Bbong 뽑는 Python

2강 조건, 반복, 함수

권태형@TaeBbong
github.com/TaeBbong

조건문

Condition Statement

조건문 Condition Statement

조건문

조건문은 if 이다.
if의 뜻은? => 만약 .. 라면

파이썬에서의 조건문(if문)은 만약 A조건이 성립한다면 a를 수행해 라는 구조를 갖는다.

어렵지 않지만, 조건문을 깔끔하고 완벽하게 처리하는게 중요하다!

조건문 Condition Statement

조건문의 구조 및 문법

```
if 조건 A:
    행동 a
if 조건 B:
    행동 b
```

조건문 Condition Statement

else

만약 A라면 a를 해라

A가 아니라면?? 일일이 조건을 써주어야 하는가? No!

else는 바로 앞의 if문의 부정, 즉 이를 문장으로 서술하면 다음과 같다.

만약 A라면 a를 해라, 그게 아니라면 b를 해라!

조건문 Condition Statement

if - else 구문

```
if 조건 A: # A라면,
    행동 a # a를 해라
else: # 그게 아니면
    행동 b # b를 해라
```

조건문 Condition Statement

elif(else if)

만약 A라면 a를 해라, 아니라면 b를 해라

세상을 두 개의 조건으로 다 나눌 수 있는가? 그렇지는 않을거다:)

elif는 if와 else의 사이에 존재하는 중간 조건으로, A가 아니라 B라면, 그게 아니라 C라면 .. 등을 담당해주는 조건문 키워드이다.

조건문 Condition Statement

if - elif - else 구문

```
if 조건 A: # A라면,
    행동 a # a를 해라
elif 조건 B: # A가 아니라 B라면
    행동 b # b를 해라
elif 조건 C: # A도 B도 아니고 C라면
    행동 C # c를 해라
else: # 그게 아니면
    행동 d # d를 해라
```

조건문 Condition Statement

if - elif - else 구문 - example

```
money = 0
card = 1

if money > 3000:
    print(“Call Taxi!”)
elif card == 1:
    print(“Call Taxi! with card”)
else:
    print(“Walk..”)
```

조건문 Condition Statement

조건문에서 유의할 점

* if - elif - else 순서가 존재하는데, 이 구조는 절대 뒤집힐 수 없다!
(하나의 if에 대해서만 다수의 elif와 하나의 else가 할당된다)

* if(만약..라면) elif(그게 아니라 .. 라면) else(그 모든게 아니라 .. 라면) 이 구조에서 elif와 else는 바로 위의 조건을 부정하는 것이 기본 정의이다.

따라서 elif 의 '그게 아니라 ..라면' 에서 '.. 라면' 에는 if의 조건이 절대 포함될 수 없다!

조건문 Condition Statement

조건문의 응용 - and or not

and: 그리고, 두 조건을 동시에 만족시켜야 할 때 사용
or: 또는, 두 조건 중 하나만 만족시켜도 괜찮을 때 사용
not: 부정, 해당 조건이 아닐 때 사용

조건문 Condition Statement

조건문의 응용 - and or not

```
if name == “TaeHyung” and age == 21:
    print(“He is TaeHyung!”)

if money < 10000 or time > 30:
    print(“그냥 걸어가자!”)

if n is not None:
    print(“n exists!”)
```

조건문 Condition Statement

조건문의 응용 - in

in 은 리스트, 튜플, 딕셔너리 등에 내가 원하는 값이 있는지 없는지에 대한 조건 처리

```
student = [‘Amy’, ‘Bob’, ‘Carl’, ‘David’]

if ‘Emily’ in student:
    print(“Hi Emily!”)
else:
    print(“Where is Emily??”)
```

조건문 Condition Statement

HW 4

Prob143 ~ Prob 156

반복문

Loop Statement

반복문 Loop Statement

반복문

컴퓨터의 가장 큰 장점은
‘엄청난 양의 연산’을 ‘엄청나게 빠른 속도로 처리할 수 있다’ 는 것이다!

이러한 장점에 가장 맞는 내용이 반복문일 것이다.
비슷한 일을 반복해서 처리할 때 사용하는 반복문에 대해 알아보자:)

- while
- for

반복문 Loop Statement

반복문 - while

while
한국어 뜻: ~하는 동안

‘특정 조건 동안 어떤 일을 반복하자’ 는 컨셉이다:)

바로 코드로 넘어가보자

반복문 Loop Statement

반복문 - while

```
age = 1

while age < 20:
        print("미성년자는 술을 구매할 수 없습니다!")
        age += 1 # 1강에서 배웠던 대입 연산자

print("성인이 되셨군요!!")
```

반복문 Loop Statement

반복문 - while 무한 Loop

언제까지 반복해야할지 모르겠어!
일단 계속 돌려놓고 내가 원할 때 멈추는게 마음이 편할거 같아!

그렇다면 while의 조건을 항상 참이 되도록 작성하자.

반복문 Loop Statement

반복문 - while 무한 Loop

```
while 1 == 1:
        print(“Infinite”)
```

반복문 Loop Statement

반복문 - while 무한 Loop

```
while 1 == 1:
        print(“Infinite”)

while True:
        print(“Infinite”)
```

반복문 Loop Statement

반복문 - while 무한 Loop

```
while 1 == 1:
        print(“Infinite”)

while True:
        print(“Infinite”)

while 1: # 0(False)가 아닌 모든 값은 True의 값을 갖게 된다.
        print(“Infinite”)
```

반복문 Loop Statement

반복문 - while 무한 Loop

대신 이렇게 반복하면 프로그램은 절대 멈추지 않는다.
(강제종료하기 전까진)

그렇다면 어떻게 종료 시켜야 하지??

break!

반복문 Loop Statement

반복문 - while 무한 Loop

break
반복문을 돌리는 중, 멈춰야 할 순간에 바로 종료시킬 수 있도록 하는 구문

```
count = 0
while True:
        print("I love you")
        count += 1
        if count > 5:
                break
```

반복문 Loop Statement

반복문 - for

for
또다른 종류의 반복문!
왜 같은 기능을 하는 키워드가 두개나 있는지, 그 차이점을 알아보자.

반복문 Loop Statement

반복문 - for

for
정해진 범위를 반복하기 위해, 혹은 정해진 자료구조 안을 전부 보기 위해 사용하는 것

어떻게 말해도 추상적이니 코드를 봐보자:)

반복문 Loop Statement

반복문 - for

```
days = [“mon”, “tue”, “wed”, “thr”, “fri”, “sat”, “sun”]
for date in days:
        print(“today is ”, date)
```

반복문 Loop Statement

반복문 - for

```
days = [“mon”, “tue”, “wed”, “thr”, “fri”, “sat”, “sun”]
for date in days: # days라는 리스트에 있는 값을 하나씩 date에 대입한다
        print(“today is ”, date) # days라는 리스트를 전부 보면 종료된다
```

for a in A: A라는 리스트(혹은 딕셔너리, 튜플, ..) 에 있는 값을 하나씩 가져와서 a에 대입한다.

반복문 Loop Statement

반복문 - for

for문의 또다른 사용법

```
for i in range(0, 10):
        print(i*i)
```

반복문 Loop Statement

반복문 - for

for문의 또다른 사용법

```
for i in range(0, 10): # range()로 범위를 만들어서 범위내의 값을 하나씩 i에 대입
        print(i*i)
```

range(a, b): a이상 b미만의 정수를 리스트로 만들어주는 함수
range(3, 6) # [3, 4, 5]

반복문 Loop Statement

반복문 - for

for문의 또다른 사용법

```
for i in range(0, 10): # range()로 범위를 만들어서 범위내의 값을 하나씩 i에 대입
        print(i*i)
```

즉 위 코드는

```
for i in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:
        print(i*i)
```

와 똑같다!!

반복문 Loop Statement

반복문 - 연습문제

여기까지 반복문을 배워봤는데, 앞서 배운 내용들과 달리 반복문은 설명 듣는 것만으로는 절대절대 부족하다.

직접 예제를 풀어보고, 실제 코드를 보면서 작동 원리를 익혀보자.

연습문제 1: 별찍기
연습문제 2: 구구단 출력하기
연습문제 3: 사칙연산 계산기 만들기

반복문 Loop Statement

반복문 - 연습문제 1: 별찍기

문제: n을 입력받아, 다음 규칙에 맞게 별을 출력하는 프로그램을 작성하라

n = 1
```
*
```

n = 2
```
*
**
```

n = 3
```
*
**
***
```

반복문 Loop Statement

반복문 - 연습문제 1: 별찍기 - 정답 예제

```python
n = int(input("n을 입력하세요: "))

for i in range(0, n):
    print("*" * i)
```

반복문 Loop Statement

반복문 - 연습문제 2: 구구단 출력하기

문제: n을 입력받아(2~9), 구구단의 n단을 출력하는 프로그램을 작성하라
n = 2

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
...
2 * 9 = 18

반복문 Loop Statement

반복문 - 연습문제 2: 구구단 출력하기 - 정답 예제

```python
n = int(input("n을 입력하세요(2~9): "))

for i in range(1, 10):
    print("{} * {} = {}".format(n, i, n * i))
```

반복문 Loop Statement

반복문 - 연습문제 3: 사칙연산 계산기 만들기

문제: 더하기, 빼기, 곱하기, 나누기 결과를 계산하는 계산기를 만들 것이다. 조건은 다음과 같다.

1. 먼저 더하기, 빼기, 곱하기, 나누기 중 어떤 연산을 할건지 입력받는다.
2. 수 두개를 입력받는다.
3. 계산한 결과를 출력하고 다시 1. 로 돌아와 새로운 입력을 받는다.
4. 만약 0 을 입력한다면 프로그램을 종료한다.

반복문 Loop Statement

반복문 - 연습문제 3: 사칙연산 계산기 만들기

```
/Users/taebbong/anaconda3/bin/python /Users/taebbong/PycharmProjects/HW3/prob89.py
------------------------------
계산기 V1
------------------------------
1. 더하기
2. 빼기
3. 곱하기
4. 나누기
9. 종료
연산을 번호로 고르세요(1, 2, 3, 4, 9): 1
첫번째 수를 입력하세요: 13
두번째 수를 입력하세요: 23
연산 결과는  36
------------------------------
계산기 V1
------------------------------
1. 더하기
2. 빼기
3. 곱하기
4. 나누기
9. 종료
연산을 번호로 고르세요(1, 2, 3, 4, 9): 3
첫번째 수를 입력하세요: 15
두번째 수를 입력하세요: 5
연산 결과는  75
------------------------------
계산기 V1
------------------------------
1. 더하기
2. 빼기
3. 곱하기
4. 나누기
9. 종료
연산을 번호로 고르세요(1, 2, 3, 4, 9): 9
계산기를 종료합니다
------------------------------

Process finished with exit code 0
```

반복문 Loop Statement

반복문 - 연습문제 3: 사칙연산 계산기 만들기 - 정답 예제

```
while True:
    print("-" * 30)
    print("계산기 V1")
    print("-" * 30)
    print("1. 더하기")
    print("2. 빼기")
    print("3. 곱하기")
    print("4. 나누기")
    print("9. 종료")

    opt = int(input("연산을 번호로 고르세요(1, 2, 3, 4, 9): "))

    if opt == 9:
        print("계산기를 종료합니다")
        print("-" * 30)
        break
```

```
a = int(input("첫번째 수를 입력하세요: "))
b = int(input("두번째 수를 입력하세요: "))

if opt == 1:
    result = a + b
elif opt == 2:
    result = a - b
elif opt == 3:
    result = a * b
elif opt == 4:
    result = a / b

print("연산 결과는 ", result)
```

반복문 Loop Statement

HW 5

Prob 167 ~ Prob 200

함수

Function

함수 Function

함수

함수란 무엇일까??

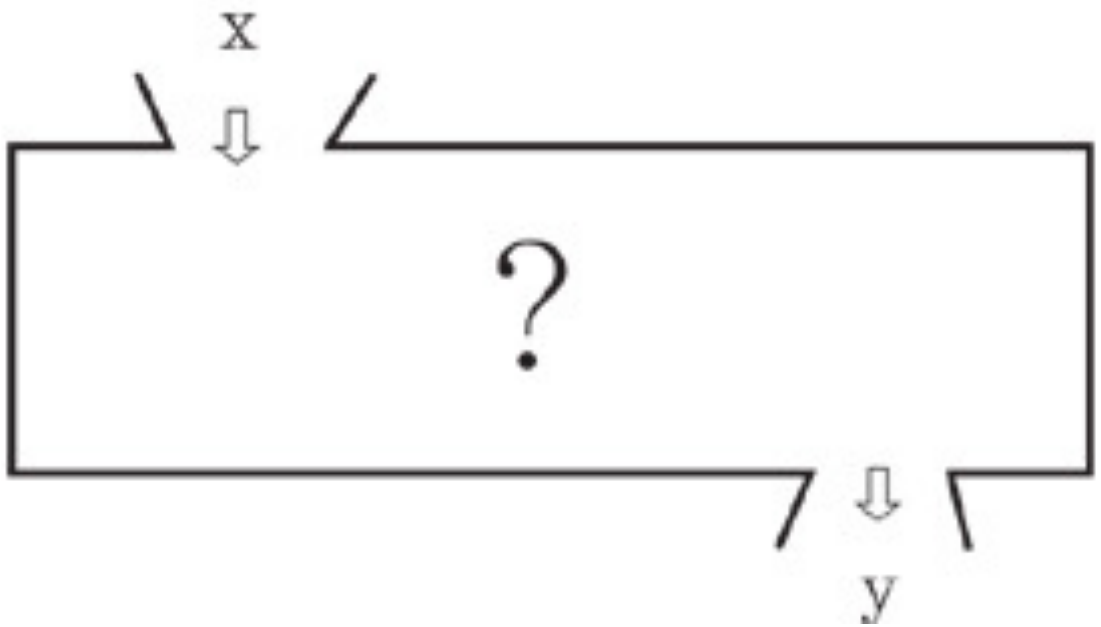
“함수는 “일련한 작업이나 기능의 단위체” 로 정의 가능 그리고 하나 더 알고 가자면 함수는 입력이 있고(parameter), 연산을 해서(function), 반환(return)해준다.”

라고 지난 1강 때 배웠었다.

함수 Function

함수의 이해

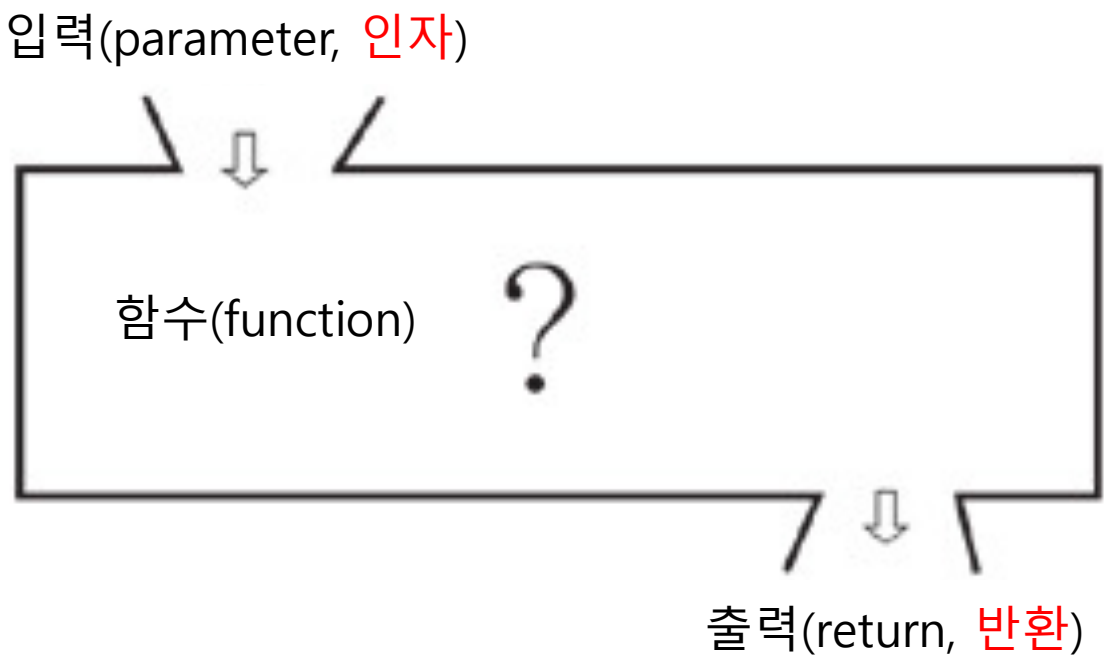
저번에 설명했던 것처럼, 함수를 수학시간에 배웠던 그림으로 이해해보자.



함수 Function

함수의 이해

저번에 설명했던 것처럼, 함수를 수학시간에 배웠던 그림으로 이해해보자.



함수 Function

함수의 예시

대표적인 함수 예시를 살펴보면 다음 함수들이 있다.

print(문자열) # 인자: 문자열, # 함수: 출력(print) # 반환: 없음
n = len(리스트) # 인자: 리스트(자료구조) # 함수: 길이 계산 # 반환: 인자의 길이

함수 Function

함수 만들기

함수는 기능들을 묶어놓은 단위체이기 때문에 별거 없다!

함수를 직접 만들어보자.

함수 Function

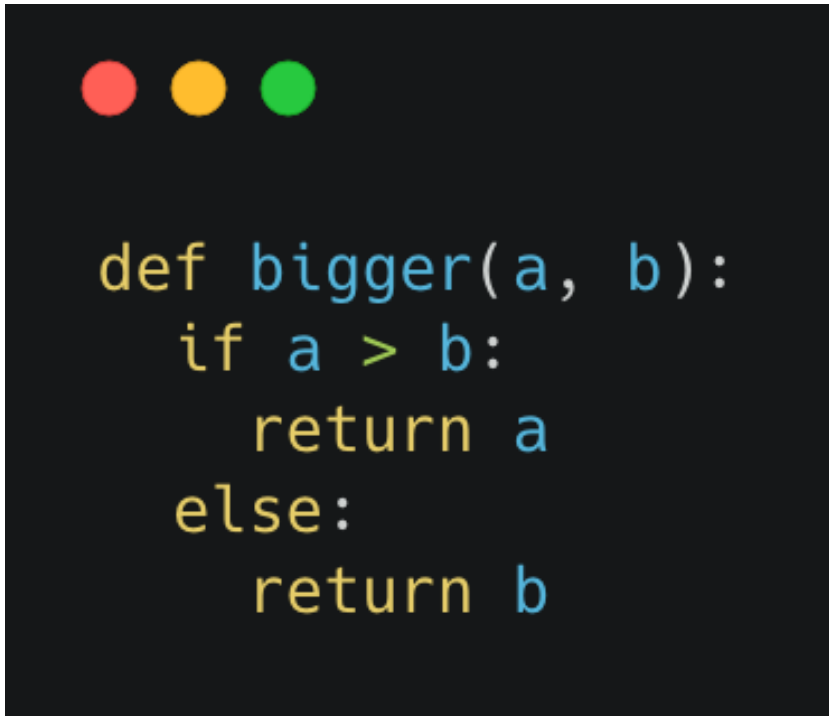
함수 만들기

```
def function_name(parameter1, parameter2, ..):
        do_something_with_parameters
        return result
```

Step1: def 로 함수를 만들겠다! 라고 시작, 함수 이름을 정한다.
Step2: parameter(인자)를 정해서 넣는다.(안 넣어도 된다!)
Step3: parameter를 활용한 기능을 수행한다.
Step4: 내가 원하는 결과물을 return(반환)한다.(return 할게 없으면 안해도 된다!)

함수 Function

함수 만들기 - 예시1 두 숫자 중 큰 숫자 반환하기



```python
def bigger(a, b):
    if a > b:
        return a
    else:
        return b
```

```python
a = 10
b = 20
n = bigger(a, b)
```

함수 Function

함수 만들기 - 예시2 두 숫자 중 큰 숫자 출력하는 함수 만들기

```python
def bigger(a, b):
  if a > b:
    print("{} is bigger!".format(a))
  else:
    print("{} is bigger!".format(b))
```

```python
a = 10
b = 20
bigger(a, b)
```

함수 Function

HW 6

Prob 204 ~ Prob 219

감사합니다:)

Thank you