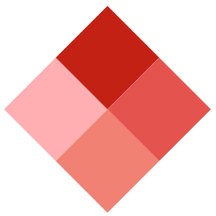


파이썬을 활용한 업무자동화

2회차: 프로그래밍 언어에 대한 이해

목차

2회차: 파이썬 필수 문법 (1)

- 복습
- 제어문
  - 조건문
    - if문이란?
  - 반복문
    - for문이란?
    - while문이란?
- 딕셔너리의 반복문 사용

복습

지난 강의 때 뭐했지..?

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’}
…	…	…

변수의 활용

- 자료형에 따른 연산: +, -, /, *, %
- 가장 중요한 **내장함수**
  - **외울 필요없고, 개발할 때 찾아보면 된다!**

제어문

드디어 내 프로그램이 '컴퓨터' 같이..

제어문이란??

- "무엇을 어떻게 해줘!" 에서 **어떻게**를 설명하기 위한 문
- 조건문과 반복문이 있음

조건문

- if문
- 특정 조건의 참, 거짓에 따라 작성한 코드가 동작할지 말지를 결정

반복문

- for문, while문
- 반복해서 문장을 수행해야 할 때 사용

조건문

if문

if문, 이럴 때 필요!

- 엑셀 필드 중 **결제 상태가 “결제 완료”인 사람**만 뽑아줘
- 접속한 **웹 사이트에 “패스트캠퍼스”가 있으면** URL을 저장해줘

문법

```
if 조건문:
    실행할 문장
elif 조건문:
    실행할 문장
else:
    실행할 문장
```

예시

```
payment_status = 'complete'

if payment_status == 'complete':
    print 'you already paid'
elif payment_status == 'inprogress':
    print 'you\'re paying'
else:
    print 'what are you doing?'
```

조건문

if문

들여쓰기

- if문을 설명하면서 실행할 문장을 들여썼는데, 매우 중요! => 영역 구분
- 공백은 tab, space 전부다 가능하지만 꼭 같은 공백을 써야함
  - tab을 썼다면 tab만, space 4칸을 썼다면 space 4칸만!

```
first  = 'something'
second = ''

if first:
    if second:
        print 'It is in second'
    print 'It is in first'
```

조건문

if문

들여쓰기

- if문을 설명하면서 실행할 문장을 들여썼는데, 매우 중요! => 영역 구분
- 공백은 tab, space 전부다 가능하지만 꼭 같은 공백을 써야함
  - tab을 썼다면 tab만, space 4칸을 썼다면 space 4칸만!

```
first  = 'something'
second = ''

if first:
    if second:
        print 'It is in second'
print 'It is in first'
```

조건문

if문

들여쓰기

- if문을 설명하면서 실행할 문장을 들여썼는데, 매우 중요! => 영역 구분
- 공백은 tab, space 전부다 가능하지만 꼭 같은 공백을 써야함

```
taehwaui-MacBook-Pro:~ Alghost$ python test.py
  File "test.py", line 7
    print 'It is in first'
                          ^
IndentationError: unindent does not match any outer indentation level
```

```
if first:
    if second:
        print 'It is in second'
print 'It is in first'
```

조건문

if문

조건문?

- 특정 조건을 표현하는 방법에 대해 알아보자
- 자료형 별 참,거짓이 있고!
- 비교연산자가 있다!

자료형 별 참 거짓

	참	거짓
숫자형	0이 아닌 수	0
문자열	빈 문자열이 아닌 문자열	""
리스트	빈 리스트가 아닌 리스트	[]
튜플	빈 튜플이 아닌 튜플	()
딕셔너리	빈 딕셔너리가 아닌 딕셔너리	{}

조건문

if문

자료형 별 참 거짓 예제 - 숫자형

```
money = 0

if money:
    print "Money!!!!"
else:
    print "No money.."

print '-'*20

money = 50000

if money:
    print "Money!!!!"
else:
    print "No money.."
```

```
taehwaui-MacBook-Pro:~ Alghost$ python test.py
No money..
--------------------
Money!!!!
```

조건문

if문

자료형 별 참 거짓 예제 - 문자열

```
name = ''

if name:
    print name
else:
    print "No name"

print '-'*20

name = 'alghost'

if name:
    print name
else:
    print "No name.."
```

```
taehwaui-MacBook-Pro:~ Alghost$ python test.py
No name
--------------------
alghost
```

조건문

if문

자료형 별 참 거짓 예제 - 리스트

```
students = []

if students:
    print students
else:
    print "No student"

print '-'*20

students = ['taehwa', 'yongseong']

if students:
    print students
else:
    print "No student.."
```

```
taehwaui-MacBook-Pro:~ Alghost$ python test.py
No student
--------------------
['taehwa', 'yongseong']
```

조건문

if문

자료형 별 참 거짓 예제 - 튜플

```
students = ()

if students:
    print students
else:
    print "No student"

print '-'*20

students = ('taehwa', 'yongseong')

if students:
    print students
else:
    print "No student.."
```

```
taehwaui-MacBook-Pro:~ Alghost$ python test.py
No student
--------------------
('taehwa', 'yongseong')
```

조건문

if문

자료형 별 참 거짓 예제 - 딕셔너리

```
info = {}

if info:
    print info
else:
    print "No info.."

print '-'*20

info = {'name' : 'taehwa'}

if info:
    print info
else:
    print "No info.."
```

```
taehwaui-MacBook-Pro:~ Alghost$ python test.py
No info..
--------------------
{'name': 'taehwa'}
```

조건문

if문

비교연산자

- 비교연산자는 조건문에 들어가는 연산자
- 우리는 이미 알고 있다! : ‘>’, ‘<’ 등..

비교연산자	설명
x < y	x가 y보다 작다
x > y	x가 y보다 크다
x == y	x와 y가 같다
x != y	x와 y가 다르다
x >= y	x가 y보다 크거나 같다
x <= y	x가 y보다 작거나 같다

조건문

if문

그 외 조건문에서 사용 가능한 연산자!

연산자	설명
x and y	x도 참이고 y도 참일 경우에 참
x or y	x와 y중 하나 이상이 참일 경우에 참
not x	x가 거짓일 경우에 참
x in 리스트	리스트에 x가 있을 경우 참
x in 튜플	튜플에 x가 있을 경우 참
x in 문자열	문자열에 x가 있을 경우 참
x not in 리스트	리스트에 x 가 없을 경우 참
x not in 튜플	튜플에 x가 없을 경우 참
x not in 문자열	문자열에 x가 없을 경우 참

조건문

if문

```
emails = ['taehwa@gmail.com', 'dlxoghk@gmail.com', 'yongseong']
desc = 'For developer'
students_count = 10
```

실습!!

- emails에 '@'가 포함되지 않은 경우 "Wrong" 출력
- desc에 developer라는 문자열이 포함되어 있을 경우

  이를 beginner로 변경하고 출력
- students_count가 5명 이상일 경우 이 값을 5로 변경하고 "Exceed" 출력

조건문

if문

실습!!

- emails에 ‘@‘가 포함되지 않은 경우 “Wrong” 출력
- desc에 developer라는 문자열이 포함되어 있을 경우

  이를 beginner로 변경하고 출력
- students_count가 5명 이상일 경우 이 값을 5로 변경하고 “Exceed” 출력

힌트

- 리스트의 인덱싱: emails[0], emails[1], emails[2]…
- 문자열 포함 확인: ‘@‘ in emails[0]
- 문자열 변경: desc = desc.replace(‘developer’, ‘beginner’)
- 비교연산자: ‘>’, ‘<‘, ‘<=‘, ‘>=‘ 등 생각을..!

조건문

if문

```
if '@' not in emails[0]:
    print 'Wrong'

if '@' not in emails[1]:
    print 'Wrong'

if '@' not in emails[2]:
    print 'Wrong'

if 'developer' in desc:
    desc = desc.replace('developer', 'beginner')
    print desc

if students_count >= 5:
    students_count = 5
    print 'Exceed'
```

반복문

for문

for문, 이럴 때 필요!

- 100명한테 이메일을 보내줘
- 뭐든 간에 반복이 필요한 부분에 사용!

문법

for 변수명 in 리스트,튜플,문자열:

    실행할 문장

    실행할 문장

예시

```
for looper in [1,2,3,4,5]:
    print looper
```

반복문

for문

range 함수를 쓰자

- for문과 매우 친한 함수이니 미리 배워보자
- 100번 반복을 위해 [1,2,3,4,…100]을 만들순 없다..
  - range가 만들어줌! => range(100): [0,1,2,3,4,…99]
    - 즉, range함수는 입력한 숫자에 맞는 리스트를 만들어 줌

예시

```
for looper in range(100):
    print looper
```

0~99까지 출력이 된다

반복문

for문

자, 이제 뭘 하고 싶죠??

- if문 예제를 개선하고 싶죠^^..
- 아래 예제는 당연히 for문으로 더 이쁘게! 변경이 가능

```
emails = ['taehwa@gmail.com', 'dlxoghk@gmail.com', 'yongseong']
```

```
if '@' not in emails[0]:
    print 'Wrong'

if '@' not in emails[1]:
    print 'Wrong'

if '@' not in emails[2]:
    print 'Wrong'
```

반복문

for문

자, 이제 뭘 하고 싶죠??

- if문 예제를 개선하고 싶죠^^..
- 아래 예제는 당연히 for문으로 더 이쁘게! 변경이 가능

```
emails = ['taehwa@gmail.com', 'dlxoghk@gmail.com', 'yongseong']
```

```
for _email in emails:
    if '@' not in _email:
        print 'Wrong'
```

반복문

while문

while문, 이럴 때 필요!

- 뭐든 간에 반복이 필요한 부분에 사용! => for문과 같은듯 다른듯..
- while문도 반복을 해주긴 하는데..
  - for문은 데이터의 수 만큼 반복해주고
  - while문은 **특정 조건**을 만족할 때까지 반복해준다

문법

while 조건문:

    실행할 문장

예시

```
user_input = ''
while user_input != 'quit':
    user_input = raw_input('Input: ')
    print user_input
```

사용자에게 계속 입력을 받고
그 값이 quit이 아니면 입력 받은 내용을 출력한다

반복문

while문

raw_input..??

- 사용자로부터 입력을 받는 함수
- 뒤에 문자열을 출력해주고 사용자의 입력을 기다림
- 엔터가 입력될때까지 입력받음

```
user_input = raw_input('Type your input: ')

print user_input
```

```
taehwaui-MacBook-Pro:~ Alghost$ python test.py
Type your input: taehwa qwer
taehwa qwer
```

반복문

while문

while문 활용 예시

- 학생 리스트에서 하나씩 꺼내면서 출력해줘!

```
students = ['taehwa', 'yongseong', 'john', 'fast', 'campus']

while students:
    print students.pop()
```

```
taehwaui-MacBook-Pro:~ Alghost$ python test.py
campus
fast
john
yongseong
taehwa
```

반복문

for, while 문

반복문에서만 쓰이는 '문'

- 반복을 하긴 하는데.. 갑자기 반복을 나가고 싶다면?
- 반복을 하긴 하는데.. 특정 경우만 스킵하고 싶다면?
- break와 continue가 있다

break

- 반복문에서 빠져 나오는 구문

```
for i in range(100):
    print i
    break
    print "!!!!"
```

continue

- 반복문에서 뒤 문장을 건너뛰는 구문

```
for i in range(100):
    print i
    continue
    print "!!!!"
```

제어문 활용

자료형과 함께 활용

딕셔너리 활용

- 딕셔너리를 효과적으로 사용해보자!
- 또 다시 찾아온 내장함수 활용

자료형	내장함수	활용
딕셔너리	keys	for key in dict.keys():
	values	for val in dict.values():
	items	for (key, val) in dict.items():
	has_key	if dict.has_key('name'):

제어문 활용

자료형과 함께 활용

딕셔너리 활용

```
person = {'name' : 'Taehwa', 'phone' : '01012345678'}

for (key,val) in person.items():
    print 'key is ' + key
    print 'value is ' + val
```

```
taehwaui-MacBook-Pro:example Alghost$ python test.py
key is phone
value is 01012345678
key is name
value is Taehwa
```

제어문 활용

자료형과 함께 활용

딕셔너리 활용

```
person = {'name' : 'Taehwa', 'phone' : '01012345678'}

for key in person.keys():
    print 'key is ' + key
    print 'value is ' + person[key]
```

```
taehwaui-MacBook-Pro:example Alghost$ python test.py
key is phone
value is 01012345678
key is name
value is Taehwa
```

제어문 활용

자료형과 함께 활용

딕셔너리 활용

```
person = {'name' : 'Taehwa', 'phone' : '01012345678'}

if not person.has_key('email'):
    print "It doesn't have email"

for key in person.keys():
    print 'key is ' + key
    print 'value is ' + person[key]
```

```
taehwaui-MacBook-Pro:example Alghost$ python test.py
It doesn't have email
key is phone
value is 01012345678
key is name
value is Taehwa
```

제어문 전체!

실습

실습!!

- 3명의 사원이 있고 이 정보는 딕셔너리의 리스트로 존재
- ceo인 경우를 제외하고 사원중에 나이가 30이상인 사람 수 세기

```
emp = []

emp.append({'name' : 'taehwa', 'age' : 30, 'position' : 'manager'})
emp.append({'name' : 'yongseong', 'age' : 21, 'position' : 'intern'})
emp.append({'name' : 'jungeun', 'age' : 32, 'position' : 'ceo'})
```

emp	name	age	position
emp[0]	taehwa	30	manager
emp[1]	yongseong	21	intern
emp[2]	jungeun	32	ceo

뭐가 필요할까?

제어문 전체!

실습

실습!!

```
emp = []

emp.append({'name' : 'taehwa', 'age' : 30, 'position' : 'manager'})
emp.append({'name' : 'yongseong', 'age' : 21, 'position' : 'intern'})
emp.append({'name' : 'jungeun', 'age' : 32, 'position' : 'ceo'})

person_count = 0
for person in emp:
    print person
    if person['position'] == 'ceo':
        print 'pass ceo'
        continue
    if person['age'] >= 30:
        print 'correct!'
        person_count = person_count + 1

print person_count
```

제어문 전체!

실습

실습!!

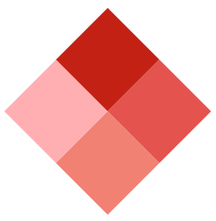
```
emp = []

emp.append({'name' : 'taehwa', 'age' : 30, 'position' : 'manager'})
em
em
```

```
taehwaui-MacBook-Pro:~ Alghost$ python test.py
{'position': 'manager', 'age': 30, 'name': 'taehwa'}
correct!
{'position': 'intern', 'age': 21, 'name': 'yongseong'}
{'position': 'ceo', 'age': 32, 'name': 'jungeun'}
pass ceo
1
```

```
pe
fo
    if person['age'] >= 30:
        print 'correct!'
        person_count = person_count + 1

print person_count
```



Good Bye

See you next time