

파이썬 프로그래밍

1회차: 프로그래밍 언어에 대한 이해

목차

1회차: 목표 공유, 왜 파이썬인가?

- 교수 소개
- 강의 목표
- 파이썬: 변수와 상수
 - ▶ 숫자형
 - ▶ 문자열
 - ▶ 리스트
 - ▶ 튜플
 - ▶ 딕셔너리

파이썬

설치

07

Python 시작! - 설치

파이썬

입문

08

코드

```
print("Hello world")
```

파이썬

변수와 상수

09

■ 변수와 상수

- 데이터를 담는 그릇 => “무엇을 어떻게 해줘”에서 무엇에 해당
- 다양한 종류의 데이터를 담을 수 있다!
 - ▶ 문자열, 숫자, 리스트 등 ...
- 프로그램에서 데이터를 다루기 위해선 무조건 그릇에 담아야 한다

■ 변수

- 변할 수 있는 값을 담는 그릇
- 이름을 지정하여 사용
 - ▶ 이름을 잘 지어야 함

■ 상수

- 변할 수 없는 값을 담는 그릇
- 파이썬엔 없다

파이썬

변수와 상수

10

코드

```
alghost = 4
```

결과

```
>>> alghost = 4
>>> print(alghost)
4
```

변수의 종류: 자료형

	설명	모습
숫자형	정수, 실수 등의 숫자를 다루는 자료형	0 or 1.25 or -123
문자열	문자열을 다루는 자료형	'alghost'
리스트	다른 자료형의 모음을 다루는 자료형	[1, 'alghost', 123]
튜플	리스트와 같지만 수정이 불가능한 자료형	(1, 'alghost', 123)
딕셔너리	키와 값으로 이루어진 자료형	{'name': 'alghost'}
...

파이썬에서 자료형은..?

- 데이터를 담는 순간!
그 변수의 자료형이 정해짐
- 자료형 마다 연산을 지원함

```
>>> alghost = 4
>>> print(alghost)
4
>>> alghost = 'awesome!'
>>> print(alghost)
awesome!
>>> alghost = [0, 100, 200]
>>> print(alghost)
[0, 100, 200]
```

파이썬

변수와 상수

12

숫자형

- 정수, 실수 등과 같은 숫자를 담는 자료형

```
>>> alghost = 1123
>>> alghost = -123
>>> alghost = 0
```

alghost 변수에 정수 넣기

숫자형-정수

```
>>> alghost = 1.0
>>> alghost = 10.0123
>>> alghost = -1.4123
```

alghost 변수에 실수 넣기

숫자형-실수

숫자형 - 연산

	기호	설명	예시	result 값
더하기	+	덧셈	result = 4 + 5	9
빼기	-	뺄셈	result = 5 - 4	1
곱하기	*	곱셈	result = 5 * 4	20
나누기	/	나눗셈	result = 5 / 4	1.25
제곱	**	제곱	result = 5 ** 3	125
나머지	%	나누었을 때 나머지	result = 11 % 6	5
몫	//	나누었을 때 몫	result = 11 // 6	1

파이썬

변수와 상수

14

문자열

- 문자열을 담는 자료형
- 파이썬의 큰 장점중 하나 => 문자열을 내맘대로 다루기가 편함!

```
>>> a = 'Python is awesome!'  
>>> a = "Python is awesome!"  
>>> a = '''Python is awesome'''  
>>> a = """Python is awesome!"""
```

문자열을 표현하는 4가지 방법

문자열

왜 4가지 방법이나 필요할까?

파이썬

변수와 상수

15

문자열

	목적	예시
'string'	문자열 안에 "를 포함시키기 위해	a = 'Send: "I am alghost"'
"string"	문자열 안에 '를 포함시키기 위해	a = "I'm tutor"
"""string"""	문자열 안에 "를 포함하고, 여러행의 문자열을 다루기 위해	a = """ From: "alghost" To: "You" """
"""string"""	문자열 안에 '를 포함하고, 여러행의 문자열을 다루기 위해	a = """ Mailto: 'test@test.com' Contents: 'test' """

너무 복잡..., 그리고 ‘ “ 둘다 넣고 싶으면?

파이썬

변수와 상수

16

문자열

	목적	예시
'string'	문자열 안에 "를 포함시키기 위해	a = 'Send I am alghost''
"string"	문자열 안에 '를 포함시키기 위해	a = "I'm tutor"
"""string"""	문자열 안에 "를 포함하고, 여러행의 문자열을 다루기 위해	a = """ From: "alghost" To: "Fastcampus" """
"""string"""	문자열 안에 '를 포함하고, 여러행의 문자열을 다루기 위해	a = """ Mailto: 'test@test.com' Contents: 'test' """

문자열

- ‘와 “중에 편한 걸 쓰자! 일관성만 있게
- 문자열에서 ‘, “ 둘다 쓰고싶다고!
 - ▶ 이스케이프 코드 약속된 기호
- 당연히..? 이 코드도 엄청나게 많지만 쓸일 없다

코드	설명
\n	개행 (줄바꿈)
\t	탭
\\	문자 그대로의 \
\'	문자 그대로의 '
\"	문자 그대로의 "

파이썬

변수와 상수

18

문자열

- 예시를 통해 확인 해보자

```
>>> a = "\"Python\" is awesome. \n\tHello Python"
>>> print(a)
"Python" is awesome.
        Hello Python
```

이스케이프 코드로 위처럼 표현이 가능하지만.. 개행 구분이 쉽지 않다

문자열

```
>>> a = """\"Python\" is awesome.
...      Hello Python"""
>>> print(a)
"Python" is awesome.
        Hello Python
```

개행시 이스케이프 코드가 아니기 때문에 개행 구분이 쉽다

문자열

파이썬

변수와 상수

19

문자열 - 연산

	기호	설명	예시
더하기	+	문자열 붙이기	<pre>>>> a = "Python" >>> b = "awesome" >>> print(a + " is " + b) Python is awesome</pre>
곱하기	*	문자열 반복하기	<pre>>>> a = "-" * 30 >>> print(a) -----</pre>

파이썬

변수와 상수

20

문자열 - 인덱싱, 슬라이싱

- 인덱싱은 문자열에서 특정 문자를 가리키는 것을 의미
- 슬라이싱은 문자열에서 특정 문자열을 가리키는 것을 의미

```
>>> a = "Python"
>>> print(a[0])
P
>>> print(a[1])
y
>>> print(a[2])
t
>>> print(a[3])
h
```

인덱싱 예제

문자열 - 인덱싱

```
>>> a = "Python"
>>> print(a[0:3])
Pyt
>>> print(a[4:6])
on
>>> print(a[:4])
Pyth
>>> print(a[2:])
thon
```

슬라이싱 예제

문자열 - 슬라이싱

파이썬

변수와 상수

21

문자열 - 내장함수

- 내장함수란 파이썬이 기본적으로 제공하는 함수
- 문자열 내장함수: 문자열 자료형이 기본적으로 제공하는 함수

아니 함수가 뭔데..

함수

- 함수는... 뭔가를 자동으로 해주는 명령..?
- 특정 입력값에 의해 정해진 동작을 수행하고 결과값을 내는 기능 이라고 간단히...
 - ▶ 결과를 반환하는 함수와 반환하지 않는 함수로 구분됨
- 3회차에 자세히!
 - ▶ (함수가 뭔지 몰라도 예제 몇 개 보면 쓰는데 문제없다...)

문자열 - 내장함수

함수명	설명	사용방법	result 값
count	특정 문자 수를 반환	<pre>a = 'Python' result = a.count('y')</pre>	1
find	특정 문자 위치를 반환	<pre>a = 'Python' result = a.find('y')</pre>	1
index	특정 문자 위치를 반환	<pre>a = 'Python' result = a.index('y')</pre>	1
join	문자 사이에 입력한 문자를 삽입	<pre>a = ',' result = a.join('abc')</pre>	a,b,c

문자열 - 내장함수

함수명	설명	사용방법	result 값
upper	대문자로 변환한 값 반환	<pre>a = 'Python' result = a.upper()</pre>	PYTHON
lower	소문자로 변환한 값 반환	<pre>a = 'Python' result = a.lower()</pre>	python
replace	문자열을 치환한 결과 반환	<pre>a = 'Python' result = a.replace('Py', 'Sy')</pre>	Sython
split	문자열 나누는 결과 반환	<pre>a = 'Python is awesome' result = a.split(' ')</pre>	['Python', 'is', 'awesome']

문자열 - 내장함수

함수명	설명	사용방법	result 값
lstrip	왼쪽 공백 제거한 값 반환	<pre>a = ' Python '</pre> <pre>result = a.lstrip()</pre>	'Python '
rstrip	오른쪽 공백 제거한 값 반환	<pre>a = ' Python '</pre> <pre>result = a.rstrip()</pre>	' Python'
strip	양쪽 공백 제거한 값 반환	<pre>a = ' Python '</pre> <pre>result = a.strip()</pre>	'Python'

파이썬

변수와 상수

25

리스트

- 데이터의 모음을 담기 위한 자료형
- 데이터의 추가, 삭제, 수정이 자유로움
- 중복된 데이터도 가능, 넣을 수 있는 데이터의 자료형도 자유!
- 인덱싱으로 데이터를 다룰 수 있음!

```
>>> a = [0, 123, 'alghost', 0, 1.1]
>>> a = []
```

리스트 자료형 예제

→ 리스트

파이썬

변수와 상수

26

리스트 - 연산

	기호	설명	예시
더하기	+	리스트 붙이기	<pre>>>> a = [1,2,3] >>> b = [6,7,8] >>> c = a + b >>> print(c) [1, 2, 3, 6, 7, 8]</pre>
곱하기	*	리스트 반복하기	<pre>>>> a = [1,2,3] >>> b = a * 3 >>> print(b) [1, 2, 3, 1, 2, 3, 1, 2, 3]</pre>

파이썬

변수와 상수

27

리스트 - 인덱싱, 슬라이싱

- 인덱싱은 리스트에서 특정 값을 가리키는 것을 의미
- 슬라이싱은 리스트에서 특정 값들을 가리키는 것을 의미
- 문자열과 같은 원리!

```
>>> a = [1, 'alghost', 123, 'test']
>>> print(a[1])
alghost
>>> print(a[0])
1
```

인덱싱 예제

리스트 - 인덱싱

```
>>> a = [1, 'alghost', 123, 'test']
>>> print(a[:3])
[1, 'alghost', 123]
>>> print(a[1:4])
['alghost', 123, 'test']
```

슬라이싱 예제

리스트 - 슬라이싱

```
>>> a = [0, ['123', 'alghost'], [0, 123, 456]]
>>> print(a[0])
0
>>> print(a[1])
['123', 'alghost']
>>> print(a[1][0])
123
>>> print(a[1][1])
alghost
```

이런 것도 가능하다!!
리스트 안의 리스트 :D

리스트 안의 리스트

리스트

인덱싱으로 삭제하기

- 리스트는 인덱싱으로 데이터 삭제가 가능!
- 튜플은 데이터 수정이 불가능하기 때문에 삭제 불가능

```
>>> data = ["Python", 0, 100]
>>> print(data)
["Python", 0, 100]
>>> del(data[0])
>>> print(data)
[0, 100]
```


리스트 - 내장함수

함수명	설명	사용방법	result 값
append	요소를 뒤에 추가	<pre>result = [1,2,3] result.append(4)</pre>	[1,2,3,4]
sort	요소들을 정렬	<pre>result = ['a', 'c', 'b'] result.sort()</pre>	['a','b','c']
reverse	요소들을 뒤집음	<pre>result = [1,10,100] result.reverse()</pre>	[100,10,1]
index	입력값의 위치를 반환 (첫번째로 찾은 위치)	<pre>a = [10,11,11,100] result = a.index(11)</pre>	1

리스트 - 내장함수

함수명	설명	사용방법	result 값
insert	특정 위치에 요소를 추가	<pre>result = [100,192,101] result.insert(1, 'a')</pre>	[100,'a',192,101]
remove	입력값을 삭제 (첫번째로 찾은 위치)	<pre>result = [10,11,100,11] result.remove(11)</pre>	[10,100,11]
pop	마지막 요소를 꺼내고 삭제	<pre>a = [10,101,102,103] result = a.pop()</pre>	103 (a: [10,101,102])
count	입력값의 갯수	<pre>a = [10,10,101,102,10,'a'] result = a.count(10)</pre>	3

파이썬

변수와 상수

31

튜플

- 리스트와 매-우 흡사
- 하지만, 리스트와 달리 수정, 삭제, 추가 불가능 => 오직 읽기만 지원
- 그렇다면 왜 필요할까?
 - ▶ 내부적으로 성능이 더 좋음 => 데이터가 많~~~을 경우 영향이 있을지도?
 - ▶ 실수 방지: 변경하면 안되는 값에 대한 보호 가능

```
>>> data = (1, 123, "Python", 0, 100)
>>> data = ()
```

튜플 자료형 예제

튜플

연산, 인덱싱, 슬라이싱 전부 리스트와 동일
내장함수의 경우 값을 변경하는 함수 제외하고 사용가능

파이썬

변수와 상수

32

딕셔너리

- 대응관계를 나타낼 수 있는 자료형
- 대응관계란?!
 - ▶ 이름: 이태화
 - ▶ 나이: 29세
- 위 처럼 'key'와 'value'가 연결된 자료형이다!
- (사람의 표현 방법에 가장 가까운 자료형이라고 생각..)

key	value
name	taehwa
age	29
height	secret

```
>>> a = {'name': 'taehwa', 'age': 30, 'height': 'secret'}
```

딕셔너리 자료형 예제

딕셔너리

위 테이블과 딕셔너리 자료형 예제는 같다고 볼 수 있다

파이썬

변수와 상수

33

■ 딕셔너리 - 연산

- 을 지원하지 않는다 :)

파이썬

변수와 상수

34

■ 딕셔너리 - 인덱싱

- 슬라이싱은 지원하지 않는다
- 앞서 설명한대로! 인덱싱은 key로 가능

```
>>> a = {'name': 'taehwa', 'age': 30}
>>> print(a['name'])
taehwa
>>> print(a['age'])
30
```

숫자가 아닌 key로 값을 가져올 수 있기 때문에
사람이 인지하기 편하다

인덱싱으로 삭제하기

- 딕셔너리는 인덱싱으로 데이터 삭제가 가능!
- 튜플은 데이터 수정이 불가능하기 때문에 삭제 불가능
- 문자열은 인덱싱을 지원하지만 인덱싱에 의한 삭제는 지원하지 않음!

```
>>> a = {'name': 'taehwa', 'age': 30}
>>> print(a)
{'name': 'taehwa', 'age': 30}
>>> del(a['name'])
>>> print(a)
{'age': 30}
```

딕셔너리 - 내장함수

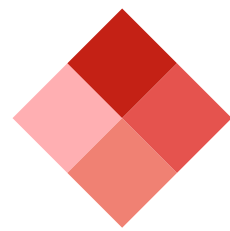
함수명	설명	사용방법	result 값
keys	딕셔너리의 key들을 반환	<pre>a = { 'a' : 123, 'b' : 456 } result = a.keys()</pre>	<code>['a', 'b']</code>
values	딕셔너리의 value들을 반환	<pre>a = { 'a' : 123, 'b' : 456 } result = a.values()</pre>	<code>[123, 456]</code>
items	key,value 쌍을 반환	<pre>a = { 'a' : 123, 'b' : 456 } result = a.items()</pre>	<code>[('a', 123), ('b', 456)]</code>
get	key에 대한 값을 반환 (값이 없는 경우 기본값 지정 이 가능)	<pre>a = { 'a' : 123, 'b' : 456 } result = a.get('c', 789)</pre>	<code>789</code>

■ 변수

- 데이터를 담을 그릇
- 여러 종류가 있음: 숫자형, 문자열, 리스트, 딕셔너리, 튜플 등
 - ▶ 기본이 되는 자료형이기 때문에 잦은 사용으로 눈,손에 익히길 권장
 - ▶ 어차피 자주 시킬(?) 예정이니 걱정 안하셔도...
 - ▶ 내장함수를 암기하면 좋지만 굳이 그럴 필요 X!! => 필요할 때 찾아보면 OK

■ 당부의 말씀

- ‘언어’는 써봐야 아는 것!
- 눈으로 보면 다 아는 것 같고..
- 단순히 따라 치는 것과 눈으로 보는 것이나 다른 게 없어 보이지만
- 말로 하는 언어와 같다고 생각하시고 자주 타이핑 해보시길 바랍니다 :D
- 최소한 수업시간 만큼은 꼭!!!



Good Bye

See you next time