

1. 파이썬의 기본요소
2. 파이썬의 자료구조
3. 파이썬의 제어문
4. 함수와 모듈
5. 과학용 라이브러리
6. 시각화
7. 클래스

## 4. 파이썬 함수와 모듈



*Yoonjoong Kim*

*Department of Computer Engineering, Hanbat National University*

*yjkim@hanbat.ac.kr*

## 4. 파이썬 함수와 모듈 내용

---

### 4.1 함수

- 함수의 구문
- 반환방법
- 파라미터, 기본값

### 4.2 모듈

- 모듈 만들기
  - 모듈의 생성 (.py)
  - 모듈의 사용
- time 모듈
- os 모듈

### 4.3 내장함수

abs, chr, ord, enumerate  
len, list, min, max, int, float,  
format

### 4.4 연습문제



## 4.1 함수

---

- “I love you.” 는 문장을 화면에 출력하여 보자

```
>>> print('I love you.')
I love you.
```

- “I love you.” 문장을 3번 화면에 출력해 보자

```
>>> print('I love you. ');print('I love you. ');print('I love you.')
I love you.
I love you.
I love you.
```

- “I love you.” 문장을 3번 화면에 출력하고, 4번 출력하는 일이 필요하다면 함수를 정의하여 효율적으로 코딩할 수 있다. => 반복되는 처리과정을 함수로 만들어 간결하게 코딩할 수 있다.

```
>>> def print_ntimes(n):
...     for i in range(n):
...         print('I Love you.')
...
>>> print_ntimes(3) #함수호출
I love you.
I love you.
I love you.
>>> print_ntimes(4)
I love you.
I love you.
I love you.
I love you.
```

## 4.1 함수(cont.)

### •1.1 함수의 구문

- 함수이름은 식별자의 규칙으로 만들어지는 이름이다.
- 매개변수(파라미터)는 함수의 처리에 필요한 자료이다.
- 함수body는 함수처리를 수행하는 프로그램의 블록이다.

### •1.2 반환 방법

- 값이 하나인 함수
  - 주식의 상한가가 이율이 15%라고하면, 주식의 상한가가는 다음과 같이 계산된다. 주식 가격이 10000이라면 상한가는  $10000+10000*0.15=11500$ 이 된다. 주가 10000,45000,81100의 상한가를 계산하자.함수가 필요하다.
- 반환 값이 두개 이상인 함수
  - 주식의 상한가가 이율이 15%라고하면, 주식의 상한가와 하한가는 다음과 같이 계산된다. 주식 가격이 10000이라면 상한가는  $10000+10000*0.15=11500$ 이고 하한가는  $10000-10000*0.15=8500$ 이다. 주가 10000,45000,81100의 상한가를 계산하자.함수가 필요하다.
- 반환 값이 두개 이상이면 리스트, 튜플형으로 만들어 반환한다.

```
>>> def sum(a,b):  
...     c=a+b  
...     return c  
...  
>>>
```

함수이름

매개변수(파라미터)

함수body  
처리내용  
결과반환

```
>>> def price_upper(price):  
...     _price = price+price*0.15  
...     return _price  
...  
>>> price_upper(10000)  
11500.0  
>>> price_upper(45000)  
51750.0
```

```
>>> def price_uplo(price):  
...     rate=0.15  
...     price_up = price+price*rate  
...     price_lo = price-price*rate  
...     return price_up, price_lo  
...  
>>> price_uplo(10000)  
(11500.0 , 8500.0)  
>>> price_uplo(45000)  
(51750.0 , 38250.0)
```

## 4.1 함수(cont.)

### • 1.3 파라미터의 기본값

- 함수의 매개변수에 기본값(default value)을 지정할 수 있다.
- 기본값이 지정된 매개변수는 함수 호출할 때 조건부 변수이다

```
>>> def price_uplo(price,rate=0.15):
...     price_up = price+price*rate
...     price_lo = price-price*rate
...     return price_up, price_lo
...
>>> price_uplo(10000)
(11500.0 , 8500.0)
>>> price_uplo(20000)
(23000.0 , 17000.0)
>>> price_uplo(10000,0.30)
(13000.0 , 7000.0)
>>> >>> price_uplo(rate=0.30,price=10000)
(13000.0 , 7000.0)
```

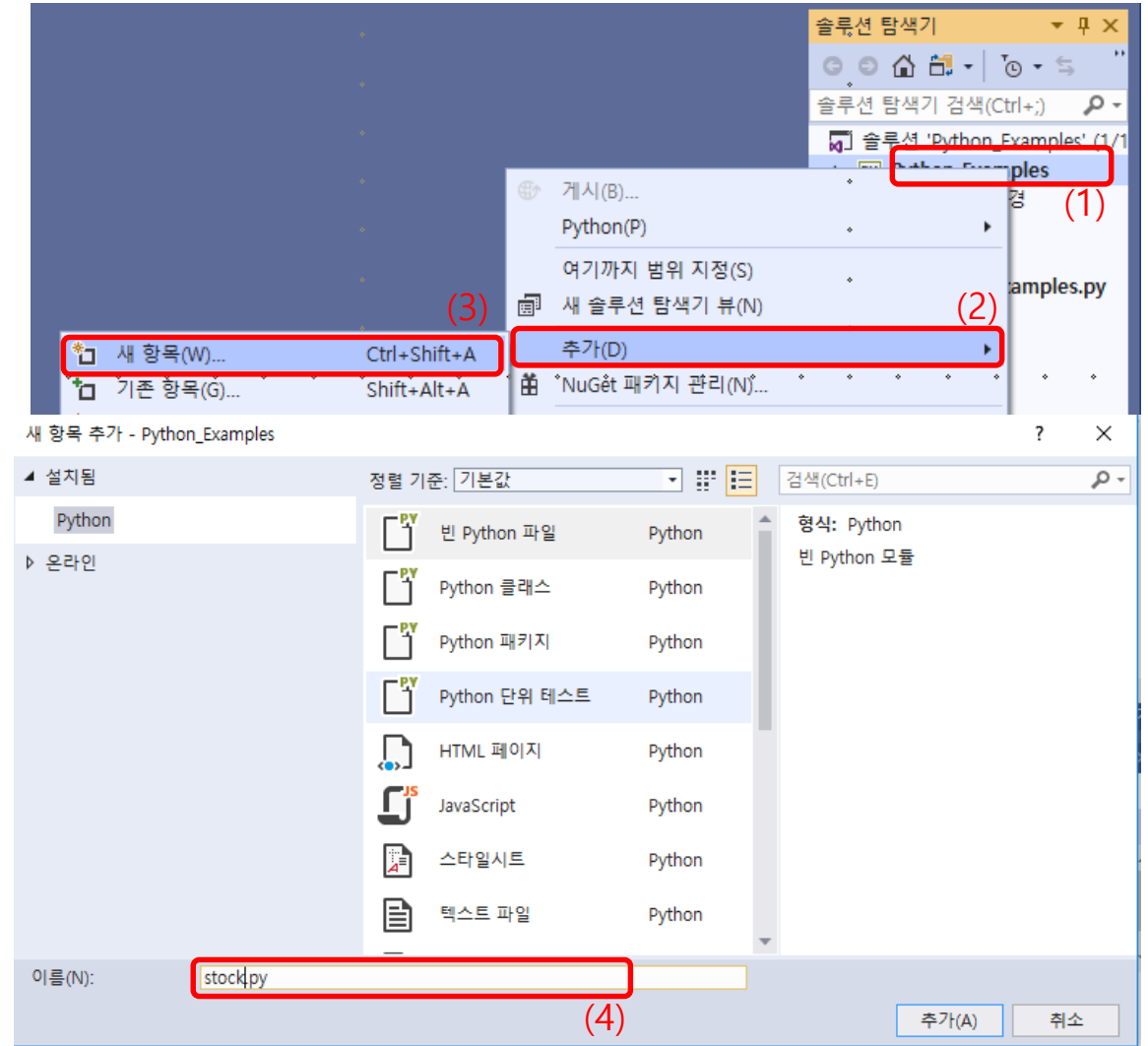
#### • 파라미터와 반환

- price, 표준 파라미터
  - 호출 시 반드시 전달되어야 한다.
- rate, 기본값 파라미터
  - 후미에 배치되어야 한다.
  - 호출 시 전달되지 않으면 기본값 rate=0.15이고 사용되고,
  - 다른 값이 전달되면 전달된 값으로 계산된다.
- price\_uplo10000)
  - print(10000,0.15)와 같이 계산된다. (8500.0, 11500.0)
- price\_uplo(10000,0.30)
  - print(10000,0.30)와 같이 계산된다. (7000.0, 13000.0)
- 반환
  - 변수 price\_lo, price\_up 저장된 값이 튜플 형으로 반환된다.

```
>>> price_uplo(10000)
(11500.0 , 8500.0)
>>> (upper,lower)= price_uplo(10000)
>>> upper
11500.0
>>> lower
8500.0
>>> upper,lower= price_uplo(10000)
>>> upper
11500.0
>>> lower
8500.0
>>> upper_lower= price_uplo(10000)
>>> upper_lower
(11500.0, 8500.0)
>>> upper_lower[0]
11500.0
>>> upper_lower[1]
8500.0
>>> upper , lower=upper_lower
>>>
```

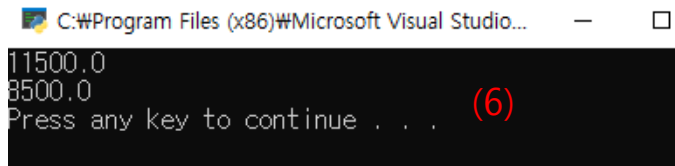
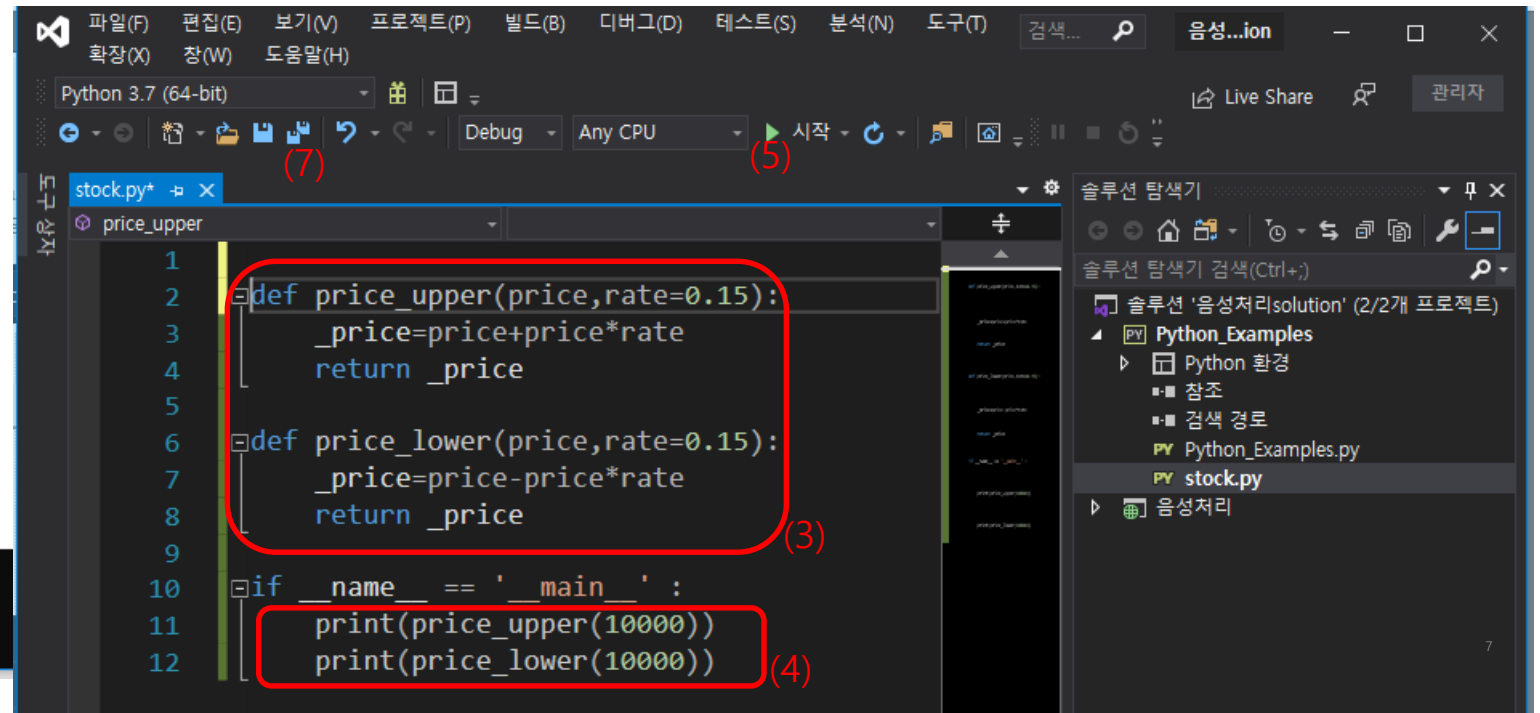
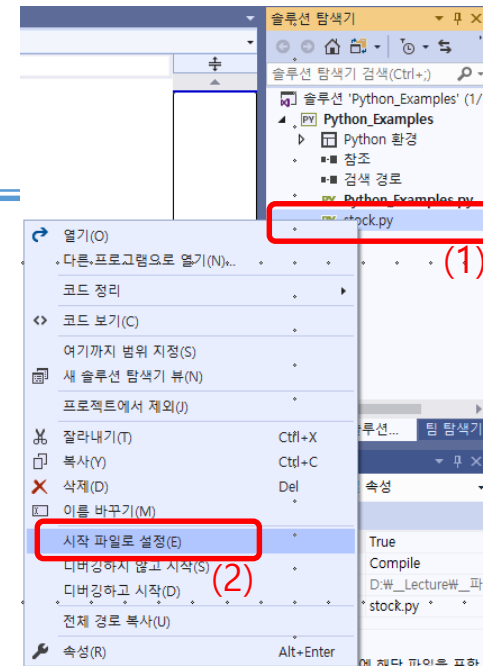
## 4.2 모듈

- 모듈은 함수, 리스트, 튜플, 사전 (dict) 들을 모아 저장한 파이썬 프로그램파일 (.py) 이다.
  - 일종의 라이브러리 이다.
  - 모듈에서 다른 모듈을 사용할 때 import 하여 사용한다.
- 
- 2.1 모듈 만들기
  - 모듈 생성
    - 솔루션 탐색기에서 빈 모듈 stock.py 만들기 (1-4)



## 4.2.1 모듈 만들기

- 모듈 (stock) 의 생성
  - 모듈 (파일) 열기 (1-2)
  - 활성화된 모듈에 함수 코드작성 (3)
  - 점검용 함수호출 코드 추가 (4)
  - 실행 (5) 및 실행결과 콘솔창에서
  - 저장 (7)



## 4.2.1 모듈 만들기(cont.)

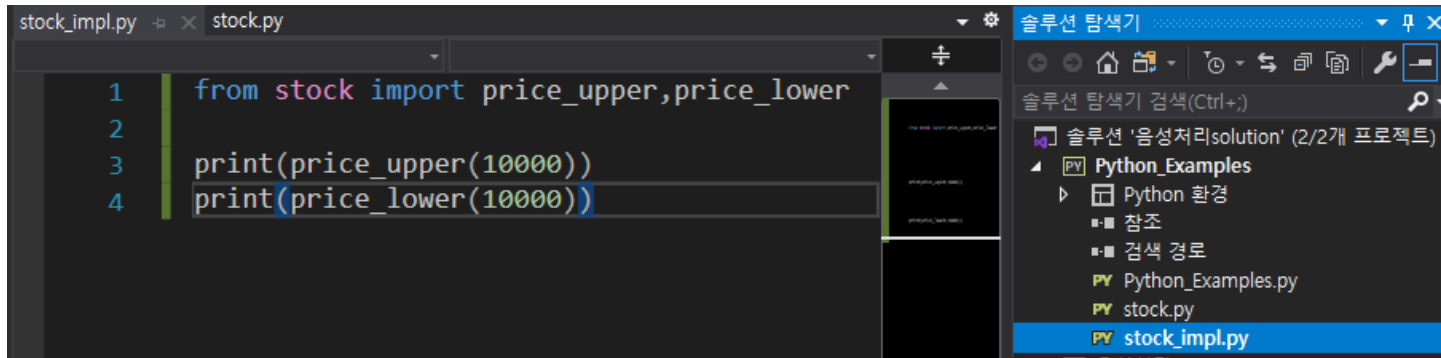
- 모듈 사용하기 - 대화형실행에서
- 대화형 Python\_examples 실행(1)
- 모듈 stock로부터 함수 price\_upper를 import 한다.
  - `from stock import price_upper, price_lower`
    - `price_upper(10000)`
    - `price_lower(10000)`
- 모듈 stock로부터 모든 함수를 import 한다.
  - `from stock import *`
    - `price_upper(10000)`
    - `price_lower(10000)`
- 모듈 stock를 import하고 사용하기
  - `import stock`
    - `stock.price_upper(10000)`
    - `stock.price_lower(10000)`
- 모듈 stock의 import하고 약어로 사용하기
  - `import stock as st`
    - `st.price_upper(10000)`
    - `st.price_lower(10000)`

```
1
2 def price_upper(price,rate=0.15) :
3     _price=price + price *rate
4     return _price
5
6 def price_lower(price,rate=0.15) :
7     _price=price - price *rate
8     return _lower
9
```

```
>>> from stock import price_upper
>>> price_upper(10000)
11500.0
>>> price_lower(10000)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'price_lower' is not defined
>>> from stock import price_upper,price_lower
>>> price_upper(10000)
11500.0
>>> price_lower(10000)
8500.0
>>> from stock import *
>>> price_upper(10000)
11500.0
>>> 11500.0
>>> price_lower(10000)
8500.0
>>> import stock
>>> stock.price_upper(10000)
11500.0
>>> stock.price_lower(10000)
8500.0
>>>
```



## • 모듈 stock 사용하기 - 다른 모듈안에서

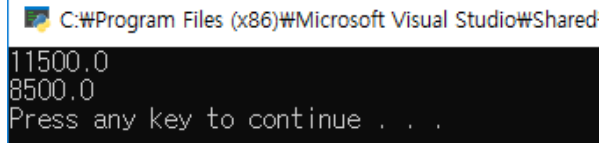


The screenshot shows the Visual Studio IDE. The main editor window displays the contents of stock.py:

```
1 from stock import price_upper, price_lower
2
3 print(price_upper(10000))
4 print(price_lower(10000))
```

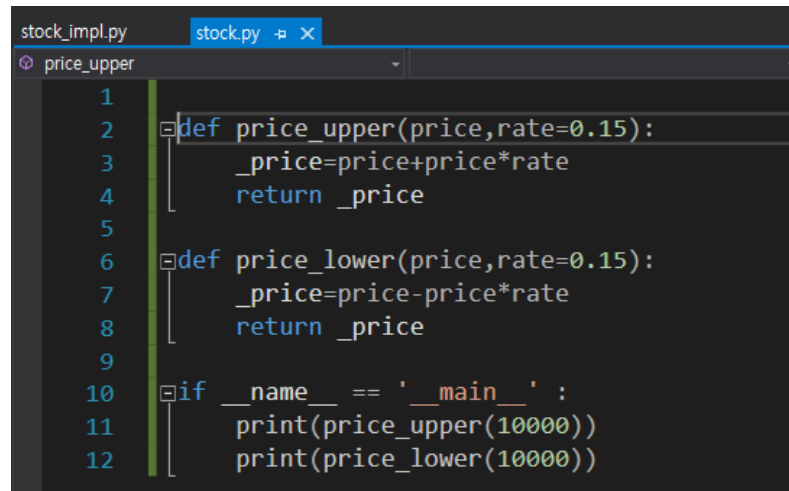
The Solution Explorer on the right shows the project structure:

- 솔루션 탐색기 (Solution Explorer)
- 솔루션 탐색기 검색 (Ctrl+.)
- 솔루션 '응성처리solution' (2/2개 프로젝트)
- Python\_Examples
  - Python 환경
  - 참조
  - 검색 경로
  - Python\_Examples.py
  - stock.py
  - stock\_impl.py



The screenshot shows the command prompt output:

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared
11500.0
8500.0
Press any key to continue . . .
```



The screenshot shows the Visual Studio IDE with the stock\_impl.py file open. The code is as follows:

```
1
2 def price_upper(price, rate=0.15):
3     _price = price + price * rate
4     return _price
5
6 def price_lower(price, rate=0.15):
7     _price = price - price * rate
8     return _price
9
10 if __name__ == '__main__':
11     print(price_upper(10000))
12     print(price_lower(10000))
```

## 4.2.2 표준모듈사용하기

---

- Python은 Python과 함께 배포되는 표준 모듈들을 포함하고 있다. Python에 포함된 표준 모듈중에서 몇 개를 소개하면 다음과 같다.
  - sys: 파이썬 시스템과 대화를 가능하게 한다.
  - os : 운영체제와 대화를 가능하게 한다.
  - string: 문자열을 처리할 수 있도록 해준다.
  - re(정규표현식regular expressions): 문자열을 정규식으로 다룰 수 있게 해준다.
  - math: 많은 수학적 함수에 접근하도록 해준다.
  - time: 시간(그리고 날짜) 함수.
  - 이외에도 Python에는 수십가지의 모듈이 제공되고 있다. 자세한 내용은 다음을 참조하기 바란다.  
<http://docs.python.org/library/index.html>
- Python 인터프리터가 동작되면 자동적으로 몇개의 모듈이 포함되는 데, 이 모듈들의 이름을 알아보려면 `dir()` 명령어를 사용하면 된다.

```
>>> dir()
['_builtins_', '__doc__', '__loader__', '__name__', '__package__', '__spec__']
>>>
```

## 4.2.3 time 모듈사용하기

- 시간처리를 위하여 파이썬에 준비되어있는 모듈은 time입니다. time모듈에는 유용한 함수들이 있다.
  - time() : 1970.01.01 00:00:00를 기준으로 현재까지의 시간을 초로 반환한다.
  - ctime() : 읽기 쉬운 형식으로 변환하여 반환한다.
  - sleep(n) : n초간 일시 실행중지
  - time : 모듈의 이름 time은 path를 요청
- dir(모듈): 모듈의 모든 구성요소를 반환한다.

```
>>> import time
>>> time.time()
1568593459.636304
>>> time.ctime()
'Mon Sep 16 09:24:29 2019'
>>> time.ctime().split(' ')[-1] 연수
'2019'
>>> time.ctime().split(' ')[2] 일
'16'
>>> time.ctime().split(' ')[1:3] 월, 일, 년
['Sep', '16']
```

```
>>> for i in range(3) :
...     print(time.ctime())
...     time.sleep(2)
...
Mon Sep 16 09:26:53 2019
Mon Sep 16 09:26:55 2019
Mon Sep 16 09:26:57 2019
>>> time #모듈 time은 내장모듈이다.
<module 'time' (built-in)>
>>> import random
>>> random #모듈 random의 패쓰 확인
<module 'random' from 'C:\\Program Files (x86)\\Micros
oft Visual Studio\\Shared\\Python36_64\\lib\\random.py
'>
>>> >>> dir(time) #모듈 time의 모든 구성요소 확인
['_STRUCT_TM_ITEMS', '__doc__', '__loader__', '__name_
__', '__package__', '__spec__', 'altzone', 'asctime', 'clock',
'ctime', 'daylight', 'get_clock_info', 'gmtime', 'localtime',
'mktime', 'monotonic', 'perf_counter', 'process_time',
'sleep', 'strftime', 'strptime', 'struct_time', 'time', 'time
zone', 'tzname']
>>>
```

## 4.2.4 os 모듈 사용하기

- os 모듈은 Operating System이 제공하는 기능을 파이썬에서 사용할 수 있게해 준다.
  - import os
  - os.getcwd()
    - 현재의 디렉토리 패스를 반환(get current directory)
  - os.listdir(path)
    - path 디렉토리의 모든 파일과 하위 디렉토리 이름을 list로 반환한다.
      - path가 주어지지 않으면 현재의 디렉토리

```
>>> import os
>>> os.getcwd()           #get current work directory
'D:\\Python_Examples\\Python_Examples'
>>> os.listdir()         #list all elements in current directory
['Python_Examples.py', 'Python_Examples.pyproj', 'stock.py', '__pycache__']
>>> fns=os.listdir('D:\\Python_Examples\\Python_Examples')
>>> fns
['Python_Examples.py', 'Python_Examples.pyproj', 'stock.py', '__pycache__']
>>> for fn in fns :
...     if fn.endswith('.py') :           #' py' 로 끝나는 이름
...         print(fn)
...
Python_Examples.py
stock.py
```

## 4.2.5 math 모듈

---

- Math와 cmath 모듈을 이용한 수학연산
  - Math 모듈은 실수의 수학함수를
  - Cmath는 복소수의 수학함수를 지원한다

```
>>> import math
>>> math.sqrt(-4)
Traceback (most recent call last):
  File "<interactive input>", line 1, in <module>
ValueError: math domain error
>>> import cmath
>>> cmath.sqrt(-4)
2j
>>>
```

```
>>> z = 1 + 1j
>>> z
(1+1j)
>>> zp = cmath.polar(z)
>>> zp
(1.4142135623730951, 0.7853981633974483)
>>> zp[0]
1.4142135623730951
>>> zp[1]
0.7853981633974483
>>> cmath.rect(zp[0], zp[1])
(1.0000000000000002+1j)
>>>
```

## 4.3 파이썬 내장함수

- 자주 사용되는 함수는 내장함수 (Built-in Functions)로 준비되어 있다. 예를 들면 절대값, 리스트의 갯수 등이다.
- Import 하지 않고 사용할 수 있다.
- `abs(x)`는 정수형 및 실수형 값의 절대값을 반환한다. `chr(n)`은 유니코드 값을 문자로, `ord(ch)`는 문자를 유니코드 로 반환한다.

```
>>> abs(-3)
3
>>> abs(-3.5)
3.5
>>>
```

<a href="#">abs()</a>	<a href="#">delattr()</a>	<a href="#">hash()</a>	<a href="#">memoryview()</a>	<a href="#">set()</a>
<a href="#">all()</a>	<a href="#">dict()</a>	<a href="#">help()</a>	<a href="#">min()</a>	<a href="#">setattr()</a>
<a href="#">any()</a>	<a href="#">dir()</a>	<a href="#">hex()</a>	<a href="#">next()</a>	<a href="#">slice()</a>
<a href="#">ascii()</a>	<a href="#">divmod()</a>	<a href="#">id()</a>	<a href="#">object()</a>	<a href="#">sorted()</a>
<a href="#">bin()</a>	<a href="#">enumerate()</a>	<a href="#">input()</a>	<a href="#">oct()</a>	<a href="#">staticmethod()</a>
<a href="#">bool()</a>	<a href="#">eval()</a>	<a href="#">int()</a>	<a href="#">open()</a>	<a href="#">str()</a>
<a href="#">breakpoint()</a>	<a href="#">exec()</a>	<a href="#">isinstance()</a>	<a href="#">ord()</a>	<a href="#">sum()</a>
<a href="#"> bytearray()</a>	<a href="#">filter()</a>	<a href="#">issubclass()</a>	<a href="#">pow()</a>	<a href="#">super()</a>
<a href="#">bytes()</a>	<a href="#">float()</a>	<a href="#">iter()</a>	<a href="#">print()</a>	<a href="#">tuple()</a>
<a href="#">callable()</a>	<a href="#">format()</a>	<a href="#">len()</a>	<a href="#">property()</a>	<a href="#">type()</a>
<a href="#">chr()</a>	<a href="#">frozenset()</a>	<a href="#">list()</a>	<a href="#">range()</a>	<a href="#">vars()</a>
<a href="#">classmethod()</a>	<a href="#">getattr()</a>	<a href="#">locals()</a>	<a href="#">repr()</a>	<a href="#">zip()</a>
<a href="#">compile()</a>	<a href="#">globals()</a>	<a href="#">map()</a>	<a href="#">reversed()</a>	<a href="#">__import__()</a>
<a href="#">complex()</a>	<a href="#">hasattr()</a>	<a href="#">max()</a>	<a href="#">round()</a>	

```
>>> ord('A'); ord('a'); ord('일')
65
97
51068
>>> chr(65); chr(97); chr(51068)
'A'
'a'
'일'
>>>
```

## 4.3 파이썬 내장함수 (cont.)

- 동물 이름을 출력하는 코드를 enumerate 함수를 이용한 예
- 시퀀스 자료형 (리스트, 튜플, 문자열) 등을 입력 받은 후 enumerate 객체를 반환한다.
- len 함수는 시퀀스 자료의 원소 갯 수를 반환한다.
- list 함수는 시퀀스, 문자열을 리스트로 변환한다.

```
>>> len('animal') # 문자열
6
>>> len(['dog', 'cat', 'monkey']) # 리스트
3
>>> len(('dog', 'cat', 'monkey')) # 튜플
3
>>> len({'dog':10, 'cat':3, 'monkey':1}) # 사전
3
```

```
>>> for i, animal in enumerate(['dog', 'cat', 'monkey']):
...     print(i, animal)
...
0 dog
1 cat
2 monkey
>>>
```

```
>>> i=0
>>> for animal in ['dog', 'cat', 'monkey']:
...     print(i, animal)
...     i=i+1
...
0 dog
1 cat
2 monkey
>>>
```

```
>>> list('animal') # 문자열
['a', 'n', 'i', 'm', 'a', 'l']
>>> list(['dog', 'cat', 'monkey']) # 튜플
['dog', 'cat', 'monkey']
>>> list({'dog':10, 'cat':3, 'monkey':1}) # 사전
['dog', 'cat', 'monkey']
>>> list({'dog':10, 'cat':3, 'monkey':1}.items()) # 사전 아이템
[('dog', 10), ('cat', 3), ('monkey', 1)]
>>>
```

## 4.3 파이썬 내장함수 (cont.)

- min, max는 리스트의 최솟값 및 최댓값을 반환한다. sorted함수는 원소를 오름차순으로 정렬하여 리스트로 출력한다.
- int, float 함수는 문자열을 정수, 실수로 변환한다.
- formatted 출력 예

```
>>> max((1,2,3))
3
>>> min((1,2,3))
1
>>> max([1,2,3])
3
>>> min([1,2,3])
1
```

```
>>> int('3')
3
>>> float('3.5')
3.5
>>> str(3)
'3'
>>> str(345.6)
'345.6'
>>> str([1,2,3, 'a', {'b':4, 'a':0}])
"[1, 2, 3, 'a', {'b': 4, 'a': 0}]"
```

```
>>> sorted((1,3,2,4))
[1, 2, 3, 4]
>>> sorted([1,3,2,4])
[1, 2, 3, 4]
>>> sorted(['a','x','b'])
['a', 'b', 'x']
```

```
>>> sorted([1,3,2,4])
[1, 2, 3, 4]
>>> sorted([1,3,2,4],key=lambda k:-k)
[4, 3, 2, 1]
>>> sorted({'a':1, 'd':4, 'b':10})
['a', 'b', 'd']
>>> sorted({'a':1, 'd':4, 'b':10}.items())
[('a', 1), ('b', 10), ('d', 4)]
>>> sorted({'a':1, 'd':4, 'b':10}.items(),key=lambda kv:-kv[1])
[('b', 10), ('d', 4), ('a', 1)]
```

```
>>> '{:4} {:7.3f}'.format(2,12.34)
' 2 12.340'
>>> '{:04} {:07.3f}'.format(2,12.34)
'0002 012.340'
>>>
```



## 4.4 연습문제

---

- 두 수의 받아 합을 계산하여 반환하는 함수를 작성하세요.
  - `def mysum(a,b):`
- 리스트를 입력 받아 리스트의 최솟값, 최댓값을 반환하는 함수를 작성하세요.
  - `def min_max(data_list):`
- 폴더의 절대경로를 입력 받아 폴더에 있는 확장자를 기본값 파라미터로 확장자를 갖는 모든 파일 이름을 반환하는 함수를 작성하세요.
  - `Def get_fns(path, endswitch= '.py' ):`
- 두수를 입력 받아 시작부터 끝까지의 수를 합을 구하여 출력하는 함수를 작성하시오.
  - `def get_sum(n_start, n_end):`
- 문자열 리스트를 입력 받아서 요소의 앞의 3 문자만으로 이루어지는 리스트로 만들고 오름차순으로 정렬하여 출력하는 함수를 작성하시오. 예 [ 'happy' , 'angree' , 'neutral' , 'sad' ]  
=> [ 'ang' , 'hap' , 'neu' , 'sad' ]
  - `def trim_sort(str_list):`
- 참고자료 [[link](#)]