CHAPTER 5. 모듈

section87. 모듈 만들기

step 1.
알고가기

- 핵심내용
  - 내가 만든 유용한 파이썬 파일을 모듈로 임포트하여 사용할 수 있다.

- 코딩하기 전에
  - 파이썬에서 모듈이란 간단히 말하면 파이썬 파일(.py)이다.
  - 우리가 지금까지 만들어 온 모든 코드 파일이 모듈이 될 수 있다
  - 모듈을 사용하는 가장 큰 이유는 바로 '재사용'이다. 재사용이라는 것은 여러 번 사용될 만한 것을 한 번만 정의해놓으면 다시 정의할 필요 없이 가져다 사용할 수 있는 것을 말한다.
  - 모듈명은 .py를 뺀 파일명과 같다.

section87. 모듈 만들기

step 2.
코딩

```
1:  # myMath.py
2:
3:  pi = 3.14
4:
5:  def oneHapN(end):     # 1부터 N까지의 합을 구해주는 함수
6:      sum = 0
7:      for i in range(end):
8:          sum += i+1
9:      return sum
10:
11: def oneGopN(end):     # 1부터 N까지의 곱을 구해주는 함수
12:     total = 1
13:     for i in range(end):
14:         total *= i+1
15:     return total
```

section87. 모듈 만들기

step 2.
코딩

```
1:  # section_087.py
2:
3:  import myMath
4:  import math
5:
6:  print('내가 만든 모듈 사용 예제')
7:
8:  print(myMath.pi)
9:  print(math.pi)
10:
11: print('1부터 10까지의 합:', myMath.oneHapN(10))
12: print('1부터 10까지의 곱:', myMath.oneGopN(10))
```

section87. 모듈 만들기

step 3.
코드분석

- myMath.py 처럼 모듈로 사용할 파일에 변수와 함수 등을 정의해 둔다.
- import myMath
  - myMath.py 파일을 모듈로 임포트하는 코드이다.
  - myMath.py에 정의된 변수와 함수를 사용하기 위해 반드시 필요한 코드이다.
- import math
  - 파이썬 내장 모듈인 math를 임포트 하였다.
- myMath.pi, myMath.oneHapN(10)
  - 임포트한 모듈에 있는 변수나 함수를 사용할 때는 모듈명과 점으로 연결해서 사용한다.

section87. 모듈 만들기

step 3.
코드분석

- 모듈 불러오는 방법
  1. import myMath
     - 해당 모듈의 모든 변수와 함수를 가져온다.
     - myMath.pi 처럼 점 표기법으로 사용한다.
  2. from myMath import pi, oneHapN
     - 해당 모듈에서 지정된 변수 또는 함수만 가져온다.
     - 여기서는 myMath 모듈에서 pi와 oneHapN만 가져와서 쓰겠다는 선언이다.
     - 장점은 print(pi)처럼 점 표기법을 생략할 수 있다.
  3. from myMath import *
     - 해당 모듈에서 모든 변수와 함수를 가져온다.
     - 장점은 print(pi)처럼 점 표기법을 생략할 수 있다.

section87. 모듈 만들기

step 4.

3줄 요약

• 3줄 요약 •

- 내가 만든 파이썬 파일을 다른 파일에서 모듈로 사용할 수 있다.
- 내가 직접 모듈을 만들어서 배포할 수도 있다.
- 모듈을 불러오는 방법 중 첫 번째 방법을 주로 사용하자.

section88. __name__ 변수

step 1.
알고가기

- 핵심내용
  - 파이썬 파일이 모듈로 임포트 되는 순간 임포트 된 파일은 실행된다.

- 코딩하기 전에
  - 파이썬은 모듈을 임포트 할 때도 해당 모듈(여기서는 apink1 모듈)을 실행한다.
  - 파이썬 파일은 직접 실행될 때도 있고, 다른 파일에서 모듈로 임포트될 때도 실행되기 때문에 이 두 가지 경우를 구별할 필요가 있다.
  - 두 가지 경우를 구분해야 할 때 __name__변수를 사용한다.

section88. __name__ 변수

step 2.
코딩

```
1:  # apink1.py
2:
3:  print('에이핑크 (Apink)는 대한민국의 6인조 걸 그룹')
4:  print('apink1.py 파일이 실행되었다.')
5:
6:  print(_ _name_ _)
```

```
에이핑크 (Apink)는 대한민국의 6인조 걸 그룹
apink1.py 파일이 실행되었다.
_ _main_ _
```

이 실행 결과는 apink1.py를 직접 실행한 결과야.

```
1:  # section_088.py
2:
3:  import apink1
```

```
에이핑크 (Apink)는 대한민국의 6인조 걸 그룹
apink1.py 파일이 실행되었다.
apink1
```

이 실행 결과는 예제 파일을 실행한 결과야.

section88. __name__ 변수

step 3.
코드분석

- 우선 apink1.py를 작성하고 실행한 결과에서 __name__ 변수의 출력 값을 확인하자.
  - apink1.py가 직접 실행됐기 때문에 __main__이라는 값을 갖는다.
- 다음 apink1을 모듈로 임포트한 후 section_88.py 파일을 실행한 결과를 확인해 보자.
  - section_088.py에는 모듈 임포트 코드 외에 아무런 코드가 없다.
  - 이 출력결과는 모듈이 임포트되면서 해당 모듈이 자동적으로 실행한 결과이다.
  - 특히 __name__ 변수의 값이 apink1임을 알 수 있다.
- __name__ 변수로 해당 파일이 직접 실행되었는지 모듈로서 실행됐는지 체크할 수 있다.

section88. __name__ 변수

step 4.

3줄 요약

• 3줄 요약 •

- 파이썬 파일은 모듈로 임포트 될 때도 실행된다.
- 모듈이 실행될 때 불필요한 출력이 생길 수 있어서 이를 구분할 필요가 있다.
- 파이썬 파일이 직접 실행되었는지 혹은 모듈로 임포트되어 실행된 것인지는 __name__변수를 통해 구별할 수 있다.

section89. if __name__ == '__main__'

step 1.
알고가기

- 핵심내용
  - if문으로 __name__의 값을 구분하여 출력문을 분류해둘 수 있다.

- 코딩하기 전에
  - 파이썬 파일을 직접 실행할 때 __name__변수의 값은 '__main__'이고,
  - 다른 파일에 임포트 돼서 실행될 때는 __name__변수값이 바로 모듈명이다.
  - 이 두 경우를 분리함으로써 모듈로 임포트되어 실행될 때 불필요한 출력이나 실행을 미연에 방지할 수 있다.

section89. if __name__ == '__main__'

step 2.
코딩

```
1:  # apink2.py
2:
3:  if __name__ == '__main__':
4:      print('에이핑크 (Apink)는 대한민국의 6인조 걸 그룹')
5:      print('apink.py 파일이 실행되었다.')
```

이 실행 결과는 apink2.py를 직접 실행한 결과야.

```
에이핑크 (Apink)는 대한민국의 6인조 걸 그룹
apink.py 파일이 실행되었다.
```

```
1:  # section_089.py
2:
3:  import apink2
```

이 실행 결과는 예제 파일을 실행한 결과야. 아무 것도 출력되지 않았어.

section89. if __name__ == '__main__'

step 3.
코드분석

- if __name__ == '__main__'
  - 만약 __name__변수의 값이 __main__이라면
  - 파일이 직접 실행된 경우이므로 이 때 필요한 코드를 if문 블록에 작성한다.
  - 만약 __name__변수의 값이 __main__이 아니라면
  - 파일이 모듈로서 실행된 경우이므로 불필요한 출력 등을 미리 배제시킬 수 있다.
  - section_089.py 실행 결과 아무것도 출력된 것이 없음을 확인할 수 있다.

section89. if __name__ == '__main__'

step 4.

3줄 요약

• 3줄 요약 •

- 파이썬 파일은 모듈로 임포트 될 때도 실행된다.
- __name__변수의 값이 __main__이면 파일이 직접 실행된 것이고, __name__변수의 값이 모듈명이면 모듈로서 임포트되면서 실행된 것이다.
- if문을 이용해서 모듈로 임포트 되었을 때와 그렇지 않을 때를 구분할 수 있다.

section90. 내장모듈연습 - keyword 모듈

step 1.
알고가기

- 핵심내용
  - keyword모듈은 파이썬 예약어 관련 기능들을 제공한다.

- 코딩하기 전에
  - 지금까지 모듈을 사용하는 방법과 모듈을 만드는 방법을 배웠다.
  - 이제 파이썬을 설치하면 바로 사용할 수 있는 내장모듈을 알아본다.
  - 어떤 프로젝트를 시작하기에 앞서 내가 구현하려는 기능이 이미 모듈로 만들어져서 배포된 것이 있는지 찾아보는 것이 좋다

section90. 내장모듈연습 - keyword 모듈

step 2.
코딩

```
1:  # section_090.py
2:
3:  import keyword
4:
5:  print(keyword.iskeyword('None'))
6:  print(keyword.iskeyword('int'))
7:  print(keyword.kwlist)
```

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

section90. 내장모듈연습 – keyword 모듈

step 3.
코드분석

- keyword 모듈
  - 이 모듈은 예약어와 관련된 내용을 담고 있다
- keyword.iskeyword('None')
  - keyword.iskeyword()는 괄호 안의 값이 예약어인지 아닌지를 True/False로 알려주는 함수이다.
- keyword.kwlist
  - kwlist 변수를 사용하면 현재 파이썬 버전의 모든 예약어 리스트를 볼 수 있다.
  - 대문자로 시작하는 예약어 False, None, True를 포함해 총 33개의 예약어를 확인할 수 있다.

section90. 내장모듈연습 - keyword 모듈

step 4.

3줄 요약

• 3줄 요약 •

- 내가 만들려는 기능이 구현된 모듈이 이미 있다면 모듈을 적극 활용하자.
- 모듈을 활용하면 개발 시간을 절약할 수 있고 스트레스도 줄일 수 있다.
- keyword 모듈은 파이썬의 예약어에 관련된 함수와 변수들을 가지고 있다.

section91. random 모듈 – 난수 만들기

step 1.
알고가기

- 핵심내용
  - random.randint() 함수로 임의의 난수를 얻을 수 있다.

- 코딩하기 전에
  - 난수란 일정한 규칙 없이 무작위로 나열되는 숫자를 말한다.
  - random 모듈로 쉽게 난수를 만들 수 있다.

section91. random 모듈 – 난수 만들기

step 2.
코딩

```
1:  # section_091.py
2:
3:  import random
4:
5:  for i in range(6):
6:      number = random.randint(1, 45)
7:      print(number, end=' ')
```

```
6 32 45 42 45 27
```

section91. random 모듈 – 난수 만들기

step 3.
코드분석

- import random
  - 난수를 생성하기 위해 임포트해준다.
- random.randint(1, 45)
  - random.randint(시작값, 끝값) 함수는 시작값과 끝값 사이에서 난수 하나를 추출해 주는 함수이다.
  - 여러 번 실행하면 중복된 값이 출력될 수 있다.

section91. random 모듈 – 난수 만들기

step 4.

3줄 요약

• 3줄 요약 •

- 난수를 만들려면 random 모듈을 임포트한다.
- random.randint(시작값, 끝값)은 시작값과 끝값 사이에서 임의의 수를 추출해 준다.
- print( ) 함수의 end 속성은 출력 후 한 줄의 마지막을 어떻게 처리할 것인지를 지정해주는 역할을 한다.

section92. random 모듈 – 리스트를 섞고 무작위로 뽑기

step 1.
알고가기

- 핵심내용
  - shuffle()은 무작위로 섞고, choice()는 무작위로 뽑아주는 함수이다.

section92. random 모듈 – 리스트를 섞고 무작위로 뽑기

step 2.
코딩

```
1:  # section_092.py
2:
3:  import random
4:
5:  card = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10',
            'J', 'Q', 'K']
6:  print(card)
7:
8:  random.shuffle(card)
9:  print(card)
10:
11: print(random.choice(card))
```

```
['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']
['9', '4', '3', '8', 'Q', 'A', '5', '6', '2', 'J', 'K', '7', '10']
K
```

section92. random 모듈 – 리스트를 섞고 무작위로 뽑기

step 3.
코드분석

- import random
  - shuffle(), choice()를 사용하기 위해 모듈을 임포트한다.
- random.shuffle(card)
  - 리스트 항목을 무작위로 섞어주는 기능을 가지고 있다.
- random.choice(card)
  - 리스트에서 무작위로 하나의 항목을 선택해준다.

section92. random 모듈 – 리스트를 섞고 무작위로 뽑기

step 4.

3줄 요약

• 2줄 요약 •

- 리스트에 random.shuffle( )을 이용하면 리스트의 값을 무작위로 섞을 수 있다.
- random.choice( ) 함수는 무작위로 하나를 선택해 주는 함수이다.

section93. random 모듈 – 여러 항목을 한꺼번에 뽑기

step 1.
알고가기

- 핵심내용
  - sample()함수는 여러 항목을 중복되지 않게 뽑아 준다.

section93. random 모듈 – 여러 항목을 한꺼번에 뽑기

step 2.
코딩

```
1:  # section_093.py
2:
3:  import random
4:
5:  myStr = "최고의영화 – 죽은시인의사회"
6:  print(random.sample(myStr, 3))
7:
8:  myList = random.sample(range(1,46), 6)
9:  myList.sort()
10: print(myList)
```

```
['고', '영', '의']
[8, 12, 25, 26, 38, 42]
```

section93. random 모듈 – 여러 항목을 한꺼번에 뽑기

step 3.
코드분석

- import random
  - sample()을 사용하기 위해 모듈을 임포트한다.
- random.sample(myStr, 3)
  - myStr에서 3개의 값을 추출해주는 기능을 가진 함수이다.
  - 이 함수는 값을 중복되지 않게 추출해준다.
- random.sample(range(1, 46), 6)
  - 1~45사이의 숫자 중에서 6개의 숫자를 중복되지 않게 추출하는 코드이다.

section93. random 모듈 – 여러 항목을 한꺼번에 뽑기

step 4.

3줄 요약

• 3줄 요약 •

- random.sample( )의 첫 번째 인수는 시퀀스형 객체 또는 집합이고 두 번째 인수는 추출하고 싶은 개수이다.
- random.sample( ) 함수는 주어진 개수만큼 무작위로 뽑아주는 기능을 가진다.
- random.sample( ) 함수는 중복되지 않은 값을 뽑아준다.

section94. random 모듈 – 실수로 구성된 난수 만들기

step 1.
알고가기

- 핵심내용
  - random()함수 또는 uniform() 함수로 실수 난수를 구할 수 있다.

section94. random 모듈 – 실수로 구성된 난수 만들기

step 2.
코딩

```
1:  # section_094.py
2:
3:  import random
4:
5:  print(random.random())
6:
7:  print(random.uniform(0, 10))
```

```
0.7744850949181107
9.925523321537058
```

section94. random 모듈 – 실수로 구성된 난수 만들기

step 3.
코드분석

- import random
  - random(), uniform()을 사용하기 위해 모듈을 임포트 한다.
- random.random()
  - random 모듈의 가장 기본이 되는 함수이다.
  - 0.0~1.0 사이의 임의의 실수를 반환해 준다.
- random.uniform(0, 10)
  - random.uniform(시작값, 끝값) 함수는 주어진 시작값과 끝값 사이의 실수를 반환해 준다.

section94. random 모듈 – 실수로 구성된 난수 만들기

step 4.

3줄 요약

• 3줄 요약 •

- random.random( ) 함수는 0.0~1.0 사이의 실수를 무작위로 추출해 준다.
- random.uniform( ) 함수는 주어진 값의 범위에서 실수를 임의로 추출해 준다.
- random 모듈에는 다양한 난수 함수가 제공되는데 이중 자주 사용하는 함수를 정리해 보자.

section95. time 모듈 – 프로그램 실행 시간 측정하기

step 1.
알고가기

- 핵심내용
  - time.time()으로 현재 시간을 초단위로 알 수 있다.

- 코딩하기 전에
  - time 모듈에는 시간과 관련된 많은 함수들이 포함되어 있다.

section95. time 모듈 – 프로그램 실행 시간 측정하기

step 2.
코딩

```
1:  # section_095.py
2:
3:  import time
4:
5:  print('현재시각:', time.time())
6:
7:  def manyloop(max):
8:      t1 = time.time()
9:      for a in range(max):
10:         pass
11:     t2 = time.time()
12:     print(t2-t1, '초 경과')
13:
14: number = int(input('숫자를 입력하세요: '))
15: manyloop(number)
```

section95. time 모듈 – 프로그램 실행 시간 측정하기

step 3.
코드분석

- import time
  - time()을 사용하기 위해 모듈을 임포트한다.
- time.time()
  - 현재 시각을 알려주는 함수이다.
  - 출력 형태는 1970년 1월 1일 00시 00분 00초 이후 지금까지의 초를 나타낸 값이다.
- 어떤 알고리즘의 실행 전 후에 시간을 체크하면 해당 알고리즘의 실행시간을 확인할 수도 있다.

section95. time 모듈 – 프로그램 실행 시간 측정하기

step 4.

3줄 요약

• 2줄 요약 •

• time.time( ) 함수는 현재 시간을 1970년 이후부터 지금까지의 초 단위로 알려준다.
• time.time( ) 함수를 이용하면 프로그램의 실행 시간을 알아낼 수 있다.

section96. time 모듈 – ctime() 함수로 년월일시분초 추출하기

step 1.
알고가기

- 핵심내용
  - time.ctime() 함수는 현재시간을 문자열로 알려준다.

section96. time 모듈 - ctime() 함수로 년월일시분초 추출하기

step 2.
코딩

```
1:  # section_096.py
2:
3:  import time
4:
5:  current = time.ctime()
6:  print(current)
7:
8:  list_cur = current.split(' ')
9:
10: for t in list_cur:
11:     print(t)
```

```
Wed Apr 20 23:57:02 2016
Wed
Apr
20
23:57:02
2016
```

section96. time 모듈 – ctime() 함수로 년월일시분초 추출하기

step 3.
코드분석

- import time
  - ctime()을 사용하기 위해 모듈을 임포트 한다.
- time.ctime()
  - 현재 시각을 문자열로 반환해준다.
  - time.time()보다 실제 시간을 아는 데는 더 유용하다.
  - 출력형식은 'Wed Apr 20 23:57:02 2016'와 같다.

section96. time 모듈 – ctime() 함수로 년월일시분초 추출하기

step 4.

3줄 요약

• 2줄 요약 •

- time.ctime( ) 함수는 현재 시간을 문자열로 알려준다.
- time.ctime( ) 함수로 얻은 시간 정보를 문자열 함수를 이용해 분리해서 사용할 수 있다.

section97. time 모듈 – 시간 딜레이 주기

step 1.
알고가기

- 핵심내용
  - time.sleep()함수는 프로그램을 잠시 쉬게 만드는 함수이다.

section97. time 모듈 – 시간 딜레이 주기

step 2.
코딩

```
1:  # section_097.py
2:
3:  import time
4:
5:  for t in range(6):
6:      print(time.ctime())
7:      time.sleep(1)
```

```
Wed Apr 20 23:37:16 2016
Wed Apr 20 23:37:17 2016
Wed Apr 20 23:37:18 2016
Wed Apr 20 23:37:19 2016
Wed Apr 20 23:37:20 2016
Wed Apr 20 23:37:21 2016
```

section97. time 모듈 – 시간 딜레이 주기

step 3.
코드분석

- import time
  - sleep()을 사용하기 위해 모듈을 임포트한다.
- time.sleep(1)
  - sleep() 함수는 프로그램을 잠시 쉬게 해주는 기능을 가지고 있다.
  - 함수에 인자값을 넣어주면 해당 초만큼 딜레이를 줄 수 있다.

section97. time 모듈 – 시간 딜레이 주기

step 4.

3줄 요약

• 2줄 요약 •

- time.sleep( ) 함수는 프로그램을 잠시 쉬게 해주는 기능을 가진 함수이다.
- time.sleep( ) 함수는 주로 반복문 안에서 사용되고, 초를 실수 값으로 입력할 수 도 있다.

section98. datetime 모듈 – 다른 방법으로 년월일시분초 추출하기

step 1.
알고가기

- 핵심내용
  - datetime 모듈에도 현재시간을 구하는 방법이 있다.

- 코딩하기 전에
  - 이 모듈은 time모듈과 date모듈을 합친 모듈로 날짜와 시간을 모두 다루고 있는 모듈이다.

section98. datetime 모듈 – 다른 방법으로 년월일시분초 추출하기

step 2.
코딩

```
1:  # section_098.py
2:
3:  import datetime
4:
5:  d = datetime.datetime.now()
6:  print(d)
7:
8:  print(d.year, d.month, d.day, sep='/')
9:  print(d.hour, d.minute, d.second, d.microsecond, sep=':')
10: print(d.weekday())
```

```
2016-04-21 00:17:43.692717
2016/4/21
0:17:43:692717
3
```

section98. datetime 모듈 – 다른 방법으로 년월일시분초 추출하기

step 3.
코드분석

- import datetime
  - datetime.now()를 사용하기 위해 모듈을 임포트한다.
- datetime.datetime.now()
  - 현재 시각을 얻을 수 있다.
  - 출력형식은 '2016-04-21 00:17:43.692717'와 같다.
- d.year
  - datetime.datetime.now()로 구한 현재 시각은 점 표기법을 이용해 년, 월, 일, 시, 분, 초, 마이크로 초를 각각 개별 추출할 수 있다.
- d.weekday()
  - 요일을 숫자로 출력해준다.
  - 0~6사이의 값은 차례대로 월요일~일요일과 대응한다.

section98. datetime 모듈 – 다른 방법으로 년월일시분초 추출하기

step 4.

3줄 요약

• 3줄 요약 •

- datetime.datetime.now( )를 이용해서 현재 시간을 구할 수 있다.
- 이때 얻은 값에서 year, month, day...와 같은 방법으로 값을 사용할 수 있다.
- weekday( )로 얻는 값은 0~6이며, 0은 월요일이고 6은 일요일이다.