파이썬 기초 실습

6장

6-1 내가 프로그램을 만들 수 있을까?

`프로그램을 만들려면 가장 먼저 '입력'과 '출력'을 생각하라.`

EX> 구구단 2단 만들기

0. 틀 만들기

```
함수이름: Gu
입력받는 값: 2 (구구단 2단 만드니까)
출력하는 값: 2단
결과 저장 형태: 연속된 자료형으로 나타나니까 리스트
```

1.
```
#Gu라는 함수에 2를 넣으면 결과가 나와야한다.
result=Gu(2)
```

2. 리스트 형으로 결과값을 받는다.
3. Gu 함수 짜기

```
def Gu(n):
    print(n)

Gu(2)
2 #입력값이 잘 출력 되는지를 확인 하는 과정이다.
```

4. 결괏값을 담을 리스트 생성하기

```
def Gu(n):
    result=[]
```

5. append함수를 사용해 리스트에 요소를 추가하는 방법도 있지만 2*n이라는 반복이 이루어 지므로 반복문을 사용하자

```
def Gu(n):
    result=[]#결괏값을 리스트 형으로 받는다.
    i=1
    while i<10:#i가 1~9까지 반복한다.
        result.append(n*i)#result= n*i의 값을 리스트로 추가한다.
        i=i+1#위의 일을 수행한 후 i에 1을 더한다.
    return result#result를 돌려준다
```

6. 마지막 테스트 해보기

```
print(Gu(2))
[2, 4, 6, 8, 10, 12, 14, 16, 18]
```

6-2 3과 5의 배수 합하기

문제:

```
10 미만의 자연수에서 3과 5의 배수를 구하면 3,5,6,9이다. 이들의 총합은 23이다. 1000미만의 자연수에서 3의 배수와 5의 배수의 총합을 구하라.
```

0. 틀짜기

```
입력받는 값: 1-999까지 (1000미만이므로)
출력하는 값: 3의 배수와 5의 배수의 총합
생각할 점
1. 3의 배수와 5의 배수를 찾기
2. 3의 배수와 5의 배수 합치기
```

1. 함수짜기

```
#1~999까지 숫자 입력하는 방법
result=0 #result값을 0으로 지정
for n in range(1,1000): #n은 1~999까지의 수 진행
    if n%3 or n%5==0: #3,5의 배수이므로 n을 3,5으로 나누면 나머지가 없다.
        result+=n #reslut값은 n의 합이다.
```

2. 결과 출력해보기

```
print(result)
365832
```

문제점

: 15의 배수는 3으로도 나뉘고 5로도 나뉘어 이중으로 값이 출력되고, 더해지지 않을 수도 있으므로, or 연산자를 사용한건데, 다음과 같이 코딩시 이중으로 더해 진다.

```
#2중으로 더해지는 문제점
result=0
for n in range(1,1000):
    if n%3==0:
        result=n
    if n%5==0:
        result+=n
print(result)
```

6-3 게시판 페이징하기

1.
```
총페이지수=(총 건수/한페이지당 보여줄 건수)+1
```

2.
```
def page(m,n): #m=게시물의 수, n=페이지당 보여 줄 게시물 수
    if m%n==0: # m/n실행후 나머지가 0이면
        return m//n #소수점을 버리는 몫을 돌려준다.
    else:
        return m//n+1
```

6-4 간단한 메모장 만들기

0. 틀짜기

```
필요한 기능: 메모장에 쓰고 읽기
입력받을 값: 내용, 실행 옵션
출력값: memo.txt
```

1. 메모 출력하는 코드

```
#C:/doit/memo.py
import sys #sys함수를 불러옴

option=sys.argv[1]
memo=sys.argv[2]
#sys.argv는 프로그램을 실행시 입력된 값을 읽어 들일 수 있는 파이썬 라이브러리
print(option)
print(memo)
```

2. memo.py저장 이후 다음명령 시행

```
C:\doit>python memo.py -a "life is too short"
-a
life is too short
```

입력 옵션과 메모 내용이 그대로 출력되는 것을 확인하기 위한 과정

3. 메모를 파일에 쓰도록 수정

```
import sys

option=sys.argv[1]

if option=='-a': #option이 -a일경우
    memo=sys.argv[2]
    f=open('memo.txt','a')#memo.txt에 파일 마지막에 새로운 내용을 추가
    f.write(memo)
    f.write('\n')
    f.close()
```

4. 입력해보기

```
C:\doit>python memo.py -a "life is too short"
C:\doit>python memo.py -a "you need python"
```

직접확인해보면 정확히 써져 있는 것을 확인 할 수 있다. 프롬프트에서 확인하려면 다음과 같이 실행해 주면 된다.

```
C:\doit>type memo.txt
    #나는 안되더라
```

5. 메모의 내용을 출력하기

```
import sys

option=sys.argv[1]

if option=='-a': #option이 -a일경우
    memo=sys.argv[2]
    f=open('memo.txt','a')#memo.txt에 파일 마지막에 새로운 내용을 추가
    f.write(memo)
    f.write('\n')
    f.close()
elif option=='-v':
    f=open('memo,txt')
    memo=f.read()
    f.close
    print(memo)
```

6. 실행해보기

```
C:\doit>python memo.py -v
#입력 내용 출력, 나는 안되더라 memo.txt가 C:/doit에 있는데
```

6-5 탭을 4개의 공백으로 바꾸기

탭을 공백 4개로 바꾸어 주는 스크립트를 작성해 보자.

```
python tabto4.py src dst
# tabto4.py는 우리가 작성해야 할 파이썬 프로그램 이름
# src는 탭을 포함하고 있는 원본 파일 이름
# dst는 파일 안의 탭을 공백 4개로 변환한 결과를 저장할 파일
ex) a.txt 파일에 있는 탭을 4개의 공백으로 바꾸어 b.txt파일에 저장하고 싶으면
python tabto4.by a.txt b.txt라 쓰면 된다.
```

1. tabto4.py파일 작성

```
#c:/doit/tabto4.py
import sys

src=sys.argv[1]
dst=sys.argv[2]

print(src)
print(dst)
#sys.argv를 사용하여 입력값을 확인하도록 만든 코드이다.
```

2. 정상 출력 확인

```
#in CMD
C:\doit> python tabto4.py a.txt b.txt
a.txt #src가 a.txt를 받아 print한 결과
b.txt #dst가 b.txt를 받아 print한 결과
```

3. 테스트 과정

```
#a.txt작성
life    is  too short
you need    python
```

4. 탭 문자 공백 4개로 변환하는 코드 짜기

```
import sys

src=sys.argv[1]
dst=sys.argv[2]

f=open(src) #src에 해당되는 파일 열기
tab_content=f.read() #tab_content가 공백을 읽은 것
f.close()

space_content=tab_content.replace("\t",""*4) #\t=tab, ""=space공백
print(space_content)
```

5. 출력 확인하기

```
python tabto4.py a.txt b.txt
life    is    too    short
you    need    python
```

6. 변경내용 b.txt에 저장하기

```
import sys

src=sys.argv[1]
dst=sys.argv[2]

f=open(src)
tab_content=f.read()
f.close()
space_content=tab_content.replace("\t"," "*4)
print(space_content)

f=open(dst,'w') #dst를 열고 w=글을 쓰기 위에 연다
f.write(space_content) #space_content내용을 적는다.
f.close
```

7. 최종 실행

```
python tabto4.py a.txt b.txt
#실행시 b.txt파일이 형성되며 4space로 나뉜 문장이 적혀 있다.
```

6-6 하위 디렉터리 검색하기

특정 디렉터리에서 하위 모든 파일 중 파이썬 파일만 출력해주는 프로그램 만들기

1.
```
#C:/doit/sub_dir_search.py

def search(dirname):
    print(dirname)

search("C:/")
#search함수를 만들고 시작 디렉터리를 입력받도록 코드를 작성
```

2. 파일 검색할 수 있도록 변경하기

```
#C:/doit/sub_dir_search.py
import os

def search(dirname):
    filenames=os.listdir(dirname)
    for filename in filenames:
        full_filename=os.path.join(dirname,filename)
        print(full_filename)

search("C:/")
```

os.listdir를 사용하면 해당 디렉터리에 있는 파일들의 리스트를 구할 수 있다.

구하는 파일 리스트는 파일 이름만 포함되어 있으므로 경로를 포함한 파일 이름을 구하기 위해 입력 받는 dirname을 앞에 붙여줌.

os 모듈에는 디렉터리와 파일 이름을 이어주는 os.path.join함수가 있으므로 이 함수를 사용하면 디렉터리를 포함한 전체 경로를 쉽게 구할 수 있다.

3. 파이썬 파일만 검색하도록 변경하기

```
#C:/doit/sub_dir_search.py
import os

def search(dirname):
    filenames=os.listdir(dirname)
    for filename in filenames:
        full_filename=os.path.join(dirname,filename)
        ext=os.path.splitext(full_filename)[-1]
        if ext=='.py':
            print(full_filename)

search("C:/")
```

   - 파일 이름에서 확장자만 추출하기 위해 사용하는 함수로 os.path.splitext는 파일 확장자를 기준으로 두 분분으로 나누어 주기 때문에 (full_filename)[-1]은 해당 파일의 확장자 이름이 된다.

4. 하위 디렉터리에서도 검색이 가능하도록 변경하기

```
#C:/doit/sub_dir_search.py
import os

def search(dirname):
    try:
        filenames=os.listdir(dirname)
        for filename in filenames:
```

```
            full_filename=os.path.join(dirname,filename)
            if os.path.isdir(full_filename):
                search(full_filename)
            else:
                ext=os.path.splitext(full_filename)[-1]
                if ext=='.py':
                    print(full_filename)
    except PermissionError:
        pass

search("C:/")
```

- try...except PermissionError로 함수 전체를 감싸면 os.listdir를 수행시 권한이 없는 디렉터리에 접근해도 그냥 수행되도록 하기 위해서
- os.path.isdir= full_filename이 디렉터리인지 파일인지 구분
  - 디렉터리 일경우=> search
  - 아닐경우-> .py파일을 찾아서 출력한다.

쉽게 만들기

os.walk= 시작 디렉터리부터 시작하여 그 하위 모든 디렉터리를 차례대로 방문하는 함수

```
import os

for(path,dir,files)in os.walk("C:/"):
    for filenane in files:
        ext=os.path.splitext(filename)[-1]
        if ext=='.py':
            print("%s/%s"%(path,filename))
```