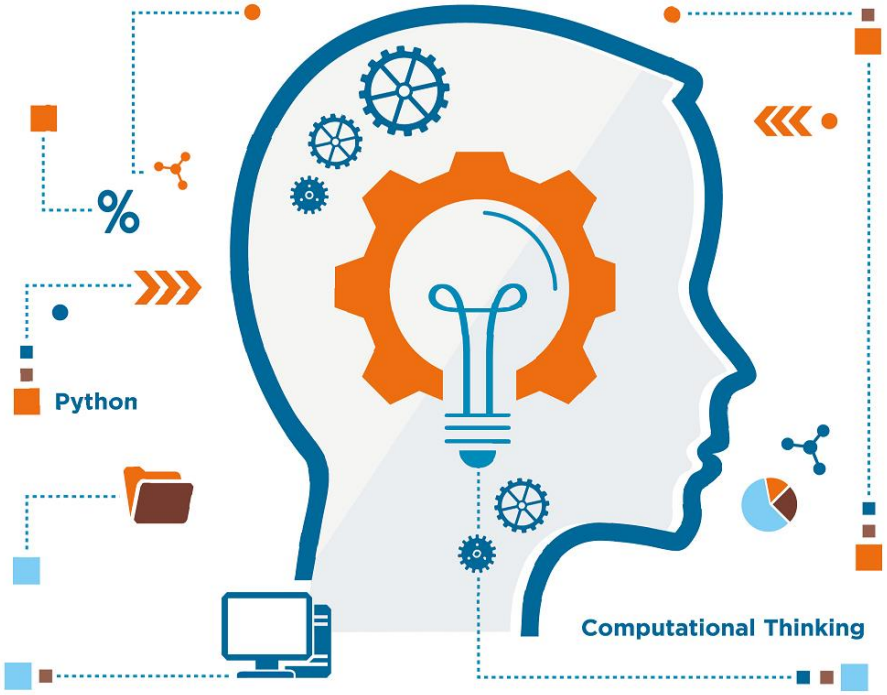


[Python]

Python으로 배우는

소프트웨어 원리

Chapter 12. 모듈

목차

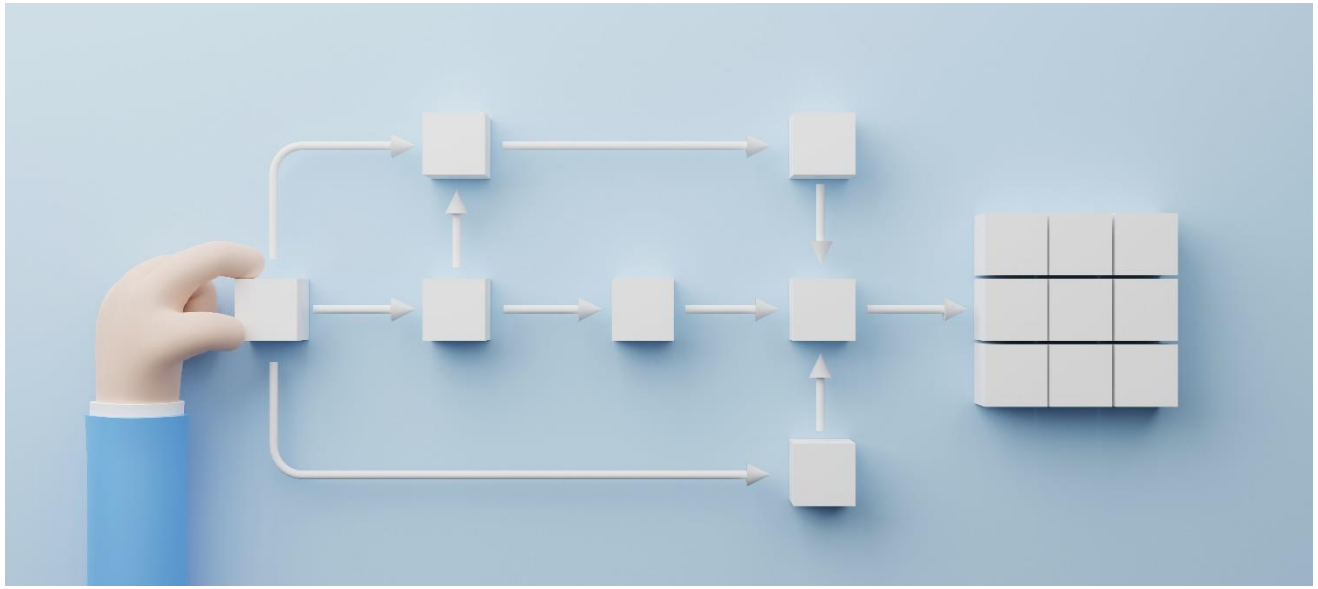
01

재사용과 모듈

01. 재사용과 모듈

[재사용(reuse)]

- 재사용은 다른 사람이 이미 만들어 놓은 코드를 가져와 사용하는 방법이다.
- 새로 만들지 않고 기존 코드를 활용하기에 효율이 좋다.



01. 재사용과 모듈

I. 재사용의 필요성

- 개발 시간이나 비용을 절감하기 위해 기존의 시스템에서 검증된 기능을 재구성하여 또 다른 시스템을 구축하는 작업

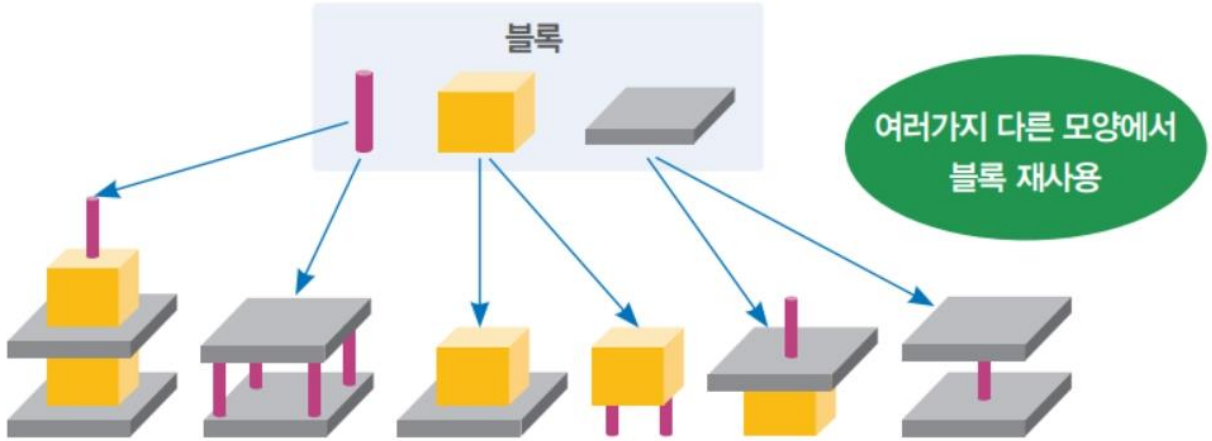


그림 12-1 재사용의 개념

01. 재사용과 모듈

II. 모듈의 개념

- 프로그램을 만들 때 자주 사용하는 코드를 별도의 파일로 만들어서 필요할 때마다 재사용할 수 있게 함

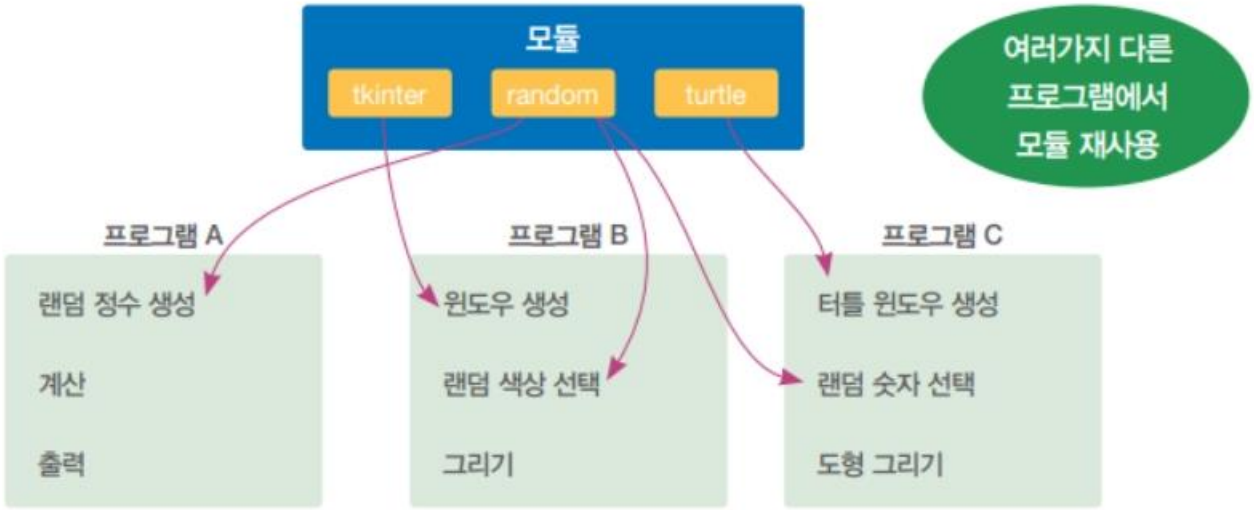


그림 12-2 모듈의 개념

II. 모듈의 개념

- **모듈의 장점**
  - 자주 사용되는 기능을 **재사용**할 수 있어서 프로그램 개발이 효율적
  - **기능의 분리**와 **복잡성의 감소**로 프로그램의 유지보수가 용이
  - 필요한 부분만 불러 사용할 수 있어서 메모리 사용을 절약
  - 오류가 발생하는 경우 파급 효과를 최소화

01. 재사용과 모듈

II. 모듈의 개념

▪ 함수와 모듈의 구분

표 12-1 함수의 구분

구분		설명	종류 및 사용 예
함수	내장 함수	파이썬에서 제공하는 함수	input(), print(), int(), type(), range(), len(), max(), pow() 등
	사용자 정의 함수	사용자가 직접 만드는 함수	def 함수명 : 명령문
	메소드	객체 이름으로 사용하는 함수	list.sort(), tuple.index(), str.split(), turtle.forward() 등

표 12-2 모듈의 구분

구분		설명	종류 및 사용 예
모듈	표준 모듈	파이썬에서 제공하는 모듈	import random random.randint(1,50)
	사용자 정의 모듈	사용자가 직접 만드는 모듈	모듈명.py 파일 작성, import 모듈명

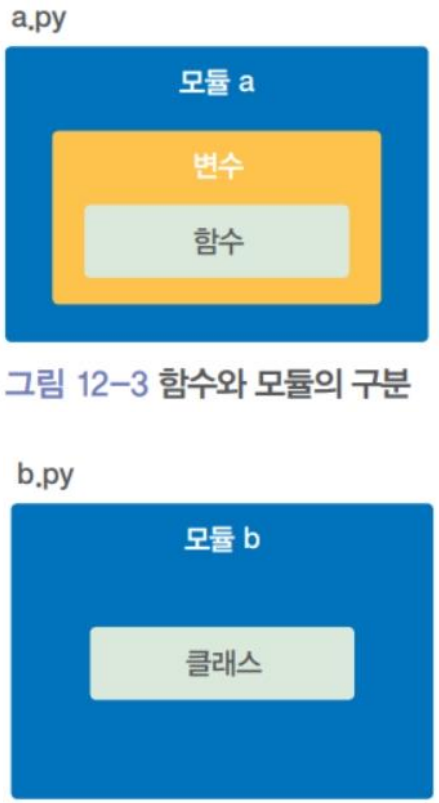


그림 12-3 함수와 모듈의 구분

02

표준 모듈

02. 표준 모듈

I. sys 모듈

실습 12-1 sys 모듈 다양하게 사용하기

① sys 모듈을 import하고, 파이썬에서 제공하는 표준 모듈의 종류를 확인

```
>>> import sys
>>> print(sys.builtin_module_names)
('_abc', '_ast', '_bisect', '_blake2', '_codecs', '_codecs_cn', '_codecs_hk', '_codecs_
iso2022', '_codecs_jp', '_codecs_kr', '_codecs_tw', '_collections', '_contextvars', '_csv',
'_datetime', '_functools', '_heapq', '_imp', '_io', '_json', '_locale', '_lsprof', '_md5', '_
```

② sys 모듈에서 사용할 수 있는 변수와 함수는 어떤 종류가 있는지 dir( )로 확인

```
>>> dir(sys)
['__breakpointhook__', '__displayhook__', '__doc__', '__excepthook__', '__
interactivehook__', '__loader__', '__name__', '__package__', '__spec__', '__stderr__', '__
stdin__', '__stdout__', '__unraisablehook__', '_base_executable',
```

```
'set_asyncgen_hooks', 'set_coroutine_origin_tracking_depth', 'setprofile',
'setrecursionlimit', 'setswitchinterval', 'settrace', 'stderr', 'stdin', 'stdout',
'thread_info', 'unraisablehook', 'version', 'version_info', 'warnoptions', 'winver']
```

02. 표준 모듈

I. sys 모듈

실습 12-1 sys 모듈 다양하게 사용하기

③ 사용하고 있는 컴퓨터의 운영체제 정보를 getwindowsversion( ) 함수로 확인

```
>>> print(sys.getwindowsversion())
sys.getwindowsversion(major=10, minor=0, build=18363, platform=2, service_pack='')
```

④ 실행 중인 파이썬 버전 정보를 확인

```
>>> print(sys.version)
3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)]
```

⑤ 파이썬이 설치된 경로 정보를 확인

```
>>> print(sys.prefix)
C:\Users\jykim\AppData\Local\Programs\Python\Python39
```

⑥ 문자열 파일을 다룰 때 필요한 기본 인코딩 방법을 getdefaultencoding() 함수로 확인

```
>>> print(sys.getdefaultencoding())
utf-8
```

02. 표준 모듈

II. os 모듈

실습 12-2 os 모듈 다양하게 사용하기

① os 모듈을 import하고, getcwd() 함수로 프로그램이 실행되고 있는 현재 경로를 확인

```
>>> import os
>>> os.getcwd()
'C:\\python'
```

② listdir( )을 사용하면 현재 경로에 존재하는 파일과 디렉토리 목록을 리스트 형태로 확인

```
>>> os.listdir()
['login.txt', 'member.txt', 'poem.txt']
```

③ rename( )으로 파일 이름을 변경하거나, mkdir( )로 디렉토리를 만들 수도 있음

```
>>> os.rename('login.txt', 'user_id.txt')
>>> os.mkdir('temp')
```

II. os 모듈

실습 12-2 os 모듈 다양하게 사용하기

③ rename( )으로 파일 이름을 변경하거나, mkdir( )로 디렉토리를 만들 수도 있음

```
>>> os.listdir()
['member.txt', 'poem.txt', 'temp', 'user_id.txt']
```

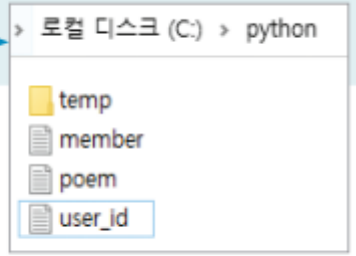


그림 12-4 탐색기에서 확인하기

II. os 모듈

실습 12-2 os 모듈 다양하게 사용하기

④ system( )은 윈도우의 명령 프롬프트나 유닉스에서 사용 가능한 시스템 명령어를 직접 실행

```
>>> os.system('date')
```

날짜를 변경하는 명령,
수정하지 않으려면 Enter를 누름

그림 12-5 날짜 변경 명령 실행

```
>>> os.system('copy member.txt member(2).txt')
0
>>> os.listdir()
['login.txt', 'member(2).txt', 'member.txt', 'poem.txt', 'temp']
```

파일을 복사하는 명령

복사본 생성

III. random 모듈

실습 12-3 중복 없는 로또 번호 생성하기 code12-03.py

① 먼저 49개의 숫자(1~49)를 내장 함수 range( )로 생성한 후 리스트에 저장

```
numbers = list(range(1, 50))
```

② random 모듈의 sample( )을 이용하면 리스트 항목 중에서 필요한 수만큼 고를 수 있어서, 중복되지 않게 번호를 조합

```
from random import *

numbers = sample(numbers, 6)          # 리스트 항목 6개를 샘플링
```

③ 6개 숫자를 5번 반복 생성하도록 for 문을 활용

```
01  from random import *
02
03  for x in range(5) :
04      numbers = list(range(1, 50))
05      numbers = sample(numbers, 6)
06      print("{0} : {1}".format(x+1, numbers))
```

```
1 : [34, 14, 33, 21, 39, 19]
2 : [40, 41, 31, 39, 11, 26]
3 : [8, 29, 13, 6, 25, 1]
4 : [32, 39, 38, 18, 26, 36]
5 : [7, 40, 21, 38, 11, 46]
```

IV. math 모듈

실습 12-4 삼각함수표 만들기 code12-04.py

① math 모듈의 사용을 선언하고, sin( ), cos( ), tan( )에 각도 x를 인수로 넣어 계산 결과를 각각 저장

```
from math import *

x = 0     # x는 각도
s = sin(radians(x))
c = cos(radians(x))
t = tan(radians(x))
```

② 반복문을 활용해 0도부터 90도까지 10도 간격으로 계산되도록 수정

```
for x in range(0, 91, 10) :          # x는 각도
    s = sin(radians(x))
    c = cos(radians(x))
    t = tan(radians(x))
```

IV. math 모듈

실습 12-4 삼각함수표 만들기 code12-04.py

③ print( )문에 문자열 포맷팅(format())으로 자릿수를 설정해서 알아보기 쉽게 출력

```
# 필드명 출력
print("{0:>3} {1:>10} {2:>10} {3:>25}".format("x", "sin(x)", "cos(x)", "tan(x)"))
print("="*52)
# 값 출력
print("{0:3d} {1:10.4f} {2:10.4f} {3:25.4f}".format(x, s, c, t))
```

④ 프로그램을 실행해서 결과를 확인

```
01  from math import *
02
03  print("{0:>3} {1:>10} {2:>10} {3:>25}".format("x", "sin(x)", "cos(x)", "tan(x)"))
04  print("="*52)
05
06  for x in range(0, 91, 10) :
07      s = sin(radians(x))
08      c = cos(radians(x))
09      t = tan(radians(x))
10      print("{0:3d} {1:10.4f} {2:10.4f} {3:25.4f}".format(x, s, c, t))          # 값 출력
```

IV. math 모듈

여기서 잠깐 **각도의 단위**

- 각도를 표시하는 또 다른 방법으로 라디안이 있음
- 호의 길이가 반지름과 같게 되는 만큼의 각을 1 라디안이라고 정의하면, 이는 약 57.3 디그리에 해당하는 값
- 모듈의 삼각함수들은 라디안 값이 인수로 사용되므로, radians()를 사용해 일반적으로 사용하는 디그리 각도를 라디안 값으로 변환해야 함

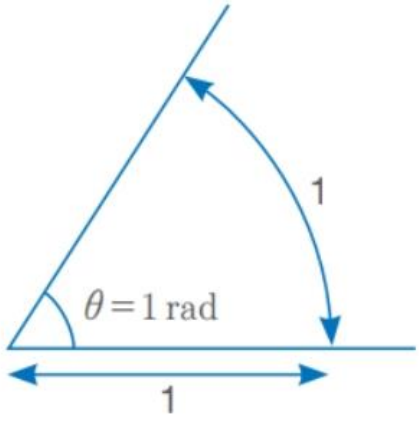


그림 12-7 라디안 단위

IV. math 모듈

여기서 잠깐 **출력 문자열의 자릿수 지정과 정렬**

- 문자열 객체의 format() 메소드는 문자열 내에 중괄호를 사용해 인덱스를 지정하고 인수 목록 중 해당 위치에 있는 값을 가져와 문자열을 구성

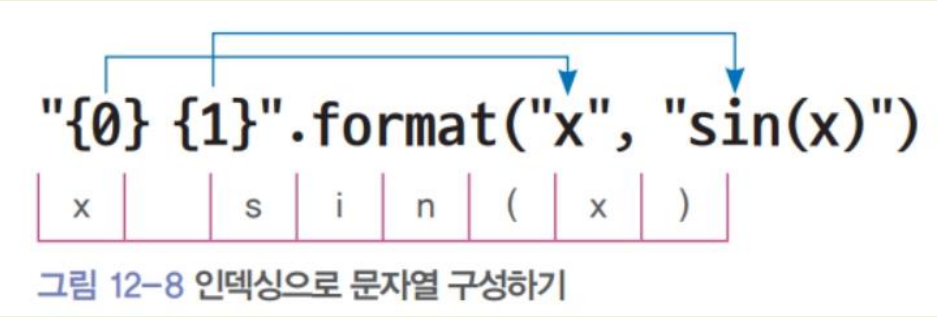


그림 12-8 인덱싱으로 문자열 구성하기

- 중괄호 안에 인덱스와 함께 콜론(:)을 사용하면 자릿수나 정렬 방법도 지정할 수 있음

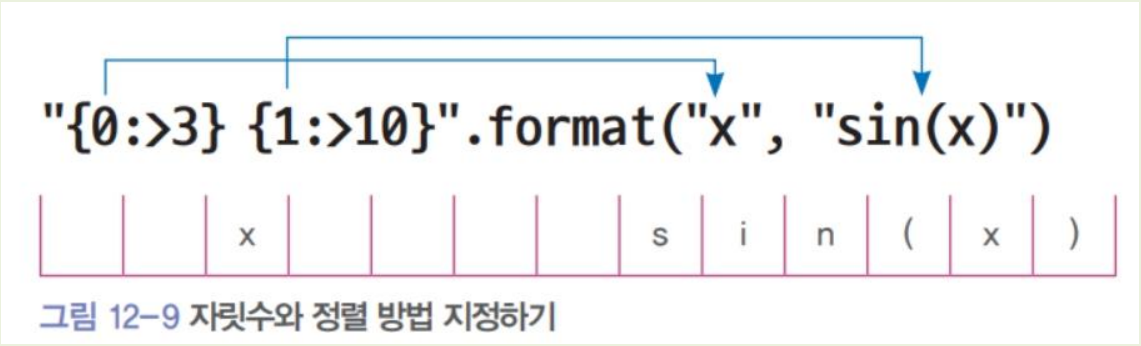


그림 12-9 자릿수와 정렬 방법 지정하기

V. datetime 모듈

실습 12-5 기념일 계산기 만들기 code12-05.py

① datetime 모듈의 사용을 선언하고 date 클래스의 today( ) 메소드로 오늘 날짜를 구함

```
from datetime import *

today = date.today()
```

② 날짜 계산에 사용할 일수를 입력

```
period = int(input("일 수 입력 : "))
```

③ 오늘 날짜에 일 수를 더하는 계산은 datetime 모듈의 timedelta( )를 이용

```
result = today + timedelta(days=period)
```

V. datetime 모듈

실습 12-5 기념일 계산기 만들기 code12-05.py

④ 계산된 날짜 정보를 년, 월, 일 단위로 구분해서 출력

```
print("오늘부터 {0}일 후는 {1}년 {2}월 {3}일입니다."
      .format(period, result.year, result.month, result.day))
```

⑤ 파일로 저장하고 실행

```
01 from datetime import *
02
03 today = date.today()
04
05 print("오늘부터 며칠 후의 날짜를 알고 싶나요?")
06 period = int(input("일 수 입력 : "))
07
08 result = today + timedelta(days=period)
09
10 print("오늘부터 {0}일 후는 {1}년 {2}월 {3}일입니다."
11       .format(period, result.year, result.month, result.day))
```

03

사용자 정의 모듈

03. 사용자 정의 모듈

I. 모듈 작성과 사용

▪ 모듈의 참조 경로

- 모듈을 찾을 때 참조하는 경로는 다음과 같이 sys 모듈의 path를 조회

```
>>> import sys
>>> sys.path
['C:/python',
'C:\\Users\\jykim\\AppData\\Local\\Programs\\Python\\Python39\\python39.zip', 'C:\\Users\\
jykim\\AppData\\Local\\Programs\\Python\\Python39\\DLLs', 'C:\\Users\\jykim\\AppData\\
Local\\Programs\\Python\\Python39\\lib', 'C:\\Users\\jykim\\AppData\\Local\\Programs\\
Python\\Python39', 'C:\\Users\\jykim\\AppData\\Local\\Programs\\Python\\Python39\\lib\\
site-packages']
```

현재 디렉토리

파이썬 라이브러리 설치 경로

- 만약 모듈을 검색하는 경로에 특정 디렉토리를 추가하려면 append( )를 사용

```
>>> sys.path.append("D:\\source")
>>> sys.path
['C:/python',
'C:\\Users\\jykim\\AppData\\Local\\Programs\\Python\\Python39\\python39.zip'
..., 'D:\\source']
```

03. 사용자 정의 모듈

I. 모듈 작성과 사용

실습 12-6 모듈 작성과 사용하기 code12-06.py

- 오늘 날짜를 가져와 출력할 때 사용하기 위한 myprint 모듈(myprint.py)을 직접 작성

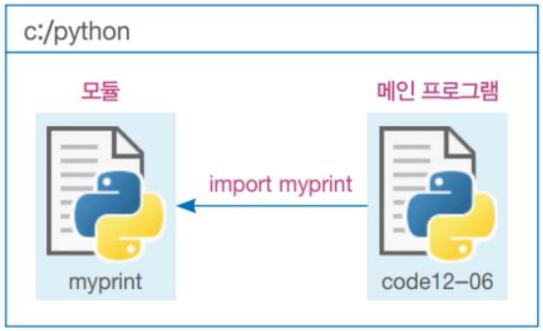


그림 12-12 모듈 작성 후 불러오기

① 코드 편집기에 다음과 같이 입력하고 'myprint.py'로 파일을 저장

```
01  # 사용자 정의 모듈 myprint
02  from datetime import *
03
04  def print_line() :
05      print('*' * 30)
06
07  def print_title() :
08      print_line()
09      print("작성일 :", date.today())
10      print_line()
```

I. 모듈 작성과 사용

여기서 잠깐 **__pycache__ 폴더**

- 모듈을 저장만 해도 되지만 F5를 누르면 코드가 컴파일 됨
- 실행 결과는 아무것도 없지만, 탐색기를 열면 '__ pycache__'가 생성되어 있음
- 만약 컴파일 과정에서 오류가 발생하면 모듈을 사용할 수 없으므로, 오류를 반드시 수정할 것

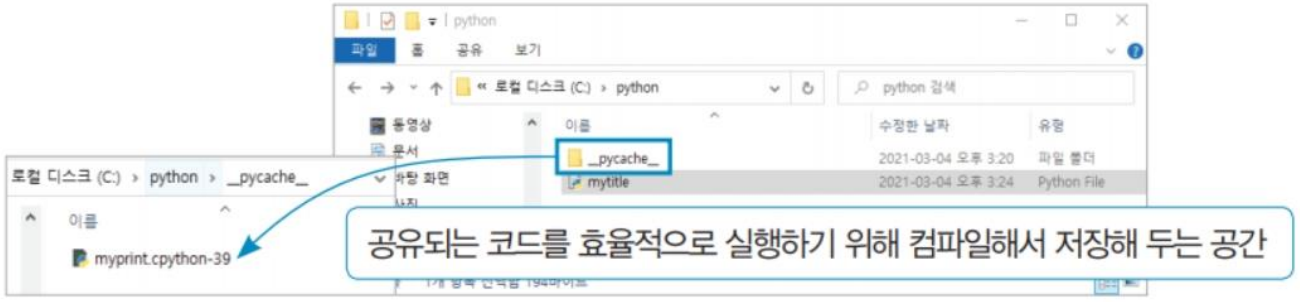


그림 12-13 __pycache__ 폴더 내부

03. 사용자 정의 모듈

I. 모듈 작성과 사용

실습 12-6 모듈 작성과 사용하기 code12-06.py

② 모듈을 가져와 사용하는 메인 프로그램을 다음과 같이 입력하고 모듈을 만들어 둔 디렉토리('C:/python')에 저장('code12-06. py')

```
01  from myprint import *              # 모듈 사용 선언
02
03  contents = input("내용을 입력하세요. : ")
04
05  print_title()                      # myprint 모듈의 print_title() 호출
06  print(contents)
07  print_line()                       # myprint 모듈의 print_line() 호출
```

③ F5를 눌러 메인 프로그램을 실행

```
내용을 입력하세요. : Beautiful is better than ugly.
******************************
작성일 : 2021-03-05
******************************
Beautiful is better than ugly.
******************************
```

03. 사용자 정의 모듈

[실습] 전화번호 지역 번호 관련 사용자 정의 모듈 Telzone.py 구성

◆ 전화번호 지역 번호 관련 모듈인 Telzone.py

- get_zone() : 지역 번호를 전달 받아 지역명(zone)을 반환
- get_no() : 지역명을 전달 받아 지역 번호(no)를 반환

❖ Telzone.py

```
zoneno = ([['010'], '휴대전화'],
        ['02', ['서울특별시', '광명시', '과천시']], ['031', '경기도'], ['032', ['인천광역시', '부천시']], ['033', '강원도'],
        ['041', '충청남도'], ['042', ['대전광역시', '계룡시']], ['043', '충청북도'], ['044', '세종특별자치시'],
        ['051', '부산광역시'], ['052', '울산광역시'], ['053', ['대구광역시', '경산시']], ['054', '경상북도'],
        ['055', '경상남도'], ['061', '전라남도'], ['062', '광주광역시'], ['063', '전라북도'], ['064', '제주특별자치도'])


def get_zone(inno):  #지역번호로 검색
    for no, zone in zoneno:
        if type(no) is list:  # 항목 형식이 리스트인지 확인
            if inno in no:   # 리스트 항목에서 찾기
                return zone
            elif no == inno:  # 일반 항목일 경우 일치 비교
                return zone
        else:
            if no == inno:
                return zone
```

03. 사용자 정의 모듈

[실습] 전화번호 지역 번호 관련 사용자 정의 모듈 Telzone.py 구성

◆ 전화번호 지역 번호 관련 모듈인 Telzone.py

- get_zone() : 지역 번호를 전달 받아 지역명(zone)을 반환
- get_no() : 지역명을 전달 받아 지역 번호(no)를 반환

❖ Telzone.py

```
def get_no(inzone):
    for no, zone in zoneno:
        if type(zone) is list:  # 항목 형식이 리스트인지 확인
            for zlist in zone:
                if inzone in zlist:
                    return no
        elif zone.find(inzone) >= 0:  # 일반 항목일 경우 일치 비교
            return no
```

03. 사용자 정의 모듈

[실습] 전화 지역번호 관련 사용자 정의 모듈 Telzone.py 사용 (1)

◆ 사용자 정의 모듈인 Telzone.py를 자신의 코드 디렉토리에 위치

- 메인 : make_menu() 함수 호출로 메뉴 선택 처리
  - 0, '종료' 1, '번호로 검색' 2, '지역명으로 검색'

❖ Telephone.py

```
import Telzone as tz
#작업 선택 종류
menuno = ([0, '종료'], [1, '번호로 검색'], [2, '지역명으로 검색'])  # 메뉴를 위한 튜플 데이터
MENUE_fn = [fpass, find_no, find_zone]  # 호출될 함수명


while True:
    selnum = make_menu(menuno)
    if selnum < 0:
        continue
    if selnum == 0:
        break
    else:
        MENUE_fn[selnum]()  # 리스트 항목에 근거한 함수 호출
```

```
>>번호로 선택: [0] 종료    [1] 번호로 검색    [2] 지역명으로 검색
No?> 1
>>전화 지역번호 입력: 032
032 ['인천광역시', '부천시']

>>번호로 선택: [0] 종료    [1] 번호로 검색    [2] 지역명으로 검색
No?> 2
>>전화번호 지역명 입력: 부천
부천 032

>>번호로 선택: [0] 종료    [1] 번호로 검색    [2] 지역명으로 검색
No?> 0
```

03. 사용자 정의 모듈

[실습] 전화 지역번호 관련 사용자 정의 모듈 Telzone.py 사용 (2)

◆ 사용자 정의 모듈인 Telzone.py를 자신의 코드 디렉토리에 위치

- 메인 : make_menu() 함수 호출로 메뉴 선택 처리
  - 0, '종료' 1, '번호로 검색' 2, '지역명으로 검색'

❖ Telephone.py

```
def make_menu(menuno):
    print("₩n>>번호로 선택: ", end='')
    for no, name in menuno:
        print("[%d] %s " % (no, name), end='  ')
    selnum = int(input("₩nNo?> "))
    if selnum >= len(menuno) or selnum < 0:
        print(">>선택 범위 초과!!")
        return -1
    else:
        return selnum
```

03. 사용자 정의 모듈

[실습] 전화 지역번호 관련 사용자 정의 모듈 Telzone.py 사용 (3)

◆ 사용자 정의 모듈인 Telzone.py를 자신의 코드 디렉토리에 위치

- 메인 : make_menu() 함수 호출로 메뉴 선택 처리
  - 0, '종료' 1, '번호로 검색' 2, '지역명으로 검색'

❖ Telephone.py

```
def find_no():  # 지역번호로 검색
    inno = input(">>전화 지역번호 입력: ")
    zone = tz.get_zone(inno)
    print(inno, zone)

def find_zone():  # 지역번호로 검색
    inzone = input(">>전화번호 지역명 입력: ")
    no = tz.get_no(inzone)
    print(inzone, no)

def fpass():  # 없는 메뉴 pass 처리
    pass
```

03. 사용자 정의 모듈

2. 사용자 정의 모듈의 활용

실습 12-7 D-Day 계산기 만들기 code12-07.py

그림 12-15 D-Day 계산기의 모양과 동작 방법

① 그래픽 사용자 인터페이스로 프로그램이 동작되도록 tkinter 표준 모듈을 사용하고 기본 윈도우를 생성

```
from tkinter import *

w = Tk()
w.title('D-Day 계산기')
```

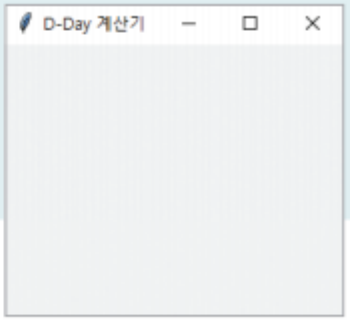


그림 12-16 기본 윈도우 생성

2. 사용자 정의 모듈의 활용

실습 12-7 D-Day 계산기 만들기 code12-07.py

② 오늘 날짜를 표시하기 위한 레이블을 만들어 윈도우에 배치

```
yy = mm = dd = 99        # 오늘 날짜를 구하는 함수를 모듈에 만들 예정이므로 임시 값을 설정
Label(w, text="오늘 날짜", width=20).grid(row=0, column=0, pady=5)
lbToday = Label(w, text="{0}년 {1}월 {2}일".format(yy, mm, dd))
lbToday.grid(row=0, column=1)
```

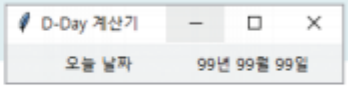


그림 12-17 날짜 표시 레이블 추가

③ D-Day를 입력하는 엔트리를 윈도우에 추가

```
Label(w, text="D-Day (예)20210301").grid(row=1, column=0, pady=5)
enDday = Entry(w)
enDday.grid(row=1, column=1)
```

03. 사용자 정의 모듈

2. 사용자 정의 모듈의 활용

실습 12-7 D-Day 계산기 만들기 code12-07.py

④ 버튼 클릭 이벤트에 대한 처리는 f_click( ) 함수로 하고, 모듈에 해당 기능을 만든 후에 함수 동작을 수정

```
def f_click() :
    pass      # '아무것도 실행하지 않음'을 의미하는 예약어
```

```
btCalc = Button(w, text="계산하기", width=20, command=f_click)
btCalc.grid(row=2, column=0, columnspan=2, pady=5)
```

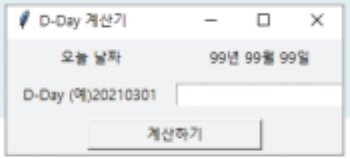


그림 12-18 엔트리와 버튼 추가

⑤ 결과를 표시할 레이블을 추가하고 이벤트 처리를 대기하는 문장을 추가

```
lbResult = Label(w, text="D-", font=("arial",15), fg="red")
lbResult.grid(row=3, column=0, columnspan=2, pady=5)

w.mainloop()
```

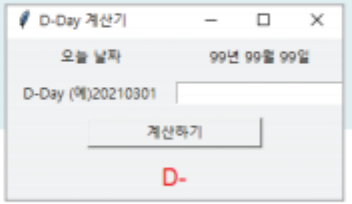


그림 12-19 결과 표시 레이블 추가

03. 사용자 정의 모듈

2. 사용자 정의 모듈의 활용

실습 12-7 D-Day 계산기 만들기 code12-07.py

⑥ 모듈 생성

```
01  from datetime import *
02
03  def get_today() :                                  # 오늘 날짜를 반환하는 함수
04      today = datetime.today()
05      return today.year, today.month, today.day      # 연도, 월, 일을 반환
06
07  def str_to_date(s) :                               # 문자열을 datetime 형으로 변환
08      try :
09          y = int(s[:4])                             # 연도, 월, 일 추출
10          m = int(s[4:6])
11          d = int(s[6:])
12      except :                                       # 날짜 형식에 맞지 않는 경우, 예외 처리
13          print("날짜 입력 오류")
14          return -1
15      else :
```

03. 사용자 정의 모듈

2. 사용자 정의 모듈의 활용

실습 12-7 D-Day 계산기 만들기 code12-07.py

⑥ 모듈 생성

```
16          return datetime(y, m, d)
17
18  def calc_days(date) :           # D-Day(예, '20210629')를 전달받아 남은 날과 시간을 계산
19      days = hours = 0
20      if str_to_date(date) != -1 :
21          result = str_to_date(date) - datetime.today()
22          days = result.days
23          hours = result.seconds//3600
24
25      return days, hours          # 남은 날과 시간 반환(날짜 오류인 경우 0, 0 반환)
```

2. 사용자 정의 모듈의 활용

실습 12-7 D-Day 계산기 만들기 code12-07.py

⑦ 메인 프로그램 수정

```
01  from tkinter import *
02  from mydays import *                                  # mydays 모듈의 사용 선언
03
04  def f_click() :
05      days, hours = calc_days(enDday.get())             # mydays 모듈의 calc_days()를 호출
06      lbResult.config(text="D- {0}일 {1}시간".format(days, hours))    # 반환 결과를 출력
07
08  w = Tk()
09  w.title('D-Day 계산기')
10
11  yy, mm, dd = get_today()                              # mydays 모듈의 get_today()를 호출
12  Label(w, text="오늘 날짜", width=20).grid(row=0, column=0, pady=5)
13  lbToday = Label(w, text="{0}년 {1}월 {2}일".format(yy, mm, dd))       # 반환 결과 출력
14  lbToday.grid(row=0, column=1)
15
```

2. 사용자 정의 모듈의 활용

실습 12-7 D-Day 계산기 만들기 code12-07.py

⑦ 메인 프로그램 수정

```
16  Label(w, text="D-Day (예)20210301").grid(row=1, column=0, pady=5)
17  enDday = Entry(w)
18  enDday.grid(row=1, column=1)
19
20  btCalc = Button(w, text="계산하기", width=20, command=f_click)
21  btCalc.grid(row=2, column=0, columnspan=2, pady=5)
22
23  lbResult = Label(w, text="D-", font=("arial",15), fg="red")
24  lbResult.grid(row=3, column=0, columnspan=2, pady=5)
25
26  w.mainloop()
```

03. 사용자 정의 모듈

2. 사용자 정의 모듈의 활용

실습 12-8 모듈을 이용한 그림판 만들기 code12-08.py

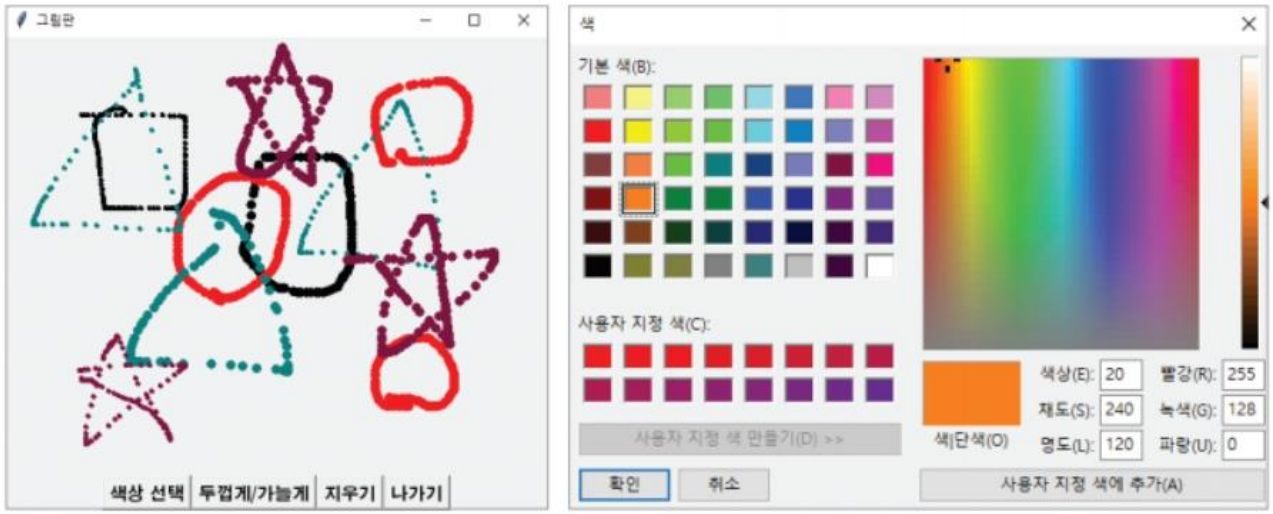


그림 12-20 실행 결과

① mypainter 모듈을 다음과 같이 작성하고 'mypainter.py'로 저장

```
01  from tkinter import colorchooser                # 색상 선택 대화 상자의 사용
02
03  def get_color():
04      color = colorchooser.askcolor()             # 선택한 색상 정보를 튜플로 반환
05      return color[1]                             # 튜플의 두 번째 항목(색상 코드)을 반환
```

03. 사용자 정의 모듈

2. 사용자 정의 모듈의 활용

실습 12-8 모듈을 이용한 그림판 만들기 code12-08.py

① mypainter 모듈을 다음과 같이 작성하고 'mypainter.py'로 저장

```
06
07  def get_xy(x, y, bold):                          # 펜 두께(bold)에 따라 타원의 크기 결정
08      if bold:
09          return (x-4, y-4, x+4, y+4)              # 타원의 좌표를 넓게 = 두껍게
10      else:
11          return (x-2, y-2, x+2, y+2)              # 타원의 좌표를 좁게 = 가늘게
```

② 메인 프로그램을 다음과 같이 작성하고 실행

```
01  from tkinter import *
02  from mypainter import *
03
04  color = 'black'                                  # 색상 초기화
05  bold = False                                     # 펜 두께 초기화(False:가늘게, True:두껍게)
```

2. 사용자 정의 모듈의 활용

실습 12-8 모듈을 이용한 그림판 만들기 code12-08.py

② 메인 프로그램을 다음과 같이 작성하고 실행

```
06
07  def draw(event):
08      x1, y1, x2, y2 = get_xy(event.x, event.y, bold)          # 좌표 계산 함수 호출
09      canvas.create_oval(x1, y1, x2, y2, fill=color, outline=color)
10
11  def clear_canvas():
12      canvas.delete("all")
13
14  def change_color():
15      global color
16      color = get_color()                                      # 색상 선택 함수 호출
17
18  def change_bold():
19      global bold
20      bold = not(bold)                          # 버튼을 누를 때마다 '두껍게↔가늘게' 전환
```

2. 사용자 정의 모듈의 활용

실습 12-8 모듈을 이용한 그림판 만들기 code12-08.py

② 메인 프로그램을 다음과 같이 작성하고 실행

```
21
22  w = Tk()
23  w.title("그림판")
24  canvas = Canvas(w, width=500, height=400)
25  canvas.pack()
26  canvas.bind("<B1-Motion>", draw)          # 마우스 왼쪽 클릭에 함수 draw()를 연결
27
28  frame = Frame(w)
29  frame.pack()
30  bColor = Button(frame, text="색상 선택", font="Tahoma", command=change_color)
31  bBold = Button(frame, text="두껍게/가늘게",
32                 font="Tahoma", command=change_bold)
33  bClear = Button(frame, text="지우기", font="Tahoma", command=clear_canvas)
34  bExit = Button(frame, text="나가기", font="Tahoma", command=w.destroy)
35  bColor.grid(row=0, column=0)
```

2. 사용자 정의 모듈의 활용

실습 12-8 모듈을 이용한 그림판 만들기 code12-08.py

② 메인 프로그램을 다음과 같이 작성하고 실행

```
36  bBold.grid(row=0, column=1)
37  bClear.grid(row=0, column=2)
38  bExit.grid(row=0, column=3)
39
40  w.mainloop()
```

Thank You !