파이썬 프로그래밍

27차시

내장 함수 활용 2

⚠ 학습개요

… 내장 함수 map(function, iterable)
… 쉘에서 사용하는 내장 함수 dir()
… 프로젝트 lab1, lab2

⚠ 학습목표

… 리스트 등 시퀀스의 항목에 동일한 함수를 적용해 결과를 활용할 수 있다.
… 쉘에서 함수 dir()을 활용할 수 있다.
… 모듈 random을 활용해 필요한 함수를 구현할 수 있다.
… 로또 복권 시뮬레이션을 구현할 수 있다.

Chapter 1.

내장 함수 map (function, iterable)

PYTHON PROGRAMMING

⚠ 함수 map(function, iterable)

+ function

- 이터러블(iterable)의 각각의 항목에 대해 **function을 적용한 후** 그 결과를 돌려주는 이터레이터(iterator)를 반환

```
>>> def addone(n):
...     return n+1
...
>>>
>>> lst = list(map(addone, [10, 30, 50, 20]))
>>> print(lst)
[11, 31, 51, 21]
```

⚠ 함수 map()

+ iterable

항목을 하나씩 차례로 반환할 수 있는 객체(object)

- 시퀀스인 **문자열, 리스트와 튜플 모두 대표적인 이터러블**
- 이터레이터는 시퀀스인 튜플이나 리스트로 변환해 항목을 활용

```
>>> def add (x, y):
...     return x + y
...
>>>
>>> lst = list(map(add, [10, 30, 50, 20], [1, 3, 5, 2]))
>>> print(1st)
[11, 33, 55, 22]
>>> lst = list(map(lambda x, y: x + y, [10, 30, 50, 20], [1, 3, 5, 2]))
>>> print(1st)
[11, 33, 55, 22]
```

⚠ 함수 map()

[코딩실습] 내장 함수 map()을 활용 난이도 응용

```
1. import math
2. clst = list(map(math.ceil, [1.1, 2.2, 3.3, 4.4, 5.5, 6.6]))
3. print(f'map(math.ceil, 리스트) : {clst}')
4.
5. lst = [1.1, 2.2, 3.3, 4.4, 5.5]
6. mlst = list(map(int, lst))
7. print(f'map(int, 리스트): {mlst}')
8.
9. lst = [1, 2, 3, 4, 5]
10.plst = list(map(lambda x, y: x ** y, lst, lst))
11.print(f'map(lambda, 리스트): {plst}')
```

결과

```
map(math.ceil, 리스트) : [2, 3, 4, 5, 6, 7]
map(int, 리스트): [1, 2, 3, 4, 5]
map(lambda, 리스트): [1, 4, 27, 256, 3125]
```

⚠ 내장 함수 map()을 활용한 원 면적 구하기

[코딩실습] 내장 함수 map()을 활용한 원 면적 구하기 난이도 응용

```
1. circle = [3, 5, 7, 10]
2. area = list(map(lambda r: r * r * 3.14, circle))
3.
4. for c, a in zip(circle, area):
5.     print('반지름 {} => 원면적 {}'.format(c, a))
```

결과

```
반지름 3 => 원면적 28.26
반지름 5 => 원면적 78.5
반지름 7 => 원면적 153.86
반지름 10 => 원면적 314.0
```

끝

Chapter 2.

쉘에서 사용하는 내장 함수 dir()

PYTHON PROGRAMMING

⚠ 함수 dir()

+ 인자 없는 함수 dir()

- 현재 정의된 변수와 함수 이름의 리스트를 반환

+ 파이썬 쉘을 처음 실행해 dir()을 입력

- 파이썬 쉘에서 사용되는 시스템 변수 리스트
- 시스템에서 사용하는 모듈이나 변수는 앞뒤에 밑줄이 2개 붙어 __builtins__처럼 정의

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

⚠ 함수 dir()

✚ 이후 우리가 정의한 함수나 변수도 dir()로 확인 가능

```
>>> def mysum (x, y):
...     return x + y
...
>>>
>>> a = mysum(10, 20)
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__',
'__name__', '__package__', '__spec__', 'a', 'mysum']
```

⚠ dir(__builtins__)

✚ 내장된 표준 함수 리스트 보기

- __builtins__: 내장을 의미하는 모듈 이름

```
>>> dir(__builtins__)
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException',
'BlockingIOError', <중간생략>, '__name__', '__package__', '__spec__', 'abs',
'all', 'any', 'ascii', 'bin', 'bool', 'bytearray', 'bytes', 'callable',
'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits',
'debugfile', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval',
'evalsc', 'exec', 'exit', 'filter', 'float', 'format', 'frozenset',
'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input',
'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list',
'locals', 'map', 'max', 'memoryview', 'min', 'next', 'object', 'oct',
'open', 'open_in_spyder', 'ord', 'pow', 'print', 'property', 'quit',
'range', 'repr', 'reversed', 'round', 'runfile', 'set', 'setattr', 'slice',
'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars',
'zip']
```

좀 더 알아봅시다!

+ 도움말 요청 함수 help()

- 함수 help([object])는 대화형 모드에서 도움말을 위한 함수다.
- 파이썬 쉘에서 인자 없이 help()로 입력하면 대화형 도움말 시스템(help utility)이 시작된다.
- 프롬프트 help> 이후에 도움을 요청할 함수나 키워드 등을 입력하면 도움말이 출력된다.
- 도움말 시스템에서 빠져 나오려면 quit를 입력한다.

```
>>> help()
Welcome to Python 3.6's help utility!
If this is your first time using Python, you should definitely
check out the tutorial on the Internet at
https://docs.python.org/3.6/tutorial/.
<중간 생략>
```

좀 더 알아봅시다!

+ 도움말 요청 함수 help()

```
help > lambda
Lambdas
******
  lambda_espr        ::= "lambda"[parameter_list] ":"expression
  lambda_expr_nocond ::= "lambda" [parameter_list]
":"expression_nocond
Lambda expressions (sometimes called lambda forms) are used to
create anonymous functions. The expression "lambda parameters:
expression"
yields a function object. The unnamed object behave like a
function
object defined with:
  def <lambda>(parameters):
      return expression
```

끝

Chapter 3.

프로젝트
lab1, lab2

PYTHON PROGRAMMING

프로젝트 lab1

1에서 100까지 10개의 정수로 간단한 함수 이용

난이도 응용

함수 setsequencce(start, end, count)에서 1에서 100까지 정수 10개를 난수로 생성해 전역 변수인 리스트 nums에 추가하자.
전역 변수 nums에 저장된 10개의 수로 합과 평균, 최댓값과 최솟값을 출력하는 프로그램을 작성해보자.
리스트 항목의 합, 최대와 최소의 내장 함수 sum(), min(), max()를 사용한다.

문제 이해 (Understanding)

난수를 사용해 일련의 수를 만들고, 생성된 리스트에서 모든 항목을 더하고,
평균을 구하고, 최대와 최소를 구하는 프로그램이다.
리스트인 전역 변수 nums를 만들고 이를 함수에서 global nums로 선언해 사용한다.

프로젝트 lab1

설계 (Design)

알고리즘(Algorithm)

① 1~100 사이의 난수를 생성하기 위해 random 모듈의 randint()함수를 사용한다.
② 함수 setsequence(start, end, count)는 인자인 start~end 사이의 정수에서 count 개수 만큼의 난수를 생성해 전역 변수 nums에 추가한다.
- global nums 필요
③ 함수 호출 setsequence(1, 100, 10)으로 1~100사이의 정수 10개를 생성해 변수 nums에 추가한다.
④ 내장 함수 sum(), len(), min(), max()를 사용해 합과 평균, 최댓값과 최솟값을 출력한다.

표준 입출력 샘플

[9, 22, 33, 35, 35, 48, 55, 59, 60, 96]
합: 452, 평균: 45.20
최대: 9, 최소: 96

프로젝트 lab1

```
1.  #%% 프로젝트 Lab 1       07-pl01-variusfunc.py
2.  # 1에서 100까지 10개의 정수로 간단한 함수 이용
3.  from random import randint
4.
5.  def setsequence(start, end, count):
6.      ''' 전역변수 nums에 start(1)~end(100) 사이의 정수 count(10)개 추가 '''
7.      global nums
8.      for _ in range(count):
9.          nums.append(randint(start, end))
10.
11. nums = []
12. setsequence(1, 100, 10)
13. print(sorted(nums))
14. print('합: %d,  평균: %.2f' % (sum(nums), sum(nums)/len(nums)))
15. print('최대: %d,  최소: %d' % (min(nums), max(nums)))
```

```
[2, 23, 29, 31, 46, 47, 77, 79, 89, 95]
합: 518,  평균: 51.80
최대: 2,  최소: 95
```

프로젝트 lab2

로또 복권 시뮬레이션 난이도 응용

로또 복권을 사면 받는 로또 번호 6개의 조합을 5개로 정할 수 있다.
이 프로젝트는에서는 함수 autolotto(), printlotto(lotto), setwinlotto(), getwinnner(lotto) 등을 만들어 로또 표와 같이 자동으로 로또 번호와 당첨 번호를 생성하고 각각의 로또 번호가 생성된 당첨 번호와 몇 개 맞는지 출력하는 프로그램을 작성하자.
자동 로또 번호는 이미 학습한 난수를 이용하며, 5개의 로또 번호에서 당첨 번호와의 당첨 개수와 번호를 알아내 출력한다.

문제 이해 (Understanding)

로또 복권은 1에서 45까지의 6개의 정수를 당첨 번호와 맞춰 보는 게임이다.
6개의 번호를 A부터 E까지 5개 만들어 출력한 후,
자동 생성시킨 당첨 번호와 비교해 당첨 번호 개수와 수를 출력한다.
중복이 없는 6개의 번호는 집합으로 처리하며, 당첨 여부는 교집합을 이용하면 편리하다.

⚠ 내장 함수 map(function, iterable)

… 리스트 등 이터러블(iterable)의 모든 항목에 function을 적용한 후
그 결과를 돌려주는 이터레이터(iterator)를 반환

⚠ 쉘에서 사용하는 내장 함수 dir()

… 현재 정의된 변수와 함수 이름의 리스트를 반환

⚠ 프로젝트 lab1, lab2