파일 다루기

- csv 파일
  - 필드를 쉼표(,)등으로 구분한 텍스트 파일
    - 쉼표, 탭, 빈칸등로 구분함
  - 엑셀 양식의 데이터를 프로그램에 상관없이 쓰기 위한 데이터 형식

```
import csv
fileName = "test.txt"
READ = "r"
with open(fileName, READ) as myFile :
    dataFromFile = csv.reader(myFile)
    for currentRow in dataFromFile :
        print(currentRow)
```

파일 다루기

- csv 파일 다루기

```
import csv
fileName = "test.txt"
READ = "r"
with open(fileName, READ) as myFile :
    dataFromFile = csv.reader(myFile)
    for currentRow in dataFromFile :
        for currentWord in currentRow :
            print(currentWord)
```

- 목록 연결해주는 함수 : Join

```
with open(filename, READ) as myFile :
    dataFromFile = csv.reader(myFile)
    for currentRow in dataFromFile :
        print(",".join(currentRow))
```