BIG DATA ANALYSIS PROGRAMMING

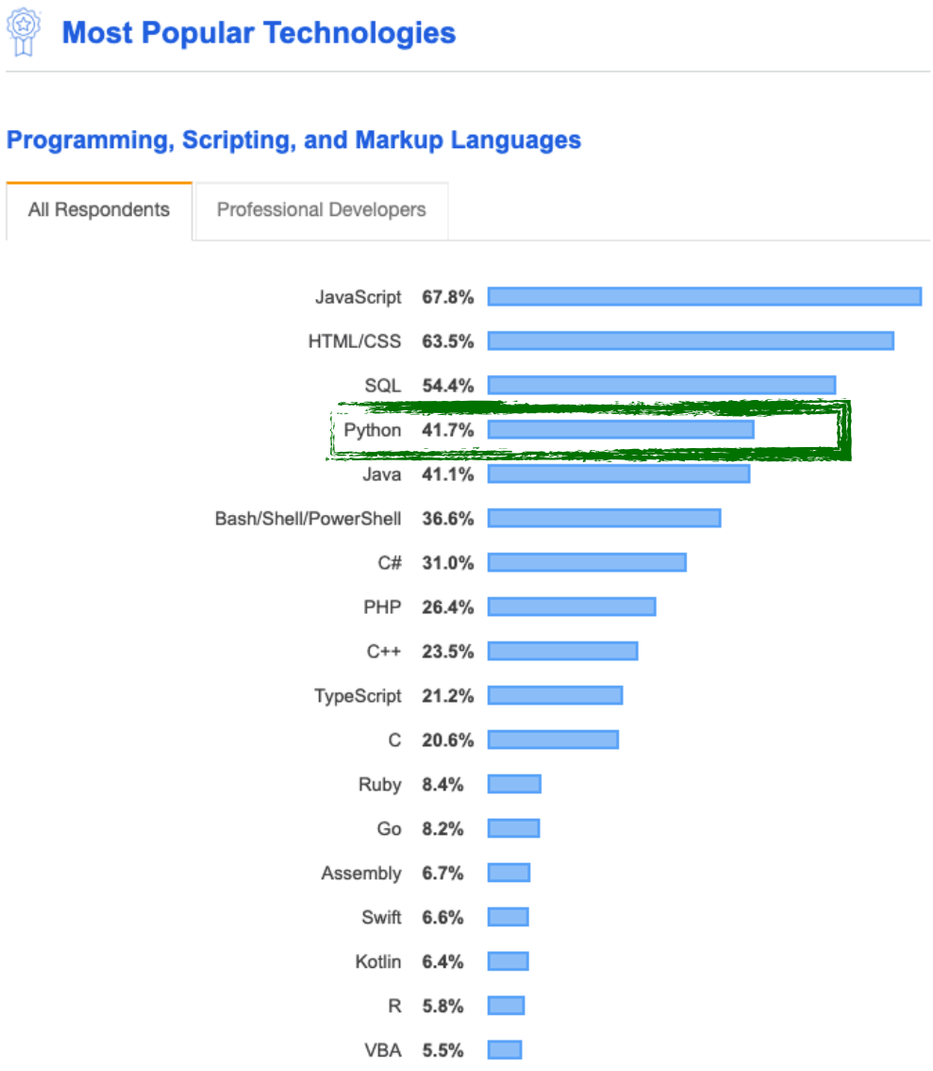
WEEK-01 | Python Basic I

Yonsei University
Jungwon Seo

Python

- 파이썬
  - Guido van Rossum가 1991년에 최초 배포
  - 코드의 가독성을 높이는 것이 파이썬의 철학

- 언어 특징
  - Interpreted
  - High Level
  - General Purpose



https://insights.stackoverflow.com/survey/2019#most-popular-technologies

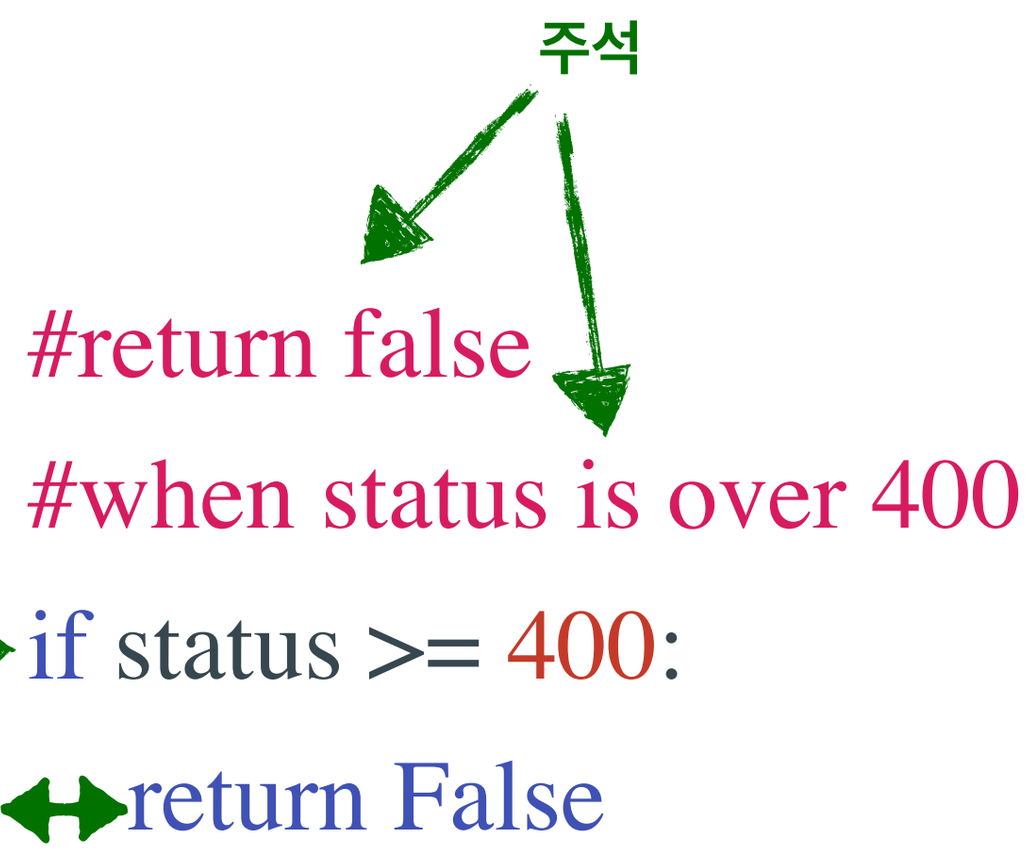
Python Keywords

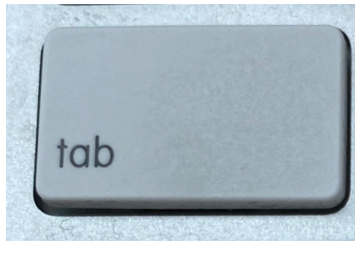
- 키워드
  - 파이썬 언어에서 예약된 (reserved) 단어들
  - 각각의 키워드들은 특정한 역할을 함
  - 변수(Variable) 명으로 키워드들을 사용할 수 없음

FALSE	class	finally	is
None	continue	for	lambda
TRUE	def	from	nonlocal
and	del	global	not
as	elif	if	or
assert	else	import	pass
break	except	in	raise
return	try	while	with
yield			

Python Terms

- Statement : 문(文)
- Comment : 주석
- Indentation : 들여쓰기

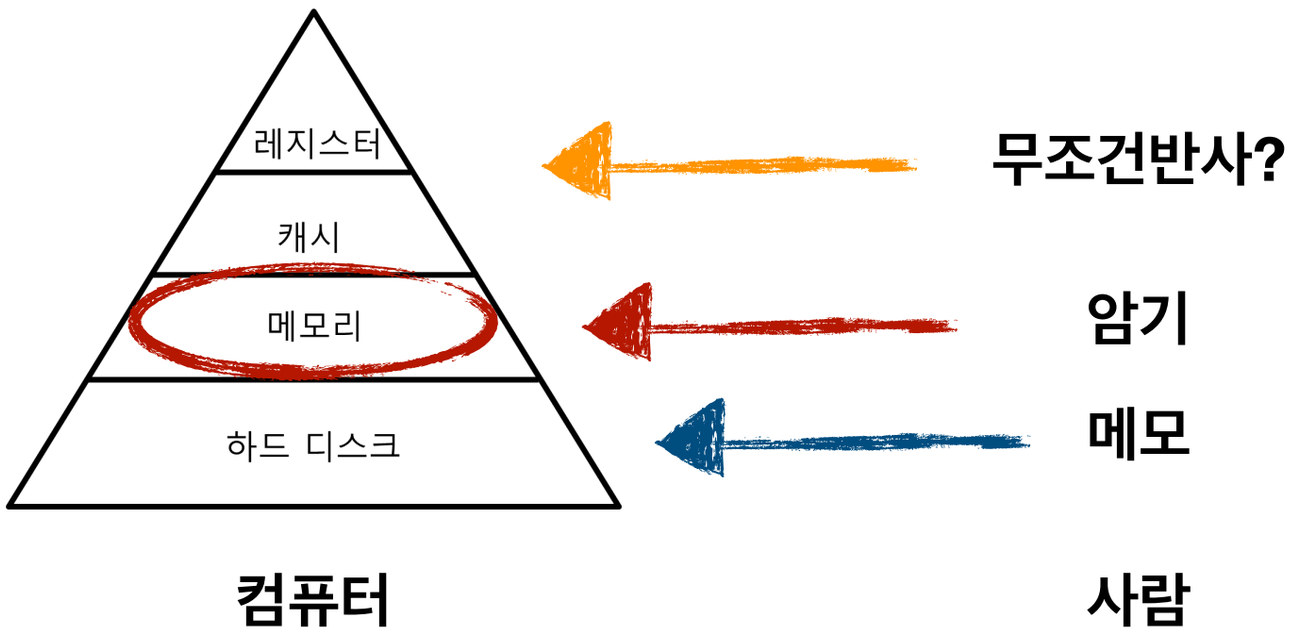




Python Variable I

- Python 변수
- Numeric: 숫자
- Boolean: 참/거짓
- String: 문자열

```
var1 = 100
var2 = True
var3 = "Hello"
var4 = var3
```



Python Variable II

- List: 배열
  - Ordered : 순서가 보장됨
  - Position based access
  - list of elements
  - String도 List의 부분집합

```
mylist = list()

mylist = [1,2,3,4]

mylist[0] = 10

print(mylist[2])
```

- dictionary: 사전
  - Unordered: 순서가 보장되지 않음
  - Key based access
  - set of key-value pairs

```
mydict = dict()

mydict = {'a':1, 'b':2}

mydict['a'] = 3

print(mydict['b'])
```

Python Variable II

- Set: 집합
  - 수학의 집합과 유사
  - 합집합, 교집합, 차집합 등의 연산을 지원
  - 모든 값은 unique하게 저장됨

```
my_set_1 = set([3,4])

my_set_2 = {1,2,3}

my_set_3 = my_set_1.intersection(my_set_2)

print(my_set_3)
```

- Tuple: 리스트와 비슷
  - List와 비슷하지만, 수정 불가(Immutable)
  - 임시로 값들을 grouping 할 때 자주 쓰임
  - 여러 변수를 하나의 변수로

```
my_tuple=tuple()

my_tuple = (1, 2, 3)

print(my_tuple[0])
```

Python Type Casting

- Casting 이란?
  - 한 타입에서 다른 타입으로 변경하는 행동
    - int를 float으로
    - int를 string으로
    - string을 list로
    - dictionary를 list로

- 숫자간 형 변환
  - 9.0 =〉 9
  - 9.1 =〉 ?

- 연산자(Operator) 의 동작 방식
  - 각각의 형(type)에 따라 연산자(+,-,*,/)가 동작하는 방식이 달라짐

Python Input/Output

- Python에서 입력과 출력

- 입력
  - 장치로 부터 (키보드, 마우스, 마이크)
  - 파일
  - DB

- 출력
  - 표준출력(stdout) : 화면에 띄어주는 방식
  - 파일
  - DB

Python Operator I

- 연산자란?
  - 산술 또는 논리 연산을 할 수 있는 특수 심벌

- Arithmetic operators : 산술 연산자
  - +, −, *, /, **, //, %

- Comparison operators : 비교 연산자
  - 〉, 〈, 〈=, 〉=, ==, !=

- Logical operators: 논리 연산자
  - and, or, not

Python Operator II

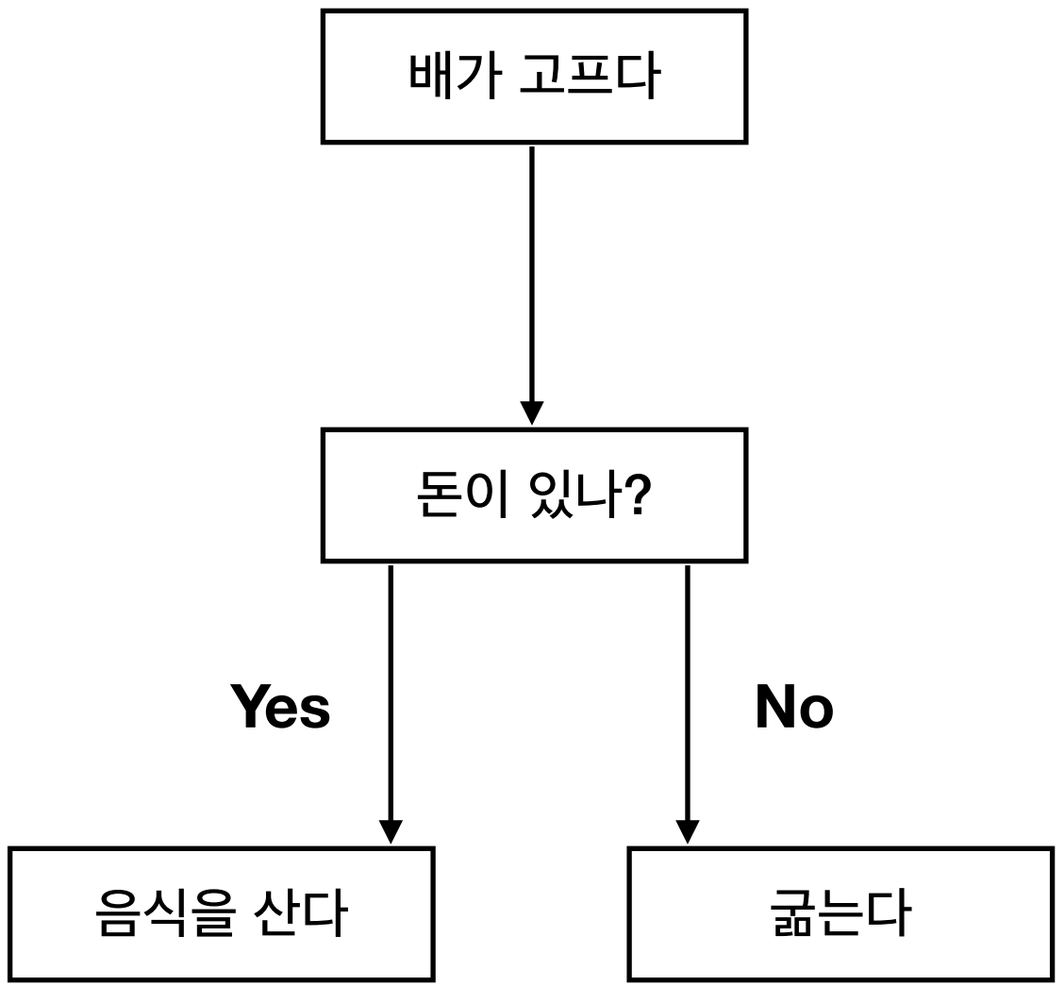
- Bitwise operators: 비트 연산자
  - &, |, ~, ^, 〉〉, 〈〈

- Assignment operators: 할당 연산자
  - =, +=, -=, *=, /=, %=, //=, **=, &=, |=, ^=, 〉〉=, 〈〈=

- Identity operators: 나는 누구?
  - is, is not

- Membership operators: 나는 어디에?
  - in, in not

Python Flow Control



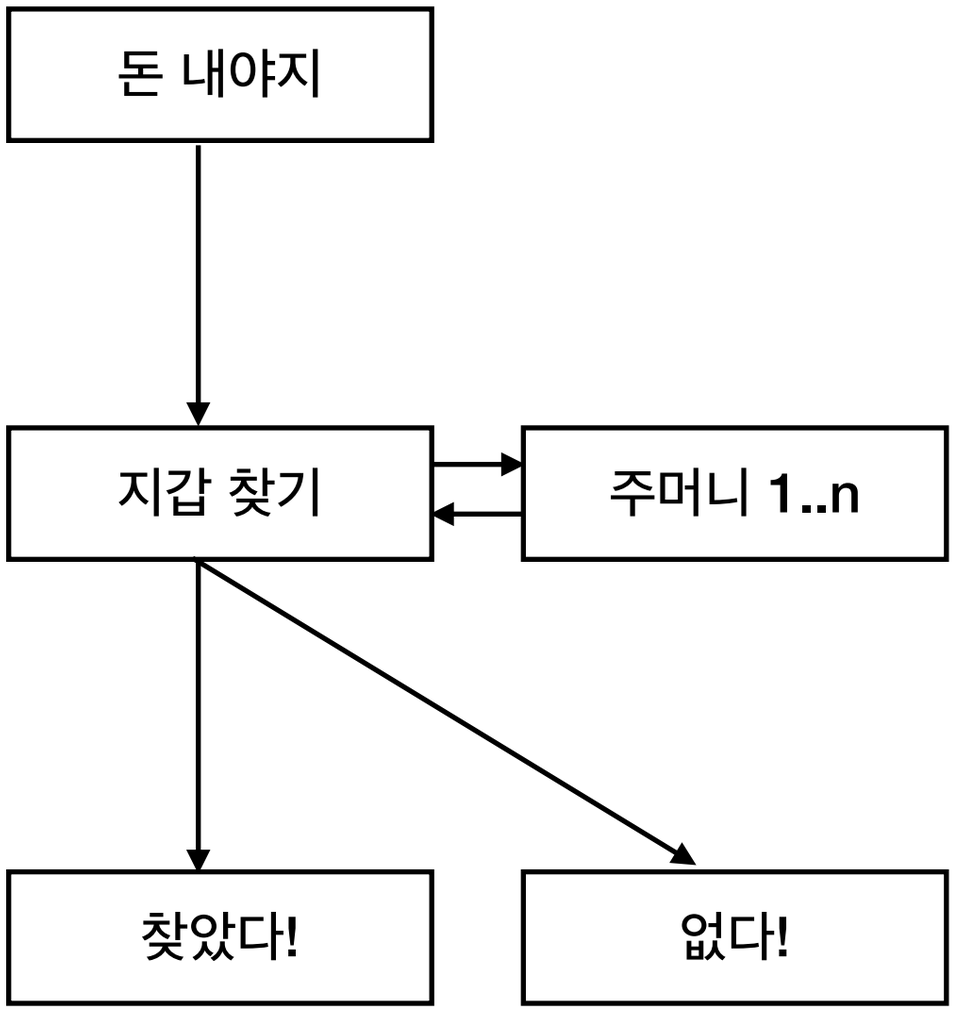

```
def hungry():
    if isMoneyEnough:
        getFood()
    else:
        starve()
```

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

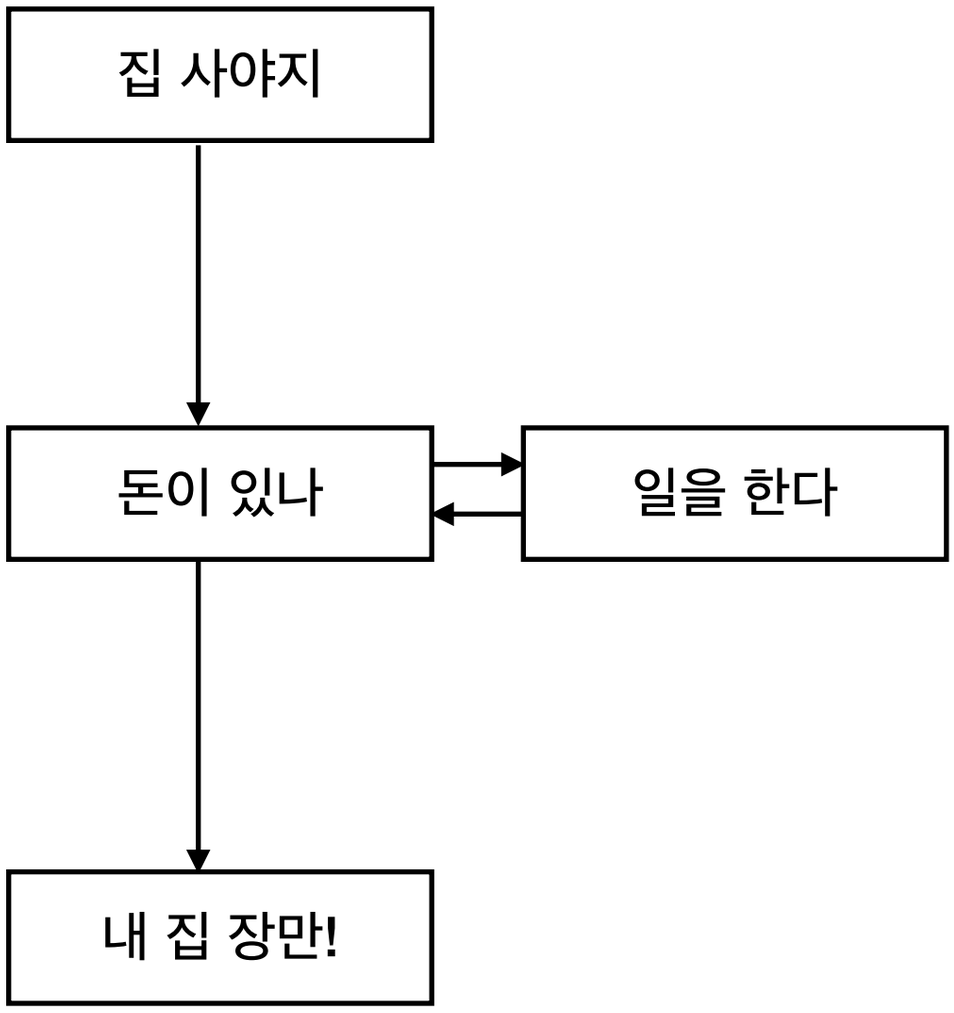
```
data = ["사과", "바나나","수박"]
for x in data:
    print(x)
```

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)
```

E.O.D