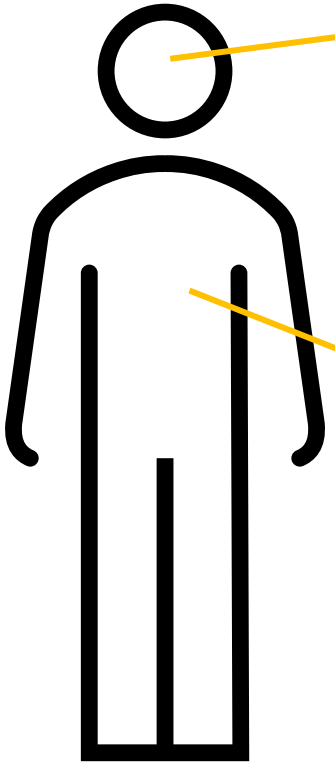
Python 기초

Session 3

NEXT X LIKELION 김지성

Intro

파이썬을 공부하는 이유



웹 서비스

뇌 == django

뼈 == HTML

피부 == CSS

근육 == JS

Intro

파이썬을 공부하는 이유

파이썬은 django를 공부하기 위한 기본기!

프로그래밍 언어

파이썬은 멋사에서 처음으로 배우는 프로그래밍 언어입니다

Dec 2021	Dec 2020	Change	Programming Language
1	3	↑	Python
2	1	↓	C
3	2		Java
4	4		C++
5	5		C#
6	6		Visual Basic
7	7		JavaScript
8	12	⇈	Assembly language
9	10	↑	SQL
10	13	↑	Swift
11	9	↓	R

파이썬이란

1989년 12월…

"크리스마스에 특별히 할 일이 없었다.
연구실은 닫혔고 집에 컴퓨터가 있길래 파이썬을
만들었다."

-귀도 반 로섬-

주의할 점

파이썬에서 들여쓰기는 필수

파이썬에서 들여쓰기(Tab)는 해당 명령의 실행 범위를 설정하는 중괄호 같은 역할!!

```
1    for i in range(10):
2        print(i)
```

```
4    def add(a, b):
5        return a + b
```

실습준비

모두 vscode에서 새 파일을 만들고 터미널을 열어주세요!

파일명은 (파일명).py로 만들어주세요

터미널을 열때는 하단 경계선 드래그 또는 Ctrl + `

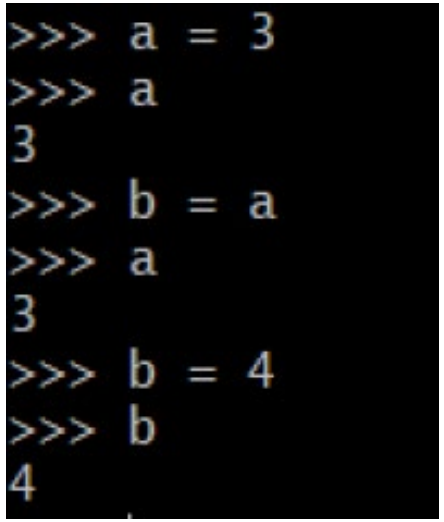
변수

프로그래밍의 기초

변수란 값을 저장하기 위한 이름

변수명 = 저장할 값

등호 오른쪽의 값을 왼쪽에 저장하겠다!
파이썬은 변수의 type을 지정해줄 필요가 없습니다.



자료형

프로그래밍의 기초

- 숫자형(int, float)
- 문자열(string)
- 불(bool)
- 리스트(list)
- 튜플(tuple)
- 딕셔너리(dict)
- 집합(set)

숫자형

사칙연산

덧셈	+
뺄셈	-
곱셈	*
나눗셈	/
제곱	**
나머지	%
몫	//

```
>>> 3 + 2
5
>>> 3 - 2
1
>>> 3 * 2
6
>>> 3 / 2
1.5
>>> 3 ** 2
9
>>> 3 % 2
1
>>> 3 // 2
1
```

숫자형

사칙연산

덧셈	+
뺄셈	-
곱셈	*
나눗셈	/
제곱	**
나머지	%
몫	//

```
1  print("+", 3+2)
2  print("-", 3-2)
3  print("*", 3*2)
4  print("/", 3/2)
5  print("**", 3**2)
6  print("%", 3%2)
7  print("//", 3//2)
```

```
+ 5
- 1
* 6
/ 1.5
** 9
% 1
// 1
```

문자열

쉬운 문자열 연산은 파이썬의 장점

문자열은 큰따옴표 또는 작은따옴표로 묶어줍니다.
(차이 없으나 문자열에 ‘ 또는 “ 가 포함될 경우 서로 다른 기호로 감싸야 함)

문자열 연결	문자열1 + 문자열2 + …
문자열 반복	문자열 * 반복횟수

```
9    a = "I like"
10   b = "python"
11   print(a + " " + b) #I like python
12   print(b * 3) #pythonpythonpython
```

문자열

인덱싱

인덱싱 (특정 위치 문자 반환)	변수이름[인덱스 번호]
슬라이싱 (일부 문자열 추출)	변수이름[시작 인덱스 : **끝 인덱스+1**]

인덱스의 시작은 항상 0입니다
0은 첫번째 인덱스, -1은 마지막 인덱스

```
b = "python"
print(b[0]) #p
print(b[3]) #h
print(b[-1]) #n
print(b[0:2]) #py
print(b[::-1]) #nohtyp
```

문자열 관련 함수

인덱싱

문자열 길이 계산	len(문자열)
문자 개수 세기	문자열.count(세고자 하는 문자)
문자 위치 찾기	문자열.find(찾고자 하는 문자) 있으면 처음 인덱스 없으면 -1 반환

```
>>> a = "likelion"
>>> a.count("l")
2
>>> a = "lion"
>>> len(a)
4
>>> b = "likelion"
>>> b.count("l")
2
>>> b.find("l")
0
>>> b.find("k")
2
>>> b.find("p")
-1
```

문자열 관련 함수

쉬운 문자열 연산은 파이썬의 장점

문자열 삽입	삽입할 문자.join(대상 문자열)
소문자를 대문자로	문자열.upper()
대문자를 소문자로	문자열.lower()

```
>>> a = "aaaaaaa"
>>> "?".join(a)
'a?a?a?a?a?a?a'
```

```
>>> b = "Likelion"
>>> b.upper()
'LIKELION'
>>> b.lower()
'likelion'
```

양쪽 공백 제거	문자열.strip()
문자열 나누기	문자열.split(기준 문자) 기준문자 안 쓰면 공백 기준 분리
문자열 바꾸기	문자열.replace(변화 전 문자열, 변화 후 문자열)

```
>>> a.strip()
'I like python'
>>> b = "a?a?a?a?a"
>>> b.split("?")
['a', 'a', 'a', 'a', 'a']
>>> b.replace("?", "")
'aaaaa'
```

* 오른쪽 공백 제거는 rstrip() 왼쪽 공백 제거는 lstrip()

실습

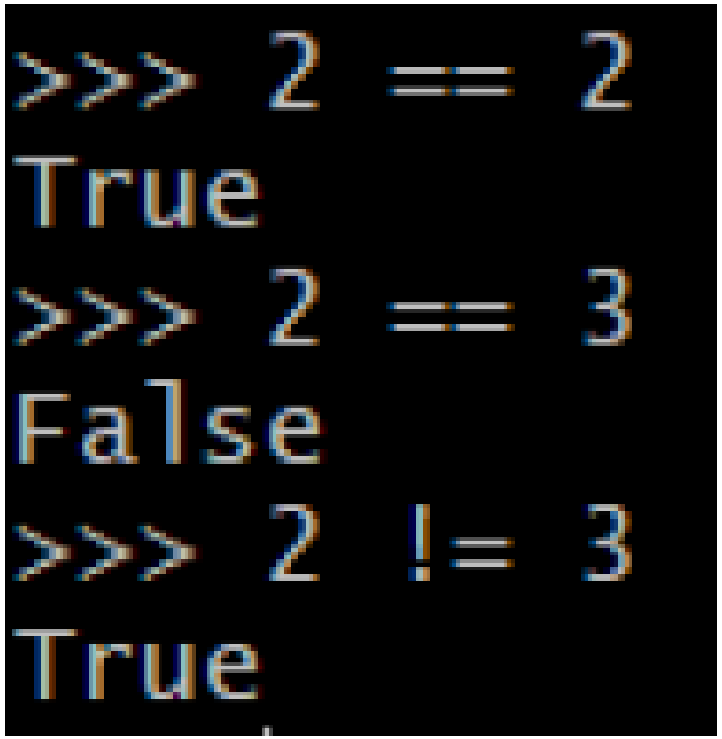
문자열 실습 문제

1. 세션 자료와 함께 포함된 string.py를 열어주세요
2. string.py에 주어진 string1, 2, 3을 조합해 answer을 만드세요

Bool 자료형

참 또는 거짓!

값 대입	=
비교문자(같다)	==
비교문자(다르다)	!=



리스트 자료형

자유롭게 수정이 가능하다

리스트 연결	리스트1 + 리스트2 + …
리스트 반복	리스트 * 반복횟수
인덱싱 (특정 위치 문자 반환)	리스트이름[인덱스 번호]
슬라이싱 (일부 문자열 추출)	리스트이름[시작 인덱스, **끝 인덱스+1**]

```
>>> a = [1,2,3,4]
>>> b = ["1","2","3","4"]
>>> c = []
>>> d = list()
```

문자열은 리스트처럼 취급할 수 있습니다.

따라서 문자열과 똑같습니다!

리스트 자료형

자유롭게 수정이 가능하다

맨 뒤에 삽입	리스트.append(삽입할 요소)
특정 위치 삽입	리스트.insert(인덱스, 삽입할 값)
맨 뒤에 삭제	리스트.pop()
특정 위치 삭제	del 리스트[인덱스] 리스트.remove[아이템]

```
>>> a = [1,2,3]
>>> a.append(4)
>>> a
[1, 2, 3, 4]
>>> a.pop()
4
>>> a
[1, 2, 3]
>>> a.insert(3,4)
>>> a
[1, 2, 3, 4]
>>> del a[3]
>>> a
[1, 2, 3]
```

리스트에 대한 함수는 매우 많습니다. 다른 함수들은 구글링!

튜플 자료형

잘 안 써요...

리스트와 유사하지만 수정, 삽입, 삭제가 불가능합니다.

수정되어서는 안될 내용을 정의할 때 주로 사용해요!

```
>>> a = (1,2,3,4)
>>> b = tuple()
>>> a[0]
1
>>> a.append(5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
```

딕셔너리 자료형

Key값 & value값

Key 값으로 접근하여 Value값을 가져올 수 있습니다.
대신 순서는 없어요!
***탐색이 매우매우 빠름**

```
21  profile = {
22      "name": "jiseong",
23      "age" : "22",
24  }
25
26  print(profile["name"]) #jiseong
27  print(profile["age"]) #22
28  profile["major"] = "ELL"
29  print(profile) #{'name': 'jiseong', 'age': '22', 'major': 'ELL'}
```

Set 자료형

중복제거

중복을 허용하지 않으므로 중복 제거에 주로 사용됩니다
순서는 보존되지 않아요

```
user_set = set(["1", "1", "1", "2", "2"])
print(user_set) #{'2', '1'}
```

자료형 타입 변환

자료형들은 서로 변환할 수 있습니다.

```
54
55    print(float(10)) #실수형 변환/ 출력 10.0
56    print(int("10")) #정수형 변환/ 출력 10
57    print(list("안녕하세요")) #리스트 변환/ 출력 ['안', '녕', '하', '세', '요']
58    print(int("10") + int("20")) #출력 30
59    print(str(10) + str(30)) #출력 1030
60
```

조건문

들여쓰기는 필수!

```
16    money = 200
17
18    if money > 5000:
19        print("택시를 타고 가세요")
20    elif money > 1250:
21        print("지하철을 타고 가세요")
22    else:
23        print("걸어가세요")
```

포함여부

In 또는 not in

```
likelion = ["10기", "9기", "8기"]
if "10기" in likelion:
    print("멋사 소속입니다.")
elif "10기" not in likelion:
    print("멋사 소속이 아닙니다.")
```

실습

조건문 실습 문제

1. 세션 자료와 함께 포함된 if.py를 풀어주세요

반복문

For/ while

While은 조건이 걸리지 않는다면 무한히 반복하는 반복문입니다.
따라서 반복문을 끝내고 빠져나갈 조건이 필요합니다.

```
count = 1

while count < 5:
    print("안녕하세요")
    count += 1 #count = count + 1과 같음
```

```
40  count = 1
41
42  while True:
43  #(또는 while 1)
44      print("안녕하세요")
45      count += 1 #count = count + 1과 같음
46      if count > 5:
47          break
```

반복문

For/ while

For문은 반복 횟수에 대한 조건을 미리 정합니다.
range(시작 인덱스, 끝인덱스+1, step(증가량))
***끝인덱스만 쓸 경우 기본값으로 0에서부터 1씩 증가함**

```
1    for i in range(1, 6, 1):
2        print("%d번째 반복 중" % i)
3        print("안녕하세요")
```

```
1    for i in range(5):
2        print("%d번째 반복 중" % i)
3        print("안녕하세요")
```

반복문

For/ while

For문은 반복 횟수에 대한 조건을 미리 정합니다.
range(시작 인덱스, 끝인덱스+1, step(증가량))
***끝인덱스만 쓸 경우 기본값으로 0에서부터 1씩 증가함**

```
for i in range(1, 6, 1):
    print("%d번째 반복 중" % i)
    print("안녕하세요")
```

```
for i in range(5):
    print("%d번째 반복 중" % i)
    print("안녕하세요")
```

```
for i in range(5):
    if i % 2 == 0: # i가 짝수라면
        continue # 이번 반복에서는 내 밑 코드는 전부 skip

    print("%d번째 반복 중" % i)
```

반복문

For/ while

For문은 리스트의 값도 뽑아낼 수 있습니다

```
48
49  test = ["안녕", "Hello", "Hola"]
50  for i in test:
51      print(i)
```

```
안녕
Hello
Hola
```

```
50
51  test = "안녕하세요"
52  for i in test:
53      print(i)
```

```
안
녕
하
세
요
```

실습

반복문 실습 문제

1. 세션 자료와 함께 포함된 repeat.py를 풀어주세요

문자열 포맷팅

문자열에 변화를 주고 싶을 때!

정수	%d
실수	%f
문자	%c
문자열	%s

Enter	\n
Tab	\t

```
26    # %포맷팅
27    print("정수 출력 %d" % 123)
28    print("실수 출력 %f" % 123.12)
29    print("문자 출력 %c \t 문자열 출력 %s" % ("호", "안녕"))
30
```

*이 외에도 f-string, format 등 여러가지 포맷팅 방법과 다양한 이스케이프 문이 있습니다.

함수

Input을 넣으면 Output이 나온다!

특정 작업들을 여러 번 반복해야 할 때, 여러 번 같은 코드를 적는 대신
1. 해당 작업들을 수행하는 함수를 작성하고
2. 필요할 때마다 함수를 호출합니다.

함수의 return

Return이 두 번 나올 수는 없어요

코드를 다 실행하거나
return을 만나면 함수가 종료됩니다.
따라서 함수의 결과값은 최대 한 개입니다!

```python
# 입력값 O, 결과값 O
def add(a, b):
    return a + b

# 입력값 O, 결과값 X
def show_add(a, b):
    print(a + b)

# 입력값 X, 결과값 O
def get_number():
    return 4

# 입력값 X, 결과값 X
def say_hello():
    print("hello")
```

함수의 선언

선언 위치에 주의하세요

함수를 쓰는 위치보다 먼저 함수를 정의해야 실행할 수 있어요
함수이름(인자)

```
61  def add(a, b):
62      return a + b
63
64  print("1과 2를 더하면 결과는 %d입니다.", add(1,2))
```

실습

함수 실습 문제

1. 세션 자료와 함께 포함된 def.py를 풀어주세요

Session3 과제1

hw1.py와 hw2.py에 있는 문제를 풀어주세요
(오늘 배운 내용을 한번에 정리할 수 있는 문제이니 검색하지 말고
직접 풀어보세요!)
마감기한 : 3월 24일까지

Session3 과제2

에브리타임 모바일 화면 클론코딩
마감기한 : 3월 28일까지