

표준 모듈

목차

- 시작하기 전에
- 모듈 사용의 기본 : math 모듈
- random 모듈
- os 모듈
- datetime 모듈
- time 모듈
- urllib 모듈
- 키워드로 정리하는 핵심 포인트
- 확인문제

[핵심 키워드] : 표준 모듈, import 구문, as 키워드, 파이썬 문서

[핵심 포인트]

조건문, 반복문을 조합해서 만들어주는 코드를 활용하는 방법을 배워본다. 표준 모듈을 사용하면서 모듈 사용 방법을 익힌다.

- 모듈 (module)
 - 코드를 분리하고 공유하는 기능
 - 표준 모듈
 - 파이썬에 기본적으로 내장된 모듈
 - 외부 모듈
 - 사람들이 만들어 공개한 모듈

```
import 모듈 이름
```

- math 모듈
 - 수학과 관련된 기능

- `import math`

자동 완성 기능으로 살펴보는 math 모듈의 변수와 함수

```
1 import math
2
3 math.
```

| | |
|----------|---|
| acos | def acos(x) |
| acosh | Return the arc cosine (measured in radians) of x. |
| asin | |
| asinh | |
| atan | |
| atan2 | |
| atanh | |
| ceil | |
| copysign | |
| cos | |
| cosh | |
| degrees | |

- math 모듈을 사용하는 코드

```
>>> import math
```

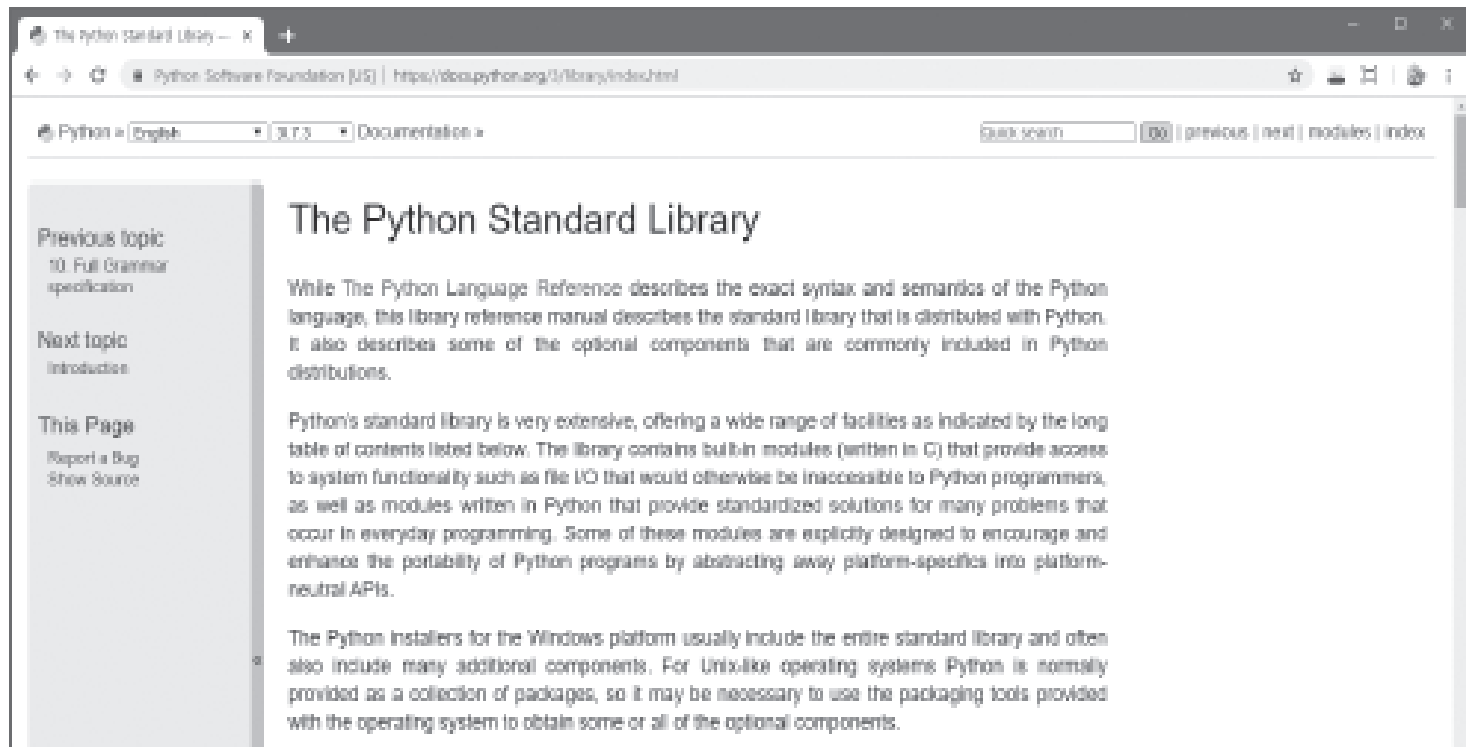
- 수학/삼각함수

```
>>> math.sin(1)      # 사인
0.8414709848078965
>>> math.cos(1)      # 코사인
0.5403023058681398
>>> math.tan(1)      # 탄젠트
1.5574077246549023
>>>
>>> math.floor(2.5)   # 내림
2
>>> math.ceil(2.5)    # 올림
3
```

- 모듈 문서
 - math 모듈은 다양한 기능 가지고 있음

| 변수 또는 함수 | 설명 |
|-----------------------------|-------------|
| <code>sin(x)</code> | 사인값을 구합니다. |
| <code>cos(x)</code> | 코사인값을 구합니다. |
| <code>tan(x)</code> | 탄젠트값을 구합니다. |
| <code>log(x[, base])</code> | 로그값을 구합니다. |
| <code>ceil(x)</code> | 올림합니다. |
| <code>floor(x)</code> | 내림합니다. |

- 파이썬 공식 문서에서 표준 모듈 등 정보 확인 가능
 - <http://docs.python.org/3/library/index.html>



- from 구문

- 다양한 함수를 계속해서 입력하는 것의 비효율성

```
from 모듈 이름 import 가져오고 싶은 변수 또는 함수
```

- '가져오고 싶은 변수 또는 함수'에 여러 개의 변수 또는 함수 입력 가능
- 이를 통해 가져온 기능은 math 붙이지 않고도 사용할 수 있음

```
>>> from math import sin, cos, tan, floor, ceil
>>> sin(1)
0.8414709848078965
>>> cos(1)
0.5403023058681398
>>> tan(1)
1.5574077246549023
>>> floor(2.5)
2
>>> ceil(2.5)
3
```

- as 구문
 - 모듈의 이름이 너무 길어 짧게 줄여 사용하고 싶은 경우

```
import 모듈 as 사용하고 싶은 식별자
```


```
>>> import math as m
>>> m.sin(1)
0.8414709848078965
>>> m.cos(1)
0.5403023058681398
>>> m.tan(1)
1.5574077246549023
>>> m.floor(2.5)
2
>>> m.ceil(2.5)
3
```

- random 모듈

```
import random
```

- random 모듈 문서

- <http://docs.python.org/3/library/random.html#examples-and-recipes>



The screenshot shows a web browser window with the title 'random — generate pseudo-random numbers'. The address bar shows the URL 'https://docs.python.org/3/library/random.html#examples-and-recipes'. The main content area is titled 'Examples and Recipes' and contains a section 'Basic examples:' with several code snippets and their corresponding comments:

```
>>> random() # Random float: 0.0 <= x < 1.0
0.37444887175646646

>>> uniform(2.5, 10.0) # Random float: 2.5 <= x < 10.0
3.1800146073117523

>>> expovariate(1 / 5) # Interval between arrivals averaging 5 seconds
5.148557573865803

>>> randrange(10) # Integer from 0 to 9 inclusive
7

>>> randrange(0, 101, 2) # Even integer from 0 to 100 inclusive
26

>>> choice(['win', 'lose', 'draw']) # Single random element from a sequence
'draw'

>>> deck = 'ace two three four'.split() # Shuffle a list
>>> shuffle(deck)
>>> deck
['four', 'two', 'ace', 'three']

>>> sample([10, 20, 30, 40, 50], k=4) # Four samples without replacement
[40, 10, 50, 30]
```

— 예시

```
01 import random
02 print("# random 모듈")
03
04 # random(): 0.0 <= x < 1.0 사이의 float를 리턴합니다.
05 print("- random():", random.random())
06
07 # uniform(min, max): 지정한 범위 사이의 float를 리턴합니다.
08 print("- uniform(10, 20):", random.uniform(10, 20))
09
10 # randrange(): 지정한 범위의 int를 리턴합니다.
11 # - randrange(max): 0부터 max 사이의 값을 리턴합니다.
12 # - randrange(min, max): min부터 max 사이의 값을 리턴합니다.
13 print("- randrange(10)", random.randrange(10))
```

```
14
15 # choice(list): 리스트 내부에 있는 요소를 랜덤하게 선택합니다.
16 print("- choice([1, 2, 3, 4, 5]):", random.choice([1, 2, 3, 4, 5]))
17
18 # shuffle(list): 리스트의 요소들을 랜덤하게 섞습니다.
19 print("- shuffle([1, 2, 3, 4, 5]):", random.shuffle([1, 2, 3, 4, 5]))
20
21 # sample(list, k=<숫자>): 리스트의 요소 중에 k개를 뽑습니다.
22 print("- sample([1, 2, 3, 4, 5], k=2):", random.sample([1, 2, 3, 4, 5], k=2))
```

실행결과

```
# random 모듈
- random(): 0.5671614057098718
- uniform(10, 20): 18.6271140555
- randrange(10) 6
- choice([1, 2, 3, 4, 5]): 2
- shuffle([1, 2, 3, 4, 5]): None
- sample([1, 2, 3, 4, 5], k=2):
```

- 5행의 random.random()처럼 random을 계속 입력하는 것은 효율적이지 못하므로 from 구문 활용해서 임포트

```
from time import random, randrange, choice
```

● sys 모듈

- 시스템과 관련된 정보 가진 모듈
- 명령 매개변수 받을 때 많이 사용

```
01  # 모듈을 읽어 들입니다.  
02  import sys  
03  
04  # 명령 매개변수를 출력합니다.  
05  print(sys.argv)
```

```
06  print("----")
07
08  # 컴퓨터 환경과 관련된 정보를 출력합니다.
09  print("getwindowsversion()", sys.getwindowsversion())
10  print("----")
11  print("copyright:", sys.copyright)
12  print("----")
13  print("version:", sys.version)
14
15  # 프로그램을 강제로 종료합니다.
16  sys.exit()
```

- 5행 sys.argv
 - 아래와 같이 실행하면 ['module_sys.py', '10', '20', '30'] 리스트 들어옴

```
> python module_sys.py 10 20 30
```

`['module_sys.py', '10', '20', '30']` → 명령 매개변수입니다. 입력한 명령어에 따라 달라집니다.

```
getwindowsversion:() sys.getwindowsversion(major=10, minor=0, build=14393,
platform=2, service_pack='')

```

```
copyright: Copyright (c) 2001-2019 Python Software Foundation.
All Rights Reserved.

```

...생략...

```
version: 3.7.3 (v3.7.3:ef4ecbed12, Mar 21 2019, 17:54:52) [MSC v.1916 32
bit (Intel)]

```


- os 모듈

- 운영체제와 관련된 기능 가진 모듈
- 새로운 폴더 만들거나 폴더 내부 파일 목록 보는 등

```
01  # 모듈을 읽어 들입니다.
02  import os
03
04  # 기본 정보를 몇 개 출력해봅시다.
05  print("현재 운영체제:", os.name)
06  print("현재 폴더:", os.getcwd())
07  print("현재 폴더 내부의 요소:", os.listdir())
08
09  # 폴더를 만들고 제거합니다[폴더가 비어있을 때만 제거 가능].
10  os.mkdir("hello")
11  os.rmdir("hello")
12
```

```
13  # 파일을 생성하고 + 파일 이름을 변경합니다.  
14  with open("original.txt", "w") as file:  
15      file.write("hello")  
16  os.rename("original.txt", "new.txt")  
17  
18  # 파일을 제거합니다.  
19  os.remove("new.txt")  
20  # os.unlink("new.txt")  
21  
22  # 시스템 명령어 실행  
23  os.system("dir")
```

현재 운영체제: nt

현재 폴더: C:\Users\hasat\sample

현재 폴더 내부의 요소: ['.vscode', 'beaut.py', 'download-png1.py', 'file.txt', 'freq.json', 'ghostdriver.log', 'iris.csv', 'lang-plot.png', 'mnist', 'mtest.py', 'newFile.xlsx', 'output.png', 'proj', 'rint.py', 'stats_104102.xlsx', 'test', 'test.csv', 'test.html', 'test.png', 'test.py', 'test.rb', 'test.txt', 'test_a.txt', 'train', 'underscore.js', 'Website.png', 'Website_B.png', 'Website_C.png', 'Website_D.png', '__pycache__']

C 드라이브의 볼륨: BOOTCAMP

볼륨 일련 번호: FCCF-6067

C:\Users\hasat\sample 디렉터리

```
2019-05-01 오전 12:18 <DIR>      .
2019-05-01 오전 12:18 <DIR>      ..
...생략...
2019-05-28 오전 04:49 <DIR>      __pycache__
                24개 파일              1,908,017 바이트
                8개 디렉터리 16,895,188,992 바이트 남음
```

→ 명령 프롬프트에서
그냥 dir을 입력했을 때의
결과와 동일합니다.
단지 파이썬에서
dir 명령어를
호출했을 뿐입니다.

- datetime 모듈

- date(날짜) 및 time(시간)과 관련된 모듈로, 날짜 형식 만들 때 자주 사용되는 코드로 구성

```
01  # 모듈을 읽어 들입니다.  
02  import datetime  
03  
04  # 현재 시각을 구하고 출력하기  
05  print("# 현재 시각 출력하기")  
06  now = datetime.datetime.now()  
07  print(now.year, "년")  
08  print(now.month, "월")  
09  print(now.day, "일")  
10  print(now.hour, "시")  
11  print(now.minute, "분")  
12  print(now.second, "초")  
13  print()  
14
```

실행결과

```
# 현재 시각 출력하기  
2019 년  
4 월  
23 일  
3 시  
51 분  
41 초  
  
# 시간을 포맷에 맞춰 출력하기  
2019.04.23 03:51:41  
2019년 4월 23일 3시 51분 41초  
2019년 04월 23일 03시 51분 41초
```

```
15  # 시간 출력 방법
16  print("# 시간을 포맷에 맞춰 출력하기")
17  output_a = now.strftime("%Y.%m.%d %H:%M:%S")
18  output_b = "{}년 {}월 {}일 {}시 {}분 {}초".format(now.year,\
19      now.month,\
20      now.day,\
21      now.hour,\
22      now.minute,\
23      now.second)
24  output_c = now.strftime("%Y{} %m{} %d{} %H{} %M{} %S{}").format(*"년월일시분초")
25  print(output_a)
26  print(output_b)
27  print(output_c)
28  print()
```

문자열, 리스트 등 앞에
요소 하나하나가 매개변수

- output_a처럼 strftime() 함수 사용하면 시간을 형식에 맞춰 출력 가능
- 그 외 다양한 시간 처리 기능

```
01  # 모듈을 읽어 들입니다.
02  import datetime
03  now = datetime.datetime.now()
04
05  # 특정 시간 이후의 시간 구하기
06  print("# datetime.timedelta로 시간 더하기")
07  after = now + datetime.timedelta(\
08      weeks=1,\
09      days=1,\
10      hours=1,\
11      minutes=1,\
12      seconds=1)
```

```
13 print(after.strftime("%Y{} %m{} %d{} %H{} %M{} %S{}").format(*"년월일시분초"))
14 print()
15
16 # 특정 시간 요소 교체하기
17 print("# now.replace()로 1년 더하기")
18 output = now.replace(year=(now.year + 1))
19 print(output.strftime("%Y{} %m{} %d{} %H{} %M{} %S{}").format(*"년월일시분초"))
```

실행결과

```
# datetime.timedelta로 시간 더하기
2019년 05월 01일 03시 39분 26초
```

```
# now.replace()로 1년 더하기
2020년 04월 23일 02시 38분
```

- timedelta() 함수 사용하면 특정한 시간의 이전 또는 이후 구함
 - "1년 후" 구할 때는 replace() 함수 사용해 날짜 값을 교체

- time 모듈

- 시간과 관련된 기능

```
import time
```

- 특정 시간 동안 코드 진행을 정지
- 정지하고 싶을 시간을 초 단위로 입력

```
01 import time
02
03 print("지금부터 5초 동안 정지합니다!")
04 time.sleep(5)
05 print("프로그램을 종료합니다")
```

실행결과

지금부터 5초 동안 정지합니다!
프로그램을 종료합니다

↓

5초 동안 정지한 이후에 출력합니다.

- urllib 모듈
 - URL 다루는 라이브러리

```
01  # 모듈을 읽어 들입니다.  
02  from urllib import request  
03  
04  # urlopen() 함수로 구글의 메인 페이지를 읽습니다.  
05  target = request.urlopen("https://google.com")  
06  output = target.read()  
07  
08  # 출력합니다.  
09  print(output)
```

```
b'<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage"
lang="ko"><head><meta content="text/html; charset=UTF-8" http-equiv="Content-
Type"><meta content="/logos/doodles/2019/amy-johnsons-114th-birthday-
5154304993263616.2-law.gif" itemprop="image">
...생략...
```

- 바이너리 데이터

- **표준 모듈** : 파이썬이 기본적으로 제공하는 모듈
- **import 구문** : 모듈 읽어 들일 때 사용하는 구문
- **as 키워드** : 모듈을 읽어 들이고 별칭 붙일 때 사용하는 구문
- **파이썬 문서** : 모듈의 자세한 사용 방법이 들어있는 문서

- 다음 중 math 모듈의 함수를 제대로 읽어들이지 못하는 코드를 고르세요.
- 파이썬 문서를 보면서 본문에서 살펴보지 않았던 모듈의 이름을 다섯 개 적어보세요.
그리고 ① import math ② import sin, cos, tan from math |보세요.
 ③ import math as m ④ from math import *

| 번호 | 모듈 이름 | 모듈 기능 |
|----|----------------------|-------------------------------|
| 0 | wave 모듈 | wav 음악 형식과 관련된 처리를 할 때 사용합니다. |
| 1 | <input type="text"/> | <input type="text"/> |
| 2 | <input type="text"/> | <input type="text"/> |
| 3 | <input type="text"/> | <input type="text"/> |
| 4 | <input type="text"/> | <input type="text"/> |