Big Data Analytics Programming

Week-04. Conditional / Loop statement

Jungwon Seo, 2022-Summer

배울 내용

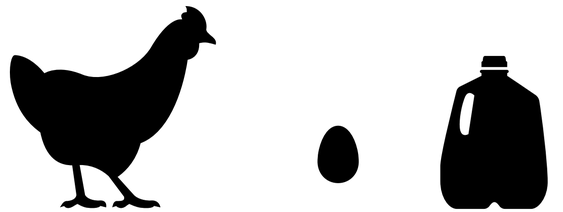
Week-04. Python Basic

- 조건문: if-statement
- 반복문: loop-statement

논리적으로 생각한다는 것은?

부모님께서 프로그래머인 나에게 심부름을 시키셨다.
"슈퍼가서 우유 하나 사와. 아, 계란 있으면 6개사와"
그래서 우유를 6개 사갔다.
"왜 우유를 6개나 샀어?"
"계란이 있길래.."

- From 인터넷 어딘가..



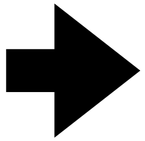
조건문(conditional statement)

What **if…** anything **else?**

- 사람은 항상 특정한 조건/상황에 기반해서 판단
- 세상에서 일어나는 모든 상황을 Logically 코드로 표현 가능

If I were a boy
Even just for a day
I'd roll outta bed in the morning
and throw on what I wanted and go

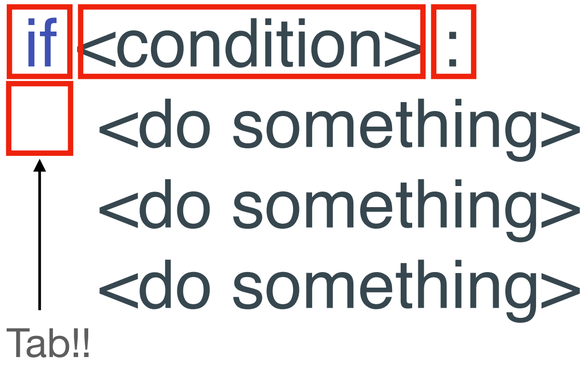
- *Beyonce, If I Were A Boy* 中



```
if user.gender == "boy" and user.duration >= "1":
    user.wake_up(from="bed", when="morning")
    user.wear(style="whatever")
    user.go_out()
```

조건문(conditional statement)
If문 사용법

- 조건문 작성법
  - if 라는 예약어로 시작
  - If 뒤에는 True(1) or False(0)로 나올 수 있는 조건
  - 해당 하는 조건의 ending 은 콜론 (:)으로 마무리
  - 그 조건문이 "만족" 됐을 때 동작할 코드는 들여쓰기(indentation) 이후 작성
  - If 안에서만 동작해야할 코드는 들여쓰기를 유지



조건문(conditional statement)

True, False?

- 기본적인 Boolean 타입은 그대로 유지 : True is True, False is False
- Boolean 타입 외에도 조건으로 사용가능

Team True!! 👻

True

All numbers except 0

All collections except
collection **with no element**

~~All functions~~

Team False!! 👽

False 0 0.0

None [ ]

( )

" " { }

조건문(conditional statement)

비교에 의한 조건

- 값들간의 비교를 통해서 조건을 만들 수 있다.
  - A가 B보다 크다 : **A > B**
    - A=5, B=3 일때의 결과는? => True
    - A=3, B=5 일때의 결과는? => False
  - A가 B보다 작다 : **A < B**
  - A가 B보다 크거나 같다: **A >=B**
  - A가 B보다 작거나 같다: **A <=B**
  - A와 B가 같다: **A == B, A is B**
  - A와 B가 다르다: **A != B, A is not B**
- 결과는 결국 True or False로 반환

```
A=5
B=3
print(A>B)
print(A<B)
print(A>=B)
print(A<=B)
print(A==B)
print(A is B)
print(A is not B)
```

조건문(conditional statement)
조건들의 조합

- 단순 1차원적인 조건이 아닌, 여러 조건을 연결 가능
- A는 B보다 크면서 0이 아니다
  - A>B and A!=0
  - 두개다 참이여야 참입니다! True and True => True
- A는 B보다 크거나 100이다
  - A>B or A==100
  - 둘중에 하나라도 참이면 참! False or False => False
- 다른 언어에서는 &&, 난 ||로 and, or 가 표현됨
  - &와 |는 bit연산으로 사용됨 1과 0에 대해서만
  - 결과적으로 각각의 조건이 True or False로 나오는 상황에서는 and, or와 동일하게 동작

```
A=5
B=3
print(A>B and A==0)
print(A>B and A!=0)
print(A>B or A==0)
print(A>B or A!=0)
print(A>B & A==0)
print(A>B | A==0)
```

조건문(conditional statement)

조건은 Optional 일 수 있고, Alternative 있다

- 단순히 특정 케이스에 대해서만 반응을 하기 위해서는
  - If와 elif (else if)로 조건문을 사용 할 수 있다.
  - 예) 만약 점수가 100점 이상이면 100점이라고 친다
  - 예) 만약 점수가 100점 이상이면 100점이라고 치되,
    0점 이하면 0점으로 친다.

```
score = 98
if score >= 100:
    score = 100
elif score <= 0 :
    score = 0
```

- 특정 케이스가 아닌 케이스에 대해서도 반응을 하기 위해서는
  - else로 전체 조건문을 마무리 할 수 있다.
  - 예) 만약 점수가 50점 이상이면 Pass라고 한다, **그게 아니라면** Fail이다.

```
score = -20
grade = None
if score >= 50:
    grade = "Pass"
else:
    grade = "Fail"
```

반복문 (iterative/loop statement)

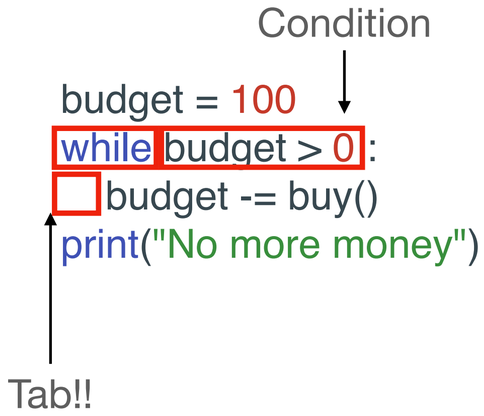
While I am… **For** how long?

- 반복문 사용의 목적
  - 어떠한 연산을 특정 횟수만큼 반복 하려는 경우
- 반복문을 사용하는 사례
  - 1,000,000개의 고객 명단이 있는데, 전화번호에 '-' 나 띄어쓰기가 있으면 없애야 하는 경우
  - 텍스트 파일을 줄 단위로 읽는데, 텍스트가 더이상 없는 경우에 멈춰야 하는 경우

반복문 (iterative statement)

While 문

- 특정 조건을 만족하는 한, 그 구간을 반복한다.
  - **While** 문을 활용
  - 예1) budget이 0이상인 동안
  - 예2) 에러가 3번 미만으로 나는 동안




```
num_of_errors = 0
my_file = None
while num_of_errors < 3:
    try:
        my_file = download_file()
        break
    except:
        num_of_errors+=1
```

반복문 (iterative statement)

For 문

- Collection 안에 있는 원소들을 각각 접근한다.
  - **For** 문을 활용
  - range를 활용한 0 부터 n까지 접근
  - List와 같은 collection 데이터의 각각의 원소에 접근


```
my_list = ['a','b','c']
for ele in my_list:
    print(ele)
```

* Tab == 4 x Space와 같지만 여기선 그냥 tab으로 통일 하겠습니다!

반복문 (iterative statement)

continue, break

- 반복문 내에서 진행 여부를 제어하는 방법
  - continue: continue 구문을 만난 시점까지만 진행하고 다음 턴으로 넘김
  - break: 반복문 종료

```
for i in range(5):
    if i == 2:
        continue
    print(i)
```

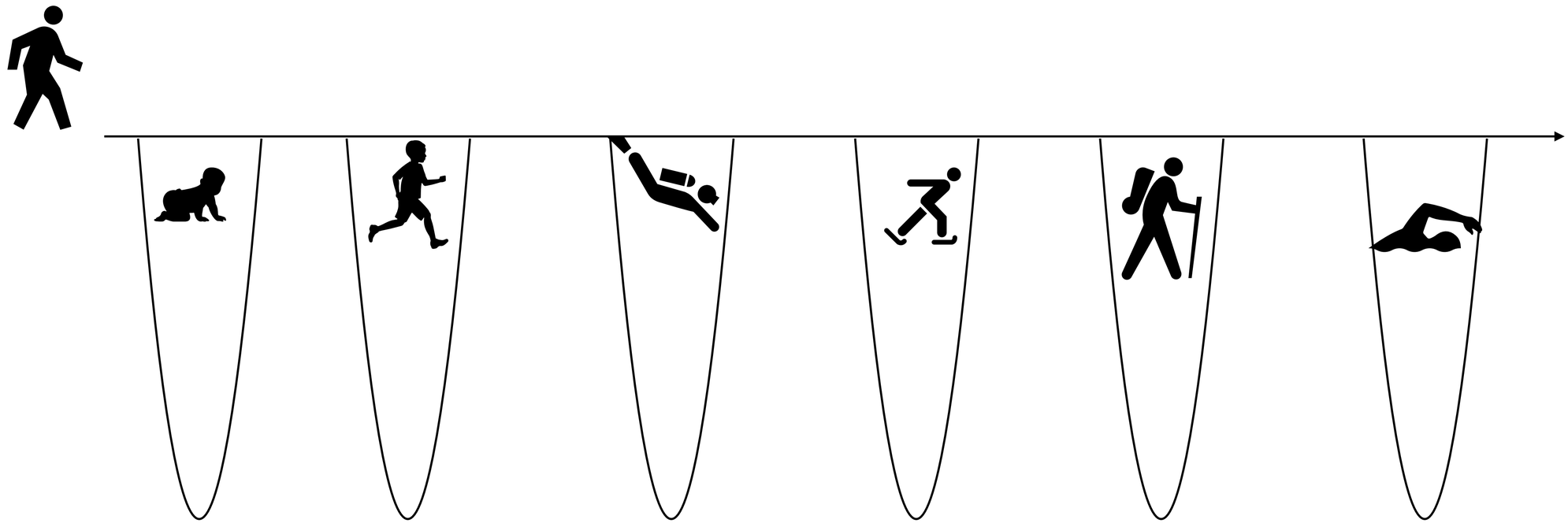
```
0
1
3
4
```

```
for i in range(5):
    if i == 2:
        break
    print(i)
```

```
0
1
```

프로그래밍의 여정

골짜기(Valley)에 매번 빠질 것입니다...



E.O.D