Chapter 12
모듈 활용하기

목차

학습목표

- 모듈의 개념과 사용시 장점을 알아봅니다.
- 모듈을 만들고 사용하는 방법을 학습합니다.
- __name__ 전역변수에 대해서 학습합니다.
- 수학, 난수, 시간과 관련된 기본 모듈에 대해서 학습합니다

Section 01

이 장에서 만들 프로그램

이 장에서 만들 프로그램

1. 로또 당첨 게임

▪사용자가 입력한 6개의 숫자와, 컴퓨터가 랜덤하게 출력한 6개의 숫자를 서로 비교하여 일치하는 숫자를 출력하는 프로그램

```
1부터 45까지의 정수 6개를 입력하세요.
Number 1 : 44
Number 2 : 45
Number 3 : 32
Number 4 : 1
Number 5 : 19
Number 6 : 22
이번주 로또 번호 [27, 1, 22, 2, 24, 19]
내가 선택한 번호 [44, 45, 32, 1, 19, 22]
일치하는 숫자 : [1, 19, 22]
```

Section 02
모듈이란?

모듈이란?

- 모듈
  - 마트의 간편식처럼 프로그램을 개발할 때도 이미 만들어져 있는 코드를 가져다 쓸 수 있는 기능
  - 즉 모듈이란 특정 기능(함수)을 포함하고 있는 파일(xxx.py)로 다른 프로그램에 이식해서 사용하는 것을 의미함

그림 12-1 이미 만들어져 있는 간편식

모듈이란?

- 모듈
  - 함수를 학습할 때 특정 기능을 함수로 만들고 저장한 파일이 모듈임
  - 즉 기능을 모아 함수가 되고 함수가 모여 모듈이 됨
  - 모듈은 파이썬에서 기본적으로 제공하는 것도 있고, 개인 혹은 단체가 만들어서 배포하는 것도 있습니다.
    - 다른 개발자가 만든 모듈을 가져와 사용하면 내가 원하는 새로운 프로그램을 쉽게 만들 수 있음

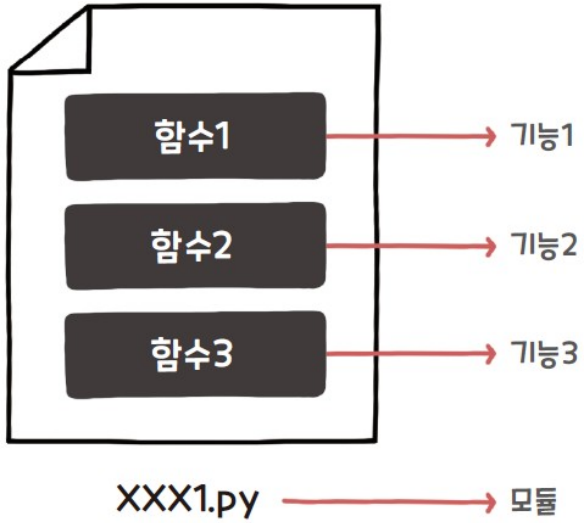


그림 12-2 함수의 모임 모듈

모듈이란?

- 모듈을 사용할 때 장점
  - 프로그램 개발 시간을 단축시킬 수 있음
  - 모듈은 이미 검증된 코드이므로 오류가 적음
  - 팀원들이 기능 구현을 분업화하고 공유할 수 있어 전체적인 작업 속도를 향상시킬 수 있음

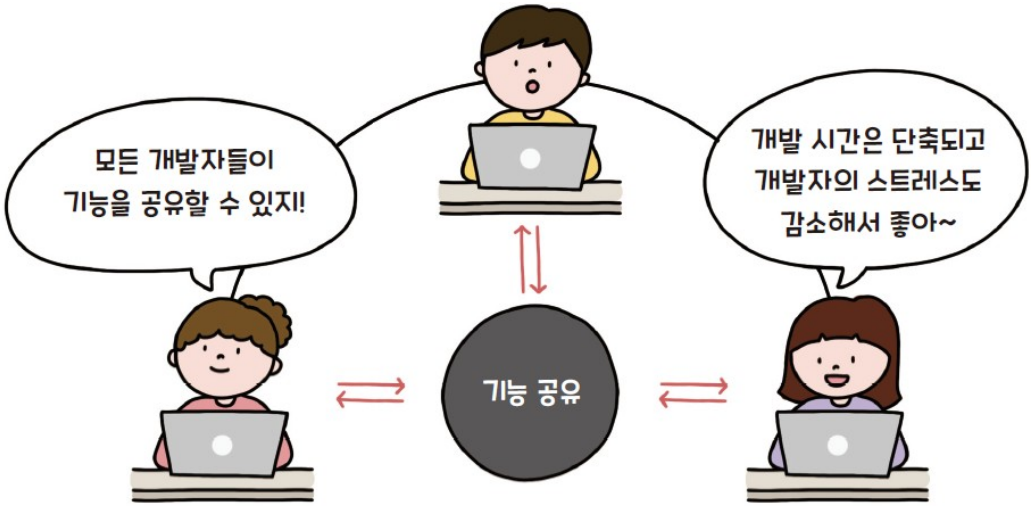


그림 12-3 모듈의 장점

확인문제

확인문제

모듈에 대한 설명으로 옳지 않은 것은 무엇인가?

① 모듈은 기능을 포함하고 있는 파이썬 파일이다.

② 모듈을 만들고 사용하려면 파이썬 재단의 인증을 받아야 한다.

③ 모듈은 누구나 만들고 사용할 수 있다.

④ 모듈을 사용하면 개발자들 사이에 기능을 공유할 수 있다.

정답

②

Section 03
모듈 제작 및 사용

모듈 제작

파이썬 모듈을 만들고 사용하기

① [ex01] 폴더를 생성한 후 'calculator.py' 파일을 만들고 코딩하기

코드 12-1 모듈 파일 — ex01\calculator.py

```
01  def addition(n1, n2):
02      print('덧셈 연산 결과 : ', n1 + n2)
03
04  def subtraction(n1, n2):
05      print('뺄셈 연산 결과 : ', n1 - n2)
06
07  def multiplication(n1, n2):
08      print('곱셈 연산 결과 : ', n1 * n2)
09
10  def division(n1, n2):
11      print('나눗셈 연산 결과 : ', n1 / n2)
12
13  def rest(n1, n2):
14      print('나머지 연산 결과 : ', n1 % n2)
15
16  def portion(n1, n2):
17      print('몫 연산 결과 : ', n1 // n2)
```

모듈 제작

- 파이썬 모듈을 만들고 사용하기

① [ex01] 폴더를 생성한 후 'calculator.py' 파일을 만들고 코딩하기

▪'calculator.py'는 산술 연산 기능을 각각의 함수로 정의한 파일로, 이것이 바로 모듈 calculator임

→ 즉 **파일의 이름이 곧 모듈의 이름**임

▪모듈 제작은 파이썬 파일을 만들면 끝임

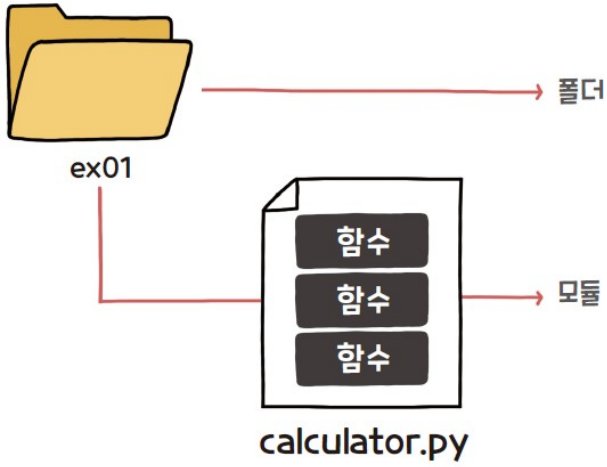


그림 12-4 [ex01] 폴더에 만들어진 calculator 모듈

모듈 사용

- 파이썬 모듈을 만들고 사용하기

② calculator 모듈이 존재하는 [ex01] 폴더에 'ch12_01.py' 파일을 코딩하기

- 'ch12_01.py' 파일은 'calculator.py'의 모든 기능을 마치 자기 것처럼 사용함

코드 12-2 실행 파일 — ex01\ch12_01.py

```
01  import calculator
02
03  calculator.addition(1, 2)
04  calculator.subtraction(1, 2)
05  calculator.multiplication(1, 2)
06  calculator.division(1, 2)
07  calculator.rest(1, 2)
08  calculator.portion(1, 2)
```

01 행: calculator 모듈을 가져온다.

03~08 행: calculator 모듈의 기능을 사용한다.

```
덧셈 연산 결과 :  3
뺄셈 연산 결과 :  -1
곱셈 연산 결과 :  2
나눗셈 연산 결과 :  0.5
나머지 연산 결과 :  1
몫 연산 결과 :  0
```

모듈 사용

■ 파이썬 모듈을 만들고 사용하기

② calculator 모듈이 존재하는 [ex01] 폴더에 'ch12_01.py' 파일을 코딩하기

▪모듈 가져오기(**1**행)

- import 키워드를 이용하여 calculator 모듈을 가지고 오는 구문
- 이때 import 뒤에 나오는 모듈의 이름은 파이썬 파일명에서 확장자(.py)를 제거하고 작성함(import calculator → ◯ / import calculator.py → ×).

▪모듈 사용하기(**3**~**8**행)

- 불러온 모듈의 기능을 사용할 때는 '모듈명.기능(함수)' 형태로 작성함
- 모듈명과 기능 사이에 도트(.) 접근자를 붙임

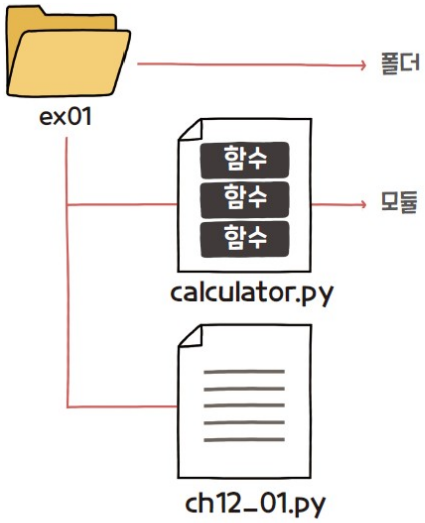


그림 12-5 calculator 모듈과 동일한 경로에 저장된 'ch12_01.py' 파일

모듈 사용

- 프로그램을 개발할 때 종종 사용하는 단위 환산 모듈 제작

코드 12-3 모듈 파일 ex02\convertUnitModule.py

```
01  def convertUnit(lenMm):
02
03      unitDic = {}
04
05      unitDic['cm'] = lenMm * 0.1
06      unitDic['m'] = lenMm * 0.001
07      unitDic['inch'] = lenMm * 0.03937
08      unitDic['ft'] = lenMm * 0.003281
09
10      return unitDic
11
12  def printLength(originalData, lengths):
13      for len in lengths.keys():
14          print(originalData, 'mm --> ', lengths[len], len)
```

```
길이(mm)를 입력하세요. 2
2 mm -->  0.2 cm
2 mm -->  0.002 m
2 mm -->  0.07874 inch
2 mm -->  0.006562 ft
```

코드 12-4 실행 파일 ex02\ch12_02.py

```
01  import convertUnitModule
02
03  inputData = int(input('길이(mm)를 입력하세요. '))
04
05  result = convertUnitModule.convertUnit(inputData)
06  convertUnitModule.printLength(inputData, result)
```

문제 해결

문제 해결 12-1 시험 통과 여부 출력하기

사용자가 3과목의 시험 점수를 입력하면 총점, 평균, 통과 여부를 출력하는 프로그램을 모듈을 이용해서 만들어봅시다. 시험 통과 여부는 평균이 60이상이고 한 과목도 40미만이 없다면 'Pass', 그렇지 않으면 'Fail'로 판정합니다.

모듈 파일 examCalcu\examCalculator.py

```
01  def getTotalScore(s1, s2, s3):
02      return s1 + s2 + s3
03
04  def getAverageScore(s1, s2, s3):
05      return getTotalScore(s1, s2, s3) / 3
06
07  def getPassFail(s1, s2, s3):
08
09      if (getAverageScore(s1, s2, s3) >= 60):
10          if(s1 >= 40 and s2 >= 40 and s3 >= 40):
11              return 'Pass'
12
13      return 'Fail'
```

문제 해결

실행 파일 examCalcu\ch12_sol_01.py

```
01  import examCalculator
02
03  score1 = int(input('1과목 점수를 입력하세요. '))
04  score2 = int(input('2과목 점수를 입력하세요. '))
05  score3 = int(input('3과목 점수를 입력하세요. '))
06
07  print('총점 : ', examCalculator.getTotalScore(score1, score2, score3))
08  print('평균 : ', examCalculator.getAverageScore(score1, score2, score3))
09  print('합격여부 : ', examCalculator.getPassFail(score1, score2, score3))
```

```
1과목 점수를 입력하세요. 65
2과목 점수를 입력하세요. 39
3과목 점수를 입력하세요. 90
총점 :  194
평균 :  64.66666666666667
합격여부 :  Fail
```

확인문제

확인문제

다음은 사용자가 정수 2개를 입력하면, 사칙연산(덧셈, 뺄셈, 곱셈, 나눗셈) 결과 값을 리스트 타입으로 반환하는 모듈이다. 빈칸을 채우시오.

모듈 파일

```
def calculator(n1, n2):
    resultList = []

    resultList.append(n1 + n2)
    resultList.append(n1 - n2)
    resultList.append(n1 * n2)
    resultList.append(n1 / n2)

    return resultList
```

실행 파일

```
import [          ]

inputNum1 = int(input('첫 번째 정수를 입력하시오. '))
inputNum2 = int(input('두 번째 정수를 입력하시오. '))
result = forCalculator.[          ](inputNum1, inputNum2)
print(result)
```

```
첫 번째 정수를 입력하시오. 10
두 번째 정수를 입력하시오. 20
[30, -10, 200, 0.5]
```

정답

forCalculataor, calculator

Section 04
슬기로운 모듈 활용법

as 키워드

- as 키워드
  - 모듈명이 길면 사용할 때마다 코드가 길어지는 불편함이 있음
  - 예를 들어 calaulator 모듈의 함수를 불러오기 위해서는 매번 앞에 calculator를 붙여 사용해야 함

```
calculator.addition()
calculator.subtraction()
calculator.multiplication()
calculator.division()
calculator.rest()
calculator.portion()
```

모듈명을 매번 중복해서 앞에 붙여야 한다.

그림 12-6 긴 모듈명 사용에 따른 번거로움

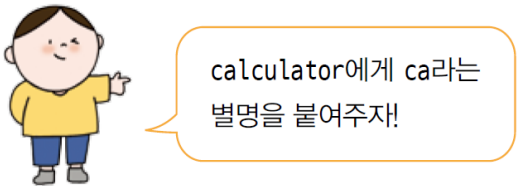
as 키워드

- as 키워드
  - 'as' 키워드를 이용하면 모듈의 이름을 다른 이름으로 치환할 수 있음
  - 'calculator'를 'ca'로 치환하면면 기존의 'calculator' 대신 'ca'를 이용할 수 있음
  - [코드 12-2]를 개선한 코드

코드 12-5 실행 파일 ex01\ch12_03.py

```
01  import calculator as ca    # calculator를 ca로 치환한다.
02
03  ca.addition(1, 2)
04  ca.subtraction(1, 2)
05  ca.multiplication(1, 2)
06  ca.division(1, 2)
07  ca.rest(1, 2)
08  ca.portion(1, 2)
```

치환된 ca를 이용해 calculator의 모듈을 불러온다.



as 키워드

- [코드 12-3]의 convertUnitModule 모듈을 cu으로 치환하여 개선한 프로그램

코드 12-6 실행 파일 ex02\ch12_04.py

```
01  import convertUnitModule as cu      # convertUnitModule을 cu로 치환한다.
02
03  inputData = int(input('길이(mm)를 입력하세요. '))
04
05  result = cu.convertUnit(inputData)
06  cu.printLength(inputData, result)
```

문제 해결

문제 해결 12-2 원의 둘레와 넓이 구하기

원의 둘레와 넓이를 구하는 모듈을 만들고, 모듈을 불러올 때 as 키워드를 사용하여 반지름이 6인 원의 둘레와 넓이를 구해봅시다.

모듈 파일 calcCircle\circleMod.py

```
01  rate = 3.14
02
03  def getCircleLength(radius):
04      return 2 * rate * radius        # 원의 둘레 반환
05
06  def getCircleArea(radius):
07      return rate * (radius ** 2)     # 원의 넓이 반환
```

실행 파일 calcCircle\ch12_sol_02.py

```
01  import circleMod as c
02
03  print('원 둘레: ', c.getCircleLength(6))
04  print('원 넓이: ', c.getCircleArea(6))
```

```
원 둘레:  37.68
원 넓이:  113.04
```

from 키워드

- from 키워드
  - 모듈은 자칫 잘못하면 필요하지 않은 기능까지 가져와야 하는 비효율적인 상황이 발생할 수 있음
  - 예를 들어 calculator 모듈의 기능 중 대학생을 위해서는 모든 기능(덧셈, 뺄셈, 곱셈, 나눗셈, 나머지, 몫)이 필요하지만, 유치원생에게는 덧셈, 뺄셈만 필요함

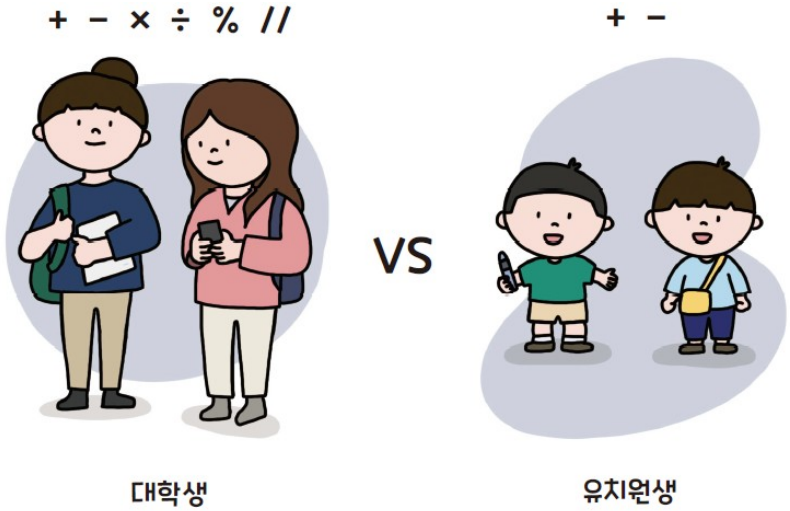


그림 12-7 상황에 따라 다른 요구사항

from 키워드

- from 키워드
  - 모듈의 전체 기능 중에서 일부 기능만 import할 수 있음
  - calculator 모듈에서 필요한 더하기, 빼기 기능만 가지고 오는 프로그램

코드 12-7 실행 파일 ex01\ch12_05.py

```
01  from calculator import addition        # calculator의 addition만 가져온다.
02  from calculator import subtraction     # calculator의 subtraction만 가져온다.
03
04  addition(1, 2)
05  subtraction(1, 2)
```

```
덧셈 연산 결과:  3
뺄셈 연산 결과:  -1
```

from 키워드

하나 더 알기 **만약 import 하지 않은 기능을 사용하면 어떻게 되나요?**

[코드 12-7]에서 addition과 subtraction 기능만 import 했기 때문에 multiplication() 또는 division()의 함수를 사용하면 에러가 발생합니다. 다음 [코드 12-8]은 multiplication()을 사용했을 때 발생하는 에러를 보여줍니다.

코드 12-8 실행 파일 ex01\ch12_06.py

```
01  from calculator import addition
02  from calculator import subtraction
03
04  addition(1, 2)
05  subtraction(1, 2)
06  multiplication(1, 2)
```

import하지 않은 기능을 호출해서 에러가 발생한다.

```
덧셈 연산 결과 :  3
뺄셈 연산 결과 :  -1
Traceback (most recent call last):
NameError: name 'multiplication' is not defined
```

콤마(,) 에스터리크(*)를 이용한 import 구문

- 콤마(,)
  - 모듈을 import할 때 콤마(,)를 이용해서 기능을 나열할 수 있음

```
from calculator import addition
from calculator import subtraction
```

→

```
from calculator import addition, subtraction
```

before　　after

- 에스터리크(*)
  - 모듈의 모든 기능을 가져 올 때는 에스터리크(*)를 이용함

```
from calculator import addition
from calculator import subtraction
from calculator import multiplication
from calculator import division
from calculator import rest
from calculator import portion
```

calculator에서 각각의 기능을 가져온다.

→

```
from calculator import *
```

calculator 모듈의
모든 기능을 가져온다.

before　　after

Section 05
실행(메인) 파일 지정하기

자동으로 실행되는 모듈 함수

- 여러 개의 모듈을 사용하는 경우 주의할 점
  - 만약 모듈에 함수를 호출하는 코드가 있다면 자동으로 함수가 실행됨

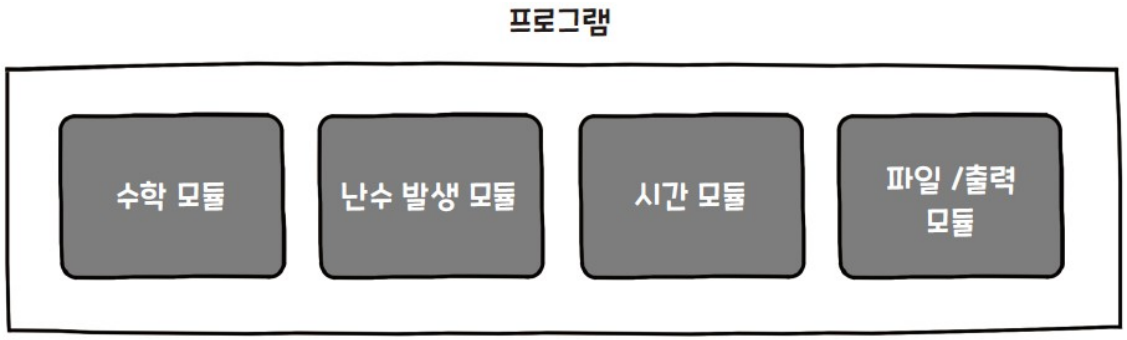


그림 12-8 여러 모듈을 사용한 프로그램

자동으로 실행되는 모듈 함수

- 1단계: 모듈 파일 생성하기
  - [ex03] 폴더를 생성하고, 폴더 내부에 'module01.py', 'module02.py', 'module03.py' 파일을 만들고 코딩함

코드 12-9 모듈 파일 — ex03\module01.py

```
01  def fun():
02      print('module01의 함수가 실행됩니다.')
03
04  fun()
```

코드 12-10 모듈 파일 — ex03\module02.py

```
01  def fun():
02      print('module02의 함수가 실행됩니다.')
03
04  fun()
```

코드 12-11 모듈 파일 — ex03\module03.py

```
01  def fun():
02      print('module03의 함수가 실행됩니다.')
03
04  fun()
```

자동으로 실행되는 모듈 함수

- 2단계: 실행 파일 생성하기
  - [ex03] 폴더에 'ch12_07.py' 파일을 만들고 다음과 같이 코딩함

코드 12-12 실행 파일 — ex03\ch12_07.py

```
01   import module01
02   import module02
03   import module03
04
05   print('실행 파일입니다.')
```

- 3단계: 출력 결과 확인하기
  - 'ch12_07.py' 파일을 실행하고 출력 결과를 확인함

```
module01의 함수가 실행됩니다.
module02의 함수가 실행됩니다.
module03의 함수가 실행됩니다.
실행 파일입니다.
```

자동으로 실행되는 모듈 함수

- 호출하지 않은 모듈의 함수가 실행됨
  - 실행한 파일은 ‘ch12_07.py’로 모듈을 import하고 ‘실행 파일입니다.’ 문자열을 출력하는 코드가 전부임
  - 다시 말해 모듈을 import만 했지 모듈의 함수를 호출하지는 않았음
  - 그런데 모듈의 함수인 fun()이 모두 실행되는 결과가 나옴
  - 실행 결과에는 오류가 없더라도 원래 의도와는 전혀 다르게 프로그램이 실행됨

자동으로 실행되는 모듈 함수

- 모듈의 함수가 자동으로 실행되지 않는 방법
  - 프로그램을 실행하는 '실행 파일'과 '모듈 파일'을 구분하여 모듈의 함수가 자동으로 실행되지 않도록 함

- 실행 파일과 모듈 파일을 구분하는 방법
  - __name__ 변수를 이용함

표 12-1 실행 파일과 모듈 파일의 구분

구분	파일 이름	내용
프로그램 실행 파일	ch12_07.py	모듈을 import 한다.
모듈 파일	module01.py module02.py module03.py	실행 파일에서 함수를 호출하기 전에는 실행되지 않는다.

__name__ 전역변수

- __name__ 전역변수
  - 파이썬에는 기본적으로 __name__이라는 전역변수에 파일 이름이 저장됨
  - 'module01.py', 'module02.py', 'module03.py' 파일의 마지막에 다음 코드를 한 줄 추가

```
print('__name__ : ', __name__)   # __name__ 전역변수 출력
```

  - 코드 추가가 끝났다면 'ch12_07.py'을 다시 실행하고 결과를 확인함

```
module01의 함수가 실행됩니다.
__name__ :  module01 ●———————— __name__에 파일 이름(module01)이 저장되어 있다.
module02의 함수가 실행됩니다.
__name__ :  module02 ●———————— __name__에 파일 이름(module02)이 저장되어 있다.
module03의 함수가 실행됩니다.
__name__ :  module03 ●———————— __name__에 파일 이름(module03)이 저장되어 있다.
실행 파일입니다.
```

__name__ 전역변수

- __name__ 전역변수
  - 이번에는 'ch12_07.py'에도 __name__을 출력하는 코드를 추가하고 다시 실행

```
module01의 함수가 실행됩니다.
__name__ :  module01
module02의 함수가 실행됩니다.
__name__ :  module02
module03의 함수가 실행됩니다.
__name__ :  module03
실행 파일입니다.
__name__ :  __main__ ●———————— __name__에 '__main__'이 저장되어 있다.
```

  - 'ch12_07.py'의 __name__에 저장된 문자열은 다른 파일의 __name__에 저장된 것과 달리 __main__이 저장되어 있음

__name__ 전역변수

- 각 파일의 __name__에 저장된 문자열

표 12-2 __name__에 저장된 문자열

구분	파일 이름	__name__에 저장된 문자열
프로그램 실행 파일	ch12_07.py	__main__
모듈 파일	module01.py	module01
	module02.py	module02
	module03.py	module03

- 이처럼 프로그램이 실행될 때 최초 실행되는 메인 파일의 __name__에 저장된 '__main__'을 이용하여 메인 파일을 구분할 수 있음

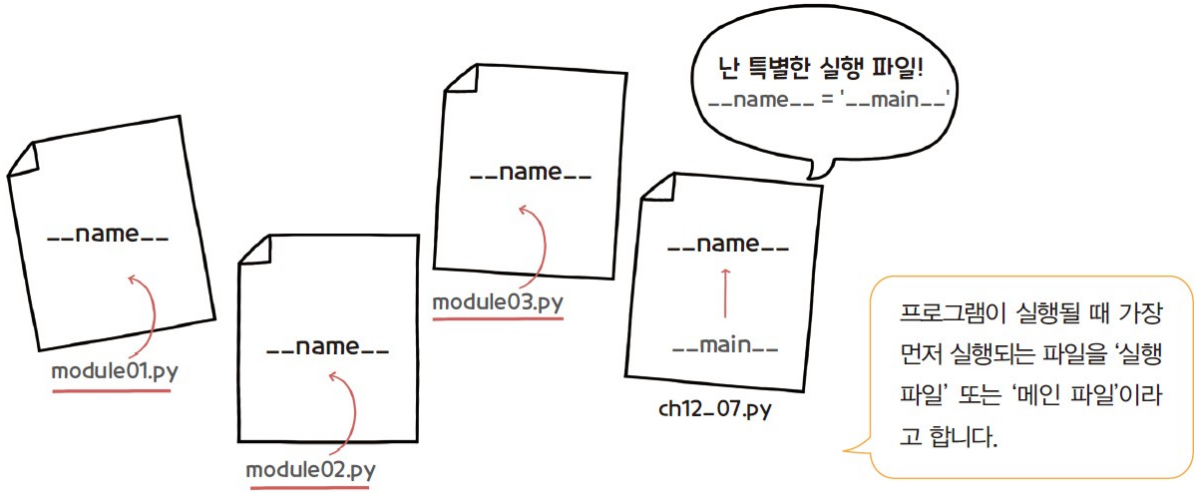


그림 12-9 __name__을 이용한 메인 파일과 모듈 파일 구분

__name__ 전역변수

- [코드 12-9]~[코드 12-12]를 메인 파일이 아니면 자동으로 실행되지 않도록 수정

코드 12-13 모듈 파일 ex03\module01_edit.py

```
01  def fun():
02      print('module01 함수가 실행됩니다.')
03
04  if __name__ == '__main__':              # 메인 파일이면 실행한다.
05      fun()
06      print('__name__ : ', __name__)
```

코드 12-14 모듈 파일 ex03\module02_edit.py

```
01  def fun():
02      print('module02 함수가 실행됩니다.')
03
04  if __name__ == '__main__':              # 메인 파일이면 실행한다.
05      fun()
06      print('__name__ : ', __name__)
```

코드 12-15 모듈 파일 ex03\module03_edit.py

```
01  def fun():
02      print('module03 함수가 실행됩니다.')
03
04  if __name__ == '__main__':              # 메인 파일이면 실행한다.
05      fun()
06      print('__name__ : ', __name__)
```

__name__ 전역변수

- [코드 12-9]~[코드 12-12]를 메인 파일이 아니면 자동으로 실행되지 않도록 수정

코드 12-16 실행 파일 ex03\ch12_08.py

```
01   import module01_edit
02   import module02_edit
03   import module03_edit
04
05   if __name__ == '__main__':              # 메인 파일이면 실행한다.
06       print('실행 파일입니다.')
07       print('__name__ : ', __name__)
```

```
실행 파일입니다.
__name__ :  __main__
```

문제 해결

문제 해결 12-3 할인율 적용 프로그램

한빛 마트에서 가을맞이 할인 행사를 진행하기로 했습니다. 상품을 많이 구입할수록 많이 할인 받도록 구입 개수에 따라 할인율이 달라지는 모듈을 만들어봅시다.

상품 구입 개수	1개	2개	3개	4개 이상
할인율	5%	10%	20%	30%

```
상품을 구매 하시겠습니까? 1.구매, 2.종료 1
구매한 상품의 금액을 입력하세요. 1500
상품을 구매 하시겠습니까? 1.구매, 2.종료 1
구매한 상품의 금액을 입력하세요. 2000
상품을 구매 하시겠습니까? 1.구매, 2.종료 1
구매한 상품의 금액을 입력하세요. 500
상품을 구매 하시겠습니까? 1.구매, 2.종료 2
할인율 :  20 %
총합계 :  3200.0 원
```

```
상품을 구매 하시겠습니까? 1.구매, 2.종료 1
구매한 상품의 금액을 입력하세요. 2000
상품을 구매 하시겠습니까? 1.구매, 2.종료 1
구매한 상품의 금액을 입력하세요. 3000
상품을 구매 하시겠습니까? 1.구매, 2.종료 2
할인율 :  10 %
총합계 :  4500.0 원
```

문제 해결

모듈 파일 dcGoods\dcGoods.py

```
01  def calcTotalPrice(gps):
02      dcRate = 0;
03      totalPrice = 0
04
05      if len(gps) == 1:
06          dcRate = 5;
07      elif len(gps) == 2:
08          dcRate = 10;
09      elif len(gps) == 3:
10          dcRate = 20;
11      elif len(gps) >= 4:
12          dcRate = 30;
13
14      for item in gps:
15          totalPrice += item * (1 -  dcRate / 100)
16
17      return [dcRate, totalPrice]
```

문제 해결

실행 파일 dcGoods\ch12_sol_03.py

```
01  import dcGoods as dg
02
03  if __name__ == '__main__':
04      flag = True
05      goodPrices = []
06
07      while flag:
08
09          purchase = int(input('상품을 구매 하시겠습니까? 1.구매, 2.종료 '))
10
11          if purchase == 1:
12              price = int(input('구매한 상품의 금액을 입력하세요. '))
13              goodPrices.append(price)
14          elif purchase == 2:
15              result = dg.calcTotalPrice(goodPrices)
16              flag = False
17
18      print('할인율 : ', result[0], '%')
19      print('총합계 : ', result[1], '원')
```

Section 06
자주 사용하는 외부 모듈

파이썬에서 제공하는 모듈

- math 모듈
  - 수학 함수와 관련된 모듈
  - 사칙연산보다 어려운 수식이 필요할 때 사용함

표 12-3 자주 쓰이는 math 모듈의 함수

함수	내용
fabs()	절댓값을 반환
ceil()	소수점 이하 올림한 값을 반환
floor()	소수점 이하 내림한 값을 반환
trunc()	소수점 이하 버림한 값을 반환
gcd()	최대공약수를 반환
factorial()	팩토리얼한 값을 반환
sqrt()	제곱근을 반환

파이썬에서 제공하는 모듈

하나 더 알기 **파이썬 내장 함수 중 수학과 관련된 함수**

math 모듈은 아니지만 파이썬 내장 함수 중에서 유용한 수학 관련 함수도 있습니다.

표 12-4 수학 관련 내장 함수

함수	내용
sum()	전체 합을 반환
max()	최댓값을 반환
min()	최솟값을 반환
abs()	절댓값을 반환
pow()	거듭제곱을 반환
round()	반올림한 값을 반환

파이썬에서 제공하는 모듈

- random 모듈
  - 난수(random number)를 발생시키는 모듈
  - 복권 추첨, 주사위 던지기와 같이 임의의 숫자가 필요한 경우에 사용

표 12-5 자주 쓰이는 ramdom 모듈의 함수

함수	내용
random()	0이상 1미만의 난수 발생
randint(n1, n2)	n1이상 n2이하의 난수 발생
randrange(n1, n2)	n1이상 n2미만의 난수 발생
sample(range(n1, n2), n3)	n1이상 n2미만의 난수 n3개 발생
choice()	무작위로 1개 아이템 선택함
shuffle()	아이템 순서 섞음

파이썬에서 제공하는 모듈

코드 12-18 ch12_10.py

```
01  import random
02
03  print('random : ', random.random())                 # 0이상 1미만 난수
04  print('randint : ', random.randint(0, 9))           # 0이상 9이하 난수
05  print('randrange : ', random.randrange(0, 9))       # 0이상 9미만 난수
06  print('sample : ', random.sample(range(0, 10), 3))  # 0이상 10미만 난수 3개
07
08  listVar = [1, 2, 3, 4, 5, 6]
09
10  # 무작위로 1개 아이템 선택
11  print('choice : ', random.choice(listVar))
12  print('choice : ', random.choice(listVar))
13
14  # 아이템 순서 섞음
15  random.shuffle(listVar)
16  print('shuffle : ', listVar)
17  random.shuffle(listVar)
18  print('shuffle : ', listVar)
```

```
random :  0.39593355510791717
randint :  3
randrange :  3
sample :  [1, 2, 6]
choice :  2
choice :  3
shuffle :  [1, 6, 5, 4, 3, 2]
shuffle :  [3, 2, 6, 1, 5, 4]
```

말 그대로 난수가 나오기 때문에
실행할 때마다 결과는 달라집니다.

파이썬에서 제공하는 모듈

- time 모듈
  - 시간 관련 모듈
  - 이 모듈은 UTC(Coordinated Universal Time)에서 정한 세계 표준시인 1970년 1월 1일 0시 0분 0초부터 초단위로 측정하는 것을 기준으로함
  - 예전에는 그리니치 표준시(GMT)로 알려졌기 때문에 UTC와 GMT는 같은 시간을 뜻함

표 12-6 자주 쓰이는 time 모듈의 함수

함수	내용
localtime()	시스템의 현재 시간을 반환
gmtime()	GMT 시간을 반환
strftime()	시간 포맷 코드에 따른 출력

파이썬에서 제공하는 모듈

- localtime() 함수를 이용하여 현재 컴퓨터에서 받아온 시간을 보여줌

코드 12-19 ch12_11.py

```
01  import time
02
03  lt = time.localtime()                # 시스템 시간
04  print('localTime : ', lt)
05
06  # 상수를 이용한 시스템 시간 출력
07  print('tm_year : ', lt.tm_year)      ← 받아온 시스템 시간에서 년도만 뽑아낸다.
08  print('tm_mon : ', lt.tm_mon)        # 월
09  print('tm_mday : ', lt.tm_mday)      # 일
10  print('tm_hour : ', lt.tm_hour)      # 시
11  print('tm_min : ', lt.tm_min)        # 분
12  print('tm_sec : ', lt.tm_sec)        # 초
13  print('tm_wday : ', lt.tm_wday)      # 요일
```

```
localTime :  time.struct_time(tm_year=2020, tm_mon=8, tm_mday=30, tm_hour=9, tm_min=21,
tm_sec=10, tm_wday=6, tm_yday=243, tm_isdst=0)
tm_year :  2020
tm_mon :  8
tm_mday :  30
tm_hour :  9
tm_min :  21
tm_sec :  10
tm_wday :  6
```

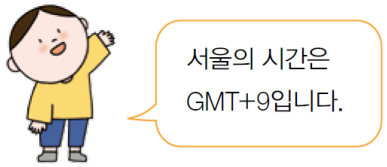
파이썬에서 제공하는 모듈

- 세계 표준시인 GMT 시간을 반환하는 gmtime() 함수를 이용한 코드

코드 12-20 ch12_12.py

```
01  import time
02
03  gt = time.gmtime()
04  print('time.gmtime() : ', gt)        # GMT 시간
05
06  # 상수를 이용한 GMT 시간 출력
07  print('tm_year : ', gt.tm_year)
08  print('tm_mon : ', gt.tm_mon)        # 월
09  print('tm_mday : ', gt.tm_mday)      # 일
10  print('tm_hour : ', gt.tm_hour)      # 시
11  print('tm_min : ', gt.tm_min)        # 분
12  print('tm_sec : ', gt.tm_sec)        # 초
13  print('tm_wday : ', gt.tm_wday)      # 요일
```

받아온 GMT 시간에서 년도만 뽑아낸다.




```
time.gmtime() : time.struct_time(tm_year=2020, tm_mon=8, tm_mday=30, tm_hour=0, tm_
min=21, tm_sec=10, tm_wday=6, tm_yday=243, tm_isdst=0)
tm_year :  2020
tm_mon :  8
tm_mday :  30
tm_hour :  0
tm_min :  21
tm_sec :  10
tm_wday :  6
```

파이썬에서 제공하는 모듈

- strftime() 함수를 이용하여 시간을 표현한 코드
  - strftime()은 gmtime()이나 localtime()에 의해 나온 튜플이나 struct_time을 지정된 포맷을 통해 문자열로 변환함
  - 포맷은 소문자와 대문자의 뜻이 다르므로 구분하여 사용해야 함

코드 12-21 ch12_13.py

```
01  import time
02
03  print(time.strftime('%H:%M:%S', time.localtime()))
04  print(time.strftime('%Y:%m:%d:%H:%M:%S', time.localtime()))
05  print(time.strftime('%H:%M:%S', time.gmtime()))
06  print(time.strftime('%Y:%m:%d:%H:%M:%S', time.gmtime()))
```

```
09:21:10
2020:08:30:09:21:10
00:21:10
2020:08:30:00:21:10
```

확인문제

확인문제

1. math 모듈을 이용해서 15과 21의 최대공약수를 출력하는 프로그램이다. 빈칸에 들어갈 함수는 무엇인가?

```
import math

result = math.[ ](15, 21)
print('최대공약수 : ', result)
```

① fabs()　② ceil()　③ gcd()　④ sqrt()

2. random 모듈을 이용해서 1부터 100까지의 정수 중 난수 5개를 발생시키는 프로그램이다. 빈칸에 들어갈 함수의 기본형은 무엇인가?

```
import random

print('난수 : ', random.[ ](range(0, 101), 5))
```

① random()　② sample(range(n1, n2), n3)
③ shuffle()　④ randint(n1, n2)

3. 다음은 time 모듈을 이용해 GMT시간을 출력하는 프로그램이다. 실행 결과를 보고 빈칸을 채우시오.

```
import time

print(time.strftime('%[ ]년 %[ ]월 %[ ]일 %[ ]시 %[ ]분 %[ ]초', time.gmtime()))
```

```
2020년 08월 30일 21시 11분 46초
```

정답

1. ③　2. ②　3. Y, m, d, H, M, S

실전 예제

실전 예제1 – 로또 당첨 게임

문제

사용자가 1부터 45까지의 정수 6개를 입력하고, 컴퓨터에서는 1부터 45까지의 난수 6개를 발생합니다. 사용자 정수와 컴퓨터 정수를 비교해서 일치하는 숫자를 출력하는 프로그램을 만들어봅시다.


```
1부터 45까지의 정수 6개를 입력하세요.
Number 1 : 44
Number 2 : 45
Number 3 : 32
Number 4 : 1
Number 5 : 19
Number 6 : 22
이번주 로또 번호 [27, 1, 22, 2, 24, 19]
내가 선택한 번호 [44, 45, 32, 1, 19, 22]
일치하는 숫자 : [1, 19, 22]
```

실전 예제1 – 로또 당첨 게임

해결

모듈 파일 lottoMahcine.py

```
01  import random
02
03  userNums = []
04  randNums = []
05  collect = []
06
07  def setUNumbers(ns):
08      global userNums
09      userNums = ns
10
11  def getUNumbers():
12      return userNums
13
14  def setRNumbers():
15      global randNums
16
17      randNums = random.sample(range(1, 46), 6)
18
19  def getRNumbers():
20      return randNums
21
22  def compareNumbers():
23      global userNums
24      global randNums
25      global collect
26
27      collect = []
28      for item in userNums:
29          if randNums.count(item) != 0:
30              collect.append(item)
31
32      return collect
```

실전 예제1 – 로또 당첨 게임

실행 파일 ch12_appEx_01.py

```
01  import lottoMachine as lm
02
03  nums = []
04
05  print('1부터 45까지의 정수 6개를 입력하세요. ')
06  nums.append(int(input('Number 1 : ')))
07  nums.append(int(input('Number 2 : ')))
08  nums.append(int(input('Number 3 : ')))
09  nums.append(int(input('Number 4 : ')))
10  nums.append(int(input('Number 5 : ')))
11  nums.append(int(input('Number 6 : ')))
12
13  lm.setUNumbers(nums)       # 사용자 번호 설정
14  lm.setRNumbers()           # 렌덤 번호 설정
15
16  print('이번주 로또 번호', lm.getRNumbers())
17  print('내가 선택한 번호', lm.getUNumbers())
18  print('일치하는 숫자 :', lm.compareNumbers())
```

Thank you!

한빛아카데미 Hanbit Academy, Inc.