

#딥베이직

20161115 김단비

1. 파이(Py) 맛보기

In [2]:

```
for countdown in 5, 4, 3, 2, 1, "hey!":
    print(countdown)
```

```
5
4
3
2
1
hey!
```

파이썬 프로그램에는 for, in, print, ,(콤마), :(콜론), ()(괄호) 등 몇 가지 특별한 단어와 기호가 있음
파이썬은 다른 언어보다 간결하고 깔끔한 구문을 가짐

다음은 리스트에서 텔레비전 뉴스 문구를 선택하여 출력하는 간단한 파이썬 프로그램임

In [1]:

```
cliches = [
    "At the end of the day",
    "Having said that",
    "The fact of the matter is",
    "Be that ad it may",
    "The bottom line is",
    "If you will",
]
print(cliches[3])
```

```
Be that ad it may
```

clichés와 같은 파이썬 리스트에는 연속적인 값이 들어있음. 리스트의 맨 처음부터 오프셋(offset)으로 접근할 수 있음.
첫 번째 값은 오프셋 0, 4번째 값은 오프셋 3

사람들은 대부분 숫자를 1부터 세서 0부터 세는 것이 이상해보일 수 있지만 위치 대신 오프셋 관점에서 생각하자

다음은 인용문을 출력하는 프로그램임

In [3]:

```
quotes = {
    "Moe": "A wise guy, huh?",
    "Larry": "Ow!",
    "Curly": "Nyuk nyuk!",
}
stooge = "Curly"
print(stooge, "says:", quotes[stooge])
```

Curly says: Nyuk nyuk!

quotes는 파이썬 딕셔너리(dictionary). 딕셔너리는 유일한 키의 모임(stooge의 이름)과 키에 해당하는 값(stooge의 이름에 해당하는 인용문)으로 이루어짐.

cliche 예제에서는 리스트를 만들기 위해 대괄호([])를 사용, stooge 예제에서는 딕셔너리를 만들기 위해 중괄호({})를 사용함

파이썬 프로그램을 작성하여 JSON 텍스트를 자료구조에 담을 수 있음.

웹사이트에서 응답하는 데이터 중 처음 6개의 동영상 제목을 출력해보자.

In []:

```
import json
from urllib.request import urlopen
url = "https://gdata.youtube.com/feeds/api/standardfeeds/top_rated?alt=json" response
contents = response.read()
text = contents.decode('utf8')
data = json.loads(text)
for video in data['feed']['entry'][0:6]:
    print(video['title']['$t'])
```

라인 1 : 파이썬 표준 라이브러리인 json의 모든 코드를 가져온다

라인 2 : 표준 urllib 라이브러리에서 urlopen 함수만 가져온다

라인 3 : 웹사이트 주소를 url 변수에 할당한다

라인 4 : URL에 해당하는 웹 서버에 연결하고 특정 웹 서비스를 요청한다

라인 5 : 요청한 데이터를 얻어서 contents 변수에 할당한다

라인 6 : 데이터를 JSON 형식으로 된 텍스트 문자열로 디코딩하고 text 변수에 할당한다

라인 7 : 동영상에 대한 자료가 담긴 text 변수를 파이썬의 자료구조로 변환해서 data 변수에 담는다

라인 8 : 한 번에 한 동영상에 대한 정보를 얻어 와서 이를 video 변수에 담는다

라인 8 : 2차원 딕셔너리(data['feed']['entry'])와 슬라이스([0:6])를 사용한다

라인 9 : 동영상의 제목을 print 함수로 출력한다

다음 코드는 requests라는 파이썬 외부 패키지를 사용해서 다시 작성한 것

In []:

```
import requests
url = "https://gdata.youtube.com/feeds/api/standardfeeds/top_rated?alt=json" response
data = response.json()
for video in data['feed']['entry'][0:6]:
    print(video['title']['$t'])
```

1.1 파이썬 활용

빠르고 쉽게 개발할 수 있어 생산성이 뛰어난 파이썬은 다음과 같이 많은 컴퓨팅 환경에서 활용 가능함

- 터미널 창의 커맨드 라인

- 웹을 포함한 GUI(Graphical User Interface)
- 서버/클라이언트 웹
- 대용량 데이터 처리를 지원하는 백엔드 서버
- 클라우드(써드파티에 의해 관리되는 서버)
- 모바일 디바이스
- 임베디드 디바이스

1.2 파이썬과 다른 언어

쉘 스크립트, C, C++, 자바, C#, Ruby, PHP 등 다른 언어와 비교하면 파이썬이 제일 간결하고 쉽다!

1.3 왜 파이썬인가?

파이썬은 범용의 고수준 언어라서 코드 읽기 아주 쉬움. 작성하기도 쉬움. 코드를 적게 작성하는 동적언어라서 생산성도 높음. 자유롭게 사용 가능.

1.4 파이썬 쓰면 안 될 때

프로그램이 계산 작업을 많이 하는 경우에는 빠르지 않을 수 있음(CPU 바운드, 수행하는 계산 작업은 CPU의 속도에 의해 결정됨)

1.5 파이썬 2와 파이썬 3

고치기 어려운 것들을 개선하기 위해 만든 것이 파이썬 3

파이썬 3가 파이썬 2와 다른 점은 `print`문의 호출방법과 유니코드 문자 처리

1.6 파이썬 설치

는 부록D 참조

1.7 파이썬 실행

1. 파이썬의 대화식 인터프리터는 작은 프로그램을 테스트하기 쉬운 환경 제공, 커맨드 라인에 코드를 입력하고 결과를 바로 볼 수 있음
2. 파이썬 프로그램을 텍스트 파일로 저장하여 실행, .py 확장자 사용, 커맨드 창에서 'python 파일 이름' 입력하여 실행

1.7.1 대화식 인터프리터 사용하기

커맨드 창에서 `python`, `python3`, 혹은 파이썬 이름을 입력해 인터프리터 실행

61 입력

`print(61)`

1.7.2 파이썬 파일 사용하기

파일에 61 입력하여 파일 실행하면 실행은 되지만 아무 것도 출력 안 함. 보통의 비대화식 파이썬 프로그램에서는 `print`함수 호출해서 출력

`print(61)`

파이썬 프로그램 파일 만들기

1. 텍스트편집기를 연다
2. print(61) 입력
3. 이 파일을 일반 텍스트 형태에서 61.py로 저장
4. GUI를 사용한다면(거의 대부분의 경우) 터미널 창을 연다
5. 다음과 같이 입력해 프로그램 실행

```
python 61.py
```

1.7.3 그 다음에는

파이썬 프로그램을 개발하는 기본적인 방법은 텍스트 편집기와 터미널 창을 사용하는 것.

파이썬을 위한 많은 통합 개발 환경(Integrated Development Environment, IDE)이 있음

1.8 파이썬 철학

```
import this
```

2. 파이 재료: 숫자, 문자열, 변수

파이썬의 데이터 타입

부울 : True 혹은 False

정수 : 42, 100000000과 같은 숫자

실수 : 3.14159 처럼 소수점이 있는 숫자, 1.0e8과 같은 지수

문자열 : 텍스트 문자들의 시퀀스

이들은 원자(atom)와 비슷, 3장에서는 이들을 어떻게 분자(molecule)로 결합하는지 볼 것

2.1 변수, 이름, 객체

파이썬에선 모든 것(부울, 정수, 실수, 문자열, 데이터 구조, 함수, 프로그램)이 객체(object)로 구현됨

객체는 데이터가 담긴 투명한 플라스틱 박스와 같음. 데이터와 함께 무엇을 처리할 수 있는지 결정하는 부울 혹은 정수 타입



타입은 박스에 포함된 데이터 값을 변경할 수 있는 변수(가변, mutable)인지 변경할 수 없는 상수(불변, immutable)인지 판단 가능

투명한 유리창으로 밀봉된 박스 안에 있는 사물(객체) 생각해보면 값을 볼 수는 있지만 변경할 수는 없음

가변 객체는 열려있는 상자와 같음. 값을 볼 수 있고 바꿀 수도 있음. 타입은 변경 불가.

파이썬은 객체의 타입을 바꿀 수 없는 강타입, 그 값이 가변인 경우에도 타입 바꿀 수 없음

변수는 컴퓨터 메모리에 있는 값을 참조하기 위한 이름. 파이썬은 변수에 값을 할당하기 위해 "=" 사용

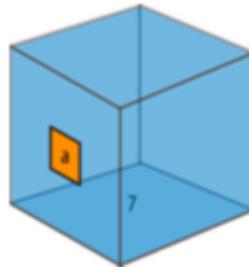
In [9]:

```
a = 7
print(a)
```

변수는 단지 이름(name), 이름을 포스트잇처럼 생각하자

할당한다는 의미는 값을 복사하는 것이 아님. 데이터가 담긴 객체에 이름을 붙이는 것!

객체 자신에 포함되는 것이 아니라 객체의 참조!



7이라는 객체에 a라는 포스트잇 붙은 것

In [11]:

```
b = a
print(b) #7이라는 객체에 b라는 포스트잇 하나 더 붙은 것
```

7

이제 변수 혹은 리터럴값의 타입을 알아보기 위해 type(변수 or 리터럴값)을 사용해보자.

In [18]:

```
type(a)
```

Out[18]:

int

In [19]:

```
type(b)
```

Out[19]:

int

In [20]:

```
type(58)
```

Out[20]:

int

In [21]:

```
type(99.9)
```

Out[21]:

float

In [22]:

type('abc')

Out[22]:

str

클래스는 객체의 정의를 의미함, 파이썬에선 **class**와 **type**은 의미가 거의 같음

변수 이름에는 다음 문자만 사용 가능

- 소문자(a ~ z)
- 대문자(A ~ Z)
- 숫자(0 ~ 9)
- 언더스코어(_)

이름은 숫자로 시작할 수 없음. 파이썬은 언더스코어로 시작하는 이름을 특별한 방법으로 처리함

아래에 있는 이름은 변수로 선언하면 안 됨, 파이썬 구문 정의하는 데 사용됨

파이썬의 예약어

False class finally is return
 None continue for lambda try
 True def from nonlocal while
 and del global not with
 as elif if or yield
 assert else import pass
 break except in raise

2.2 숫자

파이썬은 정수와 부동소수점수 지원함. 간단한 수학 연산자로 결합해서 계산 가능

Operator	Description	Example	Result
+	addition	5 + 8	13
-	subtraction	90 - 10	80
*	multiplication	4 * 7	28
/	floating point division	7 / 2	3.5
//	integer (truncating) division	7 // 2	3
%	modulus (remainder)	7 % 3	1
**	exponentiation	3 ** 4	81

2.2.1 정수

대화식 인터프리터에서 연속된 숫자는 리터럴 정수로 간주함, 숫자 0을 쓸 수는 있지만 0을 다른 숫자 앞에 넣을 순 없음

In [23]:

```
05
```

```
File "<ipython-input-23-b139c77ba8b1>", line 1
 05
 ^
SyntaxError: invalid token
```

숫자 앞에 기호가 없으면 양수를 의미, + 기호 붙여도 양수

숫자 앞에 - 기호 붙이면 음수
연산자 사용해서 계산 가능

In [24]:

```
-123
```

Out[24]:

```
-123
```

In [25]:

```
4 -10
```

Out[25]:

```
-6
```

In [26]:

```
5 + 9          +          3
```

Out[26]:

```
17
```

In [27]:

```
6 * 7 * 2 * 3
```

Out[27]:

```
252
```

나눗셈은 조금 재미진데 두 가지 방법 있음

- /는 부동소수점을 포함한 결과가 출력
- //는 소수점 이하 버리고 정수만 출력

In [28]:

9/5

Out[28]:

1.8

In [29]:

9//5

Out[29]:

1

이제까지 리터럴 정수만 사용했는데, 정수값이 할당된 변수와 리터럴 정수를 혼합해서 사용해보자

In [30]:

```
a = 95  
a - 3
```

Out[30]:

92

In [31]:

a

Out[31]:

95

a - 3의 결과값을 a에 할당하지 않았으니 a 값은 변하지 않음. a값을 바꾸고 싶다면?

In [32]:

```
a = a - 3  
a
```

Out[32]:

92

파이썬의 표현식에서는 "="의 오른쪽을 먼저 계산함! 그리고나서 왼쪽의 변수에 계산된 값 할당!

즉, "="의 오른쪽에서 빼기를 계산하고 그 결과를 기억해서 왼쪽에 있는 a에 할당!

위의 식은 산술 연산자를 결합해서 할당 가능!

In [33]:

```
a = 95  
a -= 3  
a
```

Out[33]:

92

다음은 $a = a + 8$ 과 같음

In [34]:

```
a += 8  
a
```

Out[34]:

100

다음은 $a = a * 2$ 과 같음

In [35]:

```
a *= 2  
a
```

Out[35]:

200

다음은 $a = a / 3$ 과 같음

In [36]:

```
a /= 3  
a
```

Out[36]:

66.66666666666667

a에 13을 할당한 후 축약된 $a = a // 4$ (소수점 이하 버림) 계산해보자

In [37]:

```
a = 13
a /= 4
a
```

Out[37]:

3

In [38]:

```
9 % 5 #나머지 계산
```

Out[38]:

4

In [39]:

```
divmod(9,5) #몫과 나머지 동시에 얻는 방법
```

Out[39]:

(1, 4)

divmod함수에서 인자로 정수 9와 5를 넣으면 두 항목의 결과인 튜플(tuple)이 나옴, 튜플은 3장에서 설명!

2.2.2 우선순위

파이썬에서 곱셈은 덧셈보다 높은 우선순위를 가지는데 우선순위 규칙 외우지 말고 그냥 먼저 수행하고자 하는 계산식에 괄호를 붙이세요

2.2.3 진수

정수 앞에 진수 안 붙이면 10진수로 간주함

진수는 1을 올림할 때까지 숫자를 어디까지 셀 수 있는지 나타냄

파이썬에서는 10진수 외에 세 가지 리터럴 정수 표현 가능

- 2진수(binary): 0b, 0B
- 8진수(octal): 0o, 0O
- 16진수(hex): 0x, 0X

인터프리터는 이 리터럴 정수들을 10진수로 출력해줌

8진수 10을 10진수로 출력해보자. (8곱하기1) + (1곱하기0)

In [40]:

```
0o10
```

Out[40]:

8

2.2.4 형변환

다른 데이터 타입을 정수형으로 변환하려면 `int()` 함수 사용

`int()`는 소수점을 버리고 정수를 반환함
 부울형, 부동소수점수, +,- 기호와 숫자로 이루어진 텍스트문자열, 정수 등은 변환 가능하나

 숫자가 아닌 다른 뭔가, 소수점, 지수를 포함하는 문자열은 처리하지 않음

In [43]:

```
int('99 bottles of beer on the wall')
```

```
-----
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-43-fd87007a62a6> in <module>()
----> 1 int('99 bottles of beer on the wall')
```

`ValueError: invalid literal for int() with base 10: '99 bottles of bee
r on the wall'`

In [41]:

```
int('98.6')
```

```
-----
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-41-c3b886411bc9> in <module>()
----> 1 int('98.6')
```

`ValueError: invalid literal for int() with base 10: '98.6'`

In [42]:

```
int('1.0e4')
```

```
-----
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-42-c756a59bf086> in <module>()
----> 1 int('1.0e4')
```

`ValueError: invalid literal for int() with base 10: '1.0e4'`

숫자의 타입을 섞어서 사용하면 파이썬은 자동으로 형변환함

In [44]:

```
4 + 7.0
```

Out[44]:

11.0

In [45]:

`True + 2`

Out[45]:

3

2.2.5 int의 크기

파이썬 3에서는 int의 크기가 유연해짐. 이제 10^{**100} (구꼴, googol)도 담을 수 있다. 파이썬은 문제 없이 아주 큰 정수도 처리함

2.2.6 부동소수점수

부동소수점수(float)는 소수점이 있음. 부동소수점수는 $+, -, /, //, *, \%$ 연산과 `divmod()` 함수 사용 가능해서 정수와 유사하게 처리됨

부동소수점수로 형변환하려면 `float()` 함수 사용

In [46]:

`float(False)`

Out[46]:

0.0

In [47]:

`float('99')`

Out[47]:

99.0

`int()`에서 안 되던 것들이 `float()`에서는 되네요!

In [49]:

`float('98.6')`

Out[49]:

98.6

In [48]:

`float('1.0e4')`

Out[48]:

10000.0

2.2.7 수학 함수

파이썬에는 제곱근, 코사인 등 수학 함수가 있는데 부록 C '파이 사이언스'에서 참고하세요

```
import math
```

<http://docs.python.org/3/library/math.html>

2.3 문자열

파이썬에서 문자열은 불변(immutable), 문자열 자체는 변경할 수 없으므로 문자열의 일부를 다른 문자열로 복사 가능

2.3.1 인용 부호로 문자열 생성

- 'Deepbasic'
- "Deepbasic"
- '''Deepbasic'''
- """Deepbasic""

따옴표, 쌍따옴표 1번 혹은 3번을 찍어 문자열 생성 가능

인용 부호가 두 가지 종류인 이유는 인용 부호가 포함된 문자열을 만들기 위해서임

일반적으로 3번 찍는 따옴표는 여러 줄의 문자열에 사용

In [51]:

```
poem2 = '''I do not like thee, Doctor Fell.  
...     The reason why, I cannot tell.  
...     But this I know, and know full well:  
...         I do not like thee, Doctor Fell.  
...'''
```

In [52]:

```
print(poem2)
```

```
I do not like thee, Doctor Fell.  
...     The reason why, I cannot tell.  
...     But this I know, and know full well:  
...         I do not like thee, Doctor Fell.  
...
```

In [53]:

```
poem2
```

Out[53]:

```
'I do not like thee, Doctor Fell.\n...     The reason why, I canno  
t tell.\n...     But this I know, and know full well:\n...  
I do not like thee, Doctor Fell.\n... '
```

print()의 출력 결과와 대화식 인터프리터의 결과가 다르다!

print()는 문자열에서 인용 부호를 제거한 뒤 내용을 출력

대화식 인터프리터는 단일 인용 부호 또는 \n과 같은 이스케이프 문자가 들어 있는 문자열도 출력

2.3.2 데이터 타입 변환: str()

str() 함수 사용해서 데이터 타입을 문자열로 변환 가능

문자열이 아닌 객체를 print()로 호출할 때 파이썬은 내부적으로 str()함수를 사용함

변수나 상수를 문자열 안에 삽입하는 문자열 보간(string interpolation)을 할 때도 내부적으로 사용

2.3.3 이스케이프 문자

문자열 안의 일부 문자의 의미를 다르게 해석해서 특정 효과 줄 수 있음

백슬래시() 기호를 붙여서 특별한 의미를 줌

가장 일반적인 이스케이프 시퀀스는 줄바꿈을 의미하는 \n

In [50]:

```
palindrome = 'A man,\nA plan,\nA canal:\nPanama.'
print(palindrome)
```

A man,
A plan,
A canal:
Panama.

탭이 들어있는 \t
마지막 문자열의 끝에는 착한 사람만 볼 수 있는 탭 문자가 들어있어요.

In [54]:

```
print('\tabc')
```

abc

In [55]:

```
print('a\tbc')
```

a bc

In [56]:

```
print('ab\tc')
```

ab c

In [57]:

```
print('abc\t')
```

abc

문자열에서 \' 혹은 \"을 사용하여 단일, 이중인용부호를 표현 가능

In [58]:

```
testimony = "\"I did nothing!\" he said. \"Not that either! Or the other thing.\""
print(testimony)
```

"I did nothing!" he said. "Not that either! Or the other thing."

백슬래시 입력하고 싶다면 백슬래시 두 번 입력하면 됨

In [59]:

```
speech = 'Today we honor our friend, the backslash: \\.'
print(speech)
```

Today we honor our friend, the backslash: \.

2.3.4 결합 : +

'+' 연산자 사용해서 리터럴 문자열 또는 문자열 변수 결합 가능

In [60]:

'Release the kraken! ' + 'At once! '

Out[60]:

'Release the kraken! At once! '

In [61]:

"My word! " "A gentleman caller!"

Out[61]:

'My word! A gentleman caller! '

파이썬은 문자열을 결합할 때 공백을 자동으로 붙이지 않아서 명시적으로 공백을 넣음.

`print()`는 각 인자 사이에 공백을 붙임

In [62]:

```
a = 'Duck.'
b = a
c = 'Grey Duck!'
a + b + c
```

Out[62]:

'Duck.Duck.Grey Duck! '

In [63]:

`print(a,b,c)`

Duck. Duck. Grey Duck!

2.3.5 복제하기:*

* 연산자를 이용하여 문자열을 복제할 수 있음

In [64]:

```
start = 'Na ' * 4 + '\n'
middle = 'Hey ' * 3 + '\n'
end = 'Goodbye.'
print(start + start + middle + end)
```

Na Na Na Na
Na Na Na Na
Hey Hey Hey
Goodbye.

2.3.6 문자 추출: []

문자열에서 한 문자를 얻기 위해서는 문자열 이름 뒤에 대괄호([])와 오프셋 지정

가장 왼쪽의 오프셋은 0이고 1, 2... 가장 오른쪽의 오프셋은 -1 (문자열의 수를 셀 필요가 없음)

In [65]:

```
letters = 'abcdefghijklmnopqrstuvwxyz'  
letters[0]
```

Out[65]:

```
'a'
```

In [66]:

```
letters[-1]
```

Out[66]:

```
'z'
```

In [67]:

```
letters[100]
```

```
-----  
----  
IndexError Traceback (most recent call  
last)  
<ipython-input-67-4e3c74f827ab> in <module>()  
----> 1 letters[100]
```

```
IndexError: string index out of range
```

문자열의 길이가 100개가 안 되어서 오류남

문자열은 불변하므로 특정 인덱스에 문자를 삽입하거나 변경할 수 없음

In [69]:

```
name = 'Henny'  
name[0] = 'p'
```

```
-----  
----  
TypeError Traceback (most recent call  
last)  
<ipython-input-69-7ac512f74198> in <module>()  
      1 name = 'Henny'  
----> 2 name[0] = 'p'
```

```
TypeError: 'str' object does not support item assignment
```

대신 replace()와 슬라이스와 같은 문자열 함수로 사용 가능

In [68]:

```
name = 'Henny'
name.replace('H', 'P')
```

Out[68]:

'Penny'

In [70]:

```
'p' + name[1:]
```

Out[70]:

'penny'

2.3.7 슬라이스: start:end:step]

슬라이스 사용해서 한 문자열에서 문자열의 일부를 추출 가능

대괄호를 사용해 시작(start), 오프셋(offset), 옵션으로 스텝(step) 표현으로 슬라이스 정의 가능

- [:] 처음부터 끝까지 전체 시퀀스를 추출한다
- [start:] start 오프셋부터 끝까지 시퀀스를 추출한다
- [:end] 처음부터 (end - 1) 오프셋까지 시퀀스를 추출한다
- [start:end] start 오프셋부터 (end - 1) 오프셋까지 시퀀스를 추출한다
- [start:end:step] step만큼 문자를 건너뛰면서, start 오프셋부터 (end - 1) 오프셋까지 시퀀스를 추출한다

In [71]:

```
letters[:]
```

Out[71]:

'abcdefghijklmnopqrstuvwxyz'

In [72]:

```
letters[10:]
```

Out[72]:

'klmnopqrstuvwxyz'

오프셋 12부터 14까지 추출하기

파이썬은 마지막 오프셋은 포함하지 않는다.

그러므로 14가 아닌 15 지정해야 함

In [73]:

```
letters[12:15]
```

Out[73]:

```
'mno'
```

마지막 세 문자를 추출해보자

In [74]:

```
letters[-3:]
```

Out[74]:

```
'xyz'
```

In []:

```
letters[18:-3]
```

In []:

```
letters[-6:-2]
```

1보다 큰 스텝을 원하신다고요? 두번째 콜론 이후에 숫자 명시하세요!

In [75]:

```
letters[::-7]
```

Out[75]:

```
'ahov'
```

4번째부터 19번째까지 3스텝씩 건너뛰면서 문자 추출

In [76]:

```
letters[4:20:3]
```

Out[76]:

```
'ehknqt'
```

In []:

```
letters[19::4]
```

In []:

```
letters[:21:5]
```

다시 한 번 말하지만 끝 오프셋은 실제 오프셋 + 1

끝에서 시작 지점까지 빠짐없이 문자를 추출하기

In [77]:

letters[::-1]

Out[77]:

'zyxwvutsrqponmlkjihgfedcba'

2.3.8 문자열 길이: len()

len() 함수는 문자열의 길이 나타냄

2.3.9 문자열 나누기: split()

문자열 함수는 string.function(arguments) 형태로 입력

문자열 이름(string) 입력하고 짹(.) 찍고 함수 이름(function) 입력하고 함수에 필요한 인자(arguments)도 입력

구분자를 기준으로 문자열을 작은 문자열 리스트로 나누려면 내장함수 split() 사용

In [78]:

todos = 'get gloves,get mask,give cat vitamins,call ambulance'
todos.split(',')

Out[78]:

['get gloves', 'get mask', 'give cat vitamins', 'call ambulance']

구분자를 ','로 지정하고 split()함수를 사용했음.

구분자 지정 따로 안 하면 공백문자(줄바꿈, 스페이스, 탭)를 사용

In [79]:

todos.split()

Out[79]:

['get', 'gloves,get', 'mask,give', 'cat', 'vitamins,call', 'ambulance']

2.3.10 문자열로 결합하기: join()

join() 함수는 split() 함수와 반대!!

join() 함수는 문자열 리스트를 하나의 문자열로 결합!

string.join(list) 형태

예제를 보고 이해해보자

In [80]:

crypto_list = ['Yeti', 'Bigfoot', 'Loch Ness Monster']
crypto_string = ', '.join(crypto_list)
print('Found and signing book deals:', crypto_string)

Found and signing book deals: Yeti, Bigfoot, Loch Ness Monster

리스트를 콤마와 스페이스로 구분하여 하나의 문자열로 결합한 예제

2.3.11 문자열 다루기

In [81]:

```
poem = '''All that doth flow we cannot liquid name  
Or else would fire and water be the same;  
But that is liquid which is moist and wet  
Fire that property can never get.  
Then 'tis not cold that doth the fire put out  
But 'tis the wet that makes it die, no doubt.''''
```

맨 처음 13자 출력하면?

In [82]:

```
poem[:13]
```

Out[82]:

```
'All that doth'
```

스페이스와 줄바꿈 포함해서 이 시는 몇 글자로 되어 있을까?

In [83]:

```
len(poem)
```

Out[83]:

```
270
```

이 시는 All로 시작하는가?

In [84]:

```
poem.startswith('All')
```

Out[84]:

```
True
```

이 시는 That's all, folks!로 끝나는가?

In [85]:

```
poem.endswith('That\'s all, folks!')
```

Out[85]:

```
False
```

이 시에서 첫 번째로 the가 나오는 오프셋은?

In [86]:

```
word = 'the'  
poem.find(word)
```

Out[86]:

77

마지막으로 the가 나오는 오프셋은?

In [90]:

```
poem.rfind(word)
```

Out[90]:

234

세 글자 the가 몇 번 나오는가?

In [91]:

```
poem.count(word)
```

Out[91]:

3

이 시는 글자와 숫자로만 이루어져 있는가?

In [89]:

```
poem.isalnum()
```

Out[89]:

False

시에 구두점 문자도 포함되어 있어서 False!

2.3.12 대소문자와 배치

문자열은 불변하므로 다음의 어느 예제에서도 setup 문자열을 바꿀 수 없음
단지 값을 설정하고 함수를 수행한 뒤 새로운 문자열로 결과를 반환함

첫 번째 단어를 대문자로 만들어보자

In [93]:

```
setup = 'a duck goes into a bar...'
```

In [94]:

```
setup.capitalize()
```

Out[94]:

```
'A duck goes into a bar...'
```

모든 단어의 첫 글자를 대문자로 만들어보자

In [95]:

```
setup.title()
```

Out[95]:

```
'A Duck Goes Into A Bar...'
```

글자를 모두 대문자로, 소문자로 만들어보자

In [96]:

```
setup.upper()
```

Out[96]:

```
'A DUCK GOES INTO A BAR...'
```

In [97]:

```
setup.lower()
```

Out[97]:

```
'a duck goes into a bar...'
```

대문자는 소문자로, 소문자는 대문자로 만들어보자

In [98]:

```
setup.swapcase()
```

Out[98]:

```
'A DUCK GOES INTO A BAR...'
```

문자열을 정렬(alignment)하는 함수도 보자. 문자열을 지정한 공간(여기선 30)에서 정렬해보자

In [99]:

```
setup.center(30)
```

Out[99]:

```
' a duck goes into a bar... '
```

In [100]:

setup.ljust(30)

Out[100]:

'a duck goes into a bar... '

In [101]:

setup.rjust(30)

Out[101]:

' a duck goes into a bar... '

2.3.13 대체하기: replace()

문자열의 일부를 대체하기 위해서 replace() 사용

인자로 바꿀 문자열, 대체할 새 문자열, 바꿀 문자열에 대한 횟수 입력

마지막 인자를 생략하면 첫 번째 인스턴스만 바꿈

In [102]:

setup.replace('duck', 'marmoset')

Out[102]:

'a marmoset goes into a bar... '

In [103]:

setup.replace('a ', 'a famous ', 100)

Out[103]:

'a famous duck goes into a famous bar... '

대체하고 싶은 문자열이 전체 단어인지, 한 단어의 시작의 일부인지 등 특수한 조건이 있다면 7장에서 배울 정규표현식을 사용하세요

2.3.14 문자열 함수

문자열 함수에 대한 자세한 사항은 표준 문서 웹사이트 [\(http://docs.python.org/3/library/stdtypes.html#string-methods\)](http://docs.python.org/3/library/stdtypes.html#string-methods) 참고하세요

2.4 연습문제

2.1 1시간은 몇 초인가? 대화식 인터프리터를 사용해서 계산해보라. 1시간은 60분, 1분은 60초다. 이 둘을 곱한다.

2.2 계산한 결과를 seconds_per_hour 변수에 저장하라.

2.3 1일은 몇 초인가? seconds_per_hour 변수를 사용하라.

2.4 계산한 결과를 seconds_per_day 변수에 저장하라.

2.5 부동소수점(/) 나눗셈을 사용해서 seconds_per_day를 seconds_per_hour로 나누어라

2.6 정수(/) 나눗셈을 사용해서 seconds_per_day를 seconds_per_hour로 나누어라. 이전 문제의 부동소수점 결과에서 본 .00이 있는가?