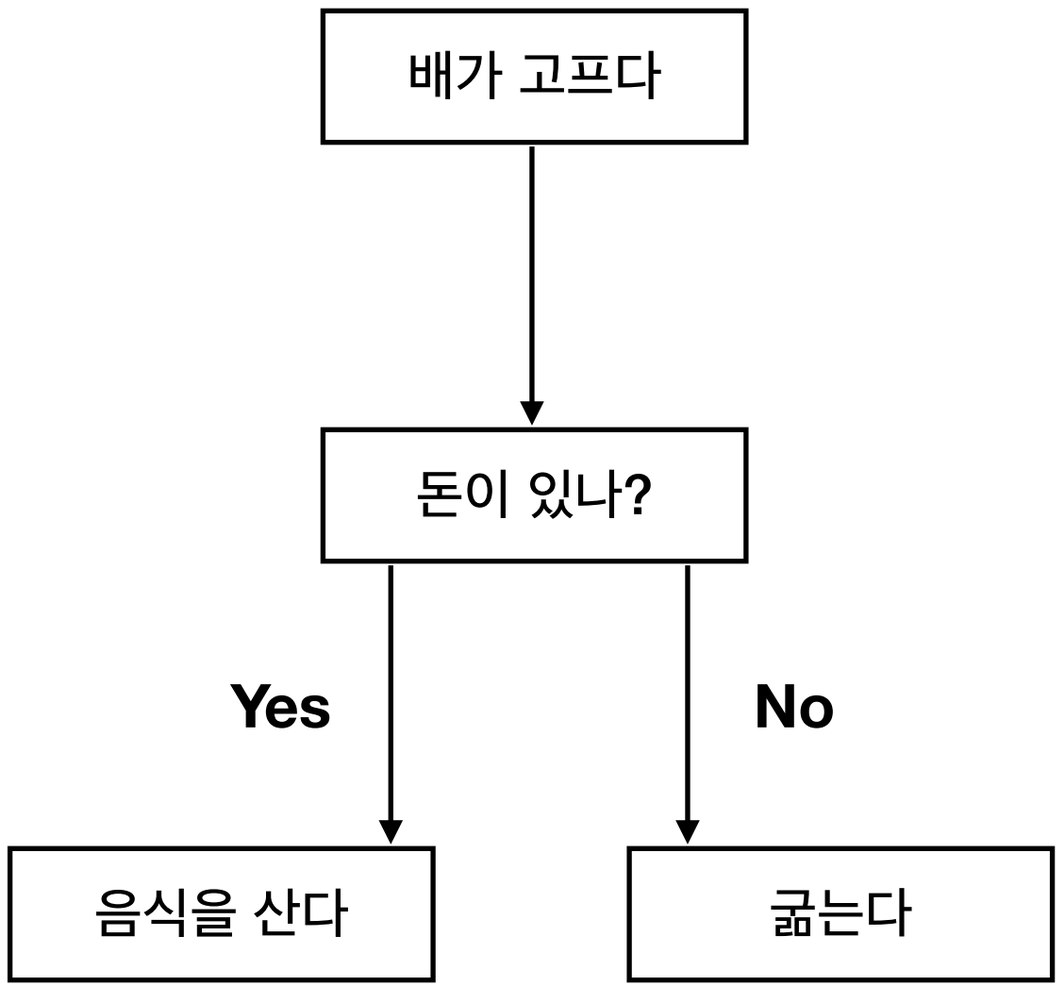
BIG DATA ANALYTICS

WEEK-02 | Python Basic II

Yonsei University
Jungwon Seo

Python Flow Control



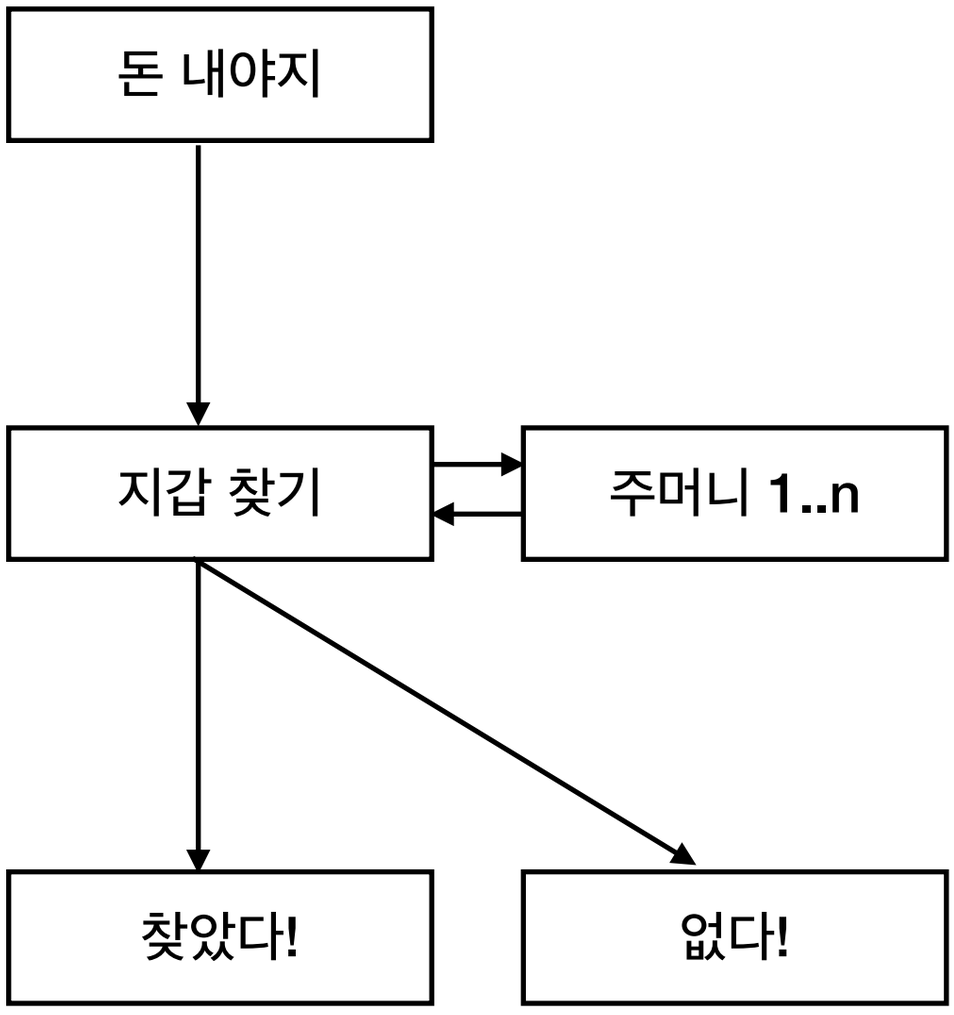

```
def hungry():
    if isMoneyEnough:
        getFood()
    else:
        starve()
```

Python IF-statement

- 조건문
- **if A**: 만일 A 하다면
- **elif** B: A하지 않고 B 하다면
- **else** : 그 외 라면

```
A = “호랑이"
if A in animals:
    print("동물!")
elif A in plants:
    print("식물!")
elif A in earth:
    print("지구 안")
else:
    print("지구 밖!")
```

Python Flow Control




```
def pay(cost = 10000):
    for pocket in pockets:
        if wallet in pocket
            return cost
    return "헐"
```

Python loop-statement: for

- for문
- for a in A : A안에 있는 값들을 다 돈다
- A는 반드시 반복 가능(iterable) 해야한다.
  - 보통 list, dict, set, tuple과 같은 그룹형 데이터

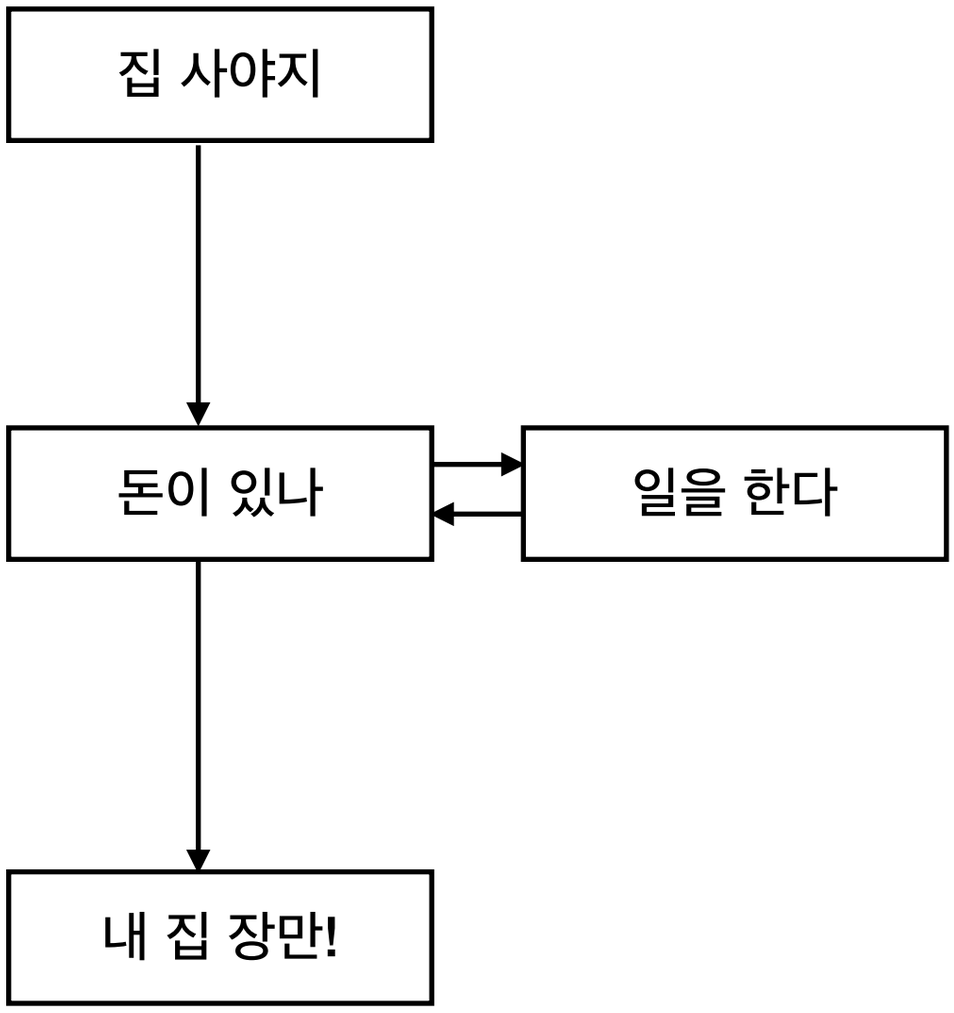
```
data = ["사과", "바나나","수박"]
for x in data:
    print(x)
```

output: 사과 바나나 수박

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

output: 0 1 2 3 4 5 6 7 8 9

Python Flow Control




```
def buy_house():
    while money<10억:
        work_hard()
    return house
```

Python loop-statement: while

- 반복문
- while A : A라는 조건이 만족하는 동안
- 주의사항
- 만약 A라는 조건이 만족되지 않을 수 있는 경우 무한루프에 빠짐

```
cnt = 0
while cnt < 10:
    print(cnt)
    cnt+=1
```

output: 0 1 2 3 4 5 6 7 8 9

~~
```
cnt = 0
while cnt < 10:
    print(cnt)
```
~~

Python 반복문 제어

- break: 반복문을 빠져나온
- continue: 현재 순서는 넘어가고 다음 순서를 시작한다
- pass: 통과 (placeholder)

```
for i in range(10):
    if i == 1:
        pass
    elif i ==2:
        continue
    elif i == 5:
        break
    print(i)
```

Python Function (함수)

- 변수는(variable) 상태를 정의하기 위한 역할
- 함수는(function) 동작을 정의하기 위한 역할
- 코드의 재사용성/가독성을 높임

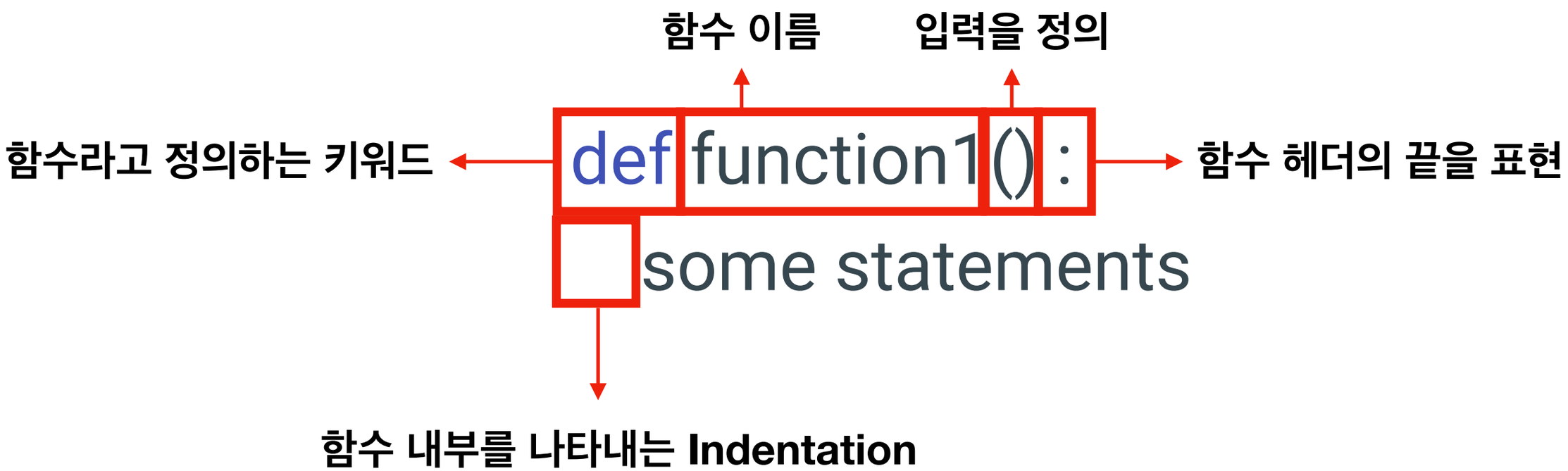
Python Function (함수)

- 코드 상에서 함수 작성법

```
def function1() :
    some statements
```

Python Function (함수)

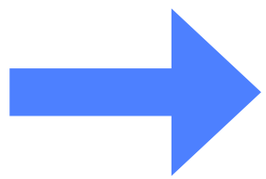
- 코드 상에서 함수 작성법



Python Function (함수)

- 함수에 입력값을 넣어 주어야 하는경우

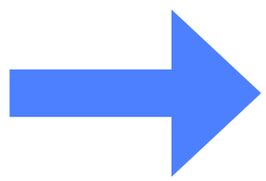
```
def fun1(x):
    print(x)
```



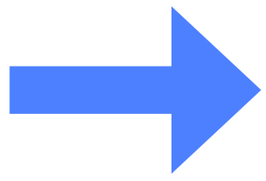
```
fun1(2)
결과: 2
```

- 입력 값의 기본값 설정시, 입력값 생략 후 호출 시 기본 값이 반영

```
def fun1(x=10):
    print(x)
```



```
fun1(2)
결과: 2
```

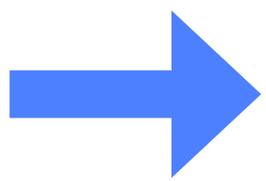


```
fun1()
결과: 10
```

Python Function (함수)

- 여러개의 입력값을 정의할 경우

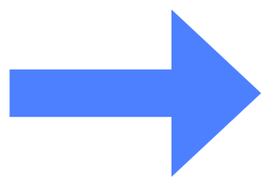
```
def fun1(x,y,z):
    print(x,y,z)
```



```
fun1(2,3,4)
결과: 2 3 4
```

- 입력값이 많아지면, 순서대신 key-value형태로 입력

```
def fun1(x,y,z):
    print(x)
```

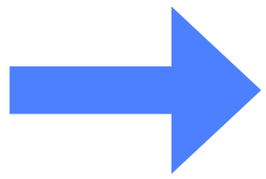


```
fun1(x=2,y=3,z=4)
결과: 2 3 4
```

Python Function (함수)

- 함수에서 값을 반환 해야 하는 경우

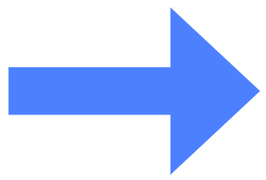
```
def fun1(x):
    return x*2
```



```
x1 = fun1(2)
```

- 함수에서 여러 값을 반환 해야 하는 경우

```
def fun1(x):
    return x*2, x+2
```



```
x1, x2 = fun1(2)
```

Python Function 작성 팁

- 필수는 아니지만, 동사형태로 함수명을 정하는 것이 권장됨
  - get(), set(), update(), remove(), parse(), etc

- 함수의 이름이 길 경우 다음과 같은 방식으로 작성

- Camel Case
  - getData()
  - getUserData()

- Snake Case
  - get_data()
  - get_user_data()

E.O.D