풀이 | 1분 코딩, 되새김 문제 및 파이썬 코딩 면허 시험

1분 코딩

129쪽

```
if 'card' not in pocket:
    print("걸어가라")
else:
    print("버스를 타고 가라")
```

139쪽

```
>>> a = 0
>>> while a < 10:
...     a = a + 1
...     if a % 3 == 0: continue
...     print(a)
```

145쪽

```
>>> a = 0
>>> for i in range(1, 101):
...     a += i
...
>>> print(a)
```

176쪽

```
f = open("C:/doit/복습.txt", 'w')
f.close()
```

219쪽

```
>>> import mod2
>>> a = mod2.Math()
>>> print(a.solv(5))
78.5398
```

02장 • 되새김 문제

116~119쪽

Q1 평균 점수 구하기

```
>>> a = 80
>>> b = 75
>>> c = 55
>>> (a + b + c) / 3
70.0
```

Q2 홀수, 짝수 판별하기

나머지 연산자를 사용하면 자연수의 홀수, 짝수를 쉽게 판별할 수 있다.

```
>>> 1 % 2
1
>>> 2 % 2
0
>>> 3 % 2
1
>>> 4 % 2
0
```

1, 2, 3, 4라는 자연수를 2로 나누었을 때의 나머지 값을 출력하는 예제이다. 결과를 보면 자연수가 홀수일 때는 1, 짝수일 때는 0을 돌려 주는 것을 확인할 수 있다.

03 주민등록번호 나누기

```
>>> pin = "881120-1068234"
>>> yyyymmdd = pin[:6]
>>> num = pin[7:]
>>> print(yyyymmdd) <-- 881120 출력
>>> print(num) <-- 1068234 출력
```

04 주민등록번호 인덱싱

```
>>> pin = "881120-1068234"
>>> print(pin[7]) <-- 1 또는 3이면 남자, 2 또는 4이면 여자
```

성별을 나타내는 숫자는 -(하이픈)을 포함하여 여덟 번째 숫자이므로 여덟 번째 자리를 인덱싱한다.

05 문자열 바꾸기

```
>>> a = "a:b:c:d"
>>> b = a.replace(":", "#")
>>> print(b) <-- a#b#c#d 출력
```

06 리스트 역순 정렬하기

```
>>> a = [1, 3, 5, 4, 2]
>>> a.sort()
>>> a.reverse()
>>> print(a) <-- [5, 4, 3, 2, 1] 출력
```

리스트의 내장 함수인 sort를 사용해 리스트 값들을 먼저 정렬한 후 reverse 함수를 사용해 순서를 뒤집는다.

Q7 리스트를 문자열로 만들기

```
>>> a = ['Life', 'is', 'too', 'short']
>>> result = " ".join(a)
>>> print(result)
```

a 리스트의 각 단어들을 한 문장으로 조립할 때 단어들 사이마다 공백을 넣어야 한다. 1개의 공백 문자(" ")를 사용해 join한다.

Q8 튜플 더하기

```
>>> a = (1, 2, 3)
>>> a = a + (4,)
>>> print(a)  <- (1, 2, 3, 4) 출력
```

a 튜플에 (4,)라는 튜플을 더하면 된다. 단, 이때 만들어지는 a + (4,)의 결과는 a 값이 변경되는 것이 아니라(튜플은 그 값을 변경할 수 없다) 새로운 튜플이 생성되고 그 값이 a 변수에 대입되는 것임을 유념하자. 다음 코드를 실행해 보면 a의 고유 주소 값이 변경됨을 확인할 수 있다.

```
>>> a = (1, 2, 3)
>>> print(id(a))  <- a의 고유 주소 값 출력
>>> a = a + (4,)
>>> print(a)
>>> print(id(a))  <- (4,) 값이 더해진 후 a의 고유 주소 값 출력
```

Q9 딕셔너리의 키

세 번째 예를 실행하면 다음과 같은 오류가 발생한다.

```
>>> a[[1]] = 'python'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

오류가 발생하는 이유는 딕셔너리의 키로 변하는 값을 사용할 수 없기 때문이다. 여기서 키로 사용된 [1]은 리스트이므로 변하는 값이다. 다른 예에서 키로 사용된 문자열, 튜플, 숫자는 변하지 않는 값이므로 딕셔너리의 키로 사용이 가능하다.

Q10 딕셔너리 값 추출하기

딕셔너리도 리스트와 마찬가지로 다음과 같이 pop 함수를 사용할 수 있다.

```
>>> a = {'A':90, 'B':80, 'C':70}
>>> result = a.pop('B')
>>> print(a)  <-- {'A':90, 'C':70} 출력
>>> print(result)  <-- 80 출력
```

'B' 키에 해당되는 값이 리턴되고 딕셔너리 a에서는 그 값이 제거되는 것을 확인할 수 있다.

Q11 리스트에서 중복 제거하기

```
>>> a = [1, 1, 1, 2, 2, 3, 3, 3, 4, 4, 5]
>>> aSet = set(a)  <-- a 리스트를 집합 자료형으로 변환
>>> b = list(aSet)  <-- 집합자료형을 리스트 자료형으로 다시 변환
>>> print(b)  <-- [1, 2, 3, 4, 5] 출력
```

리스트 자료형이 집합 자료형으로 변환되면서 중복된 값들은 사라진다. 이와 같은 성질을 사용하면 리스트 내에 중복된 값을 쉽게 제거할 수 있다.

Q12 파이썬 변수

[1, 4, 3]이 출력된다. a와 b 변수는 모두 동일한 [1, 2, 3]이라는 리스트 객체를 가리키고 있기 때문이다.

03장 • 되새김 문제

149~151쪽

Q1 조건문의 참과 거짓

결괏값으로 shirt가 출력된다.

① 첫 번째 조건: "wife"라는 단어는 a 문자열에 없으므로 거짓이다.

② 두 번째 조건: "python"이라는 단어는 a 문자열에 있지만 "you" 역시 a 문자열에 있으므로 거짓이다.

③ 세 번째 조건: "shirt"라는 단어가 a 문자열에 없으므로 참이다.

④ 네 번째 조건: "need"라는 단어가 a 문자열에 있으므로 참이다.

가장 먼저 참이 되는 것이 세 번째 조건이므로 "shirt"가 출력된다.

Q2 3의 배수의 합 구하기

3의 배수는 3으로 나누어떨어지는 수이다. 이 아이디어를 기반으로 한 파이썬 코드는 다음과 같다.

```
result = 0
i = 1
while i <= 1000:
    if i % 3 == 0:      # 3으로 나누어떨어지는 수는 3의 배수
        result += i
    i += 1

print(result)           # 166833 출력
```

Q3 별 표시하기

```
i = 0
while True:
    i += 1              # while 문을 수행할 때마다 1씩 증가
    if i > 5: break     # i 값이 5보다 크면 while 문을 벗어난다.
    print('*' * i)      # i 값의 개수만큼 *를 출력한다.
```

while 문을 수행할 때마다 i 값을 증가시킨다. 별 모양을 5번 출력해야 하므로 i 값이 5보다 클 경우 while 문을 벗어나도록 한다. 별 모양을 i 값만큼 출력하기 위해서 문자열 곱하기 기능을 사용한다.

Q4 1부터 100까지 출력하기

```
>>> for i in range(1, 101):
...     print(i)
...
1
2
3
4
5
6
7
8
9
10
(...생략...)
```

Q5 평균 점수 구하기

```
A = [70, 60, 55, 75, 95, 90, 80, 80, 85, 100]
total = 0

for score in A:
    total += score          # A학급의 점수를 모두 더한다.

average = total / len(A)    # 평균을 구하기 위해 총 점수를 총 학생 수로 나눈다.
print(average)              # 평균 79.0이 출력된다.
```

먼저 for 문을 사용해 총 점수를 구한 후 총 점수를 총 학생 수로 나누어 평균 점수를 구한다.

Q6 리스트 컴프리헨션 사용하기

문제에서 주어진 소스 코드를 리스트 컴프리헨션으로 표현하면 다음과 같다.

```
>>> numbers = [1, 2, 3, 4, 5]
>>> result = [n*2 for n in numbers if n%2 == 1]
>>> print(result)  <- [2, 6, 10] 출력
```

Q1 홀수, 짝수 판별하기

```
>>> def is_odd(number):
...     if number % 2 == 1:  <-- 2로 나누었을 때 나머지가 1이면 홀수이다.
...         return True
...     else:
...         return False
...
>>> is_odd(3)
True
>>> is_odd(4)
False
```

람다와 조건부 표현식을 사용하면 다음과 같이 간단하게 만들 수 있다.

```
>>> is_odd = lambda x: True if x%2 == 1 else False
>>> is_odd(3)
True
```

Q2 모든 입력의 평균값 구하기

```
>>> def avg_numbers(*args):  <-- 입력 개수에 상관없이 사용하기 위해 *args를 사용
...     result = 0
...     for i in args:
...         result += i
...     return result / len(args)
...
>>> avg_numbers(1, 2)
1.5
>>> avg_numbers(1, 2, 3, 4, 5)
3.0
```

Q3 프로그램 오류 수정하기 1

```
input1 = input("첫 번째 숫자를 입력하세요:")
input2 = input("두 번째 숫자를 입력하세요:")

total = int(input1) + int(input2)  <-- 입력은 항상 문자열이므로 숫자로 바꿔야 한다.
print("두 수의 합은 %s입니다" % total)
```

출력 결과는 다음과 같다.

실행 결과

```
첫 번째 숫자를 입력하세요: 3
두 번째 숫자를 입력하세요: 6
두 수의 합은 9입니다
```

Q4 출력 결과가 다른 것은?

```
>>> print("you" "need" "python")
youneedpython
>>> print("you"+"need"+"python")
youneedpython
>>> print("you", "need", "python")  <-- 쉼표(,)가 있는 경우 공백이 삽입되어 더해진다.
you need python
>>> print("".join(["you", "need", "python"]))
youneedpython
```

Q5 프로그램 오류 수정하기 2

문제의 예와 같이 파일을 닫지 않은 상태에서 다시 열면 파일에 저장한 데이터를 읽을 수 없다. 따라서 열린 파일 객체를 close로 닫아준 후 다시 열어서 파일의 내용을 읽어야 한다.

```
f1 = open("test.txt", 'w')
f1.write("Life is too short!")
f1.close() ← 열린 파일 객체를 닫는다.

f2 = open("test.txt", 'r')
print(f2.read())
f2.close()
```

또는 다음과 같이 close를 명시할 필요가 없는 with 문을 사용한다.

```
with open("test.txt", 'w') as f1:
    f1.write("Life is too short!")

with open("test.txt", 'r') as f2:
    print(f2.read())
```

Q6 사용자 입력 저장하기

기존 내용을 유지하고 새로운 내용을 덧붙이기 위해서 다음과 같이 'a' 모드를 사용해야 한다.

```
user_input = input("저장할 내용을 입력하세요:")
f = open('test.txt', 'a')     # 내용을 추가하기 위해서 'a'를 사용
f.write(user_input)
f.write("\n")                 # 입력된 내용을 줄 단위로 구분하기 위해 줄 바꿈 문자 삽입
f.close()
```

Q7 파일의 문자열 바꾸기

파일을 모두 읽은 후에 문자열의 replace 함수를 사용하여 java라는 문자열을 python으로 변경한 다음 저장한다.

```
f = open('test.txt', 'r')
body = f.read()                              # test.txt의 내용을 body 변수에 저장
f.close()

body = body.replace('java', 'python')  # body 문자열에서 "java"를 "python"으로 변경

f = open('test.txt', 'w')                    # 파일을 쓰기 모드로 열기
f.write(body)
f.close()
```

Q8 입력값을 모두 더해 출력하기

다음처럼 sys 모듈의 argv를 사용해 명령 행의 모든 입력값을 차례대로 더한다.

```
import sys

numbers = sys.argv[1:]    # 파일 이름을 제외한 명령 행의 모든 입력

result = 0
for number in numbers:
    result += int(number)
print(result)
```

Q1 클래스 상속받고 메서드 추가하기 1

다음과 같이 Calculator 클래스를 상속하는 UpgradeCalculator 클래스를 만들고 minus 메서드를 추가한다.

```
class UpgradeCalculator(Calculator):
    def minus(self, val):
        self.value -= val
```

Q2 클래스 상속받고 메서드 추가하기 2

Calculator 클래스를 상속하고 add 메서드를 오버라이딩하여 다음과 같은 클래스를 만든다.

```
class MaxLimitCalculator(Calculator):
    def add(self, val):
        self.value += val
        if self.value > 100:
            self.value = 100
```

Q3 참과 거짓 예측하기

```
>>> all([1, 2, abs(-3)-3])
False
```

abs(-3)은 -3의 절댓값이므로 3이 되어 all([1, 2, 0])이 되고, 리스트의 요솟값 중 0이 있기 때문에 all 내장 함수의 결과는 False가 된다.

```
>>> chr(ord('a')) == 'a'
True
```

ord 함수는 문자에 해당하는 유니코드 정수를 리턴한다. ord('a')의 결과는 97이 되어 chr(97)로 치환된다. chr(97)의 결과는 다시 'a'가 되므로 'a' == 'a'가 되어 True를 돌려준다.

Q4 음수 제거하기

음수를 제거하기 위한 filter의 함수로 lambda 함수를 다음과 같이 만들어 실행한다.

```
>>> list(filter(lambda x: x>0, [1, -2, 3, -5, 8, -3]))
[1, 3, 8]
```

Q5 16진수를 10진수로 변경하기

int 내장 함수를 다음과 같이 실행한다.

```
>>> int('0xea', 16)
234
```

Q6 리스트 항목마다 3 곱하여 리턴하기

입력에 항상 3을 곱하여 리턴하는 lambda 함수를 다음과 같이 만들고 map과 조합하여 실행한다.

```
>>> list(map(lambda x:x*3, [1, 2, 3, 4]))
[3, 6, 9, 12]
```

Q7 최댓값과 최솟값의 합

리스트의 최댓값은 max, 최솟값은 min 내장 함수를 사용하여 다음과 같이 구한다.

```
>>> a = [-8, 2, 7, 5, -3, 5, 0, 1]
>>> max(a) + min(a)
-1
```

Q8 소수점 반올림하기

round 내장 함수를 사용하면 다음과 같이 반올림하여 소수점 4자리까지 표시할 수 있다.

```
>>> round(17/3, 4)
5.6667
```

Q9 디렉터리 이동하고 파일 목록 출력하기

다음처럼 os 모듈의 chdir을 사용하여 C:\doit 디렉터리로 이동한다.

```
>>> import os
>>> os.chdir("c:/doit")
```

그리고 다음처럼 os 모듈의 popen을 사용하여 시스템 명령어인 dir을 수행한다.

```
>>> result = os.popen("dir")
```

popen의 결과를 출력하기 위해 다음과 같이 수행한다.

```
>>> print(result.read())
(...생략...)
abc.txt
bidusource.html
(...생략...)
```

Q10 파일 확장자가 .py인 파일만 찾기

다음과 같이 glob 모듈을 사용한다.

```
>>> import glob
>>> glob.glob("c:/doit/*.py")
['c:/doit/doit01.py', 'c:/doit/test.py']
```

Q11 날짜 표시하기

time 모듈의 strftime을 사용하여 다음과 같이 작성한다.

```
>>> import time
>>> time.strftime("%Y/%m/%d %H:%M:%S")   # %Y:년, %m:월, %d:일, %H:시, %M:분, %S:초
'2018/04/05 10:56:27'
```

Q12 로또 번호 생성하기

random 모듈의 randint를 사용하여 다음과 같이 작성한다.

```
import random

result = []
while len(result) < 6:
    num = random.randint(1, 45)     # 1부터 45까지의 난수 발생
    if num not in result:
        result.append(num)

print(result)
```

Q13 누나는 영철이보다 며칠 더 먼저 태어났을까?

```
>>> import datetime
>>> sister = datetime.date(1995, 11, 20)
>>> me = datetime.date(1998, 10, 6)
>>> (me-sister).days
1051
```

누가가 영철이보다 1051일 먼저 태어났다.

Q14 기록순으로 정렬하기

```
import operator

data = [('윤서현', 15.25),
        ('김예지', 13.31),
        ('박예원', 15.34),
        ('송순자', 15.57),
        ('김시우', 15.48),
```

```
        ('배숙자', 17.9),
        ('전정웅', 13.39),
        ('김혜진', 16.63),
        ('최보람', 17.14),
        ('한지영', 14.83),
        ('이성호', 17.7),
        ('김옥순', 16.71),
        ('황민지', 17.65),
        ('김영철', 16.7),
        ('주병철', 15.67),
        ('박상현', 14.16),
        ('김영순', 14.81),
        ('오지아', 15.13),
        ('윤지은', 16.93),
        ('문재호', 16.39)]

data = sorted(data, key=operator.itemgetter(1))
for d in data:
    print(d)
```

operator.itemgetter 모듈을 사용하여 기록순으로 정렬했다. operator.itemgetter(1)은 ("이름", "기록") 으로 구성된 튜플 데이터의 두 번째 항목인 "기록"을 의미한다. 이 코드를 실행하면 기록순으로 정렬된 데이터를 확인할 수 있다.

Q15 청소 당번 2명 뽑기

```
import itertools

students = ['나지혜', '성성민', '윤지현', '김정숙']
result = itertools.combinations(students, 2)
print(list(result))
```

itertools.combinations 모듈을 사용해 4명 중 2명을 뽑을 수 있는 경우의 수를 출력한다.

Q16 문자열 나열하기

```
import itertools

a = "abcd"
result = itertools.permutations(a, 4)
for r in result:
    print(''.join(r))
```

itertools.permutations 모듈을 사용하여 문자열 "abcd"의 문자 각각을 순열로 만들면 된다. 이때 리턴되는 순열의 항목(r)은 ('a', 'b', 'c', 'd')와 같은 튜플이므로 ''.join(r) 처럼 묶어서 출력했다.

Q17 5명에게 할 일 부여하기

```
import random
import itertools

people = ['김승현', '김진호', '강춘자', '이예준', '김현주']
duty = ['청소', '빨래', '설거지']

random.sample(people, len(people))    # 무작위로 섞는다.
result = itertools.zip_longest(people, duty, fillvalue='휴식')
for r in result:
    print(r)
```

5명을 무작위로 섞기 위해 random.sample 함수를 사용했다. 그리고 사람들과 할 일을 차례로 묶고 나머지 2명에게는 "휴식"을 부여하기 위해 itertools.zip_longest 함수를 사용했다.

018 벽에 타일 붙이기

```
import math

width = 200
height = 80

square_size = math.gcd(200, 80)
print("타일 한 선의 길이: {}".format(square_size))

width_count = width/square_size
height_count = height/square_size

print("필요한 타일의 개수: {}".format(int(width_count * height_count)))
```

실행 결과

```
타일 한 선의 길이: 40
필요한 타일의 개수: 10
```

200과 80의 최대 공약수를 구하면 벽에 붙일 수 있는 가장 큰 정사각형 타일의 길이를 구할 수 있다. 최대 공약수로 구한 타일 한 선의 길이는 40이다. 타일 한 선의 길이를 알면 가로로 붙여야 할 타일 개수와 세로로 붙여야 할 타일의 개수를 알 수 있으므로 2개의 값을 곱하여 최종적으로 필요한 타일의 개수를 구할 수 있다. 즉, 필요한 타일의 개수는 총 10개이다.

파이썬 코딩 면허 시험 20제

Q1 문자열 바꾸기

```
>>> a = "a:b:c:d"
>>> b = a.split(":")
>>> b
['a', 'b', 'c', 'd']
>>> c = "#".join(b)
>>> c
'a#b#c#d'
```

Q2 딕셔너리 값 추출하기

딕셔너리의 get 함수를 사용하면 해당 key가 없을 경우에는 두 번째 매개변수로 전달된 default 값을 대신 리턴한다.

```
>>> a = {'A':90, 'B':80}
>>> a.get('C', 70)
70
```

여기서는 'C'에 해당되는 key가 없으므로 디폴트 값으로 전달된 70을 리턴한다.

Q3 리스트의 더하기와 extend 함수

리스트 a에 +를 사용하는 경우에 대해서 먼저 살펴보자.

```
>>> a = [1, 2, 3]
>>> id(a)
4302429640
```

id 함수는 입력으로 받은 리스트 a의 주소 값을 리턴한다. 현재 a라는 리스트는 4302429640이라는 주소에 저장되어 있다.

```
>>> a = a + [4, 5]
>>> a
[1, 2, 3, 4, 5]
```

리스트 a에 +를 사용하여 [4, 5]라는 리스트를 더해 보았다. 그리고 다시 다음과 같이 리스트 a의 주소 값을 확인해 보자.

```
>>> id(a)
4302472072
```

이전에 리스트 a가 저장되어 있던 주소와 다른 값을 리턴하는 것을 확인할 수 있다. 주소 값이 다르므로 +를 사용하면 리스트 a의 값이 변하는 것이 아니라 두 리스트가 더해진 새로운 리스트가 반환된다는 것을 확인할 수 있다.

이번에는 extend 함수를 사용해 보자.

```
>>> a = [1, 2, 3]
>>> id(a)
4302429640
```

리스트 a를 생성하고 그 주소 값을 출력해 보았다.

```
>>> a.extend([4, 5])
>>> a
[1, 2, 3, 4, 5]
```

그리고 리스트 a에 extend를 사용하여 [4, 5]라는 리스트를 더해 주었다. 그리고 다시 다음과 같이 리스트 a의 주소 값을 확인해 보도록 하자.

```
>>> id(a)
4302429640
```

+를 사용하여 더한 경우와는 달리, 주소 값이 변하지 않고 그대로 유지되는 것을 확인할 수 있다.

Q4 리스트 총합 구하기

```
A = [20, 55, 67, 82, 45, 33, 90, 87, 100, 25]

result = 0
while A:                # A 리스트에 값이 있는 동안
    mark = A.pop()      # A리스트의 가장 마지막 항목을 하나씩 뽑아냄
    if mark >= 50:      # 50점 이상의 점수만 더함
        result += mark

print(result)           # 481 출력
```

Q5 피보나치 함수

피보나치 수열은 다음과 같은 순서로 결괏값을 반환한다.

① fib(0) → 0 반환

② fib(1) → 1 반환

③ fib(2) → fib(0) + fib(1) → 0 + 1 → 1 반환

④ fib(3) → fib(1) + fib(2) → 1 + 1 → 2 반환

⑤ fib(4) → fib(2) + fib(3) → 1 + 2 → 3 반환

⑥ (…생략…)

n이 0일 때는 0을 반환, 1일 때는 1을 반환한다. n이 2 이상일 경우에는 이전의 두 값을 더하여 반환한다. 재귀 호출을 사용하면 피보나치 함수를 다음과 같이 간단하게 작성할 수 있다.

```
def fib(n):
    if n == 0 : return 0          # n이 0일 때는 0을 반환
    if n == 1 : return 1          # n이 1일 때는 1을 반환
    return fib(n-2) + fib(n-1)  # n이 2 이상일 때는 그 이전의 두 값을 더하여 반환

for i in range(10):
    print(fib(i))
```

0부터 9까지의 피보나치 수열의 결괏값을 출력하여 그 값을 확인해 보았다.

Q6 숫자의 총합 구하기

```
user_input = input("숫자를 입력하세요: ")
numbers = user_input.split(",")
total = 0
for n in numbers:
    total += int(n)     # 입력은 문자열이므로 숫자로 변환해야 한다.
print(total)
```

실행 결과

```
숫자를 입력하세요: 65,45,2,3,45,8
168
```

Q7 한 줄 구구단

```
user_input = input("구구단을 출력할 숫자를 입력하세요(2~9):")
dan = int(user_input)                # 입력 문자열을 숫자로 변환
for i in range(1, 10):
    print(i * dan, end=' ')          # 1줄로 출력하기 위해 줄 바꿈 문자 대신 공백 문자를 마지막
에 출력
```

Q8 파일을 읽어 역순으로 저장하기

파일 객체의 readlines를 사용하여 모든 라인을 읽은 후에 reverse를 사용하여 역순으로 정렬한 다음 다시 파일에 저장한다.

```
f = open('abc.txt', 'r')
lines = f.readlines()     # 모든 라인을 읽음.
f.close()

lines.reverse()           # 읽은 라인을 역순으로 정렬
```

```
f = open('abc.txt', 'w')
for line in lines:
    line = line.strip()      # 포함되어 있는 줄 바꿈 문자 제거
    f.write(line)
    f.write('\n')            # 줄 바꿈 문자 삽입
f.close()
```

Q9 평균값 구하기

```
f = open("sample.txt")
lines = f.readlines()        # sample.txt를 줄 단위로 모두 읽는다.
f.close()

total = 0
for line in lines:
    score = int(line)        # 줄에 적힌 점수를 숫자형으로 변환한다.
    total += score
average = total / len(lines)

f = open("result.txt", "w")
f.write(str(average))
f.close()
```

sample.txt의 점수를 모두 읽기 위해 파일을 열고 readlines를 사용하여 각 줄의 점수 값을 모두 읽어 들여 총 점수를 구한다. 총 점수를 sample.txt 안 내용의 행 개수로 나누어 평균값을 구한 후 그 결과를 result.txt에 쓴다. 숫자 값은 result.txt 에 바로 쓸 수 없으므로 str 함수를 사용해 문자열로 변경한 후 파일에 쓴다.

Q10 계산기 만들기

```
class Calculator:
    def __init__(self, numberList):
        self.numberList = numberList

    def sum(self):
        result = 0
        for num in self.numberList:
            result += num
        return result

    def avg(self):
        total = self.sum()
        return total / len(self.numberList)

cal1 = Calculator([1, 2, 3, 4, 5])
print(cal1.sum())
print(cal1.avg())

cal2 = Calculator([6, 7, 8, 9, 10])
print(cal2.sum())
print(cal2.avg())
```

Q11 모듈을 사용하는 방법

파이썬 셸에서 mymod.py 모듈을 인식하기 위해서는 다음과 같은 3가지 방법을 사용할 수 있다.

① sys 모듈 사용하기

다음과 같이 sys.path에 C:\doit이라는 디렉터리를 추가하면 C:\doit 디렉터리에 있는 mymod 모듈을 사용할 수 있다.

```
>>> import sys
>>> sys.path.append("c:/doit")
>>> import mymod
```

② PYTHONPATH 환경 변수 사용하기

다음처럼 PYTHONPATH 환경 변수에 C:\doit 디렉터리를 지정하면 C:\doit 디렉터리에 있는 mymod 모듈을 사용할 수 있다.

```
C:\Users\home>set PYTHONPATH=c:\doit
C:\Users\home>python
>>> import mymod
```

③ 현재 디렉터리 사용하기

파이썬 셸을 mymod.py가 있는 위치로 이동하여 실행해도 mymod 모듈을 사용할 수 있다. sys.path에는 현재 디렉터리를 의미하는 '.'이 항상 포함되어 있기 때문이다.

```
C:\Users\home>cd c:\doit
C:\doit>python
>>> import mymod
```

Q12 오류와 예외 처리

70이 출력된다.

① result의 초깃값은 0이다.

② try 문 안의 [1, 2, 3][3]이라는 문장 수행 시 IndexError가 발생하여 except IndexError: 구문으로 이동하게 되어 result에 30이 더해져 30이 된다.

③ 최종적으로 finally 문이 실행되어 result에 4가 더해져 70이 된다.

④ print(result)가 수행되어 result의 최종 값인 70이 출력된다.

Q13 DashInsert 함수

다음 프로그램의 주석을 참고하자.

```
data = "4546793"
numbers = list(map(int, data))          # 숫자 문자열을 숫자 리스트로 변경
result = []
```

```
for i, num in enumerate(numbers):
    result.append(str(num))
    if i < len(numbers)-1:                          # 다음 수가 있다면
        is_odd = num % 2 == 1                       # 현재 수가 홀수
        is_next_odd = numbers[i+1] % 2 == 1         # 다음 수가 홀수
        if is_odd and is_next_odd:                  # 연속 홀수
            result.append("-")
        elif not is_odd and not is_next_odd:        # 연속 짝수
            result.append("*")

print("".join(result))
```

Q14 문자열 압축하기

먼저 입력 문자열의 문자를 확인해 동일한 문자가 들어올 경우에는 해당 문자의 숫자 값을 증가시킨다. 만약 다른 문자가 들어올 경우에는 해당 문자의 숫자 값을 1로 초기화하는 방법을 사용하여 작성한 코드이다. 상세한 설명은 다음 프로그램의 주석을 참고하자.

```
def compress_string(s):
    _c = ""             # s 문자열 중 현재 진행 중인 문자를 임시 저장하기 위한 변수
    cnt = 0             # 해당 문자가 몇 번 반복했는지 알 수 있는 카운트 변수
    result = ""         # 이 함수의 최종 리턴 문자열(예: a3b2c5a1)
    for c in s:         # 입력받은 문자열 s에서 문자 하나씩 c에 대입
        if c!=_c:       # 현재 진행 중인 문자와 c가 같지 않은 경우. 즉 새로운 문자의 시작
            _c = c      # 현재 진행 중인 문자와 같지 않으므로 현재 진행 문자는 c로 대입
            if cnt: result += str(cnt)      # 새로운 문자이므로 결과 문자열에 이전 문자의 카
                                              운트(있을 경우에만)에 해당하는 값을 더해야 함.
            result += c             # 새로운 문자이므로 결과 문자열에 새로운 문자를 더함.
            cnt = 1                 # 새로운 문자이므로 카운트는 1로 초기화
        else:                       # 현재 진행 중인 문자와 c가 같으므로 카운트 증가
            cnt += 1
    if cnt: result += str(cnt)      # for loop를 벗어날 때 이전 문자의 카운트는 마지막으로 한
                                      번 더해야 함
    return result                   # 최종 문자열 리턴

print(compress_string("aaabbccccca"))  # a3b2c6a1 출력
```

Q15 Duplicate Numbers 함수

```
def chk_dup_numbers(s):
    result = []
    for num in s:
        if num not in result:
            result.append(num)
        else:
            return False
    return len(result) == 10

print(chk_dup_numbers("0123456789"))     # True 리턴
print(chk_dup_numbers("01234"))          # False 리턴
print(chk_dup_numbers("01234567890"))    # False 리턴
print(chk_dup_numbers("6789012345"))     # True 리턴
print(chk_dup_numbers("012322456789"))   # False 리턴
```

리스트 자료형을 사용하여 중복된 값이 있는지 먼저 조사한다. 중복된 값이 있을 경우는 False를 리턴한다. 최종적으로 중복된 값이 없을 경우 0~9까지의 숫자가 모두 사용되었는지 판단하기 위해 입력 문자열의 숫자 값을 저장한 리스트 자료형의 총 개수가 10인지를 조사하여 10일 경우에는 True, 아닐 경우에는 False를 리턴한다.

016 오스 부호 해독

```
dic = {
    '.-':'A','-...':'B','-.-.':'C','-..':'D','.':'E','..-.':'F',
    '--.':'G','....':'H','..':'I','.---':'J','-.-':'K','.-..':'L',
    '--':'M','-.':'N','---':'O','.--.':'P','--.-':'Q','.-.':'R',
    '...':'S','-':'T','..-':'U','...-':'V','.--':'W','-..-':'X',
    '-.--':'Y','--..':'Z'
}

def morse(src):
    result = []
    for word in src.split("  "):
        for char in word.split(" "):
            result.append(dic[char])
        result.append(" ")
    return "".join(result)

print(morse('.... .  ... .-.. . . .--. ...  . .- .-. .-.. -.--'))
```

오스 부호 규칙 표를 딕셔너리로 작성한 후 입력에 해당되는 모스 부호 문자열을 먼저 단어(공백 문자 2개)로 구분한다. 그 후 단어를 문자(공백 문자 1개)로 구분하여 해당 모스 부호 값을 딕셔너리에서 찾아서 그 결괏값을 구한다.

017 정규식 — 기초 메타 문자

보기 중 이 조건에 해당되는 것은 B이다.

다음은 이 문제의 정규식 매치 결과를 확인해 보는 파이썬 코드이다.

```
import re

p = re.compile("a[.]{3,}b")

print(p.match("acccb"))     # None
print(p.match("a....b"))    # 매치 객체 출력
print(p.match("aaab"))      # None
print(p.match("a.cccb"))    # None
```

Q18 정규식 — 문자열 검색

정규식 [a-z]+는 소문자로 이루어진 단어를 뜻하므로 '5 python' 문자열에서 'python'과 매치될 것이다. 따라서 python 문자열의 인덱스 범위는 m.start()에서 m.end()까지이므로 10이 출력된다.

```
import re

p = re.compile('[a-z]+')
m = p.search("5 python")
print(m.start() + m.end())    # 10 출력
```

5 python 문자열에서 n이 일곱 번째 문자여서 m.end()의 값이 7이 나올것 같지만 다음과 같은 슬라이싱 규칙을 떠올려보면 왜 8이 나오는지 쉽게 이해될 것이다.

```
>>> a = "5 python"
>>> a[2:8]
'python'
>>> a[2:7]
'pytho'
```

Q19 정규식 — 그루핑

전화번호 패턴은 다음과 같이 작성할 수 있다.

```
pat = re.compile("\d{3}[-]\d{4}[-]\d{4}")
```

이 전화번호 패턴 중 뒤의 숫자 4개를 변경할 것이므로 필요한 앞부분을 다음과 같이 그루핑한다.

```
pat = re.compile("(\d{3}[-]\d{4})[-]\d{4}")
```

[illegible] 정규식 pat에 sub 함수를 사용하여 다음과 같이 문자열을 변경한다.

```
import re

s = """
park 010-9999-9988
kim 010-9909-7789
lee 010-8789-7768
"""

pat = re.compile("(\d{3}[-]\d{4})[-]\d{4}")
result = pat.sub("\g<1>-####", s)

print(result)
```

정규식 — 전방 탐색

.com과 .net에 해당되는 이메일 주소만을 매치하기 위해서 이메일 주소의 도메인 부분에 다음과 같은 긍정형 전방탐색 패턴을 적용한다.

```
pat = re.compile(".*[@].*[.](?=com$|net$).*$")
```

다음은 이 패턴을 적용한 파이썬 코드이다.

```
import re

pat = re.compile(".*[@].*[.](?=com$|net$).*$")

print(pat.match("pahkey@gmail.com"))
print(pat.match("kim@daum.net"))
print(pat.match("lee@myhome.co.kr"))
```