

세상의 속도를
따라잡고 싶다면



점프 투 파이썬

박응용 지음(위키독스 운영자)



파이썬 프로그래밍, 어떻게 시작해야 할까?



- 06-1 내가 프로그램을 만들 수 있을까?
- 06-2 3과 5의 배수를 모두 더하기
- 06-3 게시판 페이징하기
- 06-4 간단한 메모장 만들기
- 06-5 탭 문자를 공백 문자 4개로 바꾸기
- 06-6 하위 디렉터리 검색하기



■ 구구단 2단 프로그램 만들기



프로그램을 만들려면 가장 먼저 '입력' 과 '출력' 을 생각하라.

- 구구단 프로그램 중 2단을 만든다면 2를 입력값으로 주었을 때 어떻게 출력되어야 할까?

함수 이름은? gugu로 짓자

입력받는 값은? 2

출력하는 값은? 2단(2, 4, 6, 8, ..., 18)

결과는 어떤 형태로 저장하지? 연속된 자료형이므로 리스트!

■ 구구단 2단 프로그램 만들기

1. gugu 함수 구상하기

- gugu라는 함수에 2를 입력값으로 주면 result라는 변수에 곱값을 넣으라는 뜻

```
result = gugu(2)
```

2. 곱값을 어떤 형태로 받을 것인지 고민하기

- 2단 곱값 = 2, 4, 6, ... 18
 - 리스트 자료형을 활용해보면 어떨까?
 - result = [2, 4, 6, 8, 10, 12, 14, 16, 18]

■ 구구단 2단 프로그램 만들기

3. gugu 함수 만들기

- 입력값이 잘 들어오는지 확인하기 위해 입력값을 출력하도록 구현

```
def gugu(n):  
    print(n)
```

4. 곱값을 담은 리스트를 생성하도록 수정하기

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

■ 구구단 2단 프로그램 만들기

5. result에 곱값을 넣을 방법 고민하기

- 예) 리스트에 요소를 추가하는
append 함수 사용하기
- 무식한 방법이지만
입력값 2를 주었을 때 원하는 곱값을
얻을 수 있음

```
def gugu(n):  
    result = []  
    result.append(n*1)  
    result.append(n*2)  
    result.append(n*3)  
    result.append(n*4)  
    result.append(n*5)  
    result.append(n*6)  
    result.append(n*7)  
    result.append(n*8)  
    result.append(n*9)  
    return result
```

```
print(gugu(2))
```

실행 결과

[2, 4, 6, 8, 10, 12, 14, 16, 18]

■ 구구단 2단 프로그램 만들기

6. 반복문 활용하기

- `result.append(n*숫자)` 형태가 1부터 9까지 반복되고 있음
- 반복문을 만들면 해결할 수 있지 않을까?
- 반복문을 활용하여 1부터 9까지 출력하기

```
>>> i = 1
>>> while i < 10:
...     print(i)
...     i = i + 1
...
1
2
3
4
5
6
7
8
9
```

■ 구구단 2단 프로그램 만들기

7. gugu 함수 완성하기

```
def gugu(n):  
    result = []  
    i = 1  
    while i < 10:  
        result.append(n*i)  
        i = i + 1  
    return result
```

```
print(gugu(2))
```

실행 결과

```
[2, 4, 6, 8, 10, 12, 14, 16, 18]
```


- 3과 5의 배수를 모두 더하기
 - 다음 문제는 어떻게 풀면 좋을까?

10 미만의 자연수에서 3과 5의 배수를 구하면 3, 5, 6, 9이다. 이들의 총합은 23이다.
1,000 미만의 자연수에서 3의 배수와 5의 배수의 총합을 구하라.

입력받는 값은? 1부터 999까지(1000 미만의 자연수)
출력하는 값은? 3의 배수와 5의 배수의 총합
생각해 볼 것은? 하나. 3의 배수와 5의 배수는 어떻게 찾지?
둘. 3의 배수와 5의 배수가 겹칠 때는 어떻게 하지?

■ 3과 5의 배수를 모두 더하기

■ 주요 포인트

- 1,000 미만의 자연수를 구하는 방법
- 3과 5의 배수를 구하는 것

1. 1,000 미만의 자연수 구하기

- 1) 변수에 초깃값 1을 준 후 루프를 돌리며 1씩 증가시켜서 999까지 진행하는 방법

```
n = 1
while n < 1000:
    print(n)
    n += 1
```

- 2) range 함수 사용하는 방법

```
for n in range(1, 1000):
    print(n)
```

■ 3과 5의 배수를 모두 더하기

2. 3과 5의 배수를 구하는 방법

- 1,000 미만의 자연수 중 3의 배수

3, 6, 9, 12, 15, 18, ..., 999

- 3의 배수 구하기

- 1부터 1,000까지의 수 중 3으로 나누었을 때 나머지가 0인 경우(나누어 떨어지는 경우)

```
for n in range(1, 1000):  
    if n % 3 == 0:  
        print(n)
```

% 연산자 활용
(5의 배수는 $n \% 5$ 가 0이 되는 수)

■ 3과 5의 배수를 모두 더하기

3. 프로그램 완성하기

```
result = 0
for n in range(1, 1000):
    if n % 3 == 0 or n % 5 == 0:
        result += n
print(result)
```

- 3과 5의 배수에 해당하는 수를 result 변수에 계속 더해줌
- 3으로도 5로도 나누어지는 15와 같은 수를 중복으로 더하지 않도록 or 연산자 사용

■ 3과 5의 배수를 모두 더하기

3. 프로그램 완성하기

■ 잘못된 풀이의 예)

- 3의 배수도 되고 5의 배수도 되는 15와 같은 값을 이중으로 더함

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

이렇게 풀지 않도록
주의하자!



■ 게시판페이징하기

- A씨는 게시판 프로그램을 작성하는데,
게시물의 총 개수와 한 페이지에 보여 줄 게시물 수를 입력으로 주었을 때
총 페이지 수를 출력하는 프로그램이 필요하다고 함

함수 이름은? `get_total_page`

입력받는 값은? 게시물의 총 개수(m), 한 페이지에 보여 줄 게시물 수(n)

출력하는 값은? 총 페이지 수

- A씨가 필요한 프로그램을 만드는 방법은?

■ 게시판페이징하기

■ 입력값과 결과값 살펴보기

- 게시물의 총 개수가 5이고 한 페이지에 보여 줄 게시물 수가 10이면 총 페이지 수는 1
- 게시물의 총 개수가 15이고 한 페이지에 보여 줄 게시물 수가 10이면 총 페이지 수는 2

게시물의 총 개수(m)	페이지 당 보여 줄 게시물 수(n)	총 페이지 수
5	10	1
15	10	2
25	10	3
30	10	3

■ 게시판페이징하기

1. 총 페이지 수 구하는 공식 만들기

- 총 게시물 수(m)를 한 페이지에 보여 줄 게시물 수(n)로 나누고 1을 더하면 총 페이지 수

$$\text{총 페이지 수} = (\text{총 게시물 개수} / \text{한 페이지당 보여 줄 개수}) + 1$$

- 예) 총 게시물 수(m) = 5, 한 페이지에 보여 줄 게시물 수(n) = 10일 때, 총 페이지 수 = $(5/10) + 1 = 1$
- 예) 총 게시물 수(m) = 15, 한 페이지에 보여 줄 게시물 수(n) = 10일 때, 총 페이지 수 = $(15/10) + 1 = 2$

■ 게시판페이징하기

2. 공식을 적용했을 경우 총 페이지 수가 표의 값처럼 구해지는지 확인
 - m 을 n 으로 나눌 때 소수점 아래 자리를 버리기 위해 $/$ 대신 $//$ 연산자 사용

```
def get_total_page(m, n):  
    return m // n + 1          # m을 n으로 나눌 때 소수점 아래 자리를 버리기 위해 // 연산자 사용  
  
print(get_total_page(5, 10))   # 1 출력  
print(get_total_page(15, 10))  # 2 출력  
print(get_total_page(25, 10))  # 3 출력  
print(get_total_page(30, 10))  # 4 출력
```

- 네 번째 케이스는 총 개수 30, 한 페이지에 보여 줄 개수가 10인데 3이 아닌 4가 출력되어 실패!

■ 게시판페이징하기

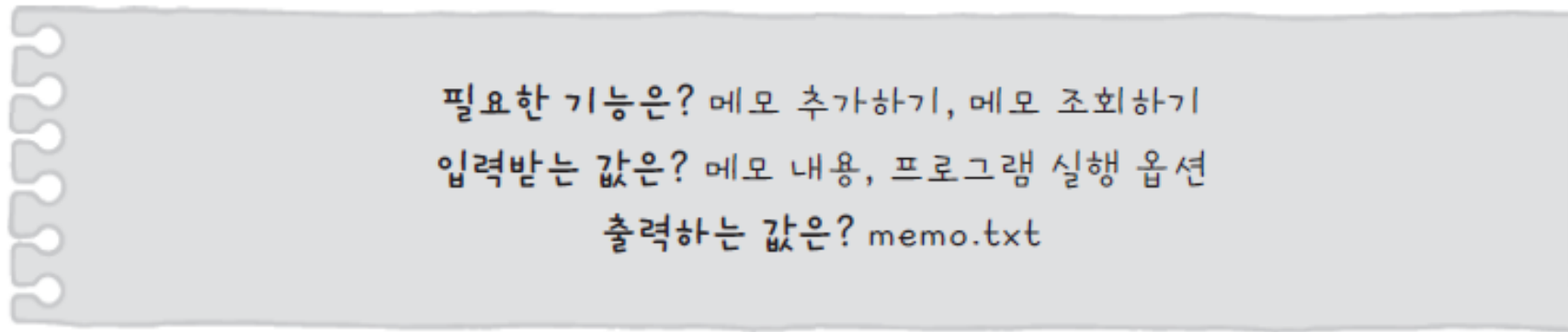
3. 실패 케이스 해결하기

- 실패 케이스는 총 게시물 수와 한 페이지에 보여 줄 게시물 수를 나눈 나머지 값이 0이 될 때 발생
- 나머지가 0인 경우는 나누기의 몫만 리턴하고 이외의 경우에는 1을 더하여 리턴하도록 변경
- 프로그램을 실행해 보면 모든 케이스가 원하던 결과를 출력함

```
def get_total_page(m, n):  
    if m % n == 0:  
        return m // n  
    else:  
        return m // n + 1  
  
print(get_total_page(5, 10))  
print(get_total_page(15, 10))  
print(get_total_page(25, 10))  
print(get_total_page(30, 10))    # 3 출력
```

■ 간단한 메모장 만들기

- 원하는 메모를 파일에 저장하고 추가 및 조회가 가능한 간단한 메모장을 만드는 프로그램



- 파이썬 프로그램 memo.py 구현하기

```
python memo.py -a "Life is too short"
```

■ 간단한 메모장 만들기

1. memo.py 작성하기

- 입력으로 받은 옵션과 메모 출력

```
import sys

option = sys.argv[1]
memo = sys.argv[2]

print(option)
print(memo)
```

- sys.argv

- 프로그램을 실행할 때 입력된 값을 읽어 들일 수 있는 파이썬 라이브러리

sys.argv[0]	파이썬 프로그램 이름 (memo.py)
sys.argv[1]	프로그램 실행 옵션 (-a)
sys.argv[2]	메모 내용

■ 간단한 메모장 만들기

2. memo.py 실행하기

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

- 입력으로 전달한 옵션 '-a'와 메모 내용 'Life is too short'이 그대로 출력됨

■ 간단한 메모장 만들기

3. 입력으로 받은 메모를 파일에 쓰도록 코드 변경하기

- 옵션이 -a인 경우에만 memo 값을 읽도록 수정
- 메모가 항상 새로운 내용이 작성되는 것이 아니라 한 줄씩 추가되어야 하므로 파일 열기 모드는 a
- 메모를 추가할 때마다 다음 줄에 저장되도록 줄바꿈 문자(\n) 추가

```
import sys

option = sys.argv[1]

if option == '-a':
    memo = sys.argv[2]
    f = open('memo.txt', 'a')
    f.write(memo)
    f.write('\n')
    f.close()
```

■ 간단한 메모장 만들기

4. 코드 실행하기

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

- 파일에 정상적으로 메모가 기입되었는지 파일 내용 확인

```
C:\doit>type memo.txt  
Life is too short  
You need python
```

■ 간단한 메모장 만들기

5. 작성한 메모 출력하기

```
python memo.py -v
```

- 메모 추가는 -a 옵션 사용,
메모 출력은 -v 옵션을 사용하도록 코드 수정
- 옵션으로 -v가 들어온 경우,
memo.txt 파일을 읽어서 출력

```
import sys

option = sys.argv[1]

if option == '-a':
    memo = sys.argv[2]
    f = open('memo.txt', 'a')
    f.write(memo)
    f.write('\n')
    f.close()
elif option == '-v':
    f = open('memo.txt')
    memo = f.read()
    f.close()
    print(memo)
```


■ 간단한 메모장 만들기

6. 코드 실행하기

```
C:\doit>python memo.py -v  
Life is too short  
You need python
```

- 입력한 메모가 그대로 출력됨

■ 탭 문자를 공백 문자 4개로 바꾸기

- 문서 파일을 읽어서 문서 안에 있는 탭 문자(tab)를 공백 문자(space) 4개로 바꾸어주는 스크립트

필요한 기능은? 문서 파일 읽어 들이기, 문자열 변경하기
입력받는 값은? 탭을 포함한 문서 파일
출력하는 값은? 탭이 공백으로 수정된 문서 파일

```
python tabto4.py src dst
```

```
python tabto4.py a.txt b.txt
```

- src : 탭을 포함하고 있는 원본 파일 이름 (예) a.txt
- dst : 파일 안의 탭을 공백 4개로 변환한 결과를 저장할 파일 이름 (예) b.txt

■ 탭 문자를 공백 문자 4개로 바꾸기

1. tabto4.py 작성하기

- sys.argv를 사용하여 입력값을 확인하는 코드

```
import sys

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

print(src)
print(dst)
```

2. tabto4.py 실행하기

```
C:\doit>python tabto4.py a.txt b.txt
a.txt
b.txt
```

- 입력으로 전달한 a.txt와 b.txt가 정상 출력됨

■ 탭 문자를 공백 문자 4개로 바꾸기

3. a.txt 만들기

- 테스트를 위한 원본 파일(탭을 포함하는 파일) a.txt 만들기

```
Life    is        too        short ← 각 단어의 사이에 탭 문자 입력
You    need    python
```

- 각 단어는 탭(Wt) 문자로 분리되도록 입력

■ 탭 문자를 공백 문자 4개로 바꾸기

4. 탭 문자를 포함한 a.txt 파일을 읽어 탭을 공백 4개로 변환하도록 코드 수정하기

- src에 해당되는 입력 파일을 읽어서
그 내용을 tab_content 변수에 저장
- 문자열의 replace 함수를 사용해
탭(\t)을 4개의 공백으로 변환

```
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)
```

- 탭 문자를 공백 문자 4개로 바꾸기

- 5. tabto4.py 실행해 보기

```
C:\doit>python tabto4.py a.txt b.txt  
Life    is    too    short  
You     need  python
```

- 탭 문자가 공백 4개로 변경되어 출력됨
 - 탭과 공백의 차이점을 눈으로 확인하기는 어려움

■ 탭 문자를 공백 문자 4개로 바꾸기

6. 변경된 내용을 b.txt 파일에 저장하도록 코드 수정하기

- 탭이 공백으로 변경된 space_content를 출력 파일인 dst에 쓰도록 코드 수정

```
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)

f = open(dst, 'w')
f.write(space_content)
f.close()
```

- 탭 문자를 공백 문자 4개로 바꾸기

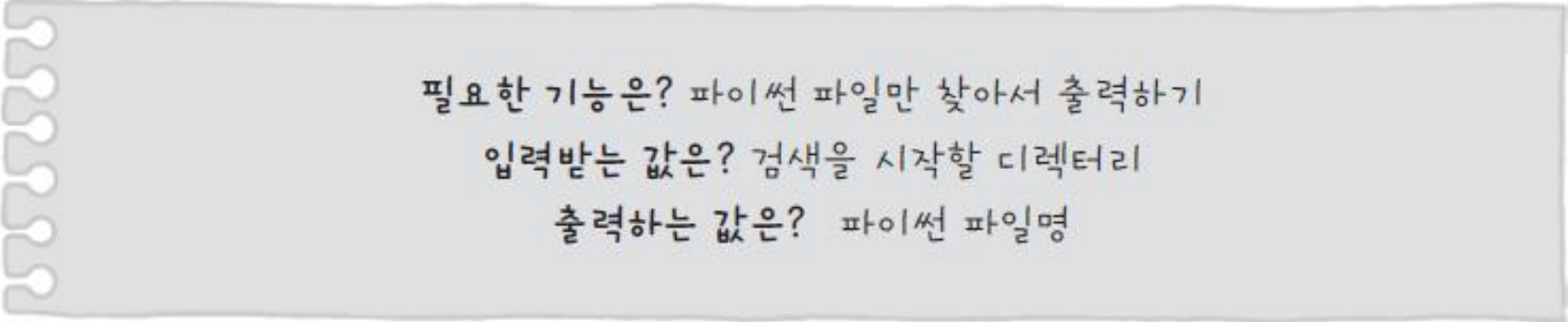
- 7. 프로그램 실행하기

```
C:\doit>python tabto4.py a.txt b.txt
```

- b.txt가 C:\doit 디렉터리에 생성됨

■ 하위 디렉터리 검색하기

- 특정 디렉터리부터 시작해서 그 하위(디렉터리 포함)의 모든 파일 중 파이썬 파일(*.py)만 출력해 주는 프로그램



필요한 기능은? 파이썬 파일만 찾아서 출력하기
입력받는 값은? 검색을 시작할 디렉터리
출력하는 값은? 파이썬 파일명

■ 하위 디렉터리 검색하기

1. sub_dir_search.py 파일 작성하기

```
def search(dirname):  
    print(dirname)  
  
search("c:/")
```

- 시작 디렉터리를 입력 받는 search 함수 작성

■ 하위 디렉터리 검색하기

2. 디렉터리에 있는 파일을 검색하도록 코드 수정하기

- `os.listdir`
 - 해당 디렉터리에 있는 파일의 리스트 반환
 - 파일 리스트는 파일 이름만 포함
 - 경로를 포함한 파일 이름을 구하려면 입력으로 받은 `dirname`을 앞에 덧붙여 주어야 함
- `Os.path.join`
 - 디렉터리와 파일 이름을 이어 주는 함수

```
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:/")
```

■ 하위 디렉터리 검색하기

2. 디렉터리에 있는 파일을 검색하도록 코드 수정하기

■ 코드 수행 결과

```
c:/$Recycle.Bin  
c:/$WINDOWS.~BT  
c:/$Windows.~WS ← 디렉터리 출력 예  
c:/adb  
c:/AMD  
c:/android  
c:/bootmgr  
c:/BOOTNXT  
(...생략...)
```

■ 하위 디렉터리 검색하기

3. 확장자가 .py인 파일만 출력하도록 코드 변경하기

- `os.path.splitext`
 - 파일 이름을 확장자를 기준으로 분리
 - `os.path.splitext(full_filename)[-1]`
 - 해당 파일의 확장자 이름
- 확장자가 .py인 경우만 출력
- 파이썬 파일이 없다면 아무것도 출력되지 않음

```
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:/")
```

■ 하위 디렉터리 검색하기

4. 하위 디렉터리(sub directory)도 검색하도록 코드 변경하기

- try-except 문으로 함수 전체 감싸기
 - os.listdir를 수행할 때 권한이 없는 디렉터리에 접근하더라도 프로그램이 오류로 종료되지 않고 그냥 수행되도록 처리
- os.path.isdir 함수
 - full_filename이 디렉터리인지 파일인지 구별
- full_filename이 디렉터리일 경우 search 함수 재귀 호출

```
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:/")
```



*"Life is too short,
You need Python!"*

인생은 너무 짧으니,
파이썬이 필요해!

감사합니다.