술술 넘기며 다 이해하는
파이썬 코딩 노트북 38

확인문제 & 실행과제 답안 및 해설

노트북_03. 파이썬, 세상과의 첫 대화

[확인문제 1]

- `print()` 함수를 이용해서 `'안녕하세요, 여러분'`이라는 한글 메시지를 출력해 보세요.

```
print('안녕하세요, 여러분')
```

- 세 개의 `print()` 함수를 이용해서 나태주 시인의 〈풀꽃〉이란 시를 출력해 보세요.

```
print('자세히 보아야 예쁘다')
print('오래 보아야 사랑스럽다')
print('너도 그렇다')
```

- 한 개의 `print()` 함수를 이용해서 같은 시를 출력해 보세요. (힌트: 줄바꾸기 문자 `\n` 을 이용하세요.)

```
print('자세히 보아야 예쁘다\n 오래 보아야 사랑스럽다\n 너도 그렇다')
```

```
자세히 보아야 예쁘다
```

[확인문제 2]

- 다음 시를 임의의 변수에 저장하고, **print()** 함수를 이용해서 출력해 보세요.

```
message = '''너를 만나 사랑하게 된 이 세상은
완전히 새로운 세상이야.
얼마나 아름답고 행복한지...'''
print(message)
```

- 인터넷에서 가져온(Ctrl+C, Ctrl+V) 여러 줄로 된 긴 글을 변수에 저장하고, **print()** 함수를 이용해서 출력해 보세요.

마우스나 키보드 등으로 복사할 텍스트 영역을 선택하고 복사하기(Ctrl-C) 키를 누른 후, 파이썬에서 다음과 같은 문자열 안에 커서를 위치시키고 붙여넣기(Ctrl-V)를 하여 `message` 변수에 저장합니다.

```
message = '''
<이곳에 키보드 커서를 위치시키고 붙여넣기(Ctrl-V)를 합니다.>
'''
print(message)
```

[확인문제 3]

- 사용자에게 '사랑해'로 삼행시를 지어 입력할 것을 요청해 보세요.

```
line1 = input('사: ')
line2 = input('랑: ')
line3 = input('해: ')
```

- 입력받은 시의 각 행을 세 개의 `print()` 함수로 출력해 보세요.

```
print(line1)
print(line2)
print(line3)
```

- 입력받은 시의 세 행을 하나의 `print()` 함수로 출력해 보세요.

```
print(line1 + '\n' + line2 + '\n' + line3)
```

실행과제

`print()` 함수를 이용하여 문자열, 숫자를 화면에 출력해 보세요.

```
print('파이썬 파이팅', 7979)
```

```
파이썬 파이팅 7979
```

키보드로 메시지를 입력받은 뒤, 다시 화면에 출력해 보세요. (힌트: `input()` 함수를 이용하세요.)

```
message = input('메시지를 입력한 후 엔터 키를 눌러 주세요: ')
print(message)
```

키보드에서 숫자를 입력받은 뒤, 그 숫자의 3 배인 수를 출력해 보세요. (힌트: `int()` 함수를 이용하세요.)

```
message = input('숫자를 입력한 후 엔터 키를 눌러 주세요: ')
number = int(message)
print(number * 3)
```

웹에서 여러 줄 텍스트를 복사하고 `msg` 변수에 저장해 보세요.

```
msg = '''
웹 브라우저에서 텍스트를 복사(Ctrl-C)한 후 여기에 붙여넣기(Ctrl-V)합니다
'''
msg
```

```
'\n웹 브라우저에서 텍스트를 복사(Ctrl-C)한 후 여기에 붙여넣기(Ctrl-V)합니다\n'
```

노트북_04. 파이썬과 좀 특별한 문자들

[확인문제 1]

- `c:\windows\fonts\arial` 의 경로 문자열을 `print()` 함수로 출력해 보세요. 어떤 결과가 나올지 미리 예측해 보세요.

```
print('c:\windows\fonts\arial')
```

```
c:#windows$fonts•rial
```

`\f` 는 출력용지를 한 페이지 넘기는 폼 피드(form feed) 코드로 인식됩니다. `\a` 는 경고음을 울리는 알람(alarm) 코드입니다. 명령 프롬프트에서 코드를 실행한다면 윈도우 경고음을 들을 수 있습니다.

- `print('ab\b')`의 출력 결과는 무엇일까요?

이 문제는 다소 혼란을 일으킬 수 있습니다. 왜냐하면 `\b` 는 백스페이스 코드이지만, 실행 환경에 따라 동작이 조금 다르기 때문입니다. 가령 IDLE 에서는 백스페이스 코드가 제대로 실행되지 않아 특수 코드로 표시됩니다.

```
>>> print('ab\b')
ab□
```

명령 프롬프트에서 이것은 커서의 위치만 왼쪽으로 이동하는 백스페이스 코드입니다. 그래서 출력 결과는 `ab` 입니다.

```
>>> print('ab\b')
ab
```

그러나 `print('ab\b ')`를 실행하면 `a` 만 출력됩니다. 커서를 왼쪽으로 이동한 후 스페이스를 하나 출력하기 때문입니다. 다음 코드가 이해에 도움이 될 수 있습니다.

```
>>> print('ab\b ')
a
```

주피터 노트북에서는 `\b` 가 왼쪽 문자를 지우는 백스페이스 코드로 동작합니다.

```
print('ab\b')
```

```
a
```

```
print('Hello\b\b')
```

```
Hel
```

- 단일 인용부호를 이용하여 Mary's Restaurant 를 출력해 보세요.

```
print('Mary\'s Restaurant')
```
```
Mary's Restaurant
```

[확인문제 2]

- 날 문자열을 이용하여 `C:\windows\fonts\arial` 경로를 출력해 보세요.

```
print(r'C:\windows\fonts\arial')
```
```
C:\windows\fonts\arial
```

[확인문제 3]

- 4 자리 유니코드(`\uxxxx`)를 이용하여 하트(♥) 특수문자를 표시해 보세요(코드값 16 진수 2665).

```
print('\u2665')
```
```
♥
```

- 8 자리 유니코드(`\Uxxxxxxxx`)를 이용하여 고양이(😺) 특수문자를 표시해 보세요(코드값 0001F63A).

```
print('\U0001F63A')
```
```
😺
```

- 웹에서 검색한 문자를 '복사-붙여넣기'하여 문자열 안에 특수문자를 표시해 보세요.

```
print('□ ■ △ ▲ ▽ ▼ → ← ↑ ↓ ↔ =')
```
```
□ ■ △ ▲ ▽ ▼ → ← ↑ ↓ ↔ =
```

- 한글 유니코드 범위를 인터넷에서 직접 검색해 보세요.

한글은 '모아쓰기하는 음소문자'인 까닭에 2 개의 유니코드 범위를 가집니다. U+1100~U+11FF 가 한글의 자모(자음&모음) 영역이며, U+AC00~U+D7AF 가 한글의 음절(소리 마디) 영역입니다.

실행과제

- 인터넷에서 유니코드 특수문자와 유니코드 이모티콘을 검색해서 다양한 특수문자를 출력해 보세요.

`print('유니코드')` 형태를 사용하면 됩니다. 문자인 만큼 따옴표를 빼놓지 마세요!

- `\uxxxx` 로 표현하는 문자와 `\Uxxxxxxxx` 로 표현하는 문자들에 어떤 차이가 있는지 설명해 보세요.

유니코드는 평면(plane)이란 이름으로 코드 영역을 설정합니다. 그 중에서 숫자가 가장 낮은 쪽의 `0000~FFFF` 영역이 평면 0(Plane 0)입니다. 이 영역을 'BMP(Basic Multilingual Plane, 기본 다국어 평면)'라고 부르고 다국어 문자를 배치하고 있습니다. 평면은 16 까지 존재하며 위로 올라갈수록 사용 빈도수가 낮은 특수 문자가 배치되어 있습니다. 더 자세한 내용은 '유니코드 문자 평면'으로 검색해 보시기 바랍니다.

- `ord()`, `hex()` 함수를 이용하여 `'가'`와 `'힣'`의 문자 코드값을 확인해 보세요.

```
ord('가')
hex(ord('가'))
ord('힣')
hex(ord('힣'))
```

```
44032
'0xac00'
55203
'0xd7a3'
```

- 한자의 유니코드 범위는 어디부터 어디까지인지 확인해 보세요.

유니코드의 한자 코드 집합은 다양한 블록에 정의되어 있어서 한 마디로 정의하기 쉽지 않지만, 한중일 통합 한자를 기준으로 한다면 **0x4E00~0x9FFF 범위**입니다.

노트북_05. 내 프로그램 작성·실행하기 _스크립트 파일

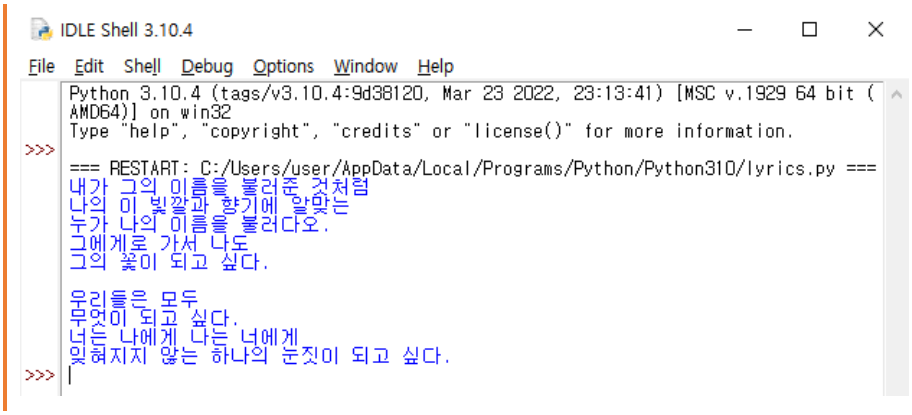
실행과제

- 여러 행으로 된 시 한 편을 출력하는 스크립트 프로그램을 만들고 실행해 보세요.

IDLE 편집기를 이용해 김춘수의 〈꽃〉 3~4 연을 출력하는 프로그램 lyrics.py 를 만들어 실행한 예입니다.

```
print('''내가 그의 이름을 불러준 것처럼
나의 이 빛깔과 향기에 알맞는
누가 나의 이름을 불러다오.
그에게로 가서 나도
그의 꽃이 되고 싶다.

우리들은 모두
무엇이 되고 싶다.
너는 나에게 나는 너에게
```

```
잊혀지지 않는 하나의 눈짓이 되고 싶다.''')
```



노트북_06. 기초 산술 연산

[확인문제 1]

- `10 = inch` 와 같은 코드가 가능하지 않은 이유는 무엇일까요?

기호 '='는 오른쪽 값을 왼쪽에 저장하라는 치환문입니다. 따라서 좌변은 값을 저장하는 변수 역할을 해야 하는데, `10` 이라는 상수는 값을 저장할 수 없기 때문입니다.

- 3 피트(feet)를 미터로 변환하는 코드를 작성해 보세요. 1 피트는 0.3048 미터입니다.

```
mpf = 0.3048
print(3 * mpf)
```

```
0.9144000000000001
```

- 80 제곱미터(㎡, 평방미터)를 평으로 환산해 보세요. 1 ㎡는 0.3025 평입니다.

```
m2p = 0.3025
print(80 * m2p)
```

```
24.2
```

- 1 갤런(gallon)은 3.785412 리터입니다. 5 갤런은 몇 리터인지 확인해 보세요.

```
lpg = 3.785412
print(5 * lpg)
```

```
18.92706
```

[확인문제 2]

오늘 점심 후식으로 마신 아메리카노 커피는 3600 원입니다. 현금으로 값을 치르려고 합니다. 이때, 다음 질문에 답해 보세요.

- 천 원짜리 몇 장을 내야 하는지 수식으로 직접 계산해 보세요. (힌트: // 연산자를 이용하세요.)

```
금액 = 3600
print(금액 // 1000)
```

```
3
```

천 원짜리는 3 장을 내야 합니다.

- 남은 금액에서 오백 원짜리는 몇 개를 내야 하는지 수식으로 직접 계산해 보세요.

```
남은금액1 = 금액 - (금액 // 1000 * 1000)
print(남은금액1, 남은금액1 // 500)
```

```
500 1
```

오백 원짜리는 1 개를 내야 합니다.

- 남은 금액에서 백 원짜리는 몇 개를 내야 하는지 수식으로 직접 계산해 보세요.

```
남은금액2 = 남은금액1 - (남은금액1 // 500 * 500)
print(남은금액2, 남은금액2 // 100)
```

```
100 1
```

백 원짜리는 1 개를 내야 합니다.

[확인문제 3]

- 2 의 64 거듭제곱이 얼마인지 확인해 봅니다. (2 의 64 거듭제곱은 64 비트 CPU 가 한 번에 연산할 수 있는 최대 크기+1 의 자연수입니다.)

```
2 ** 64       # 첫 번째 방법: ** 연산자
pow(2, 64)    # 두 번째 방법: pow() 함수
```

```
18446744073709551616
18446744073709551616
```

- 일한 대가로 오늘은 쌀 1 톨(2^0), 내일은 쌀 2 톨(2^1), 모래는 쌀 4 톨(2^2), 글피는 쌀 8 톨((2^3)……을 받는다고 할 때, 30 일째에 받는 쌀은 몇 톨일까요?

```
print(2**0, 2**1, 2**2, 2**3, 2**29)
```
```
1 2 4 8 536870912
```

오늘 받는 쌀은 전날의 두 배씩이므로 지수함수로 표현됩니다. 그래서 30 일째에 받는 쌀은 2 의 29 승이 됩니다.

- 80kg 쌀 한 가마니에 쌀이 8 백만 톨 들어 있다고 할 때, 30 일째에 받게 되는 쌀은 몇 가마니일까요?

```
(2 ** 29) / 8000000
```
```
67.108864
```

소수점 이하를 버린다고 할 때, 67 가마니입니다.

실행과제

- 총부채원리금상환비율(DSR: Debt Service Ratio)이란, 차주(돈 빌린 사람)가 보유한 금융부채의 원리금 상환액이 연소득에서 차지하는 비율이며, 다음 식으로 계산됩니다.

  DSR(총부채원리금상환비율) = (금융회사 대출의 연간 원리금 상환액) ÷ 연소득 x 100

  내 연소득이 4000 만원이고, 대출이 2000 만원이 있습니다. 대출 이자와 원금 상환액을 합친 원리금 상환액으로 40 만원씩 매월 지출한다고 할 때, DSR 을 계산해 보세요.

```
월_원리금상환액 = 400000
연소득 = 20000000
DSR = (월_원리금상환액*12) / 연소득 * 100
print(DSR)       # 24%
```
```
24.0
```

노트북_07. 실수 연산

[확인문제]

- 0.1 을 10 번 더하면 얼마가 나올지 직접 확인해 보세요. 1.0 이 나올까요? (힌트: 곱셈이 아니라 덧셈을 한 결과를 확인해야 합니다.)

```
print(0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1)
```
```
0.9999999999999999
```

- 0.125 를 10 번 더하면 얼마가 나올지 직접 확인해 보세요. 1.25 가 나올까요?

```
print(0.125 + 0.125 + 0.125 + 0.125 + 0.125 + 0.125 + 0.125 + 0.125 + 0.125 + 0.125)
```
```
1.25
```

- `pow(e, pi)` 값과 `pow(pi, e)` 값 중 어느 값이 큰지 직접 확인해 보세요. 각 변수의 값은 다음과 같이 정의됩니다.

  e = 2.7183 / i = 3.1416

```
e = 2.7183
pi = 3.141

print(pow(e, pi), pow(pi, e))
```
```
23.12746789099232 22.44810946745975
```

`math` 모듈에서 지원하는 더 정확한 `e`, `pi` 를 이용하는 방법도 있습니다.

```
from math import e, pi

print(pow(e, pi), pow(pi, e))
```
```
23.140692632779263 22.45915771836104
```

실행과제

- 위경도를 나타내는 GPS 좌표 표기법에는 D(Degree)와 DMS(Degree-Minute-Second) 두 가지 표기법이 있습니다.
  DMS 표기법인 41°24'12.2"는 41 도 24 분 12.2 초로 읽습니다. 1 도는 60 분, 1 분은 60 초로 분할됩니다. 따라서 이것을 '도(degree)'로만 계산하면 41.40338888888889 란 값이 나옵니다. 이 변환을 직접 해보길 바랍니다.

분(minute)을 도(degree)로 환산하려면 60 으로 나누면 됩니다. 초(second)을 분(minute)으로 환산하려면 60 으로 나누면 됩니다. 따라서 초를 도로 환산하려면 60*60 으로 나누어야 합니다.

```
d, m, s = 41, 24, 12.2
d + m / 60 + s / (60*60)
```
```
41.40338888888889
```

- 두 부동소수점 숫자 a=0.7, b=0.1*7 가 있습니다. 이 두 값을 비교하고 싶은데, 비교해 보니 두 숫자가 다르다고 나옵니다. 그 이유는 본문에서 언급한 바와 같이 부동소수점 오차에 의한 것입니다. 두 값을 어떻게 비교할 수 있는지 생각해 보길 바랍니다.

  * 두 값은 같다고 봐야 할까요, 다르다고 봐야 할까요?

```
a = 0.7
```

```
b = 0.1 * 7
a - b
```

```
-1.1102230246251565e-16
```

두 값은 이론적으로는 같아야 합니다. 이처럼 컴퓨터는 연산상의 오차로 인해 다르다고 판단하지만, 이론적인 판단을 원한다면 같습니다.

*** 같다고 하면, 어떻게 판단해야 할까요?**

허용 오차를 정해 두고 차이의 절대값이 허용 오차보다 작다면 같은 값으로 취급할 수 있습니다.

```
a = 0.7
b = 0.1 * 7
e = 1e-15    # 허용 오차
abs(a - b) < e
```

```
True
```

이와 같은 비교를 대신해주는 함수 `isclose()`가 모듈 `math` 에 있습니다.

```
import math

math.isclose(a, b)
```

```
True
```

`isclose()` 함수는 기본 값으로 허용 오차를 1e-09 로 설정하고 있습니다. 허용 오차를 변경하려면 `abs_tol` 인수에 값을 넘겨주면 됩니다.

```
math.isclose(a, b, abs_tol=1e-15)
```

```
True
```

노트북_08. 기초 자료형

실행과제

- `'gslee:5284:9010'` 문자열이 있을 때, 이 문자열을 `'gslee-5284-9010'`으로 바꾸는 방법을 두 가지 이상 찾아보세요.

```
s = 'gslee:5284:9010'
print(s.replace(':', '-'))        # 첫 번째 방법
print('-'.join(s.split(':')))     # 두 번째 방법
```

```
gslee-5284-9010
gslee-5284-9010
```

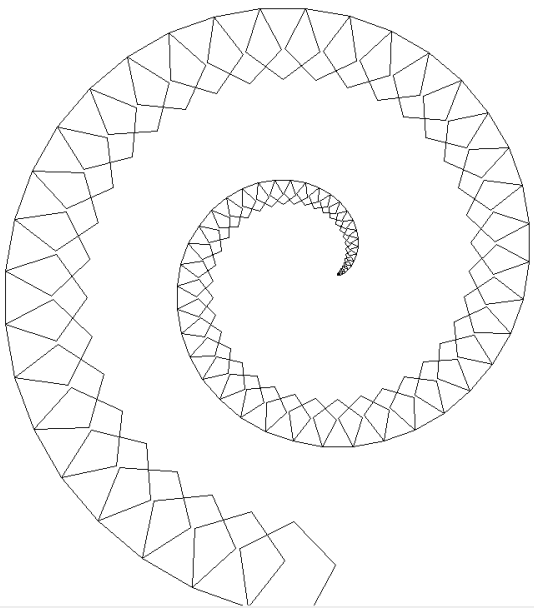
노트북_09. 터틀 그래픽과 함수 기초

실행과제

- `drawPolygon()` 함수와 `for` 문을 함께 이용해서 멋진 그림을 그려 보세요.

〈풀이(예시)〉

```
>>> import turtle
>>> def drawPolygon(t, n, length):
    for i in range(n):
            t.forward(length)
            t.left(360/n)

>>> t = turtle.Turtle('turtle')
>>> t.speed(0)
>>> for i in range(72):
    drawPolygon(t, 5, i+1)
    t.forward(i+1)
    t.left(10)
```



- 외부 변수가 함수에서 사용될 경우 어떠한 문제가 발생할 수 있는지 정리해 보세요.

함수 내부의 실행 논리가 외부 환경에 따라 좌우되는 것이 가장 큰 문제입니다. 이것은 실행 논리를 합리적으로 추적하기 어렵게 합니다. 문제가 생긴다면 원인을 함수 내에서만 찾을 수 없고 함수를 호출하는 환경까지도 점검해야 하므로 디버깅을 굉장히 어렵게 합니다. 따라서 함수에 있어서 사용되는 모든 값들은 인수를 통해 전달되는 것이 바람직합니다.

- 함수와 메서드의 차이는 무엇일까요?

함수와 메서드 모두 코드의 모음입니다. 함수는 인수를 통해 값을 전달하며 값을 반환할 수 있습니다. 함수에 전달되는 모든 데이터는 명시적으로 전달됩니다. 반면 메서드는 객체와 연결된 이름으로 호출되는 코드 모음입니다(`객체.메서드()` 형식). 메서드는 클래스 내에 저장된 데이터 및 객체에 저장된 데이터를 이용하여 작업할 수 있습니다.

노트북_10. 파이썬 만능 키, 모듈

[확인문제 1]

- 다음 형식으로 파일의 파일 크기(바이트 수)를 알 수 있습니다. 다음 코드는 계산기 파일의 크기를 확인합니다. 직접 결과를 확인하길 바랍니다.

```
import os
os.path.getsize('c:/Windows/System32/calc.exe')
```

```
27648
```

- `from os import path` 형식으로 임포트 한 후, `getsize()` 메서드를 사용해서 계산기 파일의 크기를 확인해 보세요.

```
from os import path
path.getsize('C:/Windows/System32/calc.exe')
```

```
27648
```

- 마지막으로 `from os.path import getsize` 형식으로 임포트 한 후, `getsize()` 메서드를 사용해서 계산기 파일의 크기를 확인해 보세요.

```
from os.path import getsize
path.getsize('C:/Windows/System32/calc.exe')
```

```
27648
```

노트북_11. 모듈 익히기_math & cmath

[확인문제 1]

- 몇 개의 각도에 대해서 $cos(x)$ 값을 계산해 보세요.

```
import math

for angle in [30, 45, 60, 90, 120, 180]:
    rad = math.radians(angle)
    print(angle, math.cos(rad))
```

```
30 0.8660254037844387
45 0.7071067811865476
60 0.5000000000000001
90 6.123233995736766e-17
120 -0.4999999999999998
180 -1.0
```

- 몇 개의 라디안에 대해서 $sin(x) + 0.5 \times sin(2 \times x)$ 값을 계산해 보세요.

```
from math import sin

for x in [math.pi/6, math.pi/4, math.pi/3, math.pi/2, math.pi]:
    print(x, sin(x) + 0.5*sin(2*x))
```

```
0.5235987755982988 0.9330127018922192
0.7853981633974483 1.2071067811865475
1.0471975511965976 1.299038105676658
1.5707963267948966 1.0
3.141592653589793 0.0
```

[확인문제 2]

- 바이러스의 1 일 단위 감염재생산지수가 1.17 이고 현재 감염자 수가 1,000 명일 때, 7 일 후에 새로 감염되는 사람의 수는 몇 명일지 예측해 보세요.

```
N = 1000
r = 1.17
print(int(N * pow(r, 7)))
```

```
3001
```

3001 명입니다.

[확인문제 3]

- 다음 식 $e^{i\pi} + 1 = 0$이 성립함을 직접 확인해 보세요.

`isclose()` 함수를 이용해 절대값 1e-15 오차 범위 내에서 근사함을 확인합니다.

```
from cmath import exp, pi, isclose

isclose(exp(complex(0, pi)) + 1, 0, abs_tol=1e-15)
```

```
True
```

실행과제

- 이차방정식의 해를 구하는 코드를 작성해 보세요. 실근, 중근, 허근을 직접 계산해 보세요.

```
import math
import cmath

def findRoot(a, b, c):
    d = b*b - 4*a*c
    if d >= 0:     # 중근 혹은 실근일 때
        r1 = (-b + math.sqrt(d)) / (2*a)
        r2 = (-b - math.sqrt(d)) / (2*a)
    else:          # 허근일 때
        r1 = (-b + cmath.sqrt(d)) / (2*a)
        r2 = (-b - cmath.sqrt(d)) / (2*a)
    return r1, r2

r1, r2 = findRoot(1, 2, 3)
print(r1, r2)
r1, r2 = findRoot(1, 6, 3)
print(r1, r2)
```

```
(-1+1.4142135623730951j) (-1-1.4142135623730951j)
-0.5505102572168221 -5.449489742783178
```

- 삼각함수 sin 의 도함수는 cos 입니다. 이때 수치적으로 다음 수식이 올바른지 몇 개 구간에서 직접 확인해 보세요.

$$\frac{\mathrm{d}}{\mathrm{d}x} sin\, x = cos\, x$$

도함수는 짧은 구간(dx)에서의 함수 기울기입니다. 다음은 dx 는 1e-9 으로 설정하고, 도함수와 $cos(x)$와의 오차는 1e-6 로 설정해서 테스트한 결과입니다. 0 도에서 360 사이를 20 도 간격으로 라디안으로 변환해 테스트해 봅니다.

```python
from math import sin, cos, pi, isclose

def numerical_differentiation(f, x, dx=1e-9):
    dx2 = dx / 2
    return (f(x+dx2) - f(x-dx2)) / dx

for ang in range(0, 360, 20):
    rad = math.radians(ang)
    diff = numerical_differentiation(sin, rad)
    r = isclose(diff, cos(rad), abs_tol=1e-6)
    print(ang, r, diff, cos(rad))
```

```
0 True 1.0 1.0
20 True 0.9396925459981275 0.9396926207859084
40 True 0.7660444500956487 0.766044443118978
60 True 0.5000000413701855 0.5000000000000001
80 True 0.17364820692478133 0.17364817766693041
100 True -0.17364820692478133 -0.1736481776669303
120 True -0.5000000413701855 -0.4999999999999998
140 True -0.7660445611179512 -0.7660444431189779
160 True -0.9396926570204299 -0.9396926207859083
180 True -1.000000082740371 -1.0
200 True -0.9396926570204299 -0.9396926207859084
220 True -0.7660444500956487 -0.766044443118978
240 True -0.5000000413701855 -0.5000000000000004
260 True -0.17364820692478133 -0.17364817766693033
280 True 0.17364820692478133 0.17364817766692997
300 True 0.5000000413701855 0.5000000000000001
320 True 0.7660445611179512 0.7660444431189778
340 True 0.9396927125315812 0.9396926207859084
```

함수의 인수로 또 다른 함수를 넘겨주는 것도 가능합니다. 여기에서는 `sin` 함수가 `numerical_differentiation` 함수의 첫 인수로 전달되었습니다.

노트북_12. 파이썬 연산자

[확인문제 1]

- 문자 '희'와 '히' 중 어느 쪽이 나중에 오는(큰) 문자일까요?

```
'희' < '히'
```

```
True
```

'히'가 큽니다.

[확인문제 2]

- 다음 식의 결과를 예측해 보세요.

```
1 or 0
[ ] or 1
1 and 2
not([ ] or 0)
```

```
1 or 0
```

```
1
```

```
[] or 1
```

```
1
```

```
1 and 2
```

```
2
```

```
not([] or 0)
```

```
True
```

- age 변수에 나이가 저장되어 있을 때, 20 대인지 아닌지 판단하는 식을 만들어 보세요.

#풀이 1

```
age = 25
f_twenties = 20 <= age and age < 29
```

```
print(f_twenties)
```

```
True
```

#풀이 2

```
age = 25
f_twenties = 20 <= age and age < 29
print(f_twenties)
```

```
True
```

#풀이 3

```
age = 25
f_twenties = age // 10 == 2
print(f_twenties)
```

```
True
```

- **어떤 값 x 에 대해, "x > 3 또는 x < -9 이다"라는 명제의 참/거짓을 판단해 보세요.**

'또는'은 연산자 `or` 로 판단합니다. `x` 가 1 일 때를 예시로 보겠습니다.

```
x = 1
x > 3 or x < -9
```

```
False
```

`x` 가 3 보다 크지 않고, -9 보다 크지도 않으므로 결과는 False, 즉 거짓입니다.

```
x = -11
x > 3 or x < -9
```

```
True
```

`x` 가 3 보다 크지 않지만, -9 보다 작으므로 이번에는 True, 즉 참입니다.

- **변수 c 에 주어진 문자가 알파벳인지, 숫자인지 판단하는 코드를 작성해 보세요.**

`in` 연산자를 사용해 `c` 에 주어진 문자가 '알파벳'에 속하는지 검사할 수 있습니다.

```
c = 'a'
c.upper() in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

```
True
```

`in` 연산자의 오른쪽 항에 `0~9` 까지의 정수 문자열을 넣으면, '숫자'에 속하는지 검사하는 코드가 됩니다.

```
c in '0123456789'
```

```
False
```

문자열 메서드를 이용하면 더 편리합니다. `isalpha()`입니다.

```
c = 'abc'
c.isalpha()
```

```
True
```

숫자인지 검사해 주는 메서드는 `isnumeric()`입니다.

```
c = '123'
c.isnumeric()
```

```
True
```

실행과제

- 다음 표에, 각 연산 범주에 해당하는 연산자 두 개씩을 써 넣어 보세요.

비교 연산자	<, <=, >, >=, == , !=
논리 연산자	and, or, not
멤버십 연산자	in, not in
식별 연산자	is, is not

- P→Q 연산은 논리식으로 `not (P and not Q)`로 표현됩니다. 이 논리식이 맞는지 파이썬 코드로 직접 확인해 보세요. 그리고 나중에 쉽게 활용할 수 있도록, 함수 `ifthen` 으로도 만들어 보세요.

`not (P and not Q)` 논리식은 코드로 다음과 같이 표현할 수 있습니다. `format()` 메서드를 이용해 결과를 알아보기 쉽게 출력했습니다.

```
for P in [True, False]:
    for Q in [True, False]:
        print('{}->{}: {}'.format(P, Q, not (P and not Q)))
```

```
True->True: True
True->False: False
False->True: True
False->False: True
```

`not (P and not Q)`를 돌려주는 함수 `ifthen` 을 `def` 구조로 정의해 봅니다.

```
def ifthen(P, Q):
    return not (P and not Q)

for P in [True, False]:
```

```
    for Q in [True, False]:
        print('{}->{}: {}'.format(P, Q, ifthen(P, Q)))
```

```
True->True: True
True->False: False
False->True: True
False->False: True
```

노트북_13. 복합 자료형 (1)

[확인문제]

- 리스트와 튜플의 특징을 비교·대조하여 표로 정리해 보세요.

리스트	구분	튜플
변경 가능	변경 여부	변경 불가능
변경 관련 다양한 메서드 있음	변경 메서드 유무	몇 개 없음
자료를 모으거나 수정할 때	용도	고정된 자료를 표현할 때 함수의 인수, 반환값을 나타낼 때
많음	메모리 사용량	많지 않음 (효율적)
느림	반복 속도	빠름

실행과제

- `L = [1,2,3,4,5]`일 때, 다음 두 코드의 결과가 어떻게 다를지 확인해 봅시다.

코드 1

```
L = [1, 2, 3, 4, 5]
L[1:3] = [100]
L
```

```
[1, 100, 4, 5]
```

코드 1 에서는 슬라이싱을 통해 리스트 L 의 두 번째, 세 번째 요소가 100 으로 치환됩니다.

코드 2

```
L = [1, 2, 3, 4, 5]
L[1:1] = [100]
L
```

```
[1, 100, 2, 3, 4, 5]
```

코드 2 에서는 슬라이싱의 시작 값과 끝 값이 모두 1 이므로, 요소가 대치되지 않고 리스트의 두 번째 자리에 100 이 추가됩니다.

- 다음 코드에서 b 와 c 의 차이는 무엇일까요?

b 는 중첩 리스트 없이 a 요소들을 세 번 반복하는 반면, c 는 a 리스트 전체를 요소로 가집니다.

```
a = [1,2,3]
b = a * 3
c = [a] * 3

print(b)
print(c)
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
[[1, 2, 3], [1, 2, 3], [1, 2, 3]]
```

- 다음 코드를 수행한 후, b 와 c 는 어떤 값을 갖는지 확인해 보세요.

```
a = [1,2,3]
b = a * 3
c = [a] * 3
a[0] = 0

print(b)
print(c)
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
[[0, 2, 3], [0, 2, 3], [0, 2, 3]]
```

a[0] = 0 은 [1,2,3]의 첫 번째 요소 1 을 0 으로 치환합니다. 단순히 a 가 가진 요소를 반복하기만 한 b 에는 영향이 없지만, a 리스트 자체를 요소로 가진 c 에는 이 변경이 반영되었습니다.

노트북_14. 복합 자료형 (2)

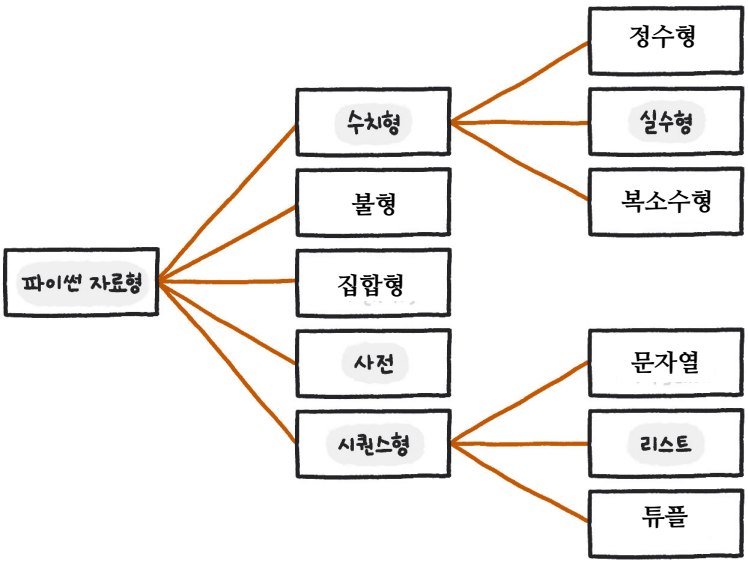
[확인문제 1]

- 리스트와 튜플, 집합의 특징을 비교해 보세요.

	변경 가능 여부	순서 유무	인덱싱/슬라이싱	중복 허용 여부
리스트	O	O	O	O

튜플	X	O	O	O
집합	O	X	X	X

- 다음 도식의 빈 곳에 적절한 자료형 이름을 써 넣어 보세요.



[확인문제 2]

- 사전의 키로 사용될 수 있는 자료형에는 어떤 것들이 있을까요?

사전의 키로는 변경 불가능한 자료형만이 올 수 있습니다. 따라서 숫자, 문자열, 튜플은 올 수 있지만 리스트, 사전등은 올 수 없습니다.

- 사전을 이용해서 'a', 'b', 'c', 'd'를 'w', 'x', 'y', 'z'로 상호 변환하는 프로그램을 작성해 보세요. 예를 들어 'cabsz'는 'ywxsd'로 변환되어야 합니다.

먼저 'abcd'의 각 문자를 'wxyz'의 각 문자와 짝지은 tran 사전을 만든 후, 두 문자열의 순서를 바꾸어 사전을 업데이트합니다.

```
tran = dict(zip('abcd', 'wxyz'))
tran.update(zip('wxyz', 'abcd'))
tran
```

```
{'a': 'w',
 'b': 'x',
 'c': 'y',
 'd': 'z',
 'w': 'a',
 'x': 'b',
 'y': 'c',
```

```
 'z': 'd'}
```

사전이 완성되었으니, `join()` 메서드를 이용해 `tran` 사전에 `'cabsz'`를 넣어 변환해 봅시다.

```
''.join([tran.get(c, c) for c in 'cabsz'])
```

```
'ywxsd'
```

노트북_15. 복합 자료형의 이해 _by.자동화

[확인문제]

- `typewrite()`, `hotkey()`, `press()` 메서드의 용도를 각각 구분해 보세요.

`typewrite()` 혹은 `write()`: `write('Hi!')`와 같이 주어진 문자열을 그대로 입력합니다.
`hotkey()`: `hotkey('ctrl', 'shift', 'esc')` 와 같이 순서대로 눌리지만 함께 눌려야 할 키 조합을 입력합니다.
`press()`: `press('f1')` 과 같이 특수 키를 입력할 때 사용합니다. `press(['left', 'left', 'left'])` 와 같이 리스트를 이용하면 순서대로 키를 누르게 할 수도 있습니다.

더 다양한 함수들과 자세한 설명은 웹페이지 https://pyautogui.readthedocs.io/en/latest/keyboard.html 를 참고해 보시기 바랍니다.

실행과제

- 이 내용을 바탕으로 계산기를 자동 실행하고 필요한 연산을 한 후, 일정 시간 기다린 후에 자동적으로 계산기 윈도우 창을 닫는 코드를 작성해 보세요.

```
import subprocess
import time
import pyautogui

subprocess.call(['calc.exe'])  # 계산기 시작
time.sleep(1)                  # 잠시 대기

title = '계산기'
wins = pyautogui.getWindowsWithTitle(title)    # 계산기를 제목으로 하는 윈도우 찾기
if wins:                       # 찾았으면
```

```
    wins[0].activate()          # 활성화 한 후
    pyautogui.write('12*34=') # 계산하기
    time.sleep(1)
    wins[0].close()             # 창 닫기
```

노트북_16. 제어문의 이해 (1)
_순차문, 선택문, 반복문

[확인문제 1]

- 수축기 혈압(high)과 이완기 혈압(low)을 알고 있을 때 건강 상태를 판단하는 `bloodPressureStatus(high, low)` 함수를 빈 칸을 채워 완성해 보세요. 그리고 `bloodPressureStatus` 함수를 실제로 호출해 보세요.

```
def bloodPressureStatus(high, low):
    if high < 120 and low < 80:
        status = '정상'
    elif 120 <= high <= 139 or 80 <= low <= 89:
        status = '고혈압 전단계'
    elif 140 <= high <= 159 or 90 <= low <= 99:
        status = '1 기 고혈압'
    elif 160 <= high <= 180 or 100 <= low <= 110:
        status = '2 기 고혈압'
    elif high > 180 or low > 110:
        status = '고혈압 위기'
    else:
        status = '???'
    return status
```

[확인문제 2]

- `result = x if a > b else y` 는 `result = (y, x)[a>b]`와 왜 동일한 표현식인지 설명해 보세요.

```
x if a > b else y
```

위 문은 a > b 일 경우 x 를 취하고 아니면 y 를 취합니다.

```
(y, x)[a>b]
```

위 문에서 a 〉 b 일 경우는 True(1)의 값을 갖고, 아니면 False(0)을 취합니다. 이 불(bool)값은 (y, x) 튜플의 인덱싱으로 사용되어 위의 구문과 동일한 결과를 냅니다.

[확인문제 3]

- `for` 문을 이용하여 `print('Hello')`를 10 회 반복해 보세요.

```
for i in range(10):
    print('Hello')
```

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

[확인문제 4]

- `while` 문을 이용하여 1 부터 20 까지의 홀수를 출력해 보세요.

```
i = 1
while i <= 20:
    print(i)
    i += 2
```

```
1
3
5
7
9
11
13
15
17
19
```

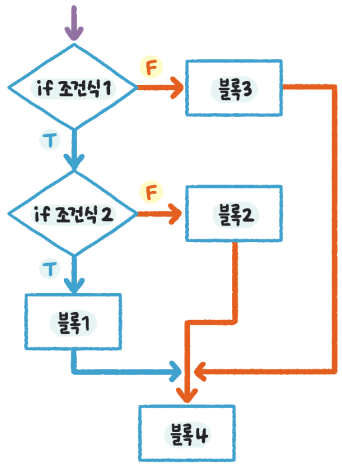
실행과제

- 다음 `minimal_three(a,b,c)` 함수는 세 값 중 가장 작은 값을 찾아내는 함수입니다. 그런데 `minimal_three(2,3,1)`을 호출할 때에는 정상 동작하지 않습니다. 이유를 찾아보세요.

```
def minimal_three(a,b,c):
    if a < b:
        if a < c:
            return (a)
    elif b < a:
        if b < c:
            return (b)
    elif c < a:
        if c < b:
            return (c)
    else:
        return '???'
```

프로그램은 위에서 아래로 진행합니다. 만일 `elif c < a` 부분부터 실행된다면 올바른 결과가 나오겠지만, 이 코드대로라면 `if a < b` 부터 진행하게 됩니다. `minimal_three(2, 3, 1)` 을 호출했을 때, **`elif b < a`** 가 False 이므로 `if` 문 블록 전체를 빠져나가게 됩니다. 즉, 실행되는 `else` 파트가 없습니다. 따라서 올바른 결과(출력)를 내지 못합니다.

- 다음 순서도와 동일한 실행 흐름을 가지도록, if 문의 구조를 작성해 보세요.




```
if 조건식 1:
    if 조건식 2:
        블록 1
    else:
        블록 2
else:
    블록 3
블록 4
```

노트북_17. 제어문의 이해 (2)
_오류 대처와 예외처리문

실행과제

- 'a.txt' 파일을 여는 `open('a.txt')` 작업을 수행하려고 합니다. `FileNotFoundError` 오류가 발생하는 경우 `open('b.txt')`를 실행하도록 `try...except` 구조를 이용하여 구현해 보세요.

```
try:
    open('a.txt')
except FileNotFoundError:
    open('b.txt')
```

- 여러분이 정의하는 사용자 정의 예외를 만들고, `raise` 와 `except` 문을 통해서 예외를 처리해 보세요.

〈예시〉

```
class MyException(Exception):
    pass

try:
    raise MyException
except MyException:
    print('ok')
```

```
ok
```

노트북_18. for 문, 반복의 모든 것

[확인문제 1]

- `range()` 함수와 `for` 문을 이용하여 문자 `'-'`를 50 개 연속으로 붙여서 한 줄에 출력해 보세요. (힌트: `end=''` 옵션을 이용하세요.)

```
for i in range(50):
    print('-', end='')
```

```
--------------------------------------------------
```

- `range()` 함수와 `for` 문을 이용하여 문자열 `'-='`를 25 회 반복해서 한 줄에 출력해 보세요.

```
for i in range(25):
    print('-=', end='')
```

```
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
```

- range() 함수와 for 문을 이용하여 숫자를 1 부터 10 까지 출력해 보세요. 숫자와 숫자 사이는 공백 대신 문자 '_'로 표시하세요.

```
for i in range(1, 10):
    print(i, end='_')
print(i+1)
```

```
1_2_3_4_5_6_7_8_9_10
```

[확인문제 2]

- range() 함수와 for 문을 이용하여 1 부터 10 사이의 홀수를 출력해 보세요.

```
for i in range(1, 10, 2):
    print(i, end=' ')
```

```
1 3 5 7 9
```

- range() 함수와 for 문을 이용하여 10 부터 1 까지의 자연수를 출력해 보세요.

```
for i in range(10, 0, -1):
    print(i, end=' ')
```

```
10 9 8 7 6 5 4 3 2 1
```

- range() 함수와 for 문을 이용하여 0.1 부터 1.0 사이의 값을 0.2 간격으로 출력해 보세요

```
for i in range(1, 11, 2):
    print(i/10, end=' ')
```

```
0.1 0.3 0.5 0.7 0.9
```

[확인문제 3]

- 은행 이자가 복리 연 2%일 때, 원금 100 만원이 10 년 동안 얼마가 되는지 확인해 보세요.

```
원금 = 100
복리 = 0.02 + 1.0     # 1.02

for i in range(11):
    print('{:2d}년후, {:.1f}만원'.format(i, 원금 * pow(복리, i)))
```

```
 0 년후, 100.0 만원
 1 년후, 102.0 만원
 2 년후, 104.0 만원
 3 년후, 106.1 만원
 4 년후, 108.2 만원
 5 년후, 110.4 만원
 6 년후, 112.6 만원
 7 년후, 114.9 만원
 8 년후, 117.2 만원
 9 년후, 119.5 만원
10 년후, 121.9 만원
```

- 은행 이자가 복리 연 3%일 때, 원금 100 만원이 10 년 동안 얼마가 되는지 확인해 보세요.

```
원금 = 100
복리 = 1.03

for i in range(11):
    print('{:2d}년후, {:.1f}만원'.format(i, 원금 * pow(복리, i)))
```

```
 0 년후, 100.0 만원
 1 년후, 103.0 만원
 2 년후, 106.1 만원
 3 년후, 109.3 만원
 4 년후, 112.6 만원
 5 년후, 115.9 만원
 6 년후, 119.4 만원
 7 년후, 123.0 만원
 8 년후, 126.7 만원
 9 년후, 130.5 만원
10 년후, 134.4 만원
```

- 은행 이자가 복리 연 5%일 때, 원금 100 만원이 10 년 동안 얼마가 되는지 확인해 보세요.

```
원금 = 100
복리 = 1.05

for i in range(11):
    print('{:2d}년후, {:.1f}만원'.format(i, 원금 * pow(복리, i)))
```

```
 0 년후, 100.0 만원
 1 년후, 105.0 만원
 2 년후, 110.2 만원
 3 년후, 115.8 만원
 4 년후, 121.6 만원
 5 년후, 127.6 만원
 6 년후, 134.0 만원
 7 년후, 140.7 만원
 8 년후, 147.7 만원
 9 년후, 155.1 만원
10 년후, 162.9 만원
```

실행과제

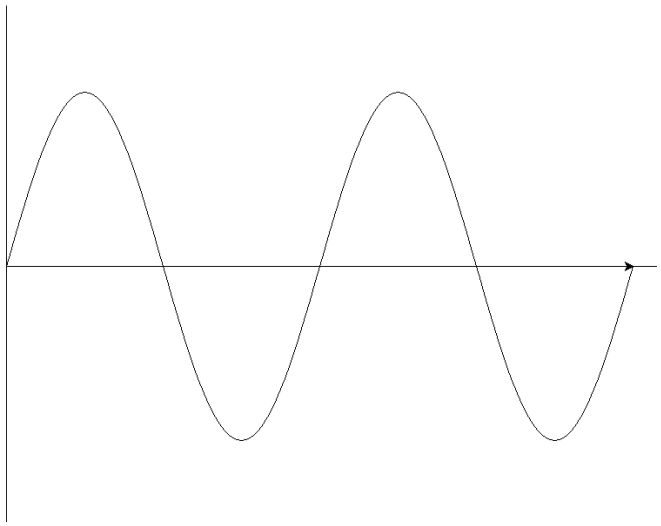
- 0 도부터 360 도까지 10 도 간격으로 사인(sin) 표를 만들어 보세요. (힌트: `math` 모듈의 `sin` 함수는 기본적으로 라디안 단위를 사용합니다.)

```
import math

for ang in range(0, 360+1, 10):
    rad = math.radians(ang)
    print('sin({:3d}) = {:.2f}'.format(ang, math.sin(rad)))
```

```
sin(  0) = 0.00
sin( 10) = 0.17
sin( 20) = 0.34
sin( 30) = 0.50
sin( 40) = 0.64
sin( 50) = 0.77
sin( 60) = 0.87
sin( 70) = 0.94
sin( 80) = 0.98
sin( 90) = 1.00
sin(100) = 0.98
sin(110) = 0.94
sin(120) = 0.87
sin(130) = 0.77
sin(140) = 0.64
sin(150) = 0.50
sin(160) = 0.34
sin(170) = 0.17
sin(180) = 0.00
sin(190) = -0.17
sin(200) = -0.34
sin(210) = -0.50
sin(220) = -0.64
sin(230) = -0.77
sin(240) = -0.87
sin(250) = -0.94
sin(260) = -0.98
sin(270) = -1.00
sin(280) = -0.98
sin(290) = -0.94
sin(300) = -0.87
sin(310) = -0.77
sin(320) = -0.64
sin(330) = -0.50
sin(340) = -0.34
sin(350) = -0.17
sin(360) = -0.00
```

- 터틀 그래픽을 이용해서 0~범위 안에서 sin 그래프를 그려 보세요. (힌트: `goto(x, y)`, `up()`, `down()` 같은 함수를 이용하면 도움이 됩니다.) 좌표 축도 함께 그려 보세요.

```
import turtle
import math

height = 300
width = 500
x_start = -300
y_scale = 200
t1 = turtle.Turtle()
t1.speed(0)

t1.up()
t1.goto(x_start, 0)
t1.down()
t1.goto(width, 0)
t1.up()
t1.goto(x_start, -height)
t1.down()
t1.goto(x_start, height)
t1.up()
t1.goto(x_start, 0)
t1.down()

for ang in range(0, 360*2+1, 5):
    rad = math.radians(ang)
    t1.goto(x_start + ang, y_scale * math.sin(rad))
```



노트북_19. for 문의 수학적인 활용 (수열과 급수연산)

[확인문제 1]

- 파이썬으로 다음 수열을 만들어 보세요.

$$1, \frac{1}{3}, \frac{1}{5}, \cdots, \frac{1}{19}$$

```
for i in range(1, 20, 2):
    print('1/{:2d}={:.4f}'.format(i, 1/i))
```

```
1/ 1=1.0000
1/ 3=0.3333
1/ 5=0.2000
1/ 7=0.1429
1/ 9=0.1111
1/11=0.0909
1/13=0.0769
1/15=0.0667
1/17=0.0588
1/19=0.0526
```

$$1, \frac{1}{3^2}, \frac{1}{5^2}, \cdots, \frac{1}{19^2}$$

```
sign = 1
for i in range(1, 20, 2):
    print('1/({:2d}^2)={:.4f}'.format(i, sign*1/(i*i)))
    sign *= -1
```

```
1/( 1^2)=1.0000
1/( 3^2)=-0.1111
1/( 5^2)=0.0400
1/( 7^2)=-0.0204
1/( 9^2)=0.0123
1/(11^2)=-0.0083
1/(13^2)=0.0059
1/(15^2)=-0.0044
1/(17^2)=0.0035
1/(19^2)=-0.0028
```

$$1, -\frac{1}{3^2}, \frac{1}{5^2}, \cdots, -\frac{1}{19^2}$$

```
sign = 1
for i in range(1, 20, 2):
```

```
    print('1/({:2d}^2)={:.4f}'.format(i, sign*1/(i*i)))
    sign *= -1
```

```
1/( 1^2)=1.0000
1/( 3^2)=-0.1111
1/( 5^2)=0.0400
1/( 7^2)=-0.0204
1/( 9^2)=0.0123
1/(11^2)=-0.0083
1/(13^2)=0.0059
1/(15^2)=-0.0044
1/(17^2)=0.0035
1/(19^2)=-0.0028
```

[확인문제 2]

- 1 부터 100 사이의 홀수의 합을 계산해 보세요. (힌트: `range(1, 100, 2)`를 이용하세요.)

```
acc = 0
for i in range(1, 100, 2):
    acc += i
acc
```

```
2500
```

- 100 이하 자연수의 7 의 배수의 합을 계산해 보세요.

```
acc = 0
for i in range(7, 100, 7):
    acc += i
acc
```

```
735
```

- 1 부터 20 이하의 모든 수를 곱해 보세요. 얼마나 큰 수가 나올까요? (힌트: 덧셈 연산 대신 곱셈 연산을 적용하면 됩니다. 덧셈의 항등원은 0 이지만 곱셈에 대한 항등원은 1 인 것에 주의하시기 바랍니다.)

```
fac = 1
for i in range(1, 20):
    fac *= i
fac
```

```
121645100408832000
```

[확인문제 3]

- 다음 수식의 값을 직접 확인해 보세요.

$$\zeta(3) = 1 + \frac{1}{2^3} + \frac{1}{3^3} + \cdots = 1.202$$

```
acc = 0
for i in range(1, 1000000):
    acc += 1/pow(i,3)
acc
```

```
1.202056903150321
```

- 다음 수식의 값을 직접 확인해 보세요.

$$\zeta(5) = 1 + \frac{1}{2^5} + \frac{1}{3^5} + \cdots = 1.036$$

```
acc = 0
for i in range(1, 1000000):
    acc += 1/pow(i,5)
acc
```

```
1.036927755143338
```

- 다음 수식의 값을 직접 확인해 보세요. 오차가 어느 정도 되는지도 확인해 보세요.

$$\zeta(8) = 1 + \frac{1}{2^8} + \frac{1}{3^8} + \cdots = \frac{\pi^8}{9450}$$

```
import math

acc = 0
for i in range(1, 1000):
    acc += 1/pow(i,8)
diff = abs(acc - pow(math.pi, 8) / 9450)
diff
```

```
8.881784197001252e-16
```

실행과제

- 본 실행과제에서 제시한 ζ(8) 수식을 1000 항까지 계산해 보되, 일반 부동소수점 연산 결과와 Decimal() 함수를 이용한 연산 결과를 비교해서 어느 정도 오차가 발생하는지 직접 확인해 보세요.

`demical` 모듈을 `from... import...` 문으로 가져옵니다. 정확도를 50 으로 조정해 줍니다.

```
from decimal import *

getcontext().prec = 50
```

$\zeta(8) = 1 + \frac{1}{2^8} + \frac{1}{3^8} + \cdots = \frac{\pi^8}{9450}$ 수식을 일반 부동소수점 연산으로 계산해 봅니다.

```
acc1 = 0
for i in range(1, 1000):
    acc1 += 1/pow(i,8)

acc1
```

```
1.004077356197943
```

$\zeta(8) = 1 + \frac{1}{2^8} + \frac{1}{3^8} + \cdots = \frac{\pi^8}{9450}$ 수식의 Decimal 연산 결과는 다음과 같습니다.

```
acc2 = Decimal('0.0')
for i in range(1, 1000):
    acc2 += Decimal('1')/pow(i,8)

acc2
```

```
Decimal('1.0040773561979443393785418 78')
```

두 연산 값의 차이는 얼마일까요?

```
acc2 - Decimal(acc1)
```

```
Decimal('1.31209573788464048306621441504020 79375E-15')
```

노트북_20. if 문과 함께 for 문 사용하기

[확인문제 1]

- 10 개의 0 혹은 1 값을 난수로 생성하고, 0 의 개수를 세어 보세요.

풀이 1

```
import random

cnt = 0
for i in range(10):
    n = random.randint(0, 1)
    if n == 0:
        cnt += 1
print(cnt)
```

```
5
```

#풀이 2

```
import random

numbers = [random.randint(0, 1) for i in range(10)]
print(numbers.count(0))
```

```
5
```

#풀이 3

```
import random

[random.randint(0, 1) for i in range(10)].count(0)
```

```
5
```

#풀이 4

```
import random

sum([1 for i in range(10) if random.randint(0, 1) == 0])
```

```
5
```

- 문자열에서 한글만 추출하는 코드를 작성해 보세요. (힌트: 한글의 유니코드 범위는 `0xAC00~0xD7FF` 입니다.)

```
s = '풀어 보면 scores 리스트의 score 값(for score in scores)에 대해'
```

풀이 1

```
s = '풀어 보면 scores 리스트의 score 값(for score in scores)에 대해'
hs = ''
for c in s:
    if 0xAC00 <= ord(c) <= 0xD7FF:
        hs += c
print(hs)
```

```
풀어보면리스트의값에대해
```

풀이 2

```
s = '풀어 보면 scores 리스트의 score 값(for score in scores)에 대해'
hs = ''.join([c for c in s if 0xAC00 <= ord(c) <= 0xD7FF])
print(hs)
```

```
풀어보면리스트의값에대해
```

[확인문제 2]

- 10 개의 0 혹은 1 값을 난수로 생성하고, 0 이면 앞면, 1 이면 뒷면이라고 출력해 보세요.

풀이 1

```
import random
```

```
ht = []
for i in range(10):
    if random.randint(0, 1):
        ht.append('뒷면')
    else:
        ht.append('앞면')
print(ht)
```

```
['앞면', '뒷면', '앞면', '앞면', '앞면', '앞면', '앞면', '앞면', '뒷면', '뒷면']
```

풀이 2

```
import random

ht = []
for i in range(10):
    ht.append('뒷면' if random.randint(0, 1) else '앞면')
print(ht)
```

```
['뒷면', '뒷면', '앞면', '앞면', '뒷면', '뒷면', '앞면', '앞면', '뒷면', '뒷면']
```

풀이 3

```
import random

ht = ['뒷면' if random.randint(0, 1) else '앞면' for i in range(10)]
print(ht)
```

```
['뒷면', '뒷면', '뒷면', '앞면', '앞면', '뒷면', '앞면', '뒷면', '뒷면', '뒷면']
```

- 리스트 `scores` 에 학점이 담겨 있다고 할 때, 학점을 A(90-100), B(80-89), C(70-79), F(0-69)로 구분해서, 각 구간에 몇 명이나 있는지 확인하는 코드를 작성해 보세요.

방법 1

```
import random

scores = [random.randint(0, 100) for i in range(100)]
count_A = count_B = count_C = count_F = 0

for s in scores:
    if 90 <= s <= 100:
        count_A += 1
    elif 80 <= s < 90:
        count_B += 1
    elif 70 <= s < 80:
        count_C += 1
    else:
        count_F += 1
print(count_A, count_B, count_C, count_F)
```

```
6 10 2 82
```

방법 2

```
import random

scores = [random.randint(0, 100) for i in range(100)]
count_A = sum([90 <= s <= 100 for s in scores])
count_B = sum([80 <= s < 90 for s in scores])
count_C = sum([70 <= s < 80 for s in scores])
count_F = sum([s < 70 for s in scores])
print(count_A, count_B, count_C, count_F)
```

```
9 15 15 61
```

실행과제

- 다음 주어진 문자열에서 한글만으로 구성된 문자열을 만들어 보세요. 한글은 그대로 모으고, 알파벳은 공백으로 대치합니다. (힌트: 한글의 문자 범위는 '가' <= c <= '힣'을 이용합니다.)

```
s = 'HIV 는 complex retrovirus 로 여러 가지 효소를 virion 내에 포함하고 있기 때문에 단순히 cDNA 를 transfaction 해서는 감염이 되지 않는다.'
```

방법 1

```
h = ''

for c in s:
    if '가' <= c <= '힣':
        h += c
print(h)
```

```
는로여러가지효소를내에포함하고있기때문에단순히를해서는감염이되지않는다
```

방법 2

```
''.join([c for c in s if '가' <= c <= '힣'])
```

```
'는로여러가지효소를내에포함하고있기때문에단순히를해서는감염이되지않는다'
```

- 다음 문자열은 3·1 독립 선언서(三一獨立宣言書)의 첫 문장입니다. 이 문자열에서 한자만 추출해 보 세요. 한자는 그대로 모으고 한글은 공백으로 대치합니다. (힌트: 한자의 문자 범위는 '\u4E00' <= c <= '\u9fff'를 이용합니다.)

```
s = '吾等은 玆에 我 朝鮮의 獨立國임과 朝鮮人의 自主民임을 宣言하노라 此로써 世界萬邦에 告하야 人類平等의 大義를 克明하며 此로써 子孫萬代에 誥하야 民族自存의 正權을 永有케하노라.'
```

```
h = ''

for c in s:
    if '가' <= c <= '힣':
        h += ' '
```

```
    elif '\u4E00' <= c <= '\u9fff':
        h += c

print(h)
```

```
吾等 玆 我朝鮮 獨立國  朝鮮人 自主民  宣言   此  世界萬邦 告  人類平等 大義 克明  此  子孫萬代 誥  民族自存 正權 永有
```

노트북_21. 문자열 형식화 (원하는 형식으로 출력하기)

[확인문제 1]

- 10 진수 100 이 8 진수와 16 진수로는 각각 얼마인지 포맷 문자열로 확인해 보세요.

```
# 방법 1
print('8 진수: {:o}, 16 진수: {:x}'.format(100, 100))
print('8 진수: {0:o}, 16 진수: {0:x}'.format(100))
```

```
8 진수: 144, 16 진수: 64
8 진수: 144, 16 진수: 64
```

```
# 방법 2
print(f'8 진수: {100:o}, 16 진수: {100:x}')
```

```
8 진수: 144, 16 진수: 64
```

- `math.pi` 값을 소수점 5 자리까지 문자열로 출력해 보세요.

```
import math
'{:.5f}'.format(math.pi)
```

```
'3.14159'
```

- 0 부터 200 까지의 정숫값을 50 간격으로 출력해 보세요. 단, 자릿수 3 자리를 확보하고 앞 빈 자리가 있다면 0 을 채워서 출력해 보시기 바랍니다.

```
for i in range(0, 201, 50):
    print('{:03d}'.format(i))
```

```
000
050
100
150
```

```
200
```

- f'' 포맷 문자열을 이용하여 1, 1/2, 1/3, ⋯, 1/10 수열을 형식에 맞게 출력해 보세요.

```
for i in range(1, 11):
    print('1/{:2d}={:.3f}'.format(i, 1/i))
```

```
1/ 1=1.000
1/ 2=0.500
1/ 3=0.333
1/ 4=0.250
1/ 5=0.200
1/ 6=0.167
1/ 7=0.143
1/ 8=0.125
1/ 9=0.111
1/10=0.100
```

[확인문제 2]

- 다음 양식에 format() 메서드를 키워드 인수 방식으로 호출하여, 최저 기온 15 도, 최고 기온 23 도를 알리는 문자열을 만들어 보세요.

```
"오늘 오전 최저 기온은 {temp_low}도이고, 낮 최고 기온은 {temp_high}도입니다.".format(temp_low=15, temp_high=23)
```

```
'오늘 오전 최저 기온은 15 도이고, 낮 최고 기온은 23 도입니다.'
```

- 앞 문제와 같은 양식에 대해 필요한 데이터를 사전에 저장해 두고, 사전을 이용해 format() 메서드를 호출하고 최종 문자열을 만들어 보세요.

```
info = {'temp_low': 15, 'temp_high': 23}
"오늘 오전 최저 기온은 {temp_low}도이고, 낮 최고 기온은 {temp_high}도입니다.".format(**info)
```

```
'오늘 오전 최저 기온은 15 도이고, 낮 최고 기온은 23 도입니다.'
```

실행과제

- 1 부터 100 까지의 정숫값을 10 진수, 8 진수, 16 진수, 2 진수로 각각 출력해 보세요. 출력 열이 보기 좋게 잘 정렬되도록, 출력 자릿수를 조절해 보세요.

```
print('10 진수 8 진수 16 진수')
for i in range(1, 101):
    print(' {0:3d} {0:4o}  {0:3x}'.format(i))
```

```
10진수 8진수 16진수
   1    1    1
   2    2    2
   3    3    3
   4    4    4
   5    5    5
   6    6    6
   7    7    7
   8   10    8
   9   11    9
  10   12    a
  11   13    b
  12   14    c
  13   15    d
  14   16    e
  15   17    f
  16   20   10
  17   21   11
  18   22   12
  19   23   13
  20   24   14
  21   25   15
  22   26   16
  23   27   17
  24   30   18
  25   31   19
...
```

- 1 부터 100 까지의 정숫값을 10 진수 3 자리, 8 진수 3 자리, 16 진수 2 자리, 2 진수 8 자리로 각각 출력해 보세요. 단, 남는 자리는 0 으로 채우도록 합니다.

```
print('10 진수 8 진수 16 진수 2 진수')
for i in range(1, 101):
    print('  {0:3d}  {0:3o}  {0:3x} {0:8b}'.format(i))
```

```
10진수 8진수 16진수 2진수
    1    1    1        1
    2    2    2       10
    3    3    3       11
    4    4    4      100
    5    5    5      101
    6    6    6      110
    7    7    7      111
    8   10    8     1000
    9   11    9     1001
   10   12    a     1010
   11   13    b     1011
   12   14    c     1100
   13   15    d     1101
   14   16    e     1110
   15   17    f     1111
   16   20   10    10000
   17   21   11    10001
   18   22   12    10010
   19   23   13    10011
   20   24   14    10100
   21   25   15    10101
   22   26   16    10110
   23   27   17    10111
   24   30   18    11000
   25   31   19    11001
...
```

- 현재 달러당 원 가격이 1,250 원이라고 합니다. 한화 1,000 원부터 10,000 원까지 천 단위 액수들이 각각 몇 달러 몇 센트(예: $12.56)가 되는지, 표 형식으로 열을 맞추어서 출력해 보세요.

```
wpd = 1250
for won in range(1000, 10000+1, 1000):
    print('{:5d}원 = ${:.2f}'.format(won, won / wpd))
```

```
 1000원 = $0.80
 2000원 = $1.60
 3000원 = $2.40
 4000원 = $3.20
 5000원 = $4.00
 6000원 = $4.80
 7000원 = $5.60
 8000원 = $6.40
 9000원 = $7.20
10000원 = $8.00
```

노트북_22. 리스트에 결과 저장 (출력을 입력으로)

[확인문제 1]

- 다음 수식의 1 부터 100 항까지의 값을 리스트에 저장해 보세요.

$$1, \frac{1}{2^3}, \frac{1}{3^3} \ldots$$

풀이 1

```
data = []
for i in range(1, 101):
    data.append(1/pow(i,3))

data
```

```
[1.0,
 0.125,
 0.037037037037037035,
 0.015625,
 0.008,
...
1.1302806712962962e-06,
 1.0956826815299675e-06,
 1.0624824690392609e-06,
 1.0306101521283646e-06,
```

```
 1e-06]
```

풀이 2

```
data = [1/pow(i,3) for i in range(1, 101)]
data
```

```
[1.0,
 0.125,
 0.037037037037037035,
 0.015625,
 0.008,
...
1.1302806712962962e-06,
 1.0956826815299675e-06,
 1.0624824690392609e-06,
 1.0306101521283646e-06,
 1e-06]
```

[확인문제 2]

- 1 항까지의 합, 2 항까지의 합, 3 항까지의 합... 의 방식으로 다음 급수의 100 항까지의 합을 리스트로 만들어 보세요.

$$1+\frac{1}{2^3}+\frac{1}{3^3}+\ldots$$

풀이 1

```
L = []
acc = 0

for i in range(1, 101):
    acc += 1/pow(i, 3)
    L.append(acc)

print(L)
```

```
[1.0,  1.125,  1.162037037037037,  1.177662037037037,  1.185662037037037,  1.1902916666666665,
1.1932071185617104,  1.1951602435617104,  1.1965319856741932,  1.197531985674193,
1.1982833004750946,  1.1988620041787983,  1.1993171703144379,  1.1996816018013183,
1.1999778980976146,  1.2002220387226146,  1.2004255803468766,  1.200597048110937,
1.2007428419584365,  1.2008678419584364,  1.200975821658253,  1.2010697360083655,
1.2011519255374195,  1.2012242635003825,  1.2012882635003825,  1.2013451592673374,
1.2013959645307628,  1.201441518466623,  1.2014825205577295,  1.2015195575947666,
1.2015531247794868,  1.2015836423576118,  1.2016114688317192,  1.201636911534752,
1.2016602351499124, 1.20168166862042, 1.201701410787715, 1.2017196350186525, 1.2017364930236762,
1.2017521180236763, 1.201766627389472, 1.201780124851949, 1.2017927023608477, 1.2018044416546119,
1.2018154155915117,  1.2018256892826436,  1.2018353210598027,  1.2018443633051732,
1.2018528631649255,  1.2018608631649255,  1.2018684017436019,  1.2018755137144712,
```

```
1.2018822306687356,       1.2018885813266638,       1.201894591845071,       1.2019002860870536,
1.2019056858591832,       1.2019108111205716,       1.201915680167553,       1.2019203097971827,
1.2019247154522816,       1.2019289113503717,       1.201932910598513,       1.2019367252957787,
1.2019403666248638,       1.2019438449341273,       1.2019471698111899,       1.201950350149069,
1.2019533942057006,       1.2019563096575956,       1.2019591036482804,       1.2019617828320939,
1.2019643534138422,       1.201966821184754,       1.2019691915551245,       1.2019714695839916,
1.2019736600061517, 1.2019757672567797, 1.2019777954938968, 1.2019797486188968, 1.20198163029532,
1.2019834439660444,       1.201985192869045,       1.2019868800518547,       1.2019885083848487,
1.2019900805734611, 1.201991599169428, 1.2019930665811487, 1.201994485083239, 1.2019958568253515,
1.2019971838403243,       1.2019984680517157,       1.2019997112807794,       1.2020009152529243,
1.2020020816037043,       1.2020032118843755,       1.202004307567057,       1.2020053700495261,
1.2020064006596782, 1.2020074006596781]
```

풀이 2

본문에는 없는 내용이지만 고급 해법을 소개합니다.

```
from itertools import accumulate

data = [1/pow(i,3) for i in range(1, 101)]
L = list(accumulate(data))

print(L)
```

```
[1.0,  1.125,  1.162037037037037,  1.177662037037037,  1.185662037037037,  1.1902916666666665,
1.1932071185617104,       1.1951602435617104,       1.1965319856741932,       1.197531985674193,
1.1982833004750946,       1.1988620041787983,       1.1993171703144379,       1.1996816018013183,
1.1999778980976146,       1.2002220387226146,       1.2004255803468766,       1.200597048110937,
1.2007428419584365,       1.2008678419584364,       1.200975821658253,       1.2010697360083655,
1.2011519255374195,       1.2012242635003825,       1.2012882635003825,       1.2013451592673374,
1.2013959645307628,       1.201441518466623,       1.2014825205577295,       1.2015195575947666,
1.2015531247794868,       1.2015836423576118,       1.2016114688317192,       1.201636911534752,
1.2016602351499124, 1.20168166862042, 1.201701410787715, 1.2017196350186525, 1.2017364930236762,
1.2017521180236763, 1.201766627389472, 1.201780124851949, 1.2017927023608477, 1.2018044416546119,
1.2018154155915117,       1.2018256892826436,       1.2018353210598027,       1.2018443633051732,
1.2018528631649255,       1.2018608631649255,       1.2018684017436019,       1.2018755137144712,
1.2018822306687356,       1.2018885813266638,       1.201894591845071,       1.2019002860870536,
1.2019056858591832,       1.2019108111205716,       1.201915680167553,       1.2019203097971827,
1.2019247154522816,       1.2019289113503717,       1.201932910598513,       1.2019367252957787,
1.2019403666248638,       1.2019438449341273,       1.2019471698111899,       1.201950350149069,
1.2019533942057006,       1.2019563096575956,       1.2019591036482804,       1.2019617828320939,
1.2019643534138422,       1.201966821184754,       1.2019691915551245,       1.2019714695839916,
1.2019736600061517, 1.2019757672567797, 1.2019777954938968, 1.2019797486188968, 1.20198163029532,
1.2019834439660444,       1.201985192869045,       1.2019868800518547,       1.2019885083848487,
1.2019900805734611, 1.201991599169428, 1.2019930665811487, 1.201994485083239, 1.2019958568253515,
1.2019971838403243,       1.2019984680517157,       1.2019997112807794,       1.2020009152529243,
1.2020020816037043,       1.2020032118843755,       1.202004307567057,       1.2020053700495261,
1.2020064006596782, 1.2020074006596781]
```

[확인문제 3]

- 첫 문자가 'D' 혹은 'd'인 경우만 나오도록, 네모 칸 안을 채워 넣어 식을 완성해 보세요.

```
>>> names = ['C#','D#','Eb','db','Gb','Ab']
>>> new_names = [name for name in names if          ]
>>> new_names
['D#', 'db']
```

다양한 풀이가 가능합니다. 어떻게 해결할 수 있는지 하나씩 잘 살펴보세요.

풀이 1

```
names = ['C#','D#','Eb','db','Gb','Ab']
new_names = [name for name in names if name[0] == 'D' or name[0] == 'd']
new_names
```

```
['D#', 'db']
```

풀이 2

```
names = ['C#','D#','Eb','db','Gb','Ab']
new_names = [name for name in names if name[0] in 'Dd']
new_names
```

```
['D#', 'db']
```

풀이 3

```
names = ['C#','D#','Eb','db','Gb','Ab']
new_names = [name for name in names if name.upper().startswith('D')]
new_names
```

```
['D#', 'db']
```

- 다음 문자열에서 한글 및 알파벳만 추출하는 코드를 작성해 보세요.

```
s = '본 위키는 공신력(public confidence, 公信力)을 얻을 수 없는 위키(wiki)이다.'
```

문자 범위를 이용하는 것과 `isalpha()` 함수를 이용하는 것 두 가지 방법을 소개합니다.

풀이 1

```
result = ''.join([c for c in s if '가' <= c <= '힣' or 'A' <= c.upper() <= 'Z'])
```

```
'본위키는공신력 publicconfidence 公信力을얻을수없는위키 wiki 이다'
```

풀이 2

```
result = ''.join([c for c in s if '가' <= c <= '힣' or c.isalpha()])
result
```

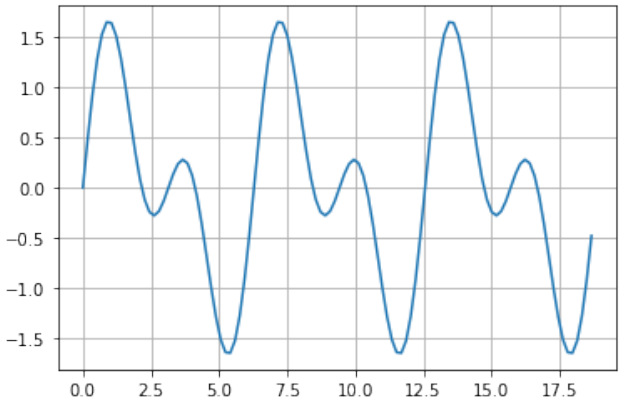
```
'본위키는공신력 publicconfidence 公信力을얻을수없는위키 wiki 이다'
```

노트북_23. 연산 결과 시각화 _Matplotlib (그래프 그리기)

[확인문제]

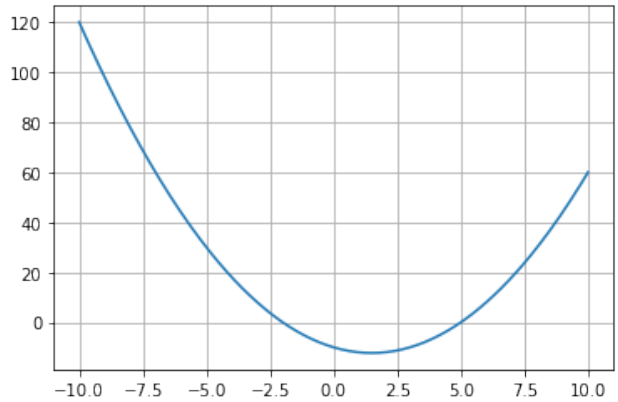
- $sin(x) + 0.9sin(2x)$ 식의 그래프를 numpy 모듈을 이용해서 그려 보세요.

```
import numpy as np
import matplotlib.pyplot as plt

angles = np.arange(0, 360*3, 10)
xs = np.radians(angles)
ys = np.sin(xs) + 0.9*np.sin(2*xs)
plt.plot(xs, ys)
plt.grid()
```



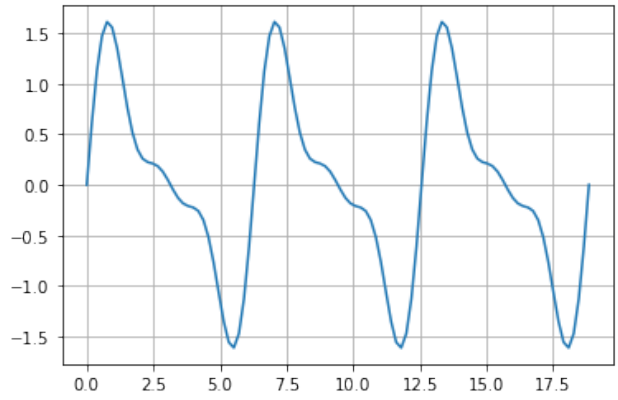
- 2 차 함수 $y = ax^2 + bx + c$ 의 그래프를 $a = 1$, $b = -3$, $c = -10$ 에 대해서 $10 \le x \le 10$ 구간에서 그려 보세요. x 축 간격은 0.1 로 합니다.

```
import numpy as np
import matplotlib.pyplot as plt

def f(a, b, c, x):
    return a*x*x + b*x + c

xs = np.arange(-10, 10+0.1, 0.1)
ys = f(1, -3, -10, xs)
plt.plot(xs, ys)
plt.grid()
```



- 삼각함수 $sin(x) + 0.7sin(2x) + 0.3sin(3x)$의 그래프를 $0 \le x \le 6\pi$ 구간에서 그려 보세요. x 축 구간은 100 개로 합니다. (힌트: `np.linspace()`를 이용하세요.)

```
import numpy as np
import matplotlib.pyplot as plt

xs = np.linspace(0, np.pi*6, 100)
ys = np.sin(xs) + 0.7*np.sin(2*xs) + 0.3*np.sin(3*xs)
plt.plot(xs, ys)
plt.grid()
```



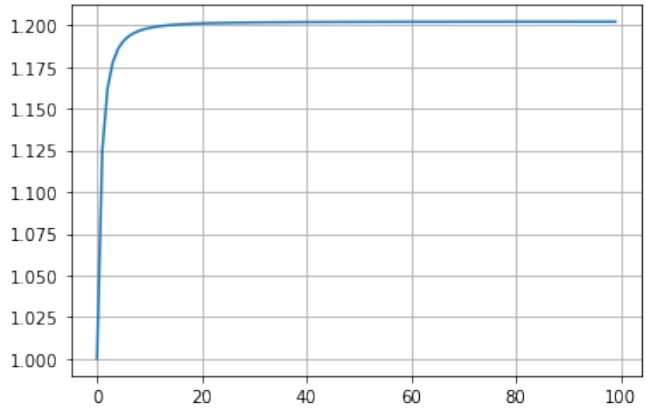
[확인문제 2]

- 다음 수식의 누적 과정을 그래프로 그려 직접 확인해 보세요(100 항까지).

$$\zeta(3) = 1 + \frac{1}{2^3} + \frac{1}{3^3} + \cdots = 1.202\ldots$$

```
import numpy as np
from itertools import accumulate
import matplotlib.pyplot as plt

xs = np.arange(1, 101)
```

```
terms = 1 / pow(xs, 3)
ys = list(accumulate(terms))
plt.plot(ys)
plt.grid()
```

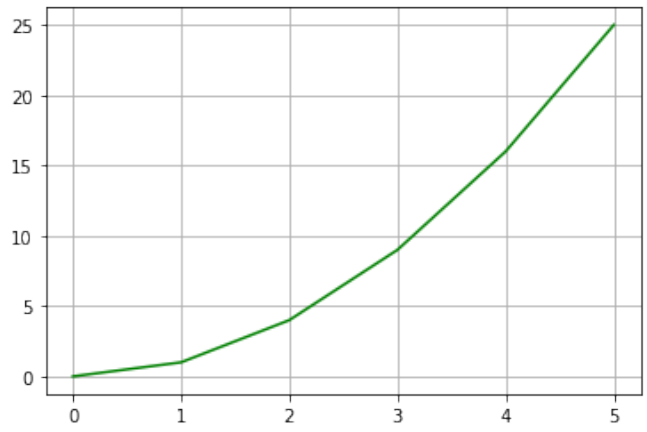


실행과제

- 다음 리스트 x, y 에 대하여 선 그래프를 그려 보세요. 이때 선의 색상은 녹색으로 지정하세요.

```
x = [0, 1, 2, 3, 4, 5]
y = [0, 1, 4, 9, 16, 25]
```

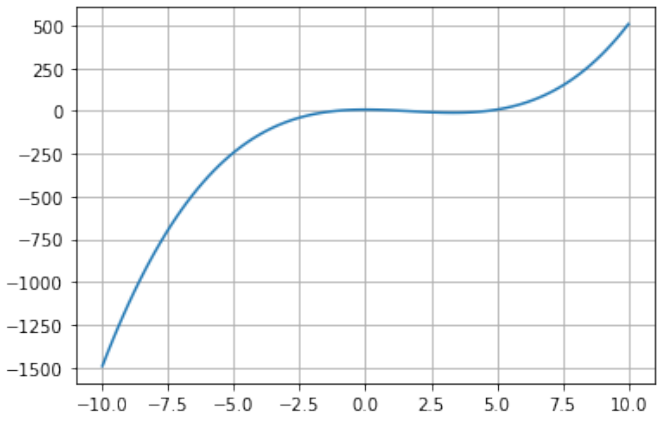
```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5]
y = [0, 1, 4, 9, 16, 25]
plt.plot(x, y, c='g')
plt.grid()
```



- 다음 3 차 함수에 대한 그래프를 $-10 \le x < 10$ 구간에서 그려 보세요.

$$y = x^3 - 5x^2 + 8$$

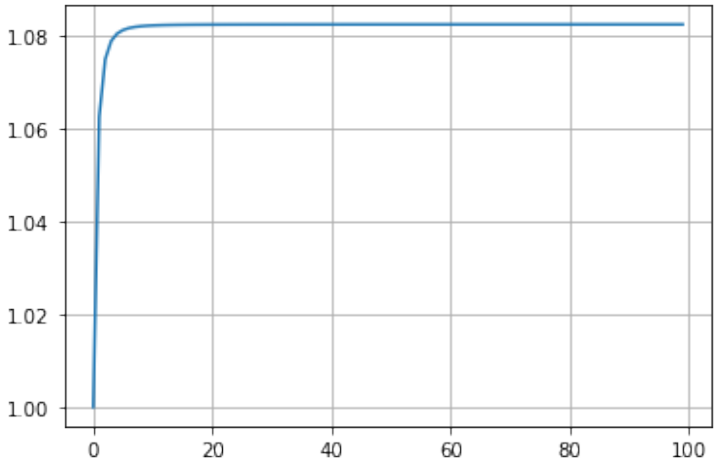
```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-10, 10, 100)
y = pow(x, 3) - 5 * pow(x, 2) + 8
plt.plot(x, y)
plt.grid()
```



- 다음 수식의 누적 과정을 그래프로 그려 직접 확인해 보세요(100 항까지).

$$1+\frac{1}{2^4}+\frac{1}{3^4}+\cdots=1.082\ldots$$

```
import numpy as np
from itertools import accumulate
import matplotlib.pyplot as plt

xs = np.arange(1, 101)
ys = list(accumulate(1 / pow(xs, 4)))
plt.plot(ys)
plt.grid()
```



노트북_24. 리스트 정렬 (원하는 순서로 출력하기)

[확인문제 1]

- 다음 날짜 문자열을 어떻게 정렬할 수 있을까요? 람다 함수를 이용해 보세요.

```
L = ['25/May/2015', '27/May/2014', '27/Apr/2015', '27/Jan/2015', '07/May/2015',
'22/May/2015', '16/Jan/2015']
```

```
month_lookup = ['', 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',
'Oct', 'Nov', 'Dec']
L = ['25/May/2015', '27/May/2014', '27/Apr/2015', '27/Jan/2015', '07/May/2015',
'22/May/2015', '16/Jan/2015']

L.sort(key=lambda e: (int(e.split('/')[2]), month_lookup.index(e.split('/')[1]),
int(e.split('/')[0])))
L
```

```
['27/May/2014',
 '16/Jan/2015',
 '27/Jan/2015',
 '27/Apr/2015',
 '07/May/2015',
 '22/May/2015',
 '25/May/2015']
```

[확인문제 2]

- `sorted()` 함수를 이용하여 다음 시각 리스트를 정렬한 새로운 리스트를 생성해 보세요.

```
L = ['7:27', '14:22', '6:9', '6:11', '2:19']
```

```
L = ['7:27', '14:22', '6:9', '6:11', '2:19']
sorted(L, key=lambda e: (int(e.split(':')[0]), int(e.split(':')[1])))
```

```
['2:19', '6:9', '6:11', '7:27', '14:22']
```

실행과제

- 다음과 같은 자료가 있습니다. 요소는 (파일 이름, 파일 생성 시각, 파일 크기) 형식으로 구성되어 있습니다. 리스트를 파일 이름을 기준으로 정렬해 보세요.

```
data = [('a1.pdf', 1604371920.0, 142488661),
        ('a2.pdf', 1597387946.0, 20365352),
        ('a3.pdf', 1619502356.0, 121225390),
        ('a4.pdf', 1602193158.0, 35801775),
        ('a5.pdf', 1455978788.0, 94522324),
        ('a6.pdf', 1329119770.0, 166388969),
        ('a7.pdf', 1639215016.0, 24283843),
        ('a8.pdf', 1347616254.0, 84165004),
        ('a9.pdf', 1348489154.0, 82744960),
        ('a10.pdf', 1348489160.0, 136499377)]

data.sort(key=lambda e: e[0])
data
```

```
[('a1.pdf', 1604371920.0, 142488661),
 ('a10.pdf', 1348489160.0, 136499377),
 ('a2.pdf', 1597387946.0, 20365352),
 ('a3.pdf', 1619502356.0, 121225390),
 ('a4.pdf', 1602193158.0, 35801775),
 ('a5.pdf', 1455978788.0, 94522324),
 ('a6.pdf', 1329119770.0, 166388969),
 ('a7.pdf', 1639215016.0, 24283843),
 ('a8.pdf', 1347616254.0, 84165004),
 ('a9.pdf', 1348489154.0, 82744960)]
```

노트북_25. 파이썬으로 파일과 폴더 다루기

[확인문제]

- `glob.glob()` 등으로 얻은 파일 목록이 있습니다. 이 파일들 중에서 확장자가 `.bak` 인 파일은 모두 삭제하는 코드를 작성해 보세요.

```
import glob
import os

flist = glob.glob('*.*')
for fpath in flist:
    if fpath.lower().endswith('.bak'):
        print('deleting', fpath)
        os.remove(fpath)
```

모든 파일을 얻은 뒤에('*.*'), 그중 .bak 으로 끝나는(endswith) 파일을 찾아, os 모듈의 remove() 메서드로 제거하는 코드입니다. 잘 동작하고 있는지 알기 위해, 'deleting' 메시지를 출력하도록 요청했습니다.

실행과제

- `os.walk(top_dir)`는 하위 디렉토리를 재귀적으로(recursively) 탐색하면서 파일 목록을 얻게 해 줍니다. 반환 형식은 튜플 `(dirpath, dirnames, filenames)`입니다. 이를 이용해 하위 디렉토리의 모든 파일을 검색해서 현재 시간으로부터 24 시간 이내에 수정된 파일 목록을 출력해 보세요. (힌트: 현재 시각은 `time` 모듈의 `time.time()`, 파일 수정 시각은 `os.path.getmtime(fpath)`로 얻으면 됩니다.)

```
import os
import time

ref_time = time.time() - 24*60*60

for curdir, dirs, files in os.walk('.'):
    for fname in files:
        fpath = os.path.join(curdir, fname)
        try:
            os.path.getmtime(fpath) > ref_time:
            print(fpath)
        except FileNotFoundError:
            pass
```

`time.time()`의 현재 시간에서 24 시간을 뺀 것을 기준 시간(`ref_time`)으로 삼습니다. 60 을 두 번 곱해준 것은 윈도우의 시각 기준이 초(sec)이기 때문에, 시간을 초로 변환해준 것입니다. 코드를 실행하면, `os.walk` 로 얻은 파일 목록 가운데, 파일 수정 시각(`os.path.getmtime`)이 `ref_time` 보다 큰 파일의 경로와 이름을 반환해 줍니다. 간혹 파일이나 경로 문제로 `FileNotFoundError` 가 발생할 수 있으니, `try...except` 구조로 보완해 주었습니다.

노트북_26. 파일 및 자료형의 입출력

실행과제

- `shelve` 모듈을 이용해서도 이처럼 함수를 저장할 수 있는지, 잘 동작하는지 확인해 보세요.

`shelve` 도 파이썬 객체(함수 포함)를 저장할 수 있습니다. 본 실행과제의 코드와 비교해 무엇이 다른지 알아보세요.

```
import shelve

def add(a, b):
    return a+b

shelve_test = shelve.open('shelvetest.db')
```

```
shelve_test['add'] = add

add2 = shelve_test['add']
add2(2, 3)
```

```
5
```

노트북_27. 정규식

[확인문제 1]

- 다음 주어진 문자열에서 금액(숫자)만 추출해 보세요.

```
'금액:150,000 만원 금액:2600 만원, 금액:45,000 만원, 금액:500 만원'
```

```
import re

s = '금액:150,000 만원 금액:2600 만원, 금액:45,000 만원, 금액:500 만원'
re.findall('(\d+,?\d*)', s)
```

```
['150,000', '2600', '45,000', '500']
```

[확인문제 2]

- re.match() 함수를 이용해서 주어진 전화번호가 010-1234-5678 형식에 맞는지 검사하는 코드를 작성해 보세요. (힌트: {} 메타 문자를 이용해 보세요.)

```
import re

s = '010-1234-5678'
m = re.match('\d{3}-\d{4}-\d{4}$', s)
m
```

```
<re.Match object; span=(0, 13), match='010-1234-5678'>
```

위 코드에서 $는 문자열의 끝을 나타내는 메타 문자입니다. 더 이상 숫자가 나와서는 안 된다는 것을 지시합니다.

- 입력한 주민등록번호가 형식에 맞는지 검사하는 코드를 작성해 보세요. (힌트: xxxxxx-xxxxxxx 형식의 숫자 로 구성되면 됩니다. {} 메타 문자를 이용해 보세요.)

```
import re

s = '300312-1012345'
m = re.match('\d{6}-\d{7}$', s)
m
```

```
<re.Match object; span=(0, 14), match='300312-1012345'>
```

실행과제

- `re.sub()` 함수를 이용하여 다음 문장 중 'metaverse' 단어에만 `<bold>metaverse</bold>`와 같이 태그를 붙여 보세요.

```
In futurism and science fiction, the metaverse is a hypothetical iteration of the Internet as a single, universal and immersive virtual world that is facilitated by the use of virtual reality (VR) and augmented reality (AR) headsets.
```

```
import re

def f(m):
    return '<bold>{}</bold>'.format(m.group())

s = 'In futurism and science fiction, the metaverse is a hypothetical iteration of the Internet as a single, universal and immersive virtual world that is facilitated by the use of virtual reality (VR) and augmented reality (AR) headsets.'
result = re.sub('metavers', f, s)
result
```

```
'In futurism and science fiction, the <bold>metavers</bold>e is a hypothetical iteration of the Internet as a single, universal and immersive virtual world that is facilitated by the use of virtual reality (VR) and augmented reality (AR) headsets.'
```

- `{'야곱': 'Jacob', '에서': 'Esau'}` 사전을 이용하여, 다음 문장에서 '야곱'은 '야곱(Jacob)'으로, '에서'는 '에서(Esau)'로 문자열을 변경해 보세요.

```
'야곱과 에서는 집에서 나와 들로 나갔습니다.'
```

```
import re

def f(m):
    return '<bold>{}</bold>'.format(m.group())

names = {'야곱': 'Jacob', '에서': 'Esau'}
s = '야곱과 에서는 집에서 나와 들로 나갔습니다.'
pat = '|'.join(['\\b' + e for e in names])
result = re.sub(pat, f, s)
result
```

```
'<bold>야곱</bold>과 <bold>에서</bold>는 집에서 나와 들로 나갔습니다.'
```

노트북_28. 정규식 더 알아보기

[확인문제 1]

- 다음 문자열에서 `title` 속성에 해당하는 값들만 추출해 보세요. (힌트: 최소 매칭과 플래그를 활용하세요.)

```
import re

s = '''<span strong="3568" title="[nyn] 명사
지금">vũv</span><span strong="2631" title="[katakrima] 명사
형벌">κατάκριμα</span>'''

re.findall('<span.*?title="(.*?)">', s, re.IGNORECASE | re.DOTALL)
```

```
['[nyn] 명사 \n 지금', '[katakrima] 명사 \n 형벌']
```

[확인문제 2]

- 앞서 `compile()` 없이 진행했던 예제들을, `compile()` 함수를 적용해서 다시 풀어 보세요.

한 예제의 `compile()` 버전 풀이를 예시로 보여드립니다. 다른 예제는 직접 실행해 보세요.

```
import re

s = '''<span strong="3568" title="[nyn] 명사
지금">vũv</span><span strong="2631" title="[katakrima] 명사
형벌">κατάκριμα</span>'''

p = re.compile('<span.*?title="(.*?)">', re.IGNORECASE | re.DOTALL)
p.findall(s)
```

```
['[nyn] 명사 \n 지금', '[katakrima] 명사 \n 형벌']
```

[실행과제]

- `New date: 02-01-2026` 과 같은 문자열이 주어져 있을 때, `DD-MM-YYYY` 형식의 날짜를 `YYYY-MM-DD` 형식의 날짜로 변경하는 정규식을 작성하고 실행해 보세요. (힌트: 그룹을 지정하고, 메타 문자 `\\1, \\2, \\3` 을 이용해 보세요.)

```
import re

s = 'New date: 02-01-2026'
re.sub('(\d{2})-(\d{2})-(\d{4})', '\\3-\\2-\\1', s)
```

```
'New date: 2026-01-02'
```

노트북_29. 문자열 통계 처리 (소설이 궁금해!)

실행과제

《이상한 나라의 앨리스》로 몇 가지 문제를 더 풀어 봅시다.

- 소설에서 알파벳 A 다음에 B 가 올 수 있는 조건부 확률을 계산해 보세요. 영문자 'AB'가 나타난 횟수를 'A'가 나타난 횟수로 나누어 주면 됩니다.

```
import requests    # requests 모듈 가져오기

url = 'http://www.umich.edu/~umfandsf/other/ebooks/alice30.txt'
r = requests.get(url)    # url 에서 받은 응답을 r 에 저장
text = r.text            # r 의 text 속성 값만 추려 text 에 저장

text.upper().count('AB') / text.upper().count('A')
```

```
0.024343078148106018
```

- 이와 같이 A 다음에 올 수 있는 모든 알파벳에 대해서 확률을 계산해 보세요.

```
for c in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ':
    print('A->{}'.format(c), text.upper().count('A'+c) / text.upper().count('A'))
```

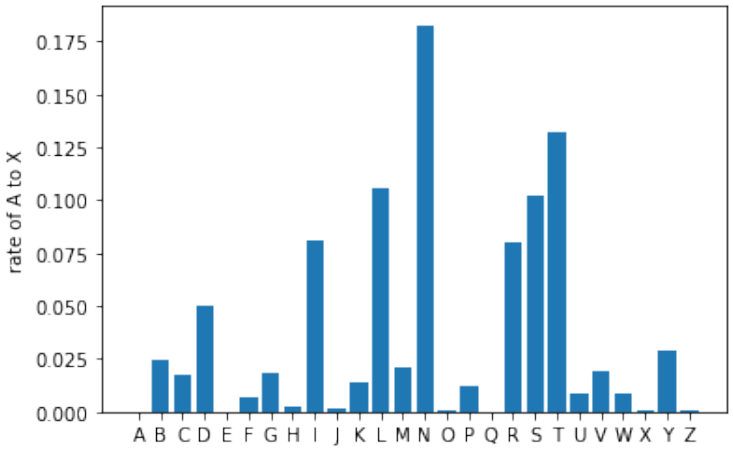
```
A->A 0.0
A->B 0.024343078148106018
A->C 0.017859174155386192
A->D 0.05027869411898533
A->E 0.0
A->F 0.007166420202479809
A->G 0.018200432260266183
A->H 0.0028438175406665907
A->I 0.0813331816630645
A->J 0.0013650324195199637
A->K 0.014219087703332954
```

```
A->L 0.10567625981117051
A->M 0.020816744397679447
A->N 0.1826868388124218
A->O 0.0003412581048799909
A->P 0.012399044477306337
A->Q 0.0
A->R 0.08019565464679786
A->S 0.10226367876237061
A->T 0.1322943919918098
A->U 0.008645205323626436
A->V 0.01911045387327949
A->W 0.008531452621999773
A->X 0.00045501080650665453
A->Y 0.029234444318052555
A->Z 0.0005687635081333182
```

- 결과를 막대 그래프로 그려 보세요.

```
import matplotlib.pyplot as plt

upper = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
cnts = []
for c in upper:
    cnts.append(text.upper().count('A'+c) / text.upper().count('A'))

plt.bar(list(upper), cnts)      # 막대 그래프 그리기
plt.ylabel('rate of A to X')    # y 축 이름 삽입하기
```

```
Text(0, 0.5, 'rate of A to X')
```



노트북_30. 사전을 이용한 웹 정보 표현

실행과제

이 매물 사전에 대해, 다음 문제들을 풀어 보세요.

```
[{'매물번호': 'AA7328082',
'공급/전용면적': '공급 137.58 ㎡/전용 115.26 ㎡',
'전용율': '84%',
'방향': '남향',
'해당층/총층': '14 층/총 14 층',
'방수/욕실수': '4 개/2 개',
'현관구조': '계단식',
'실구입자금': '150,000 만원'},
{'매물번호': 'EA1490360',
'계약면적': '공급 154.75 ㎡(계약면적 기준)/전용 80.96 ㎡',
'방향': '서향(거실방향기준)',
'해당층/총층': '고층/총 13 층',
'방수/욕실수': '3 개/2 개',
'복층여부': '단층',
'현관구조': '계단식',
'실구입자금': '65,000 만원'}]
```

- '방수'와 '욕실수' 정보를 분리해서 사전에 재저장해 보세요.

```
import re

매물목록 = [{'매물번호': 'AA7328082',
'공급/전용면적': '공급 137.58 ㎡/전용 115.26 ㎡',
'전용율': '84%',
'방향': '남향',
'해당층/총층': '14 층/총 14 층',
'방수/욕실수': '4 개/2 개',
'현관구조': '계단식',
'실구입자금': '150,000 만원'},
{'매물번호': 'EA1490360',
'계약면적': '공급 154.75 ㎡(계약면적 기준)/전용 80.96 ㎡',
'방향': '서향(거실방향기준)',
'해당층/총층': '고층/총 13 층',
'방수/욕실수': '3 개/2 개',
'복층여부': '단층',
'현관구조': '계단식',
'실구입자금': '65,000 만원'}]

for 매물 in 매물목록:
    m = re.match('(\d+)개/(\d+)개', 매물.get('방수/욕실수', ''))
    if m:
        매물['방수'] = m.group(1)
        매물['욕실수'] = m.group(2)
        del 매물['방수/욕실수']

print(매물목록)
```

```
[{'매물번호': 'AA7328082', '공급/전용면적': '공급 137.58 ㎡/전용 115.26 ㎡', '전용율': '84%', '방향': '남향', '해당층/총층': '14 층/총 14 층', '현관구조': '계단식', '실구입자금': '150,000 만원', '방수': '4', '욕실수': '2'}, {'매물번호': 'EA1490360', '계약면적': '공급 154.75 ㎡(계약면적 기준)/전용 80.96 ㎡', '방향': '서향(거실방향기준)', '해당층/총층': '고층/총 13 층', '복층여부': '단층', '현관구조': '계단식', '실구입자금': '65,000 만원', '방수': '3', '욕실수': '2'}]
```

- 해당층과 총층 정보를 분리해서 사전에 재저장해 보세요.

```
for 매물 in 매물목록:
    m = re.match('(.+?)/(.+)', 매물.get('해당층/총층', ''))
    if m:
        매물['해당층'] = m.group(1)
        매물['총층'] = m.group(2)
        del 매물['해당층/총층']

print(매물목록)
```

```
[{'매물번호': 'AA7328082', '공급/전용면적': '공급 137.58 ㎡/전용 115.26 ㎡', '전용율': '84%', '방향': '남향', '현관구조': '계단식', '실구입자금': '150,000 만원', '방수': '4', '욕실수': '2', '해당층': '14 층', '총층': '총 14 층'}, {'매물번호': 'EA1490360', '계약면적': '공급 154.75 ㎡(계약면적 기준)/전용 80.96 ㎡', '방향': '서향(거실방향기준)', '복층여부': '단층', '현관구조': '계단식', '실구입자금': '65,000 만원', '방수': '3', '욕실수': '2', '해당층': '고층', '총층': '총 13 층'}]
```

- 실구입자금을 정수화해서 저장해 보세요.

```
for 매물 in 매물목록:
    m = re.match('(\d+(,\d+)*)만원', 매물['실구입자금'])
    매물['실구입자금'] = int(m.group(1).replace(',', '')) * 10000

print(매물목록)
```

```
[{'매물번호': 'AA7328082', '공급/전용면적': '공급 137.58 ㎡/전용 115.26 ㎡', '전용율': '84%', '방향': '남향', '현관구조': '계단식', '실구입자금': 1500000000, '방수': '4', '욕실수': '2', '해당층': '14 층', '총층': '총 14 층'}, {'매물번호': 'EA1490360', '계약면적': '공급 154.75 ㎡(계약면적 기준)/전용 80.96 ㎡', '방향': '서향(거실방향기준)', '복층여부': '단층', '현관구조': '계단식', '실구입자금': 650000000, '방수': '3', '욕실수': '2', '해당층': '고층', '총층': '총 13 층'}]
```

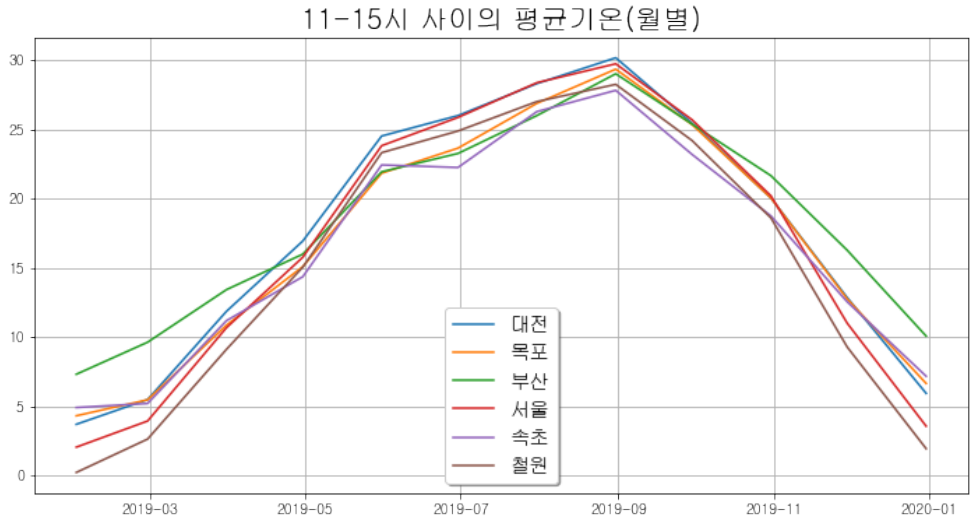
노트북_32. 데이터 처리 (열대야 일수 계산)

실행과제

- 주어진 기상정보 2019.xlsx 파일을 읽고, 오전 11 시부터 오후 3 시 사이 최고 기온의 월평균 그래프를 지역별로 그려 비교해 보세요.

```
import re
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl              # 한글 폰트를 위한 모듈1
import matplotlib.font_manager as fm  # 한글 폰트를 위한 모듈2


# 한글 폰트를 위한 코드 블록
mpl.rcParams['axes.unicode_minus'] = False
mpl.rcParams['axes.titlesize'] = 20     # 제목 크기 설정
path = r'C:/Windows/Fonts/gulim.ttc'    # r'폰트 파일 경로 / 굴림
font_name = fm.FontProperties(fname=path).get_name()
plt.rc('font', family=font_name)
plt.figure(figsize=(12, 6))    # 전체 크기 변경


df = pd.read_excel('data/기상정보 2019.xlsx', index_col='일시')
names = [re.sub('[^가-힣]', '', name) for name in df.columns]
df.columns = names
df.drop(['강수량', '풍속', '습도', '현지기압', '일조', '적설'], axis=1, inplace=True)

df.fillna(method='ffill', inplace=True)
df1 = df.between_time('11:00', '15:00')

# groupby()를 이용하면 지점명별로 별도의 데이터프레임 구성 가능
for name, group_df in df1.groupby('지점명'):
    mean = group_df['기온'].resample('M').mean()
    plt.plot(mean, label=name)

# 그래프에 식별 요소 추가
plt.title('11-15 시 사이의 평균기온(월별)')   # 그래프 제목 입력
plt.grid()
plt.legend(loc='lower center', shadow=True, fontsize='x-large')
```

```
<matplotlib.legend.Legend at 0x15084f2b730>
```



노트북_33. 이름 공간과 이름의 선언

[확인문제]

- 다음 코드의 출력 값은 얼마일까요?

```
a = 10
def function(a, b):
    print(a + b)
a = 12
b = 38
function(a, b)
```

```
50
```

함수가 리턴하는 값은 없고(None), 출력(print)하는 값은 50 입니다.

- 다음 코드의 출력 값은 얼마일까요?

```
def outer_fun(a, b):
    def inner_fun(c, d):
        return c + d
    return inner_fun(a, b)

result = outer_fun(5, 10)
print(result)
```

```
15
```

코드를 보면, `outer_fun(a, b)` 함수는 `inner_fun(a, b)`를 돌려줍니다. 따라서 `a`, `b` 가 그대로 `inner_fun(c, d)` 함수에서 연산됩니다. 이 함수는 두 요소(`c`, `d`)를 더할 것을 정의하고 있으므로, `outer_fun(5, 10)`의 출력은 5+10, 15 가 됩니다.

[확인문제 2]

- 다음 코드의 출력 값은 얼마일까요?

```
var1 = 50
def fun():
    var1 = 10
    def gun():
        nonlocal var1
        var1 = var1 + 10
        print(var1)
    gun()
fun()
```

```
20
```

`gun()` 함수 코드 블록을 보면, `var1` 변수를 `nonlocal` 선언하고 있습니다. 따라서 이 함수에서 참조하는 `var1` 은 전역의 `var1 = 50` 이 아니라, 내포 영역에 해당하는 `fun()` 함수의 `var1 = 10` 입니다. 그러므로 마지막 `fun()`이 출력하는 것은 `var1(10) + 10`, 즉 20 이 됩니다.

실행과제

- 다음 세 개의 코드를 비교해 보고, 그 결과를 예측해 보세요.

코드 1

```
x = 0
def outer():
    x = 1
    def inner():
        x = 2
        print("inner:", x)
    inner()
    print("outer:", x)
outer()
print("global:", x)
```

```
inner: 2
outer: 1
global: 0
```

코드 1 에는 `nonlocal` 선언이나 `global` 선언이 없으므로, 함수의 영역만 잘 살펴보면 됩니다. 각 `x` 값이 그대로 유지됩니다.

코드 2

```
x = 0
def outer():
    x = 1
    def inner():
        nonlocal x
        x = 2
        print("inner:", x)
    inner()
    print("outer:", x)
outer()
print("global:", x)
```

```
inner: 2
outer: 2
global: 0
```

`inner()` 함수 내에서 `nonlocal` 선언이 있었으므로, `outer()` 함수의 `x` 값이 2 가 됩니다. 따라서 inner 와 outer 의 `x` 는 모두 2 입니다. 전역 영역에는 영향이 없기 때문에 global 은 그대로 0 입니다.

코드 3

```
x = 0
def outer():
    x = 1
    def inner():
        global x
        x = 2
        print("inner:", x)
    inner()
    print("outer:", x)
outer()
print("global:", x)
```

```
inner: 2
outer: 1
global: 2
```

이번에는 `inner()` 함수 내에서 `global` 선언이 있었습니다. 전역 영역의 `x` 가 `inner()`의 `x` 값으로 대치됩니다. 내포 영역(`outer()`)에는 영향이 없으므로, 1 이 유지됩니다.

노트북_34. 이름과 객체의 참조 구조

[확인문제 1]

- 다음 코드에서 최종 y 값의 출력이 어떨지 예상해 보세요.

```
x = {'hello': 'world'}
y = x
x = None
y
```

```
{'hello': 'world'}
```

`x` 가 `None` 이 되었지만, 이미 `y` 에 `{'hello': 'world'}` 값이 할당되어 있습니다. `x` 의 참조 주소 변경은 `y` 에 영향을 미치지 않으므로, 값은 유지됩니다.

[확인문제 2]

- 〈복합 자료형의 참조 구조〉 리스트의 최종 상태에서, 다음 코드는 무엇을 출력할까요?

```
# 〈복합 자료형의 참조 구조〉 리스트의 최종 상태
X = [100, 2, 3]
Y = [10, X, 30]
Z = Y

X[1] = 200          # X 의 두 번째 요소 200 으로 변경 ⇒ X = [100, 200, 3]
print(Y[1][1])      # Y = [10, [100, 200, 3], 3]에서 두 번째 요소의 두 번째 요소 출력
print(Z[1][1])      # Z = [10, [100, 200, 3], 3]에서 두 번째 요소의 두 번째 요소 출력
```

```
200
200
```

[확인문제 3]

- 다음 코드에서 사용된 복사는 어떤 종류의 복사일까요(얕은 복사/깊은 복사)?

```
>>> a1 = [[10, 20], [30, 40], [50, 60]]
>>> a2 = list(a1)
>>> a2
[[10, 20], [30, 40], [50, 60]]
```

본문 마지막과 비슷하게 `a1` 과 `a2` 의 요소끼리 비교하고, 리스트 자체끼리 비교해 봅니다.

```
a1[0] == a2[0], a1 is a2
```

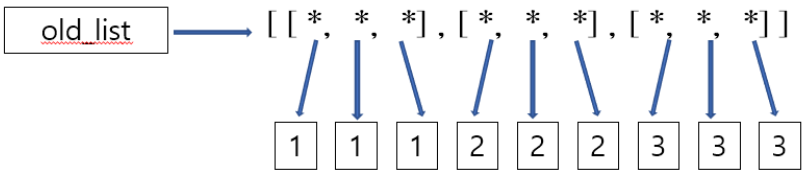
```
(True, False)
```

요소의 값은 같은데(`True`), `a1` 과 `a2` 는 다른 개체입니다(`False`). 따라서 얕은 복사를 한 것으로 볼 수 있습니다.

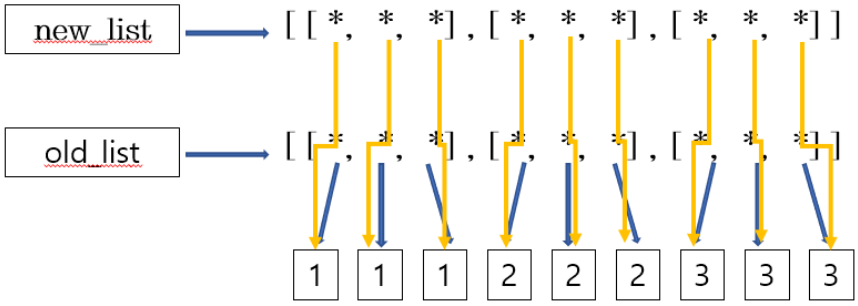
- 다음 코드를 실행한 결과 생성되는 `old_list` 와 `new_list` 의 참조 구조를 각각 그려 보세요.

```
import copy
old_list = [[1, 1, 1], [2, 2, 2], [3, 3, 3]]
new_list = copy.deepcopy(old_list)
```

리스트 3 개가 모인 중첩 리스트 `old_list` 의 참조 구조는 다음과 같습니다.



이 `old_list` 를 깊은 복사한 `new_list` 의 참조 구조는 어떨까요?



실행과제

- 다음 코드들이 출력할 결과를 예측해 보세요.

코드 1

```
s = "foo"

def test(s):
    s = "bar"
    print("Inside Function:", s)

test(s)
print("Outside Function:", s)
```

```
Inside Function: bar
Outside Function: foo
```

`print("Inside Function:", s)` 명령은 `test()` 함수의 지역 변수 `s` 값을 출력합니다(`bar`). `print("Outside Function:", s)` 명령은 전역 변수 `s` 값을 출력합니다(`foo`).

코드 2

```
def foo(a):
    a[0] = "Nothing"

bar = ['Hi', 'how', 'are', 'you', 'doing']

foo(bar)
print(bar)
```

```
['Nothing', 'how', 'are', 'you', 'doing']
```

`foo(bar)`는 `bar` 리스트의 첫 번째 항목을 "Nothing"으로 바꿉니다.

노트북_35. 메인 모듈과 모듈 임포트

[확인문제]

- `__name__` 변수의 역할은 무엇입니까? 이 변수는 어떤 용도로 활용할 수 있나요?

`__name__` 변수는 모듈의 이름을 갖는 특수 변수입니다. 이 이름은 파이썬에서 자동으로 설정됩니다. 임포트 되는 모듈의 이름은 모듈(혹은 패키지)의 이름과 `__name__` 변수의 이름이 같습니다. 하지만 가장 먼저 실행되는 스크립트의 이름은, 그 파일 이름이 무엇이든 간에 `__main__`이란 이름을 갖습니다. 그래서 임포트 되는 모듈과 가장 먼저 실행되는 스크립트의 구분이 이 변수를 통해 가능해지게 됩니다. 보통 다음 형식으로 최초 스크립트로 실행되는 경우에만 어떤 코드를 실행하도록 만드는 코드를 뒤쪽에 추가하는 것이 하나의 패턴입니다.

```
if __name__ == '__main__':
    main()
```

실행과제

- 표준 모듈(예: `re`)의 저장 경로와 `pip` 로 설치된 외부 모듈(예: `numpy`)의 저장 경로 간 차이점은 무엇인가요?

```
#표준 모듈의 저장 경로
re.__file__
'C:\\Users\\user\\AppData\\Local\\Programs\\Python\\Python310\\lib\\re.py'

#외부 모듈의 저장 경로
numpy.__file__
'C:\\Users\\user\\AppData\\Local\\Programs\\Python\\Python310\\lib\\
```

```
site-packages\\numpy\\__init__.py'
```

하나의 파일로 된 모듈인 경우의 경로는 `re.py` 와 같은 파이썬 스크립트 파일이지만, 여러 모듈을 하나의 폴더로 묶어 만든 패키지인 경우는 폴더 이름(`numpy`)이 패키지 이름이 되며, 초기화를 위한 파일은 패키지 폴더 안의 `__init__.py` 파일이 됩니다. 그래서 파일 경로의 이름이 좀 달라 보입니다.

노트북_36. 객체와 클래스

[확인문제 1]

- 자료형과 객체는 어떻게 다른가요? 두 객체가 is-a 관계를 갖고 있는지 어떻게 확인할 수 있을까요?

자료형은 보편자, 객체는 개별자에 해당합니다. 객체는 자료형으로부터 만들어지는 것입니다. 두 관계는 `isinstance(객체, 자료형)` 형식으로 확인 가능합니다.

[확인문제 2]

- 숫자 12 가 `int` 의 인스턴스인지 어떻게 확인할 수 있나요?

`isinstance()` 함수를 이용하면 됩니다.

```
isinstance(12, int)
```
```
True
```

[확인문제 3]

- 앞서 정의한 `Turtle` 클래스를 이용해 다음과 같이 코드를 실행하면 오류가 생깁니다. 왜 그런지 설명해 보세요.

```
class Turtle:
    pi = 3.141592              # 클래스 멤버
    def forward(self, steps): # 메서드
        self.x += steps        # self.x 인스턴스 멤버
```

```
t1 = Turtle()
t1.forward(10)
```

```
AttributeError: 'Turtle' object has no attribute 'x'
```

이 코드에서 `forward()` 메서드는 `x` 라는 멤버를 참조하려고 하는데, 이 멤버는 정의된 적이 없습니다. 그 때문에 AttributeError: 'Turtle' object has no attribute 'x'라는 오류가 발생하는 것입니다.

노트북_37. 클래스 작성 방법

[확인문제]

- 다음과 같이 주어진 클래스에 `speak` 라는 새로운 메서드를 여러분의 방식으로 추가해서 동작시켜 보세요. 그리고 서로 다른 이름과 나이를 갖는 인스턴스를 두 개 만들어 보세요.

예시로 '콩이', '보리'라는 강아지들이 '왈왈'이라고 짖는 `speak` 메서드를 추가해 보았습니다.

```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def speak(self)      # '왈왈'을 출력하는 speak 메서드 정의
        return '왈왈'

자식1 = Dog('콩이', 3)
자식2 = Dog('보리', 5)

자식1.speak()
```

```
왈왈
```

실행과제

- 파이썬은 `__init__` 외에도 미리 정의된 메서드를 다수 가지고 있습니다. 그중 `__repr__`은 문자열을 돌려주는데, 이것이 인스턴스 객체들이 `print()`로 출력할 때 보여줄 메시지가 됩니다. 이 메서드를 어떻게 사용하는지 확인해 보고, 직접 클래스에 구현해서 실행해 보세요.

〈예시〉

```
class Dog:
    def __init__(self, name, age):
```

```
        self.name = name
        self.age = age
    def __repr__(self):
        return '[저는 Dog 입니다. 이름은 {}, 나이는 {}살이에요]'.format(self.name, self.age)

자식1 = Dog('콩이', 3)
print(자식1)
```

```
[저는 Dog 입니다. 이름은 콩이, 나이는 3 살이에요]
```

노트북_38. 클래스 설계의 예

[확인문제]

- 다음 코드가 출력하는 것은 무엇일까요?

```
class Dog:
    def walk(self):
        return "걷습니다"
    def speak(self):
        return "멍멍!"

class Dachshund(Dog):
    def speak(self):
        return "왈왈!"

bobo = Dachshund()
bobo.walk()
```

```
"걷습니다"
```