



01-1 파이썬 시작하기

- 귀도 반 로섬(Guido Van Rossum)이 개발한 인터프리터 언어
- 구글에서 만들어진 소프트웨어의 50% 이상이 파이썬
- 드롭박스(Dropbox), 장고(Django), 인스타그램
- 공동 작업과 유지 보수가 매우 쉽고 편하다



01-2 파이썬의 특징[1]

- 파이썬은 인간다운 언어이다

```
if 4 in [1,2,3,4]: print ("4가 있습니다")
```

- 파이썬은 문법이 쉬워 빠르게 배울 수 있다
 - 파이썬을 공부한 지 단 하루 만에 자신이 원하는 프로그램을 작성 (Eric Raymond)
 - 1주일이면 충분 (프로그래밍 유 경험자)



01-2 파이썬의 특징[2]

- 파이썬은 무료이지만 강력하다
 - 사용료 걱정없이 언제 어디서든 파이썬을 다운로드하여 사용
 - 파이썬과 C는 찰떡궁합
 - 파이썬 라이브러리들 중에는 C로 만들어진 것도 많다.



01-2 파이썬의 특징(3)

- 파이썬은 간결하다

```
# simple.py
languages = ['python', 'perl', 'c', 'java']

for lang in languages:
    if lang in ['python', 'perl']:
        print("%6s need interpreter" % lang)
    elif lang in ['c', 'java']:
        print("%6s need compiler" % lang)
    else:
        print("should not reach here")
```

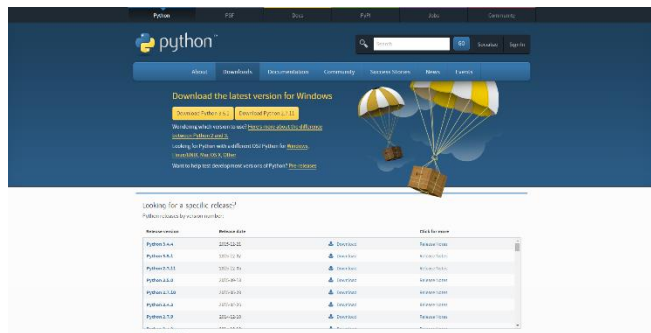
- 파이썬은 가장 좋은 방법 1가지만 이용하는 것을 선호
- 실행이 되게 하려면 꼭 줄을 맞추어야 한다.



01-3 파이썬으로 무엇을 할 수 있을까?[1]

- 파이썬으로 할 수 있는 일
 - 시스템 유틸리티 제작
 - GUI 프로그래밍
 - C/C++와의 결합
 - 웹 프로그래밍
 - 수치 연산 프로그래밍
 - 데이터베이스 프로그래밍
 - 데이터 분석, 사물 인터넷

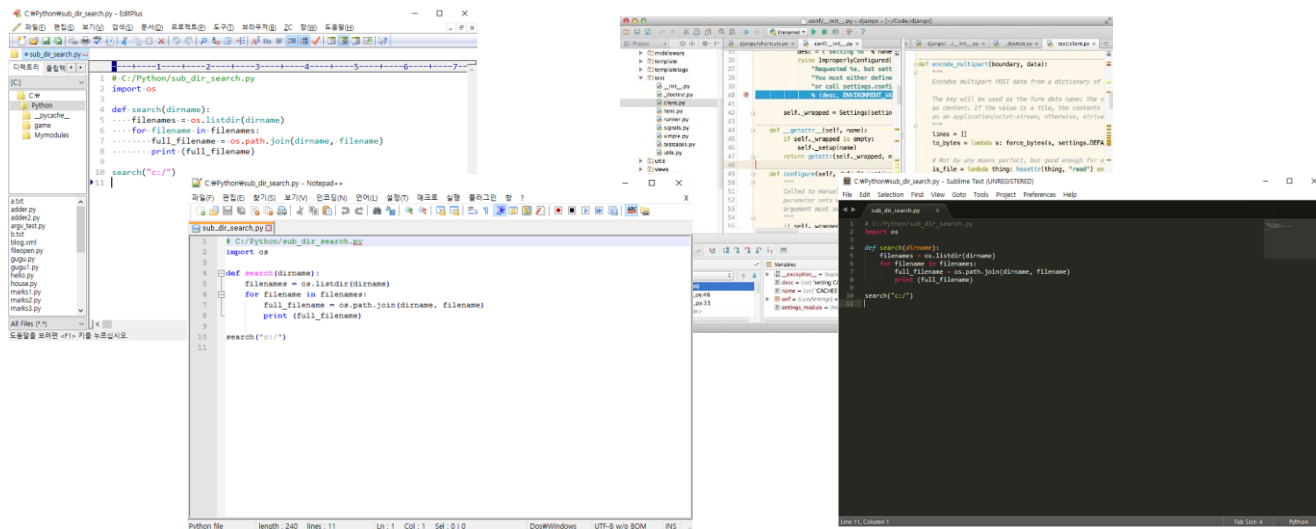
01-4 파이썬 설치하기



- 파이썬 언어 패키지를 다운로드한다.
- 파이썬이 어느 곳에서든지 실행될 수 있도록 "Add Python 3.5 to PATH" 옵션을 선택하도록 하자.

01-5 파이썬 둘러보기(2)

- 추천에디터



- 에디트 플러스, 파이참, 노트패드++, 서브라임텍스트3



파이썬 프로그래밍의 기초, 자료형

자료형을 알고 있다면 그 언어의
절반을 터득한 것



02-1 숫자형

사칙연산

```
>>> a = 3
>>> b = 4
>>> a + b
7
>>> a * b
12
>>> a / b
0.75
```



02-1 숫자형

제 곱

```
>>> a = 3
>>> b = 4
>>> a ** b
81
```

% 연산자

```
>>> 7 % 3
1
>>> 3 % 7
3
```



02-1 숫자형

// 연산자

```
>>> 7 / 4  
1.75  
>>> 7 // 4  
1
```



02-2 문자열 자료형

문자열 자료형 만드는 4가지 방법

```
"Hello World"
```

```
'Python is fun'
```

```
"""Life is too short, You need python"""
```

```
'''Life is too short, You need python'''
```



02-2 문자열 자료형

문자열 더해서 연결하기 (Concatenation)

```
>>> head = "Python"  
>>> tail = " is fun!"  
>>> head + tail  
'Python is fun!'
```

문자열 곱하기

```
>>> a = "python"  
>>> a * 2  
'pythonpython'
```



02-2 문자열 자료형

인덱싱(Indexing)

```
>>> a = "Life is too short, You need Python"
>>> a[0]
'L'
>>> a[12]
's'
>>> a[-1]
'n'
```

파이썬은 0부터 숫자를 센다



02-2 문자열 자료형

슬라이싱(Slicing)

```
>>> a = "Life is too short, You need Python"
>>> a[0:4]
'Life'
```

```
>>> a = "20010331Rainy"
>>> date = a[:8]
>>> weather = a[8:]
>>> date
'20010331'
>>> weather
'Rainy'
```



02-2 문자열 자료형

문자열 포매팅

```
>>> "I eat %d apples." % 3
'I eat 3 apples.'
>>> number = 10
>>> day = "three"
>>> "I ate %d apples. so I was sick for %s days." % (number,
day)
'I ate 10 apples. so I was sick for three days.'
```




02-2 문자열 자료형

문자열 개수 세기(count)

```
>>> a = "hobby"  
>>> a.count('b')  
2
```

위치 알려주기1(find)

```
>>> a = "Python is best choice"  
>>> a.find('b')  
10  
>>> a.find('k')  
-1
```



02-2 문자열 자료형

문자열 삽입(join)

```
>>> a= ","  
>>> a.join('abcd')  
'a,b,c,d'
```

소문자를 대문자로 바꾸기(upper)

```
>>> a = "hi"  
>>> a.upper()  
'HI'
```



02-2 문자열 자료형

대문자를 소문자로 바꾸기(lower)

```
>>> a = "HI"  
>>> a.lower()  
'hi'
```

양쪽 공백 지우기(strip)

```
>>> a = " hi "  
>>> a.strip()  
'hi'
```



02-2 문자열 자료형

문자열 바꾸기(replace)

```
>>> a = "Life is too short"
>>> a.replace("Life", "Your leg")
'Your leg is too short'
```

문자열 나누기(split)

```
>>> a = "Life is too short"
>>> a.split()
['Life', 'is', 'too', 'short']
>>> a = "a:b:c:d"
>>> a.split(':')
['a', 'b', 'c', 'd']
```



02-3 리스트 자료형

1, 3, 5, 7, 9라는 숫자 모음

```
>>> odd = [1, 3, 5, 7, 9]
```

리스트명 = [요소1, 요소2, 요소3, ...]

```
>>> a = [ ]  
>>> b = [1, 2, 3]  
>>> c = ['Life', 'is', 'too', 'short']  
>>> d = [1, 2, 'Life', 'is']  
>>> e = [1, 2, ['Life', 'is']]
```



02-3 리스트 자료형

리스트의 인덱싱

```
>>> a = [1, 2, 3]
>>> a[0]
1
>>> a[0] + a[2]
4
>>> a[-1]
3
```



02-3 리스트 자료형

리스트의 슬라이싱

```
>>> a = [1, 2, 3, 4, 5]
>>> a[0:2]
[1, 2]
>>> b = a[:2]
>>> c = a[2:]
>>> b
[1, 2]
>>> c
[3, 4, 5]
```



02-3 리스트 자료형

[] 사용해 리스트 요소 삭제하기

```
>>> a = [1, 'a', 'b', 'c', 4]
>>> a[1:3] = []
>>> a
[1, 'c', 4]
```

del 함수 사용해 리스트 요소 삭제하기

```
>>> a
[1, 'c', 4]
>>> del a[1]
>>> a
[1, 4]
```




02-3 리스트 자료형

리스트에 요소 추가(append)

```
>>> a = [1, 2, 3]
>>> a.append(4)
>>> a
[1, 2, 3, 4]
```

리스트 정렬(sort)

```
>>> a = [1, 4, 3, 2]
>>> a.sort()
>>> a
[1, 2, 3, 4]
```



02-3 리스트 자료형

리스트 뒤집기(reverse)

```
>>> a = ['a', 'c', 'b']
>>> a.reverse()
>>> a
['b', 'c', 'a']
```

위치 반환(index)

```
>>> a = [1,2,3]
>>> a.index(3)
2
>>> a.index(1)
0
```



02-3 리스트 자료형

리스트에 요소 삽입(insert)

```
>>> a = [1, 2, 3]
>>> a.insert(0, 4)
[4, 1, 2, 3]
```

리스트 요소 제거(remove)

```
>>> a = [1, 2, 3, 1, 2, 3]
>>> a.remove(3)
[1, 2, 1, 2, 3]
```



02-3 리스트 자료형

리스트 요소 끄집어내기(pop)

```
>>> a = [1,2,3]
>>> a.pop()
3
>>> a
[1, 2]
```

리스트에 포함된 요소 x의 개수 세기(count)

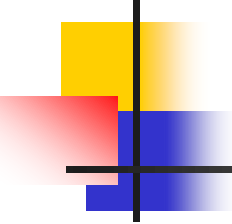
```
>>> a = [1,2,3,1]
>>> a.count(1)
2
```



02-3 리스트 자료형

리스트 확장(extend)

```
>>> a = [1,2,3]
>>> a.extend([4,5])
>>> a
[1, 2, 3, 4, 5]
>>> b = [6, 7]
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5, 6, 7]
```



02-4 튜플 자료형

튜플 요소값 삭제 시 오류

```
>>> t1 = (1, 2, 'a', 'b')
>>> del t1[0]
Traceback (innermost last):
  File "", line 1, in ?del t1[0]
TypeError: object doesn't support item deletion
```



02-4 튜플 자료형

튜플 요소값 변경 시 오류

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[0] = 'c'
Traceback (innermost last):
  File "", line 1, in ?t1[0] = 'c'
TypeError: object doesn't support item assignment
```



02-5 딕셔너리 자료형

- 연관 배열(**Associative array**) 또는 해시(**Hash**)
- 단어 그대로 해석하면 사전이라는 뜻

```
>>> dic = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
```




02-5 딕셔너리 자료형

딕셔너리 쌍 추가하기

```
>>> a = {1: 'a'}  
>>> a[2] = 'b'  
>>> a  
{2: 'b', 1: 'a'}
```

딕셔너리 요소 삭제하기

```
>>> del a[1]  
>>> a  
{'name': 'pey', 3: [1, 2, 3], 2: 'b'}
```



02-5 딕셔너리 자료형

딕셔너리에서 **Key** 사용해 **Value** 얻기

```
>>> grade = {'pey': 10, 'julliet': 99}
>>> grade['pey']
10
>>> grade['julliet']
99
```

딕셔너리 만들 때 주의할 사항

```
>>> a = {1:'a', 1:'b'}
>>> a
{1: 'b'}
```



02-5 딕셔너리 자료형

Key 리스트 만들기(keys)

```
>>> a = {'name': 'pey', 'phone': '0119993323', 'birth':  
'1118'}  
>>> a.keys()  
dict_keys(['name', 'phone', 'birth'])
```

Value 리스트 만들기(values)

```
>>> a.values()  
dict_values(['pey', '0119993323', '1118'])
```



02-5 딕셔너리 자료형

Key, Value 쌍 얻기(items)

```
>>> a = {'name': 'pey', 'phone': '0119993323', 'birth':  
'1118'}  
>>> a.items()  
dict_items([('name', 'pey'), ('phone', '0119993323'),  
('birth', '1118')])
```

Key: Value 쌍 모두 지우기(clear)

```
>>> a.clear()  
>>> a  
{}
```



02-5 딕셔너리 자료형

Key로 Value얻기(get)

```
>>> a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}  
>>> a.get('name')  
'pey'  
>>> a.get('phone')  
'0119993323'
```



02-5 딕셔너리 자료형

해당 **Key**가 딕셔너리 안에 있는지 조사하기(**in**)

```
>>> a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}  
>>> 'name' in a  
True  
>>> 'email' in a  
False
```



02-6 집합 자료형

- 집합에 관련된 것들을 쉽게 처리하기 위해 만들어진 자료형
- 중복을 허용하지 않는다.
- 순서가 없다(Unordered).



02-6 집합 자료형

집합 자료형

```
>>> s1 = set([1,2,3])  
>>> s1  
{1, 2, 3}
```

순서가 없고 중복이 허용되지 않는다

```
>>> s2 = set("Hello")  
>>> s2  
{'e', 'l', 'o', 'H'}
```




02-6 집합 자료형

값 1개 추가하기(add)

```
>>> s1 = set([1, 2, 3])
>>> s1.add(4)
>>> s1
{1, 2, 3, 4}
```

값 여러 개 추가하기(update)

```
>>> s1 = set([1, 2, 3])
>>> s1.update([4, 5, 6])
>>> s1
{1, 2, 3, 4, 5, 6}
```



02-6 집합 자료형

특정 값 제거하기(remove)

```
>>> s1 = set([1, 2, 3])  
>>> s1.remove(2)  
>>> s1  
{1, 3}
```



02-7 자료형의 참과 거짓

값	참 or 거짓
"python"	참
""	거짓
[1, 2, 3]	참
[]	거짓
()	거짓
{}	거짓
1	참
0	거짓
None	거짓



02-7 자료형의 참과 거짓

자료형의 참과 거짓은 어떻게 사용되나?

```
>>> a = [1, 2, 3, 4]
>>> while a:
...     a.pop()
...
4
3
2
1
```



프로그램의 구조를 쌓는다! 제어문

자료형을 바탕으로 제어문을
이용하여 프로그램의 구조를
만들어 보자



03-1 If 문

돈이 있으면 택시를 타고, 돈이 없으면 걸어 간다

```
>>> money = 1
>>> if money:
...     print("택시를 타고 가라")
... else:
...     print("걸어 가라")
...
택시를 타고 가라
```



03-1 If 문

if문의 기본구조

```
if 조건문:  
    수행할 문장1  
    수행할 문장2  
    ...  
else:  
    수행할 문장A  
    수행할 문장B  
    ...
```



03-1 If 문

들여쓰기

if 조건문:

수행할 문장1

수행할 문장2

수행할 문장3



03-1 If 문

조건문

```
>>> money = 1  
>>> if money:
```



03-1 If 문

비교 연산자

비교연산자	설명
$x < y$	x가 y보다 작다
$x > y$	x가 y보다 크다
$x == y$	x와 y가 같다
$x != y$	x와 y가 같지 않다
$x \geq y$	x가 y보다 크거나 같다
$x \leq y$	x가 y보다 작거나 같다



03-1 If 문

비교 연산자

```
>>> x = 3
>>> y = 2
>>> x > y
True
>>> x < y
False
>>> x == y
False
>>> x != y
True
```



03-1 If 문

만약 3000원 이상의 돈을 가지고 있으면 택시를 타고 그렇지 않으면 걸어 가라

```
>>> money = 2000
>>> if money >= 3000:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
걸어가라
```



03-1 If 문

and, or, not

연산자	설명
x or y	x와 y 둘중에 하나만 참이면 참이다
x and y	x와 y 모두 참이어야 참이다
not x	x가 거짓이면 참이다



03-1 If 문

돈이 3000원 이상 있거나 카드가 있다면 택시를 타고 그렇지 않으면 걸어 가라

```
>>> money = 2000
>>> card = 1
>>> if money >= 3000 or card:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
```



03-1 If 문

x in s, x not in s

```
>>> 1 in [1, 2, 3]
```

```
True
```

```
>>> 1 not in [1, 2, 3]
```

```
False
```



03-1 If 문

만약 주머니에 돈이 있으면 택시를 타고, 없으면
걸어 가라

```
>>> pocket = ['paper', 'cellphone', 'money']
>>> if 'money' in pocket:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
>>>
```




03-1 If 문

조건문에서 아무 일도 하지 않게 설정하고 싶다면?

```
>>> pocket = ['paper', 'money', 'cellphone']
>>> if 'money' in pocket:
...     pass
... else:
...     print("카드를 꺼내라")
...
```



03-1 If 문

다중 조건 판단 elif

```
>>> pocket = ['paper', 'cellphone']
>>> card = 1
>>> if 'money' in pocket:
...     print("택시를 타고가라")
... elif card:
...     print("택시를 타고가라")
... else:
...     print("걸어가라")
...
택시를 타고가라
```



03-2 while 문

while문의 기본 구조

```
while <조건문>:  
    <수행할 문장1>  
    <수행할 문장2>  
    <수행할 문장3>  
    ...
```



03-2 while 문

열 번 찍어 안 넘어 가는 나무 없다

```
>>> treeHit = 1
>>> while treeHit <= 10:
...     print("나무를 %d번 찍었습니다." % treeHit)
...     if treeHit == 10:
...         print("나무 넘어갑니다.")
...     ... treeHit = treeHit +1
```

나무를 1번 찍었습니다.

나무를 2번 찍었습니다.

나무를 3번 찍었습니다.

...

나무 넘어갑니다.



03-2 while 문

continue

```
>>> a = 0
>>> while a < 10:
...     a = a+1
...     if a % 2 == 0: continue
...     print(a)
...
1
3
5
7
9
```



03-2 while 문

무한 루프

```
while True:  
    수행할 문장1  
    수행할 문장2  
    ...
```



03-3 for 문

for문의 기본 구조

```
for 변수 in 리스트(또는 튜플, 문자열):  
    수행할 문장1  
    수행할 문장2  
    ...
```



03-3 for 문

전형적인 for문

```
>>> test_list = ['one', 'two', 'three']
>>> for i in test_list:
...     print(i)
...
one
two
three
```




03-3 for 문

60점이 넘으면 합격이고 그렇지 않으면 불합격

```
marks = [90, 25, 67, 45, 80]

number = 0
for mark in marks:
    number = number + 1
    if mark >= 60:
        print("%d번 학생은 합격입니다." % number)
    else:
        print("%d번 학생은 불합격입니다." % number)
```



03-3 for 문

for문과 continue

```
marks = [90, 25, 67, 45, 80]

number = 0
for mark in marks:
    number = number + 1
    if mark < 60: continue
    print("%d번 학생 축하합니다. 합격입니다. " % number)
```



03-3 for 문

for와 함께 자주 사용하는 range함수

```
>>> sum = 0
>>> for i in range(1, 11):
...     sum = sum + i
...
>>> print(sum)
55
```



03-3 for 문

구구단

```
>>> for i in range(2,10):  
...     for j in range(1, 10):  
...         print(i*j, end=" ")  
...     print('')  
...  
2 4 6 8 10 12 14 16 18  
3 6 9 12 15 18 21 24 27  
4 8 12 16 20 24 28 32 36  
5 10 15 20 25 30 35 40 45  
6 12 18 24 30 36 42 48 54  
7 14 21 28 35 42 49 56 63  
8 16 24 32 40 48 56 64 72  
9 18 27 36 45 54 63 72 81
```



03-3 for 문

리스트 내포(List comprehension)

```
>>> result = [num * 3 for num in a]
>>> print(result)
[3, 6, 9, 12]
```

```
>>> result = [num * 3 for num in a if num % 2 == 0]
>>> print(result)
[6, 12]
```



프로그램의 입력과 출력은 어떻게 해야 할까?

함수, 입력과 출력, 파일 처리 방법



04-1 함수

파이썬 함수의 구조

```
def 함수명(입력 인수):  
    <수행할 문장1>  
    <수행할 문장2>  
    ...
```



04-1 함수

일반적인 함수

```
def sum(a, b):  
    result = a + b  
    return result
```

```
>>> a = sum(3, 4)  
>>> print(a)  
7
```




04-1 함수

입력값이 없는 함수

```
>>> def say():  
...     return 'Hi'  
...  
>>>
```

```
>>> a = say()  
>>> print(a)  
Hi
```



04-1 함수

결과값이 없는 함수

```
>>> def sum(a, b):  
...     print("%d, %d의 합은 %d입니다." % (a, b, a+b))  
...  
>>>
```

```
>>> sum(3, 4)  
3, 4의 합은 7입니다.
```



04-1 함수

입력값도 결과값도 없는 함수

```
>>> def say():  
...     print('Hi')  
...  
>>>
```

```
>>> say()  
Hi
```



04-1 함수

입력 인수에 초깃값 미리 설정하기

```
def say_myself(name, old, man=True):  
    print("나의 이름은 %s 입니다." % name)  
    print("나이는 %d살입니다." % old)  
    if man:  
        print("남자입니다.")  
    else:  
        print("여자입니다.")
```

```
say_myself("박응용", 27)  
say_myself("박응용", 27, True)
```



04-1 함수

함수 입력 인수에 초깃값을 설정할 때 주의할 사항

```
def say_myself(name, man=True, old):  
    print("나의 이름은 %s 입니다." % name)  
    print("나이는 %d살입니다." % old)  
    if man:  
        print("남자입니다.")  
    else:  
        print("여자입니다.")
```



04-2 사용자 입력과 출력

input의 사용

```
>>> a = input()  
Life is too short, you need python  
>>> a  
'Life is too short, you need python'  
>>>
```



04-2 사용자 입력과 출력

프롬프트를 띄워서 사용자 입력 받기

```
>>> number = input("숫자를 입력하세요: ")
숫자를 입력하세요: 3
>>> print(number)
3
>>>
```



04-3 파일 읽고 쓰기

파일 생성하기

```
f = open("새파일.txt", 'w')  
f.close()
```

파일열기모드	설명
r	읽기모드 - 파일을 읽기만 할 때 사용
w	쓰기모드 - 파일에 내용을 쓸 때 사용
a	추가모드 - 파일의 마지막에 새로운 내용을 추가시킬 때 사용



04-3 파일 읽고 쓰기

파일을 쓰기 모드로 열어 출력값 적기

```
f = open("C:/Python/새파일.txt", 'w')
for i in range(1, 11):
    data = "%d번째 줄입니다.\n" % i
    f.write(data)
f.close()
```



04-3 파일 읽고 쓰기

readline() 함수

```
f = open("C:/Python/새파일.txt", 'r')
line = f.readline()
print(line)
f.close()
```

```
f = open("C:/Python/새파일.txt", 'r')
while True:
    line = f.readline()
    if not line: break
    print(line)
f.close()
```



04-3 파일 읽고 쓰기

파일에 새로운 내용 추가하기

```
f = open("C:/Python/새파일.txt",'a')
for i in range(11, 20):
    data = "%d번째 줄입니다.\n" % i
    f.write(data)
f.close()
```



04-3 파일 읽고 쓰기

with문과 함께 사용하기

```
f = open("foo.txt", 'w')  
f.write("Life is too short, you need python")  
f.close()
```

```
with open("foo.txt", "w") as f:  
    f.write("Life is too short, you need python")
```