




# 모듈과 나만의 함수

## Python Programming

국민대학교 경영대학원 AI빅데이터전공  
문현실 (hsmoon@kookmin.ac.kr)







## Ch.01 패키지(package)의 사용

# 모듈? 라이브러리?

- 프로그램에서 독립적인 단위
- import문을 이용해 불러와서 사용할 수 있음

```
import random
```

- 모듈 호출시 이름 지정 가능

```
import pandas as pd
```

홀로코스터

4 시간 전

라이브러리하고 프레임워크가 뭐임? 이거 참고해가면서 일하면 얼마나 편함?

0 0 ...

Ausfaller

4 시간 전

@홀로코스터 라이브러리만 봐도 좀 긴 함수를 짜놓고 그걸 설정해놓은 단순한 한줄짜리 명령어로 불러올 수 있는데 그런게 여러개인데다 자주 쓰이면 효율성이 장난 아니지

0 0 ...

톤스

4 시간 전

@홀로코스터 라이브러리 안썼을때 : 좌우 고관절을 33도로 틀면서 종아리 근육을 20퍼센트의 힘으로 수축시키면서 달팽이관의 기울기에 비례하여 상체의 척추와 등근육을 팽창시키고 안구의 간상세포로 기울기를 보정하며 다리사이로 충분한 높이의 사물을 확인한뒤 뒤통치뼈에 체중을 실어 넘어짐을 방지하면서 무릎을 발목보다 수직 선상앞으로 이동시킨후 체중의 70퍼센트가 뒤로 쏠렸을때 대퇴근을 이완시키며 어찌고저찌고

라이브러리 썼을때 : 앓아



# 모듈.함수

- 불러온 모듈의 함수를 모듈.함수 형태로 쓸 수 있음
  - 모듈의 함수 사용은 메서드와 사용법이 비슷

```
random.randint(1,6)
```

# import의 사용법

- random 모듈 전체를 임포트

```
import random
```

- random 모듈에서 randint 함수만 임포트
  - random.randint가 아닌 randint 형태로 호출

```
from random import randint
```

# from ... import \*

- from ... import \* 형태로 모듈 내 모든 함수를 임포트
  - 여러 모듈을 임포트하면 어떤 함수를 임포트했는지 알 수 없는 문제 발생

```
from random import *
```

# 패키지

- 여러 개의 모듈들로 구성된 덩어리
- pip와 conda를 이용해 설치
  - pip : Python 표준 패키지 설치 프로그램
  - conda : 아나콘다 패키지 설치 프로그램
- conda -> pip 순으로 시도
  - 일반적으로 conda의 버전이 낮은 편



# TurtleGraphics

- 컴퓨터 프로그래밍의 원리를 가르치기 위한 교육 방법
- 거북이를 움직여서 다양한 그림을 그려보자
  - 패키지 로딩 : `import turtle`
  - 색 변경 : `color('색이름')`
  - 앞으로 이동 : `forward(이동거리)`
  - 왼쪽으로 회전 : `left(각도)`
  - 그리기 종료 : `done()`
  - 거북이 모양으로 바꾸기 : `shape('turtle')`

# Wrap-up Exercise

- 다음의 도형을 그려보자
  - 정사각형
  - 직사각형
  - 정오각형
  - 별



# Wrap-up Exercise

- 리스트를 활용해서 오륜기 그리기

## Ch.02 사용자 함수의 작성

# 여러 명이 프로그램을 함께 개발한다면?

- 다같이 모여 토론하며 한 줄 한 줄 작성하기
- 가장 잘하는 사람이 혼자 작성하기
- 필요한 부분을 나누어 작성한 후 합치기
  - 함수, 객체, 모듈

# 함수의 개념과 장점

- 함수(Function)
  - 어떤 일을 수행하는 코드의 덩어리, 또는 코드의 묶음
- 장점
  - 필요할 때마다 호출 가능
    - 반복적으로 수행해야 하는 업무를 단 한번만 작성하여 필요할 때마다 호출하여 사용
  - 논리적 단위로 분할 가능
  - 코드의 캡슐화
    - 인터페이스가 잘 정의된다면 다른 사람이 쉽게 가져가다가 사용할 수 있는 특징

# 함수의 선언

```
def 함수 이름 (인자1, 인자2, ...):  
    수행문 1  
    수행문 2  
    ...  
    return <반환값>
```

- 인자(argument) ≡ 매개변수(parameter)

# 함수의 형태

매개변수 유무 반환값 유무	매개변수 없음	매개변수 있음
반환값 없음	함수 안 수행문만 수행	매개변수를 사용하여 수행문만 수행
반환값 있음	매개변수 없이 수행문을 수행한 후, 결과값 반환	매개변수를 사용하여 수행문을 수행한 후, 결과값 반환

```
def a_rectangle_area():  
    print(5*7)  
def b_rectangle_area(x, y):  
    print(x*y)  
def c_rectangle_area():  
    return(5*7)  
def d_rectangle_area(x,y ):  
    return(x*y)
```





# Wrap-up Exercise

- 사각형 그리기 함수

# 변수의 사용범위

- 지역 변수(Local variable) : 함수 안에서만 사용
- 전역변수(Global variable) : 프로그램 전체에서 사용
  - 함수 내에서 전역 변수를 사용하기 위해서는 global 사용

```
def test(t):  
    print(x)  
    t = 20  
    print("In function:", t)  
  
x = 10  
test(x)  
print("In Main:", x)  
print("In Main:", t)
```

```
def f():  
    s = "I love London!"  
    print(s)  
  
s = "I love Paris!"  
f()  
print(s)
```

```
def f():  
    global s  
    s = "I love London!"  
    print(s)  
  
s = "I love Paris!"  
f()  
print(s)
```

# 함수 개발 가이드라인

- 함수 이름
  - 함수는 가능하면 짧게 작성할 것 : 외부에 공개하는 함수일 경우, 줄임말을 사용하지 않고 짧고 명료한 이름 사용
  - 함수 이름에 함수의 역할과 의도를 명확히 드러낼 것
  - 소문자로 입력 권장
  - 띄어쓰기를 할 경우에는 \_기호를 사용 ex) save\_model
  - 행위를 기록하므로 동사와 명사를 함께 사용 ex) find\_number
- 함수의 역할
  - 하나의 함수에는 유사한 역할을 하는 코드만 포함
  - 한 가지 역할을 명확히 하도록 작성
- 함수를 만드는 경우
  - 공통으로 사용되는 코드를 함수로 변환
  - 복잡한 로직이 사용되었을 때, 식별 가능한 이름의 함수로 변환

# Wrap-up Exercise

- FizzBuzz
  - 시작값과 끝값을 인수로 하여 해당 수열의 값이 3의 배수면 Fizz, 5의 배수면 Buzz라고 출력하고 15의 배수면 FizzBuzz라고 출력하는 함수를 작성
  - 예를 들어 1과 100을 인자로 받으면 실행결과는 다음과 같아야 한다.
    - 1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz

# Wrap-up Exercise

- BMI 계산기
  - BMI=몸무게/(키\*키)

구분	BMI(이상~미만)
저체중	<18.5
정상	18.5~23
과체중	23~25
경도비만	25~30
고도비만	30이상



# **Q&A**

Thank you for your listening

