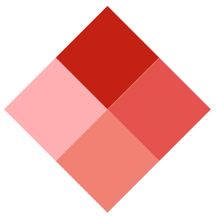


파이썬을 활용한 업무자동화

1회차: 프로그래밍 언어에 대한 이해

목차

1회차: 캠프의 목표 공유, 왜 파이썬인가?

- 강사 소개
- 강의 목표
- 프로그래밍을 한다?
- 파이썬: 변수와 상수
  - 숫자형
  - 문자열
  - 리스트
  - 튜플
  - 딕셔너리

강사 소개

이태화

2015.02 ~ 현재: (주)글루시스 재직

2013.02 ~ 2015.02: 한양대학교 컴퓨터 소프트웨어학과 박사 수료

2011.02 ~ 2013.02: 한양대학교 전자컴퓨터통신공학 석사 졸업

2007.02 ~ 2011.02: 안양대학교 컴퓨터학 학사 졸업

Contacts

Facebook: https://www.facebook.com/alghostman

email: alghost.lee@gmail.com

kakao: @alghostlee

강의 목표

내 일을 컴퓨터가 한다!

자기 업무에 자동화 끼얹기

1. 프로그래밍에 흥미 얻기
2. 준비된 업무 자동화 익히기
3. 업무 중에 필요한 “자동화”가 커리큘럼에 없으면 요청해서 배워가기
4. 회사에서 프로그램 돌리면서 일하는 척 하기

프로그래밍을 한다?

어떻게?

컴퓨터와 대화

- 컴퓨터가 이해할 수 있는 언어! : 프로그래밍 언어
- 프로그래밍 언어를 통해 컴퓨터가 해야할 일을 작성
  - “**무엇**을 **어떻게** 하고, **무엇**을 **어떻게 어떻게** … 해줘!”

프로그래밍 언어

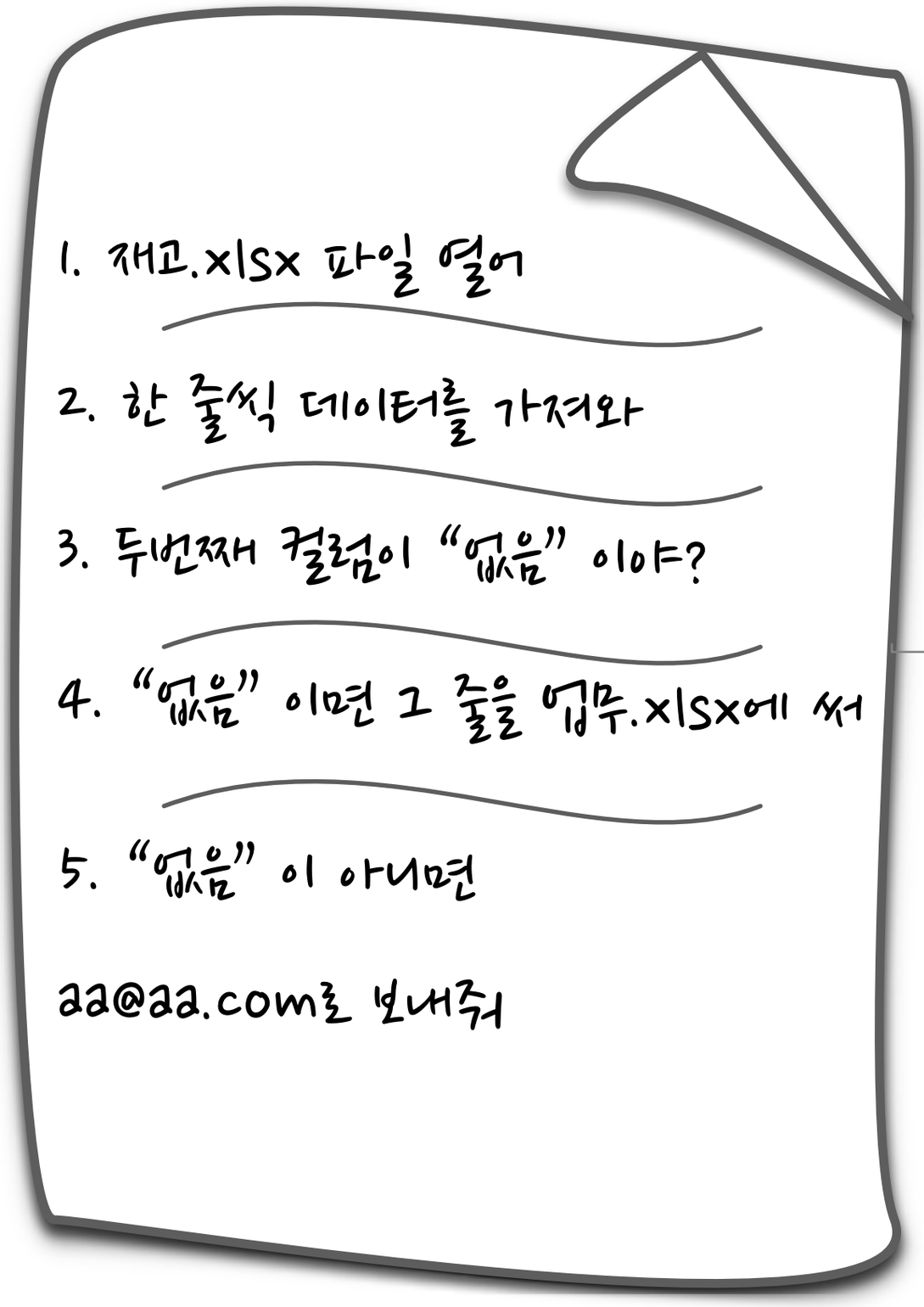
- 프로그래밍 언어는 굉~장히 다양
  - C언어, Ruby, C++, C#, Python, Java 등..
- 즉, 이 모든 언어들로 우리의 목표를 이룰 수 있음
- 따라서 우리는 쉬운 걸 택하고 시작 : **Python**

예: 영어만 이해할 수 있는 직원에게 업무 지시

- 이해할 수 있는 언어: 영어
- 영어를 통해 직원이 해야할 일을 지시

프로그래밍을 한다?

어떻게?



이렇게 작성할 수 있다면
당신도
프로그래밍을 할 수 있다!

Programming

프로그래밍을 한다?

어렵지 않을까..

언어는 당연히 어렵다

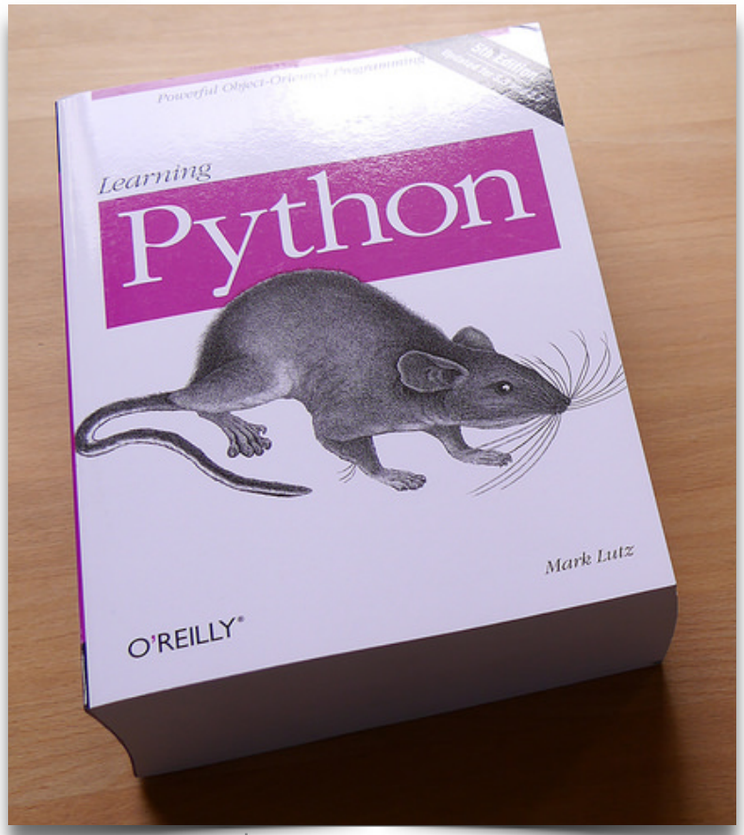
- 프로그래밍 언어도 “언어”
- 당연히! 굉장히 많은 요소를 포함
  - 프로그래밍 전공 서적 두께가..

하지만 우리는

- 아마도? 아직은? 프로그래밍이 주 업무가 아님!
- 또 다른 업무나 아르바이트가 아닌 **도구로서의 프로그래밍이 필요**
- 당연히 내가 하고싶은 작업에 **필요한 것만 배우고 싶음!**

그래도 우리는

- “언어”를 배우기 때문에 영어 공부하듯, **잦은 사용이 중요**
- 어려움을 인정하고 **강사를 귀찮게 해야함**



책 두께를 보라..
당연히 많은 내용이 있고,
우린 다 배울필요가 없다

Learning Python

파이썬

파이썬이란?

파이썬의 기본

- 버전이 2가지가 있고, 사용법이 약간 다름
  - **우리가 사용하는 버전은 3.x**
- 스크립트 언어
  - 우리가 한줄 한줄 작성할 때마다 실행이 됨
  - 장점
    - 작성한 코드를 실행가능한 프로그램으로 만들(컴파일) 필요가 없음
    - 스크립트를 실행해줄 프로그램(파이썬)만 설치하면 어디든 실행가능
    - 배우기가 (상대적으로) 쉬움
  - 단점
    - 성능이 느림

Python 시작! - 설치

파이썬

입문

코드

```
print(“Hello fastcampus”)
```

결과

```
>>> print("Hello fastcampus")
Hello fastcampus
```

파이썬

변수와 상수

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

코드

```
alghost = 4
```

결과

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

파이썬

변수와 상수

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

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

파이썬

변수와 상수

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

숫자형 - 연산

- [주의1]정수와 실수를 연산하면 결과는 실수
- [주의2]실수를 나누면 몫이 아니라

  나머지를 포함한 실수가 나옴
- [주의3]음수를 나누었을 때 나머지가 있는 경우

  -1가 추가된 값이 나옴

```
>>> a = 123 + 1.1
>>> print(a)
124.1
```

[주의1] 예시

숫자형 - 연산

```
>>> a = 1.3 / 1.1
>>> print(a)
1.1818181818181817
```

[주의2] 예시

숫자형 - 연산

```
>>> a = -11 // 2
>>> print(a)
-6
```

[주의3] 예시

숫자형 - 연산

파이썬

변수와 상수

문자열

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

```
>>> a = 'Fastcampus is awesome'
>>> a = "Fastcampus is awesome"
>>> a = '''Fastcampus is awesome'''
>>> a = """Fastcampus is awesome"""
```

문자열을 표현하는 4가지 방법

문자열

왜 4가지 방법이나 필요할까?

파이썬

변수와 상수

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’ ””””

너무 복잡…, 그리고 ‘ “ 둘다 넣고 싶으면?

파이썬

변수와 상수

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’ ”””

파이썬

변수와 상수

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

문자열

- 예시를 통해 확인 해보자

```
>>> a = "\"Fastcampus \" is awesome. \n\tI want \'this course\' to be helpful"
>>> print(a)
"Fastcampus " is awesome.
        I want 'this course' to be helpful
```

이스케이프 코드로 위처럼 표현이 가능하지만.. 개행 구분이 쉽지 않다

문자열

```
>>> a = """\"Fastcampus \" is awesome.
...     I want \'this course\' to be helpful"""
>>> print(a)
"Fastcampus " is awesome.
        I want 'this course' to be helpful
```

개행시 이스케이프 코드가 아니기 때문에 개행 구분이 쉽다

문자열

파이썬

변수와 상수

문자열 - 연산

	기호	설명	예시

더하기	+	문자열 붙이기	`>>> a = 'Fastcampus'`
			`>>> b = 'awesome'`
			`>>> print(a + ' is ' + b)`
			`Fastcampus is awesome`

곱하기	*	문자열 반복하기	`>>> a = '-' * 30`
			`>>> print(a)`
			`------------------------------`

파이썬

변수와 상수

문자열 - 인덱싱, 슬라이싱

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

```
>>> a = 'Fastcampus'
>>> print(a[0])
F
>>> print(a[1])
a
>>> print(a[2])
s
>>> print(a[3])
t
```

인덱싱 예제

문자열 - 인덱싱

```
>>> print(a[0:3])
Fas
>>> print(a[4:10])
campus
>>> print(a[:4])
Fast
>>> print(a[4:])
campus
```

슬라이싱 예제

문자열 - 슬라이싱

파이썬

변수와 상수

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…)

파이썬

변수와 상수

문자열 - 내장함수

함수명	설명	사용방법	result 값
count	특정 문자 수를 반환	a = ‘Fastcampus’ result = a.count(‘a’)	2
find	특정 문자 위치를 반환	a = ‘Fastcampus’ result = a.find(‘a’)	1
index	특정 문자 위치를 반환	a = ‘Fastcampus’ result = a.index(‘a’)	1
join	문자 사이에 입력한 문자를 삽입	a = ‘,’ result = a.join(‘abc’)	a,b,c

파이썬

변수와 상수

문자열 - 내장함수

함수명	설명	사용방법	result 값
upper	대문자로 변환한 값 반환	a = ‘Fastcampus’ result = a.upper()	FASTCAMPUS
lower	소문자로 변환한 값 반환	a = ‘Fastcampus’ result = a.lower()	fastcampus
replace	문자열을 치환한 결과 반환	a = ‘Fastcampus’ result = a.replace(‘Fast’, ‘Slow’)	Slowcampus
split	문자열 나눈 결과 반환	a = ‘Fastcampus is awesome’ result = a.split(‘ ‘)	[‘Fastcampus’, ‘is’, ‘awesome’]

파이썬

변수와 상수

문자열 - 내장함수

함수명	설명	사용방법	result 값
lstrip	왼쪽 공백 제거한 값 반환	a = ‘ Fastcampus ‘ result = a.lstrip()	‘Fastcampus ‘
rstrip	오른쪽 공백 제거한 값 반환	a = ‘ Fastcampus ‘ result = a.rstrip()	‘ Fastcampus’
strip	양쪽 공백 제거한 값 반환	a = ‘ Fastcampus ‘ result = a.strip()	‘Fastcampus’

파이썬

변수와 상수

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 = []
```

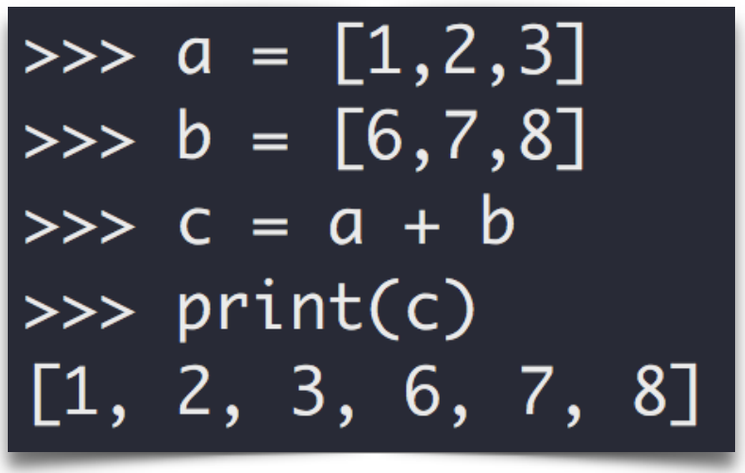
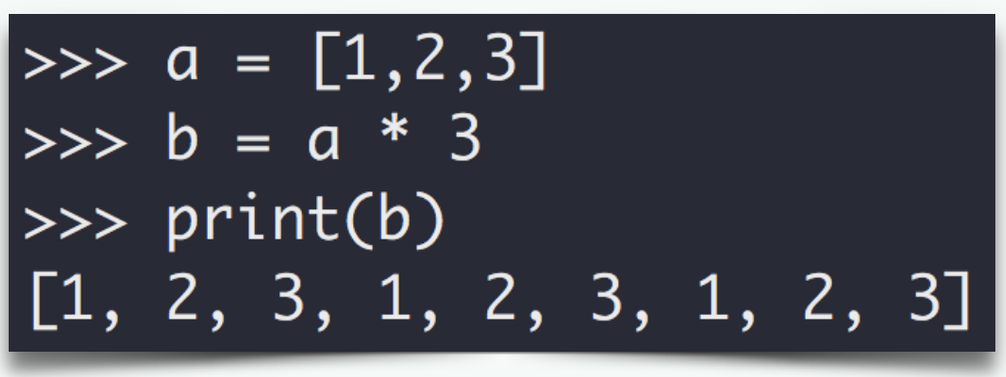
리스트 자료형 예제

리스트

파이썬

변수와 상수

리스트 - 연산

	기호	설명	예시
더하기	+	리스트 붙이기	
곱하기	*	리스트 반복하기	

```
>>> a = [1,2,3]
>>> b = [6,7,8]
>>> c = a + b
>>> print(c)
[1, 2, 3, 6, 7, 8]
```

```
>>> a = [1,2,3]
>>> b = a * 3
>>> print(b)
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

파이썬

변수와 상수

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

- **리스트**는 인덱싱으로 데이터 삭제가 가능!
- 튜플은 데이터 수정이 불가능하기 때문에 삭제 불가능

```
>>> data = ['fast', 'campus', 0, 100]
>>> print(data)
['fast', 'campus', 0, 100]
>>> del(data[0])
>>> print(data)
['campus', 0, 100]
```

파이썬

변수와 상수

리스트 - 내장함수

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

파이썬

변수와 상수

리스트 - 내장함수

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

파이썬

변수와 상수

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

```
>>> a = (1, 123, 'fastcampus', 'alghost')
>>> a = ()
>>> a = (1, 123, ('alghost', 123), 'taehwa')
```

튜플 자료형 예제

튜플

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

파이썬

변수와 상수

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

딕셔너리 - 연산

- 을 지원하지 않는다 :)

파이썬

변수와 상수

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

파이썬

변수와 상수

인덱싱으로 삭제하기

- **딕셔너리**는 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}
```

파이썬

변수와 상수

딕셔너리 - 내장함수

함수명	설명	사용방법	result 값
keys	딕셔너리의 key들을 반환	a = { ‘a’ : 123, ‘b’ : 456 } result = a.keys()	[‘a’, ‘b’]
values	딕셔너리의 value들을 반환	a = {‘a’ : 123, ‘b’ : 456 } result = a.values()	[123, 456]
items	key,value 쌍을 반환	a = {‘a’ : 123, ‘b’ : 456} result = a.items()	[(‘a’, 123), (‘b’, 456)]
get	key에 대한 값을 반환 (값이 없는 경우 기본값 지정이 가능)	a = {‘a’ : 123, ‘b’ : 456} result = a.get(‘c’, 789)	789

파이썬

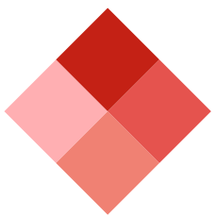
마무리

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