

- day 1

1. python 설치

1. python 사이트(<https://www.python.org/>)에 접속한다.
2. 페이지를 내리다 보면 downloads 와 latest : python 3.64가 보이고 클릭하여 들어간다.
3. files version에서 Windows x86-64 executable installer 항목을 선택하여 설치한다.
4. 설치가 완료되면 Installer가 뜨며 next를 누르며 python 설치를 진행한다.
(이때 path를 자동으로 설정해준다는 항목을 선택 후 설치를 진행한다.
하지 않는다면 직접 path를 설정해야한다.)

2. sublime text(version 3) 설치 과정

1. sublime text 사이트(<https://www.sublimetext.com/>)에 접속한다.
2. 오른쪽 위에서 download를 찾고 그 페이지에 들어간다.
3. OS X, windows, windows 64bit, linux 중 windows를 선택한다.
(손상시킬 수 있는 파일이라고 뜰 수 있는데 그냥 계속 다운로드를 한다.)
4. 계속 next를 누르다가 설치 될 위치를 d드라이브로 설정하고 다운로드를 완료한다.

3. hello world 출력

- 새 python 파일 생성
 1. sublime text를 연다(실행한다).
 2. 아무것도 하지 않은 상태에서 ctrl + s(저장)를 눌러 저장한다.
(이때 파일명은 자신이 원하는 명으로 하며 확장자는 py(python)로 파일명.py로 저장한다.)
 3. 새 파일이 생성 되면 자신이 원하는 코드를 입력하고 저장한다.

-code 짜기

: python에서의 출력은 print()를 사용하며 ;(세미콜론)를 사용하지 않는다. 문자열을 쓸 때는 c처럼 ""안에 쓰며 hello world 출력을 위한 문장은 print("hello world") 이다.

python code)

```
print("hello world")
```

result)

```
hello world
```

4. python 파일 실행시키기

: 자신이 python 파일을 만든 폴더에 들어가 shift + 오른쪽 클릭(마우스)을 하면 '여기에 powershell 창 열기' 항목이 나온다.(그냥 오른쪽 클릭하면 나오지 않는다.) shell 창을 열고 'python 파일명.py'를 치면 "hello word" 가 출력된다.

python 파일이 있는 폴더에 들어가기 -> shift + 오른쪽 클릭 -> 여기에 powershell 창 열기
-> python 파일명.py

5. 주석

: python에서의 주석은 '#'을 사용하여 쓰며 즉, #adfdfdf 라고 쓰면 주석처리가 된다. 하나하나 #을 써서 주석하기 번거로울 때는 ctrl + /를 사용하면 커서가 있는 줄에 주석처리가 된다. 여러 줄을 주석처리하고 싶을 때는 시작할 부분에 """ 또는 ''' 를 쓰고 마칠 부분에도 ''' 를 사용하면 주석처리가 된다.

1. #
2. ctrl + /
3. ''' or """

python code)

```
"""this is juseok  
ctrl + / is also juseok  
is also juseok(many code)"""
```

- day 2

1. 파이썬

: 스크립트 언어이며, 스크립터 언어란 **컴파일** 없이 실행 가능한 언어이다.

* **컴파일** : 어떤 언어의 코드를 다른 언어로 변환

: 원시 언어(고급언어)를 목적 언어(저급언어, 기계어)로 바꿔줌

2. 컴파일러 vs 인터프리터

- 컴파일러

1. 프로그램 단위로 번역
2. 번역 속도 느림
3. 실행 속도 빠름
4. 큰 메모리 필요(*메모리 부족 에러)

- 인터프리터

1. 명령 줄 단위 번역
2. 번역 속도 빠름
3. 실행 속도 느림
4. 적은 메모리 필요

즉, 프로그램이 커지면 클수록 컴파일러가 좋고 반대로 작으면 작을수록 인터프리터가 좋다.

3. 자료형과 연산자

- 자료형

: 데이터를 변수에 담는 순간 자료형이 정해지며 자료형마다 연산을 지원한다.

python code)

```
a=3
print(a)
print(type(a))
```

result)

```
3
<class 'int'>
```

- 연산자

1. 더하기 : +
2. 빼기 : -
3. 곱하기 : *
4. 나누기 : /
5. 제곱 : **
6. 나머지 : %
7. 몫 : //

* 정수와 실수의 연산 결과는 실수 형태이고 자료형 또한 실수이다.

python code)

```
a=3.14
b=6
c= a +b
print(type(c))
```

result)

```
<class 'float'>
```

* 음수를 나누었을 때 나머지가 있는 경우에는 몫에 -1이 추가되어 나온다.

python code)

```
a =-11
b =2
c= a //b
print(c)
```

result)

```
-6
```

: $-11//2 = -6$ (원래의 몫은 -5이지만 -1이 추가되었다.)

4. 문자열

- 문자열 표현

1. "(문자열)"
2. '(문자열)'

- 따옴표 출력하기

1. ' '를 같이 출력하고 싶으면 " (문자열) " 으로 " 안에 ' '를 넣으면 가능하며 그 반대도 가

능하다..

python code)

```
a="고양이"  
print(a)
```

result)

```
'고양이'
```

: “ 안에 넣은 ’도 같이 출력된 것을 볼 수 있다.

2. 이스케이프 문자를 사용하여 표현 할 수도 있다.

: \’ 또는 \“ 를 사용하면 ‘와 ’를 출력할 수 있다.

python code)

```
print('\')
```

```
print('\')
```

result)

```
'  
"
```

3. """ """ 사용하기

: """ """ 안에 넣은 글 즉, 엔터, ' , ", 모두 쓴 대로 적용된다. ('' '''도 마찬가지이다.)

python code)

```
a=""" 가나다라  
마바"사 " 아자차 """  
print(a)
```

result)

```
가나다라  
마바"사 " 아자차
```

- tab을 적용시키기

: ctrl+tab을 하면 된다.

- * 그냥 tab 키를 누를 경우 탭이 아닌 해당 문자로 시작하는 함수 리스트(?)가 나타난다.
- * 단, python IDLE에서 적용되는 이야기이다.
- * sublime text에서는 그냥 tab키를 눌러도 탭이 적용된다.

5. 문자열 연산

- 붙이기

: '문자열' + '문자열'을 해주면 두 문자열이 하나의 문자열로 되어 출력된다.

python code)

```
a="바나나"  
b="우유"  
c=a +b  
print(c)
```

result)

```
바나나우유
```

- 곱하기

'문자열' * 숫자를 해주고 출력을 해주면 곱해준 숫자만큼 문자열이 출력된다.

python code)

```
a="바나나"
b=a *5
print(b)
```

result)

바나나바나바나바나바나바나바나

- 인덱싱

1. 문자열에서 특정 '문자'를 가리키는 것을 의미한다.
2. 인덱스 값은 0부터 시작한다.
(공백도 인덱스 값 들어감. 즉, a[1]이 공백이 될 수도 있다.)
3. 배열처럼 생각한다.

python code)

```
a="banana milk"
print(a[0])
print(a[3])
print(a[0]+a[3]+a[6]+a[8])
```

result)

b
a
ba i

- 슬라이싱

1. 자르기라고 할 수 있다.
2. 문자열에서 특정 '문자열'을 가리키는 것을 의미한다.
3. a[0:4] 라도 할 때 인덱스 4번 전(3번까지)까지 출력한다.

python code)

```
a="2018-03-13"
print(a[0:4])
```

result)

2018

6. 리스트와 딕셔너리

- 리스트

1. 자료형이 달라도 같은 리스트에 넣을 수 있다.
2. 각 항목을 +로 연결하여 출력하지 않는다.(,(콤마)로)
* 각각 자료형이 다르기 때문에 +로 붙일 수 없다.
3. 슬라이싱이 가능하다.

python code)

```
a=[0,123,'aaa',111]
print(a[0],'and',a[1])
print(a[2:])
```

result)

```
0 and 123
['aaa', 111]
```

- 딕셔너리

1. 대응관계를 나타낼 수 있는 자료형이다.
2. key와 value가 연결된 자료형이다.
3. 사전 같은 느낌이다.
4. 딕셔너리 자체의 연산은 되지 않는다.
5. 인덱싱만 되고 슬라이싱은 되지 않는다.
6. {}로 묶고 :로 key와 value를 나눈다.

python code)

```
dic={'1':'min','2':'gyu'}
print(dic['1'])
```

result)

```
min
```

7. 내장 함수 알아보기

- count() : 특정 문자(찾고자 하는)의 수를 반환
 - index() : 찾고자 하는 문자의 위치(인덱스) 반환
 - * 가장 첫 번째 위치 반환함
 - find() : index와 같음
 - * find는 찾고자하는 문자가 없으면 -1을, index는 error를 반환한다.
 - join() : 문자열의 각각의 문자 사이에 문자 또는 문자열을 삽입한다.
 - upper() : 소문자를 대문자로 바꿔준다.
 - lower() : 대문자를 소문자로 바꿔준다.
 - replace() : (바뀔 문자열, 바꿀 문자열)의 형식으로, 특정 값을 다른 문자열로 치환해준다.
 - split() : 괄호 안에 어떤 값도 넣어주지 않으면 공백(스페이스, 탭, 엔터 등)으로 문자열을 나눠주고 특정 값이 있는 경우 그 값으로 나눠준다.
 - lstrip() : 문자열 중 가장 왼쪽에 있는 한 칸 이상의 연속된 공백을 모두 지운다.
 - rstrip() : 문자열 중 가장 오른쪽에 있는 한 칸 이상의 연속된 공백을 모두 지운다.
 - strip() : 문자열 양쪽에 있는 한 칸 이상의 연속된 공백을 모두 지워준다.
 - type() : 입력값의 자료형이 무엇인지 알려준다.
 - int() : 문자열 형태의 숫자나 소수점이 있는 숫자 등을 정수 형태로 리턴 한다.
 - str() : 문자열 형태로 객체를 변환하여 리턴 한다.
 - ord() : 인수로 '문자'를 입력받으면 그에 해당하는 아스키코드 값을 리턴 한다.
 - chr() : ord 함수와 반대로 입력받은 정수의 해당 문자를 출력한다.
-

- day 3

1. 문자열 함수

- count

- : count 함수가 메소드로 지정된 변수의 문자열에서 count 함수 인자 개수를 반환한다.
- : 만약 a에 숫자를 담고 count에 숫자 인자를 넣으면 문자열 함수이기 때문에 실행 불가능하다.

python code)

```
a = 'programming'
res = a.count('m')
print(res)
```

result)

2

- find

- : find 함수가 메소드로 지정된 변수의 문자열에서 find 함수 인자의 인덱스를 반환한다.
- : 만약 찾는 문자가 없다면 -1을 출력한다.
- : 또한 반환되는 -1이 정수이기 때문에 연산이 가능하다.

ex) if(res==-1) 또는 res+1

python code)

```
a = 'programming'
res = a.find('m')
print(res)
```

result)

6

- index

- : find 함수처럼 메소드로 지정된 변수의 문자열에서 index 함수 인자의 인덱스를 반환한다.
- : 다만 find 함수와는 다르게 찾는 문자가 없다면 에러를 출력한다.

python code)

```
a = 'programming'
res = a.index('m')
print(res)
```

result)

6

- join

- : join 함수가 메소드로 지정된 변수의 문자를 join 함수의 인자(문자)사이에 삽입한다.
- : 문자열 함수이기 때문에 숫자를 삽입하는 것은 불가능하다.
 - * 대신 숫자를 문자로 형 변환 하면 가능하다.
- : 한 줄씩 띄우게 하고 싶을 땐 변수의 문자를 '\n'로 지정해주면 된다.

python code)

```
a = '_m-_-m_'
res = a.join('ABC')
print(res)
```

result)

```
A_m-_-m_B_m-_-m_C
```

* 타임(쉬어가기)

: 그냥 리스트를 출력해주면 []도 출력되기 때문에 이를 없애기 위해 콤수를 쓸 수 있다.

python code)

```
a = [1,'sfsdfsd',123,'ssdfs']
print(a)
print(str(a[1:4])[1:-1])
print(str(a)[1:-1])
```

result)

```
'sfsdfsd', 123, 'ssdfs'
1, 'sfsdfsd', 123, 'ssdfs'
[1, 'sfsdfsd', 123, 'ssdfs']
```

- upper

: 대문자로 변환한 값을 반환한다.

python code)

```
a = 'programming'
res = a.upper()
print(res) => PROGRAMMING
```

result)

```
PROGRAMMING
```

- lower

: 소문자로 변환한 값을 반환한다.

python code)

```
a = 'PROGRAMMING'
res = a.lower()
print(res) => programming
```

result)

```
programming
```

* '특수'문자에 대해서는 upper, lower 둘 다 적용 되지 않는다.

- replace

: 문자열을 치환한 결과를 반환한다.

: 공백을 다른 문자로 치환하기 위해서는 replace(' ','1')로 쓰면 된다.

: 공백으로 치환하기 위해서는 replace('g','')로 쓰면 된다.

: 띄어쓰기를 다른 문자로 치환하기 위해서는 replace(' ','1')로 쓰면 된다.

: 없는 문자열은 치환이 불가능하기 때문에 원본 그대로 출력한다.

python code)

```
a = 'programmer'
res = a.replace('.', '1')
print(res)
```

result)

```
1p1r1o1g1r1a1m1m1e1r1
```

- split

: 문자열을 나눈 결과를 반환하며 결과 값의 자료형은 리스트이다.
: 결과가 리스트이므로 나눈 결과를 결합하려면 인덱스를 활용한다.

python code)

```
a = 'alp1p1l1e'
res = a.split('1')
print(res[0]+res[1]+res[2]+res[3]+res[4])
```

result)

```
1p1r1o1g1r1a1m1m1e1r1
```

- lstrip

: 문자열 중 가장 왼쪽에 있는 한 칸 이상의 연속된 공백을 모두 지운다.

python code)

```
a = '    programming    '
res = a.lstrip()
print(res)
print(res + '1') # 왼쪽 공백만 제거되었는지 확인
```

result)

```
programming
programming1
```

-rstrip

: 문자열 중 가장 오른쪽에 있는 한 칸 이상의 연속된 공백을 모두 지운다.

python code)

```
a = '    programming    '
res = a.rstrip()
print(res)
print(res + '1') # 오른쪽 공백만 제거되었는지 확인
```

result)

```
programming
programming1
```

- strip

: 문자열 양쪽에 있는 한 칸 이상의 연속된 공백을 모두 지워준다.

python code)

```
a = '      programming      '  
res = a.strip()  
print(res)  
print('1'+res+'1')
```

result)

```
programming  
1programming1
```

- type

- : 입력값의 자료형이 무엇인지 알려준다.
- : type 함수는 반환 값이 있기 때문에 결과 값을 res에 넣고 출력할 수 있다.
- : 단, 반환 값이 없는 함수 결과 값을 인자에 넣고 출력하면 none 이 뜬다.

python code)

```
a = 'programming'  
res = type(a)  
print(res)
```

result)

```
<class 'str'>
```

- str

- : 숫자를 문자열로 바꿔준다.
- : 문자열인지 판단하기 위해서 *10을 해주면 문자열이 10번 반복 되어 출력된다.
- : 문자를 문자열로 바꾸는 것도 가능하다.

python code)

```
a = 123  
res = str(a)  
print(res * 10)
```

result)

```
123123123123123123123123123123123
```

- int

- : 문자로 표현된 숫자를 정수형으로 바꿔준다.
- : 순수 문자를 정수로 바꾸는 것은 불가능하다.
- : 바꾼 숫자로 계산 가능하다.

* ex) res+1

python code)

```
a = '123'  
res = int(a) # 실수로 바꾸려면 => float(a)  
print(res)
```

result)

```
123
```

- ord

: 문자를 아스키코드(정수)로 바꿔준다.

python code)

```
a = 'A'
res = ord(a)
print(res)
```

result)

65

- chr

: 정수에 해당하는 아스키코드 문자를 반환한다.

: a = 34로 하고 chr(a*2)로 해도 가능하다.

: 단, a = 68로 하고 chr(a/2)로 하면 불가능하다.

* 이유는 a/2의 자료형이 float 이기 때문이다.

python code)

```
a = 65
res = chr(a//2) # 몫만 나오게 해도 되고
int() 함수를 써서 정수로 바꿔줘도 된다..
print(res)
```

result)

A

2. 리스트 함수

- append

: append의 인자를 문자열 뒤에 추가한다.

: 단, append는 반환 값이 있는 함수가 아니다.

* 즉, res = a.append(4)를 하면 res에는 none 들어간다.

python code)

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

result)

None
[1, 2, 3, 4]

- sort

: 리스트를 오름차순으로 정렬해준다.

: reverse = True를 써주면 내림차순으로 정렬해준다.

* True라고 정확히 써줘야 한다.

: sort 함수는 반환 값이 없지만 sorted 함수(외부정렬)는 반환 값이 있다.

: reverse 함수는 차순을 바꿔주는 함수이다. 즉, 내림차순으로 바꿔주며 반환 값이 없다.

: sorted는 딕셔너리 정렬도 가능하다.

python code)

```
res1 = ['e','a','h']
res2 = [1,6,2]
res1.sort()
res2.sort(reverse=True) # True라고 정확히
입력해야 함
print(res1)
print(res2)
res3 = sorted(res1) #외부정렬(반환값 있음)
res4 = sorted(res1,reverse = True)
res1.reverse() # 차순 바꾸기 # 당연히 반환
값이 없는 함수
print(res1)
print(res2)
print(res3)
print(res4)
# sorted는 딕셔너리 정렬이 가능하다
a = {'2':'B','1':'A','3':'U'}
a1 = sorted(a)
print(a1)
```

result)

```
['a', 'e', 'h']
[6, 2, 1]
['h', 'e', 'a']
[6, 2, 1]
['a', 'e', 'h']
['h', 'e', 'a']
['1', '2', '3']
```

- insert

: 특정 인덱스의 값이 되도록 요소를 추가한다. (특정 인덱스 자리에 요소를 추가한다.)

: 계산값을 넣어도 된다.

: (인덱스, 값)형태로 사용한다.

python code)

```
res = [100,123,523]
res.insert(1,2)
print(res)
```

result)

```
[100, 2, 123, 523]
```

- remove

: 함수의 인자값을 찾아서 삭제한다.

: 값이 여러 개라면 가장 첫 번째 요소를 삭제한다.

python code)

```
res = [10,20,30,40,10]
res.remove(10)
print(res)
```

result)

```
[20, 30, 40, 10]
```

- pop

: 마지막 요소를 삭제하는 함수이다.

: 반환 값이 있으므로 변수에 pop한 값을 저장할 수 있음

python code)

```
res = [10,20,30,40]
a = res.pop() # 반환값이 있는 함수
print(a)
```

result)

```
40
```

- count

: 함수의 인자 값을 찾아서 개수를 센다.

: 내부의 계산 값을 넣어도 정수이므로 가능하다.

* ex) res = a.count(5+5) => 가능

python code)

```
a = [10,10,101,102,10,'ab']
res = a.count(10)
print(res)
```

result)

```
3
```

3. 딕셔너리 함수

- keys

: 딕셔너리의 key들을 반환한다.

python code)

```
a = {'a':123,'b':456}
res = a.keys()
print(res)
```

result)

```
dict_keys(['a', 'b'])
```

- values

: 딕셔너리의 값들을 반환한다.

python code)

```
a = {'a':123,'b':456}
res = a.values()
print(res)
```

result)

```
dict_values([123, 456])
```

- items

: 딕셔너리의 키와 값들을 반환한다. (즉, 딕셔너리 안의 내용을 반환한다.)

python code)

```
a = {'a':123,'b':456}
res = a.items()
print(res)
```

result)

```
dict_items([('a', 123), ('b', 456)])
```

* dict~~ 라는 말을 없애고 싶다면 ? => list를 활용한다.

- get

: print(a['t'])와의 차이로 get 함수를 쓸 때 키가 없는 값을 찾으려면 none을 반환하지만

: 그냥 딕셔너리에서 키 값으로 찾았을 때 없으면 오류가 난다.

: get 함수는 키 값이 있다면 그 값을 반환하고 없으면 기본 값을 지정하여 반환할 수 있다.

python code)

```
a = {'q':123,'w':456}
res = a.get('t',789) #ㅇ 없는 값이면 789
반환 / 있으면 해당 값 반환
print(res)
```

result)

```
789
```

- in

: 키 값이 있는지 없는지 검사한다.

: 값이 있으면 True, 없으면 False

: 이 결과를 가지고 if(~~==True) 로 사용가능하다.

python code)

```
a = {'q':123,'w':456}
print('q' in a)
```

result)

```
True
```

- day 4

1. 코드 축약

- 키보드를 사용하여 두 정수를 입력하면 덧셈, 뺄셈, 곱셈, 나눗셈, 몫, 나머지가 출력되는 계산기 프로그램을 작성하시오.

ex) 입력 : 30 8

출력 : 38, 22, 3.75, 3, 6

python code 1번째)

```
a=input()
b=input()
a=int(a)
b=int(b)
print(a +b)
print(a -b)
print(a *b)
print("%.2f"%(a/b)) #formatting
print(int(a /b))
print(a%b)
```

: 입력을 한 줄에 입력하기 위해 split 함수를 사용하고 각 변수에 정수로 매핑해주기 위해 map 함수를 사용한다.

python code 2번째)

```
a,b=map(int,input().split()) # 한줄 입력이 가능해진다.
# # 입력을 받아서 각 변수에 정수로 mapping
```

: 출력하는 print문을 한 줄로 써줌으로써 코드를 축약한다.

: 이 때, sep = '\n'을 써서 각 결과 값을 한 줄씩 출력한다.

python code 최종)

```
a,b=map(int,input().split())
# 입력을 받아서 각 변수에 정수로 mapping
print(a +b,a -b,a *b,"%.2f"%(a /b),a //b,a%b,sep ='\n')
```

result)

: 입력은 첫째줄

```
10 5
15
5
50
2.00
2
0
50 5 10 5 10
```

2. 조건 구조

- age에 나이를 입력 받고 20이상이면 party tonight를 출력하고, 20미만이면 study tonight를 출력한다.

: 파이썬의 코드 구분은 “들여쓰기”로 수행된다.

: 즉, 함수, 조건, 반복 구조 등 내포가 필요한 구문은 콜론(:)으로 구분한다.

python code)

```
age = input("나이를 입력하세요 : ")
if int(age) >=20: #괄호를 써도 되고 안 써도 됨
    print("Party tonight")
else: print("Study tonight")
```

: 코드를 축약하기 위해 삼항 연산자를 사용한다.

* format : 참일 때의 명령문 if 조건 else 거짓일 때의 명령문

python code 축약 형태)

```
age = input("나이를 입력하세요 : ")
# print("Party tonight") if int(age)>=20 else print("Study tonight")
print("Party tonight"if int(age)>=20 else "Study tonight")
```

result)

```
나이를 입력하세요 : 18
Study tonight
50 5 10 5 10
```

3. 반복구조

- a라는 변수에 아무 입력이나 계속 받고(숫자 또는 문자), a가 1이 아닐 때는 “This is not one”을 출력하고, a가 1일 때는 “The end”를 출력한다.

: 무한 루프를 돌기 위해선 while True를 사용한다.

* True라고 정확히 써줘야 한다.

python code)

```
while True:
    a=input("숫자나 문자를 입력하시오 : ")
    if a == '1' and int(a) == 1: #문자(열)을 숫자로 바꾸고 숫자와 비교하는 것이 사실 불가능
        print("The end")
        break
    else :
        print("This is not one")
```

result)

```
숫자나 문자를 입력하시오 : 12
This is not one
```

- for문으로 구구단 출력하기

python code)

```
for i in range(2,10):
    for j in range(1,10):
        print(i *j, end = " ")
    print("")
```

result)

```
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
```

- 리스트에 for문 넣기

python code)

```
list_a =[3 ,6 ,9 ,12 ]
res2=[n -1 for n in list_a]
print(res2)
```

result)

```
[2, 5, 8, 11]
```

- 이를 활용해 구구단 출력하기

python code)

```
gugudan=[i *j for i in range(2,10) for j in range(1,10)]
print(gugudan)
```

result)

```
[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]
```

4. 내가 문제 만들기

- * 문제 작성 - 문제해결 논리에 의한 정상 코드 - 파이썬 기능을 활용하여 축약 코드 제시

- 문제 만들기) 두 수를 입력받아 공약수를 찾고 리스트에 넣어 출력

1. 두 수를 입력한다.
2. 두 수 중 더 작은 값을 min으로 설정한다.
3. 2부터 min까지의 수에서 공약수를 찾고 공약수이면 리스트에 넣고 아니면 넣지 않는다
4. 리스트를 출력한다.

python code 1번째)

```
a,b=input().split()
a=int(a)
b=int(b)
```

python code 2번째)

```
a,b=map(int,input().split())
min=0
if a <b : min = a
else : min = b
```

python code 3번째)

```
a,b=map(int,input().split())
min= a if a <b else b
c=[]
for i in range(2,min +1):
    if a%i ==0 and b%i ==0 : c.append(i)
    else None
print(c)
```

python code 4번째)

```
a,b=map(int,input().split())
min= a if a <b else b
c=[]
for i in range(2,min +1):
    c.append(i) if a%i ==0 and b%i ==0 else None
print(c)
```

result)

```
10 40
[2, 5, 10]
```

- day 5

1. 문제풀기

1. 추상화 : 문제 분해, 핵심 요소 추출

- 현재 상태 정의
- 목표 상태 정의
- 문제 분해
- 핵심요소(조건) 추출

2. 알고리즘 : 절차적 사고 표현

- 0개 이상의 입력
- 1개 이상의 결과
- 명확성/유한성/실행 가능

3. 자동화 : 프로그래밍

- 코딩
- 코오딩
- cooooooooooding
- + 시뮬레이션

2. 하노이 탑

1. 추상화

- 현재 상태 : 탑이 모두 a기둥에 있다.
- 목표 상태 : 탑이 모두 c기둥에 있다.
- 문제 분해 : ????????????? 모르겠다.
- 핵심 요소 추출 : ????????????? 모르겠다.

: 이것만으로는 어떻게 문제를 풀어야 하는지 모르겠음
따라서, **문제 분해**를 통해 푼다.

문제 분해 1

- 현재 상태 : 탑이 모두 a기둥에 있다.
 - 목표 상태 : 제일 큰 원판이 a기둥에 있고, 나머지 원판들이 b기둥에 있다.
(제일 큰 원판을 남기고 나머지 원판을 b기둥에 옮긴다. -> 상태가 아니라 행동이므로 x)
 - 문제 분해 : 위의 과정이 문제 분해이다.
 - 핵심 요소 : 제일 큰 원판 -> n
나머지 원판 -> 1~ n-1
- * 큰 원판 이외의 나머지 원판들을 인접 기둥에 옮기고 큰 원판을 마지막 기둥에 옮긴 후 나머지 원판을 옮긴다. 이 과정을 반복한다.

문제 분해 2

- 현재 상태 : 제일 큰 원판이 a기둥에 있고, 나머지 원판들이 b기둥에 있다.
- 목표 상태 : 제일 큰 원판이 c기둥에 있고, 나머지 원판들이 b기둥에 있다.
- 문제 분해 : 위의 과정이 문제 분해이다.
- 핵심 요소 : 제일 큰 원판 옮기는 횟수 => +1

문제 분해 3

- 현재 상태 : 제일 큰 원판이 c기둥에 있고, 나머지 원판들이 b기둥에 있음
- 목표 상태 : 모든 원판이 c기둥에 있음
- 문제 분해 : 위의 과정이 문제 분해임
- 핵심 요소 : 나머지 기둥 옮기는 횟수 -> n-1
n-1이 1이면 종료

python code)

```
def hanoi(h_num):
    if h_num == 1: return 1 #유한성, 조건
    else : return hanoi(h_num - 1)+1 +hanoi(h_num - 1) #입력 0개 이상, 출력 1개 이상
h_num =int(input("하노이 원판의 개수를 입력하시오 : "))
print(hanoi(h_num))
```

result)

```
하노이 원판의 개수를 입력하시오 : 6
63
```

python code 간단히)

```
h_num =int(input("하노이 원판의 개수를 입력하시오 : "))
print(2 **h_num -1)
# 문제 분해를 통해 구조를 알아낸 후 이를 수학적으로 구현한다.
```

: 결과 값은 같다.

3. 직접 문제풀기

- n개의 계단을 오를 때 한 번에 1계단 또는 2계단으로 오를 수 있는 방법의 수 구하는 문제
- : n=3일 때 계단을 오르는 방법은 세 가지이다.

문제 분해 1

- 현재 상태 : 올라갈 계단이 1칸이다.
- 목표 상태 : 계단을 모두 올라간다.
- 핵심 요소 : 올라가는 경우의 수는 1가지이다.

문제 분해 2

- 현재 상태 : 올라갈 계단이 2칸이다.
- 목표 상태 : 계단을 모두 올라간다.
- 핵심 요소 : 올라가는 경우의 수는 2가지이다.

python code 간단히)

```
def stair(s_num):
    '''
        if s_num==1 : return 1
        elif s_num==2 : return 2
        줄일 수 있음
    '''
    if s_num <3:return s_num
    else : return stair(s_num -1)+stair(s_num -2)
s_num =int(input("계단의 개수를 입력하시오 : "))
print(stair(s_num))
```

: 이 문제는 피보나치 수열을 구하는 것과 같다.

: 1과 2일 때는 자기 자신을 반환하며 3부터 자기 자신의 -1한 값과 -2한 값을 더해서 반환한다.

result)

```
계단의 개수를 입력하시오 : 5
8
```

4. 직접 문제 풀기 2

- 코드표를 통한 암호해독 문제

ex) 코드표 예시

l o h c g p d a b k
0 1 2 3 4 5 6 7 8 9

암호문 예시

cdp -> 365

- 중복되지 않은 10개의 코드를 가진 암호코드표가 주어지고, 각각의 암호 코드에는 0부터 9까지의 숫자가 매칭 된다.
- 암호문이 주어졌을 때 이 암호코드표를 기반으로 암호문을 복호화 하는 알고리즘(파이썬 코드)을 작성하시오.
- * 암호문은 공백을 허용 한다.

문제분해

현재 상태 : 암호코드표가 입력된 상황이다.

목표 상태 : 암호코드에서 암호문의 문자를 찾아 인덱스를 출력한다.

핵심요소1 : 공백이 입력되면 공백을 반환한다.

핵심요소2 : 문자가 입력되면 암호문을 순차 탐색하여 인덱스를 반환한다.

python code 최종)

```
a=input("코드표를 입력하시오 : ")
b=input("암호문을 입력하시오 : ")

for i in b:
    print(a.find(i) if i != ' ' else i, end = "")
```

before 과정

python code 1번째)

```
while(i < len(b)):
    if b[i] != ' ':
        print(a.find(b[i]), end = "")
    else:
        print(b[i], end = "")
    i=i +1
```

python code 2번째)

```
while(i < len(b)):
    print(a.find(b[i]) if b[i] != ' ' else b[i],
    end = "")
    i=i +1
```

result)

```
코드표를 입력하시오 : lohcgpdabk
암호문을 입력하시오 : cdp
365
```

- day 6

1. about 함수

- : 입력 값을 받아서, 특정 연산(작업)을 수행한 후에 결과를 출력하는 것이다.
- : 입력 값을 받아서, 특정 연산(작업)을 수행한 결과이다.

다만...

- : 프로그래밍에서는 조금 다른 의미를 가진다.
- : 결과보다는 어떤 기능을 하느냐가 더 주안점이다.

2. 사용자 함수

- 주의사항

- : 함수 호출 전까지는 함수 안의 문장들은 수행되지 않는다.
- : 함수는 호출 되기 전에 먼저 만들어져야 한다.

* 함수의 정의 => def 함수이름(함수의 인수..) :

- 동작과정

- : 반환을 하는지 안 하는지 생각하기

python code 반환 안함)

```
# function that no return value
def func1(a):
    print(a * 3)
print('start')
func1('agagd')
```

result)

```
start
agagdagagd
```

python code 반환 함)

```
# function that return value
def sum(a,b):
    return a + b
a,b=input().split()
res=sum(int(a),int(b))
print(res)
```

result)

```
10 50
60
```

3. 만들어보자

1. 문자열을 입력받는다.
2. 문자열 내에 'skip'이라는 단어가 있으면 'rejected'를 출력한다.
3. 'skip'이라는 단어가 없으면 그대로 출력한다.
4. 매 출력마다 '-'*10 출력해서 구분한다.
5. 사용자가 'quit'을 입력하여 끝낼 때까지 프로그램을 무한 반복한다.

python code)

```
def print_without_skip(str):
    if 'skip' in str:
        print('rejected')
        return
    else:
        print(str)
    print('-'*10)

# quit을 입력할 때까지 반복하여 사용자 입력을 받음
user_input = ''
while user_input != 'quit':
    user_input = input('input: ')
    print_without_skip(user_input)
```

result)

```
input: skip
skip
-----
input: skip
rejected
input: quit
quit
-----
```

4. 가상의 이메일 전송 함수

- : 보내는 메일주소, 받는 메일주소, 제목, 내용을 매개변수로 갖는 send_mail 함수를 만든다.
 - * 각 매개변수를 출력한다.
- : users 라는 리스트 안에 name 과 email 주소를 키 값으로 갖는 딕셔너리를 만들고 삽입한다.
- : for 문으로 users 리스트에서 각 딕셔너리에서 이름과 이메일 주소를 가져와서 메일을 보낸다.
(send_mail 함수로)

python code)

```
# 가상의 이메일 전송 함수
def send_mail(from_mail, to_mail, subject, contents ):
    print("From: \t"+from_mail)
    print("To: \t"+to_mail)
    print("Subject: \t"+subject)
    print("*"*20)
    print(contents)
    print("*"*20)
    print("*"*20)

my_email = "jiw021538@gmail.com"
users=[]
users.append({'name':'Boo','email':'jiw021538@gmail.com'})
users.append({'name':'John','email':'jiw021538@gmail.com'})
contents="""This is DSM
my leader is Eo young Bo young
i love you
"""

for user in users:
    title='Dear. '+user['name']
    send_mail(my_email,user['email'],title,contents)
```

- day 7

1. 클래스

- : 일종의 템플릿이다.
- : c언어의 구조체와 유사하다

- 차이점은?

- : 구조체는 변수만 담을 수 있지만 클래스는 함수까지 담을 수 있다.

즉, 클래스 = 변수 (합집합) 함수

- : 형식

```
class 클래스명():
    변수명 = 변수값
    def 함수명(self, 인자"", "")
```

2. 만들어보자

- : c언어의 구조체처럼 .을 써서 클래스 내부를 활용한다.

python code)

```
class SimpleTest():
    a = 0
    postfix = '\t DSM'
    def print_with(self, string):
        print(string)
        print(self.a)
        print(str(self.a)+string +self.postfix)

#클래스 변수를 생성
s1 = SimpleTest()
s2 = SimpleTest()
print(s1.a)
print(s2.a)
s1.a = 10
s2.a = 20
print(s1.a)
print(s2.a)
s1.print_with('KSH')
s2.print_with('UYBY&HG HG')
```

result)

```
10
20
KSH
10
10KSH    DSM
UYBY&HG HG
20
20UYBY&HG HG    DSM
```

- 알게 된 점

: 클래스도 함수처럼 호출되기 전에는 수행이 되지 않는다.

- self란?

: 클래스의 변수에 접근하기 위해 파이썬이 제공하는 변수이다.

: 클래스 내에서 함수를 정의할 때 잊지 말고 꼭 쓰자!

- 생성자

: 클래스 변수가 생성될 때 자동으로 호출되는 함수이다.

: 클래스 내부에 정의된 변수 등을 초기화 할 때 사용한다.

: simple = SimpleTest()처럼 클래스 변수를 만들 때 __init__ 함수가 자동으로 실행된다.

python code)

```
class SimpleTest():
    my_data = 0
    def __init__(self):
        self.my_data = 100
        print('Call init!')
simple = SimpleTest() # Call init!가 출력됨
simple.__init__() #실행 됨
print(simple.my_data)
```

result)

```
Call init!
Call init!
100
```

3. 이메일 보내기

- 필요한 설정 정보

SMTP 서버 : 메일을 보내는 서버

POP3 서버 : 메일을 받는 서버

*** (서버 주소 / 포트번호)**

계정 정보 : ID/PW

보내기만 할 것이므로 SMTP만 사용한다.

라이브러리 : email, smtplib

: 기본 라이브러리로 제공한다.

: 따로 설치하지 않아도 된다.

- 라이브러리 내부

MIME : 전자 우편을 위한 인터넷 표준 포맷

MIMEText

MIMEMultipart

*** SMTP가 사용하는 양식에 맞춰서 내용을 작성해주는 클래스이다.**

python code)

```
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import smtplib

#상수 변수
SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = 465
SMTP_USER = 'jiw021538@gmail.com'
SMTP_PASSWORD = (비밀번호)
def send_mail(name, addr):
    msg = MIMEMultipart()
    # 메일 전송
    msg['From'] = SMTP_USER
    msg['To'] = addr
    msg['Subject'] = name + '님에게 메일이 도착하였습니다 ^o^'
    contents = name + '''님에게 메일이 도착하였습니다.
안녕하세요 ^^ 자동화로 니 컴퓨터 해킹할거야 아이조아'''
    #인코딩이 안됨
    #msg['text'] = contents
    text = MIMEText(_text =contents,_charset = 'utf-8')
    msg.attach(text)

    #SMTP로 접속할 서버 정보를 가진 클래스 변수를 생성한다.
    smtp = smtplib.SMTP_SSL(SMTP_SERVER,SMTP_PORT)
    #계정 정보로 로그인
    smtp.login(SMTP_USER,SMTP_PASSWORD)
    #메일 전송
    smtp.sendmail(SMTP_USER,addr,msg.as_string())
    print("\n 메일이 전송되었습니다 \n")
    smtp.close()
send_mail('한지우','aldohq02@gmail.com')
```

result)



jiw021538@gmail.com

한지우님에게 메일이 도착하였습니다 ^o^

한지우님에게 메일이 도착하였습니다. 안녕하세요 ^^

4. 엑셀 활용

- pip install openpyxl

: openpyxl 라이브러리를 설치한다.

- 직접 엑셀 접근 방법

1. 사람이 엑셀에서 데이터가 들어있는 파일 실행한다.
2. 데이터가 들어있는 시트로 이동한다.
3. 데이터가 있는 위치의 데이터를 활용한다.

- 프로그램으로 엑셀 접근 방법

1. 프로그램이 엑셀에서 데이터가 들어있는 파일명으로 클래스 변수를 생성한다.
2. 클래스 변수에서 시트이름을 활용하여 시트로 이동 한다.
3. 데이터가 있는 위치의 데이터를 활용한다.

python code 1번째)

```
from openpyxl import load_workbook
# load_workbook 함수를 이용하여 엑셀 클래스 변수 생성
wb = load_workbook('test.xlsx')
# 활성화 된 시트를 sheet 변수로 설정
sheet = wb.active
print(sheet['A1'].value)
print(sheet['A2'].value)
print(sheet['B1'].value)
print(sheet['C1'].value)
print(sheet['B2'].value)
```

result)

```
dfd
dfd
dfa
gdffa
ada
```

: 엑셀 파일을 저장하기 위해서는 wb.save('파일명')를 사용한다.

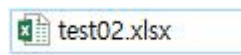
: 해당 파일이 있으면 덮어쓰기 되고 없으면 새로운 엑셀 파일을 생성한다.

python code 2번째)

```
from openpyxl import load_workbook
# load_workbook 함수를 이용하여 엑셀 클래스 변수 생성
wb = load_workbook('test.xlsx')
# 활성화 된 시트를 sheet 변수로 설정
sheet = wb.active
sheet['A1'] = 'Number'
sheet['B1'] = 'Name'
sheet['A2'] = 1
sheet['B2'] = 'AAA'
sheet['A3'] = 2
sheet['B3'] = 'BBB'
# test02라는 파일이 있으면 덮어쓰기 됨
# 없으면 새로운 엑셀파일 생성
wb.save('test02.xlsx')
```

result)

| | A | B | C |
|---|--------|------|------|
| 1 | Number | Name | gdfa |
| 2 | 1 | AAA | daf |
| 3 | 2 | BBB | df |



: 시트 이름을 title을 사용하여 바꿀 수 있다.

python code 3번째)

```
wb = load_workbook('test.xlsx')
# 활성화 된 시트를 sheet 변수로 설정
sheet = wb.active
sheet.title = 'My_class'
# 시트 이름을 title을 써서 바꿀 수 있다.
sheet.append(['Number', 'name'])
for i in range(10):
    sheet.append([i, chr(i+65)*3])
wb.save('test03.xlsx')
```

result)

| Number | name |
|--------|------|
| 0 | AAA |
| 1 | BBB |
| 2 | CCC |
| 3 | DDD |
| 4 | EEE |
| 5 | FFF |
| 6 | GGG |
| 7 | HHH |
| 8 | III |
| 9 | JJJ |

: 시트 안에 있는 내용을 출력할 때 행 번호를 일일이 확인할 수 없다.

: 따라서 셀이 입력되어 있는 구간을 알아서 인식하도록 하면 좋다. => `iter_rows()`

python code 4번째)

```
rows = sheet['1:11']
# 행 번호를 일일이 확인할 수 없으므로
# 셀이 입력되어 있는 구간을 알아서
인식하도록 하면 좋다.
for row in sheet.iter_rows():
    print(row[0].value,row[1].value)
```

result)

```
dfd dfa
dfd ada
adf adf
Number name
0 AAA
1 BBB
2 CCC
3 DDD
4 EEE
5 FFF
6 GGG
7 HHH
8 III
9 JJJ
```

- day 8

1. pyautogui 라이브러리 설치

1. easy_install.exe Pillow를 통해 Pillow를 먼저 설치해준다.
*그냥 `pip install pyautogui`를 하면 되지 않았다.,
2. easy_install.exe pyautogui를 통해 pyautogui를 성공적으로 설치해준다.

2. pyautogui 라이브러리

- size

: 화면의 크기를 반환하는 함수이다.

python code)

```
import pyautogui
scrWD, scrHE = pyautogui.size()
print(scrWD, scrHE)
```

result)

```
1366 768
```

- moveTo

: 원하는 위치(절대좌표)로 마우스를 이동하는 함수이다.

: 마우스의 위치를 확인하고 싶다면 ctrl 옵션을 설정하면 된다.

python code)

```
import pyautogui
scrWD, scrHE = pyautogui.size()
pyautogui.moveTo(scrWD,scrHE)
pyautogui.moveTo(scrWD/2,scrHE /2)
```

- moveRel

- : 원하는 위치(상대좌표)로 마우스를 이동하는 함수이다.
- : 이동하지 않으려면 none 값으로 인자를 설정한다.

python code)

```
pyautogui.moveRel(-150,-150)
for i in range(0,10):
    pyautogui.moveRel(150,None)
    pyautogui.moveRel(None,150)
    pyautogui.moveRel(-150,None)
    pyautogui.moveRel(None,-150)
```

- click

- : 클릭하는 함수이다.
- : 옵션을 통해 횟수와 마우스 버튼을 지정 가능하다.
- : x , y 는 마우스 위치를 지정한다.
- : clicks 는 마우스의 클릭 횟수를 지정한다.
- : interval 은 클릭 간격을 조정한다. (값은 second, 초 단위로 지정한다.)
- : button 은 마우스 버튼 위치를 선택한다. (left 또는 right)

python code)

```
scrWD, scrHE = pyautogui.size()
pyautogui.click(x= scrWD /3, y =scrHE /2, clicks =2, interval =3, button ='right')
pyautogui.click(x= scrWD /3, y =scrHE /2, button ='left')
```

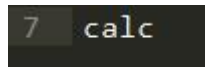
- typewrite

- : 키를 입력하는 함수이다.
- : 지금은 영어만 입력할 수 있다.
- : 커서가 있는 자리에 입력한다.

python code)

```
pyautogui.typewrite('calc')calc
```

result)



- **press**

: 특수키를 입력할 때 사용하는 함수이다.

python code)

```
pyautogui.press('enter')
```

result

```
6 pyautogui.press('enter')
7
8 |
```

- **locateCenterOnScreen**

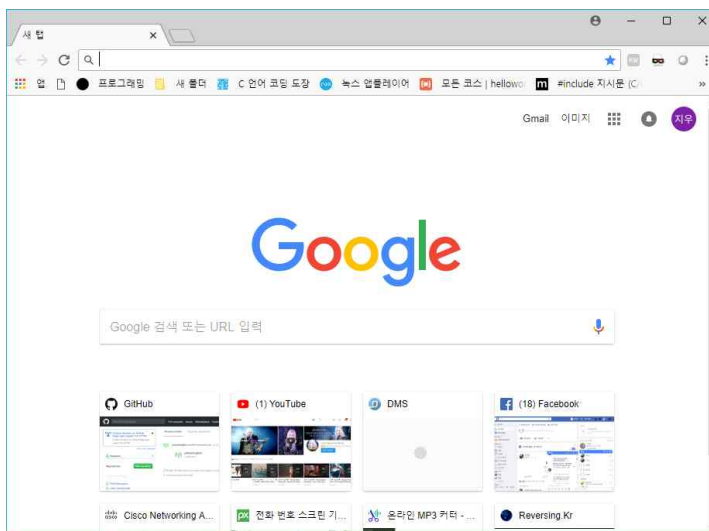
: 그림과 일치하는 위치의 좌표를 반환하는 함수이다.

: 그림파일을 png로 설정해야한다.

python code)

```
lx, ly = pyautogui.locateCenterOnScreen('test2.png')
pyautogui.moveTo(lx, ly)
pyautogui.click(clicks=2, button='left')
```

result



3. 계산기 활용

: pyautogui를 이용해서 프로그램이 계산기로 계산을 하도록 한다.

- **필요 라이브러리**

subprocess : 계산기를 작동시키기 위해 필요

opencvloc : 해당 이미지에 맞는 마우스 커서 값을 가져오기 위해 필요

time : sleep() 함수를 위해 필요

- 라이브러리 내부

run (subprocess)
locateCenterImage (opencvloc)

- 라이브러리 설치

1. pip install numpy를 통해 opencv를 설치하기 전에 numpy를 설치해준다.
2. pip install opencv_python을 통해 opencv를 설치해준다.

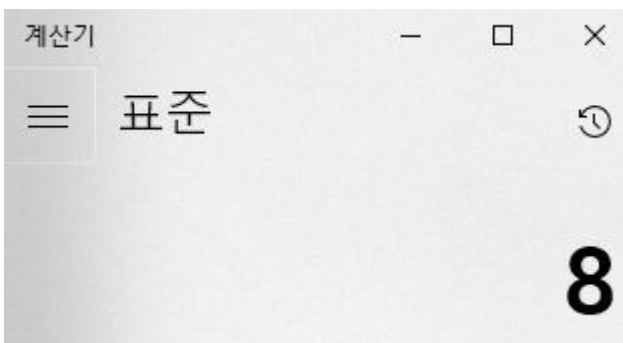
*또는 opencvloc.py 파일을 통해 라이브러리를 사용할 수 있다.

: import ~ as ~을 통해 라이브러리 명을 다른 이름으로 대체 할 수 있다.

python code)

```
from subprocess import run
from opencvloc import locateCenterImage
import pyautogui as py # 라이브러리 대체
import time
run(['calc'])
time.sleep(1)
x,y=locateCenterImage('5.png')
py.click(x,y)
x,y=locateCenterImage('plus.png')
py.click(x,y)
x,y=locateCenterImage('3.png')
py.click(x,y)
x,y=locateCenterImage('equal.png')
py.click(x,y)
```

result)



4. 1에서 9까지 더하기

: 1부터 9까지 더하는 프로그램을 작성한다.

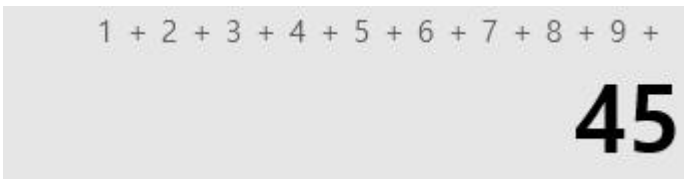
python code)

```
from subprocess import run
from opencvloc import locateCenterImage
import pyautogui as py # 라이브러리 대체
import time
run(['calc'])
time.sleep(1)
for i in range(1,10):

    i=str(i)
    x,y=locateCenterImage(i+'.png')
    py.click(x,y)
    if int(i)!=9:

x,y=locateCenterImage('plus.png')
    py.click(x,y)
x,y=locateCenterImage('equal.png')
py.click(x,y)
```

result)



1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 +
45

- day 9

1. 크롤링 원리

- 웹사이트도 결국 문서

- : 브라우저 별로 내부에서 쓰는 드라이버가 있다.
- : 예를 들어 chromedriver.exe 와 같은

- 웹 드라이버의 역할

- : 웹문서를 분석하고 이를 활용하여 화면을 구성한다.
- : 웹문서에 이벤트를 전달하고 결과 값을 받는다.
- : 웹 드라이버가 제공하는 방법으로 서로 주고받아야 한다.

- 웹 드라이버를 직접 다루는 것

: 한마디로 브라우저를 만드는 것이다.

: 역시 라이브러리를 활용하자

- selenium

: 일종의 서버 프로그램으로 라이브러리로 제공된다.

: 다양한 브라우저의 웹 드라이버 컨트롤을 가능하게 해준다.

* 라이브러리를 통해 웹드라이버 컨트롤 가능

2. selenium 설치

1. pip install selenium을 통해 selenium을 설치한다.

2. 안되면 python -m pip install --upgrade pip을 사용한다.

3. 웹 페이지 분석

: 일관된 화면을 위해 이번엔 크롬으로 활용한다.

- 필요 프로그램 다운로드

: <https://sites.google.com/a/chromium.org/chromedriver/downloads>

* 내가 만들 파이썬 파일이 저장되는 곳에 다운 받은 exe파일이 있어야 한다.

4. 코딩

- try ~ except ~ finally

: try 안의 코드를 수행하다가 에러가 발생하면 발생한 시점 이후의 코드는 수행하지 않고 except로 가서 코드를 수행하라는 예외처리 문이다.

: finally는 에러 여부와 상관없이 무조건 수행한다.

- 네이버 뉴스 페이지에서 최근 뉴스들의 제목을 출력하는 프로그램을 작성하시오.

python code)

```
# 크롬 브라우저를 띄우기 위해 selenium 으로 웹드라이버를 가져옴
from selenium import webdriver
# 크롬 드라이버로 크롬 브라우저를 실행
driver = webdriver.Chrome('chromedriver')
try:
    # 네이버 뉴스 페이지로 이동
    driver.get('http://news.naver.com')
    # 네이버 뉴스임을 알 수 있도록 현재 타이틀 출력
    print(driver.title)
    # 최근 뉴스 목록을 가진 div id 태그를 가져옴
    title_id = driver.find_element_by_id('right.ranking_contents')
    # 위 div_id 안에 li 태그로 구분되어있는 정보를 가져와 리스트로 저장
    news_list = title_id.find_elements_by_tag_name('li')
    # 가져온 태그들에 대해 반복문을 수행하면서 각각의 문자열을 출력
    for news in news_list:
        print(news.text)
except Exception as e:
    print(e)
finally:
    # 브라우저 종료
    driver.quit()
```

result)

```
네이버 뉴스
1 '통일농구' 방북단 평양行...조명균 "평화 진전 계기 됐으면" (종합)
2 文대통령 "성평등, 여가부 의무만이 아냐...각 부처가 책임질 고유업무"
3 김어준 "김부선 스캔들, 적절할 시점에 밝힐 것...필요하면 법정 증언"
4 文대통령 "靑·정부, 기업 현장방문 적극 해달라"
5 [단독] "쿠데타 JP는 훈장 받고 독립운동가 장준하 자식은 숨어살고...공정한 세상 멀었다"
6 한국당 구원투수에 83세 이회창?
7 "한국당이 변했어요"...한국당의 잇단 '감성 발언'
8 '비핵화 시간표' 들고 방북하는 폼페이오, 北 '화답' 끌어낼까
9 이를 남긴 폼페이오 방북...차후 한반도·동북아 정세 '가능자'
10 "이회창 전 총재, 한국당 비대위원장설에 불쾌감"
```

- dms 페이지에서 오늘의 식단을 출력하는 프로그램을 작성하시오.

python code)

```
# 크롬 브라우저를 띄우기 위해 selenium 으로 웹드라이버를 가져옴
from selenium import webdriver
# 크롬 드라이버로 크롬 브라우저를 실행
driver = webdriver.Chrome('chromedriver')
try:
    # 네이버 뉴스 페이지로 이동
    driver.get('http://news.naver.com')
    # 네이버 뉴스임을 알 수 있도록 현재 타이틀 출력
    print(driver.title)
    # 최근 뉴스 목록을 가진 div id 태그를 가져옴
    title_id = driver.find_element_by_id('right.ranking_contents')
    # 위 div_id 안에 li 태그로 구분되어있는 정보를 가져와 리스트로 저장
    news_list = title_id.find_elements_by_tag_name('li')
    # 가져온 태그들에 대해 반복문을 수행하면서 각각의 문자열을 출력
    for news in news_list:
        print(news.text)
except Exception as e:
    print(e)
finally:
    # 브라우저 종료
    driver.quit()
```

result)

```
DMS
Breakfast
기장밥, 감자수제비국, 떡갈비두부강정, 요구르트, 배추김치, 파리고추메추리알조림
Lunch
기장밥, 짜글이찌개, 닭봉데리야끼오븐구이, 콩나물무침, 고구마그라탕, 열무김치
Dinner
기장밥, 단배추된장국, 영양돼지갈비찜, 도토리묵상추무침, 배추김치, 제주감귤쥬스
```

- day 10

1. 크롤링 단계

- selenium 설정

: 네이버로 이동 후 분석한다.

: 검색을 위해 검색어 창의 태그를 분석한다.

* 실제 브라우저에서 보기 (ctrl+shift+c 또는 f12 로 볼 수 있다.)

=> id 값은 query 로 되어있다.

: 검색어 창으로 원하는 입력 값을 전송한다.

: 검색 후 결과 창을 분석한다.

: 블로그, 카페, 뉴스 등 여러 정보가 있다.

: 클래스 구분이 띄어쓰기인 것을 알 수 있다.

* ctrl+f => 클래스 명으로 고유한 값인지 확인

: 고유한 클래스 명으로 사용해야만 정확한 요소에 접근할 수 있다.

- 블로그 제목과 url 출력하기

: clear()를 통해 검색 창을 초기화 해준다.

: 간혹 미리 써져 있는 값이 있을 수 있기 때문이다.

python code)

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
driver= webdriver.Chrome('chromedriver')
try :
    driver.get('http://www.naver.com')
    print(driver.title)
    elem = driver.find_element_by_id('query')
    elem.clear()
    elem.send_keys('대덕소프트웨어마이스터고')
    elem.send_keys(Keys.RETURN)
    blogs = driver.find_element_by_class_name('_blogBase')
    blogs_list = blogs.find_elements_by_tag_name('li')
    # blogs_list의 자료형은 list가 된다
    for post in blogs_list:
        post_title = post.find_element_by_class_name('sh_blog_title')
        print(post_title.text)
        print('-'*20)

except Exception as e:
    print(e)
finally:
    driver.close()
```

result)

```
NAVER
어제 대덕소프트웨어마이스터고 면접 마치고 왔습니다 + 후기
-----
2018년 대덕소프트웨어 마이스터고 전형요강을 살펴...
-----
대덕소프트웨어 마이스터고 준비하는 법
-----
```

- : ... 으로 생략되어 있는 부분의 제목이 나오지 않음을 볼 수 있다.
- : li 태그를 자세히 보면 속성 title 값으로 제목이 들어있다.
- : 이를 통해 제목을 생략없이 출력하도록 한다.
- : 또한 href 속성을 이용하여 url도 출력해준다.

python code)

```
print (post_title.get_attribute('title'))
print (post_title.get_attribute('href'))
```

result)

```
NAVER
어제 대덕소프트웨어마이스터고 면접 마치고 왔습니다 + 후기
https://bman4.blog.me/221132349707
-----
2018년 대덕소프트웨어 마이스터고 전형요강을 살펴 보겠습니다.
http://wondangcom.com/103
-----
대덕소프트웨어 마이스터고 준비하는 법
https://blog.naver.com/frogramo?Redirect=Log&logNo=220895343332
-----
```

- 뉴스 제목과 url 출력

- : 제목들을 포함하고 있는 li 태그와 그 하위의 뉴스 리스트인 li 태그가 하나로 묶여있다.
- *그냥 li 태그로 요소를 찾아서 리스트로 저장하면 오류가 발생하게 된다.**
- : xpath를 활용하여 경로를 지정해준다.
- : ul 태그 밑에 있는 li 태그 즉, 제목을 포함하고 있는 li 태그만을 가져오도록 한다.

python code)

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
driver= webdriver.Chrome('chromedriver')
try :
    driver.get('http://www.naver.com')
    print(driver.title)
    elem = driver.find_element_by_id('query')
    elem.clear()
    elem.send_keys('대덕소프트웨어마이스터고')
    elem.send_keys(Keys.RETURN)
    news = driver.find_element_by_class_name('news')
    news_list = news.find_elements_by_tag_name('li')
    news_list = news.find_elements_by_xpath('./ul/li')

    for news in news_list:
        news_title = news.find_element_by_class_name('_sp_each_title')
        print(news_title.text)
        print (news_title.get_attribute('href'))
        print('-'*20)

except Exception as e:
    print(e)
finally:
    driver.close()
```

result)

```
NAVER
2018 상위17개 대학 '일반고 선방'... '학종시대' 맞아 특목고 감소
http://www.veritas-a.com/news/articleView.html?idxno=120378
-----
SW 업계, 대덕SW마이스터고와 산학협력 강화
http://www.edaily.co.kr/news/newspath.asp?newsid=03381686619243688
-----
대전교육청, 4차 산업혁명 대비 소프트웨어 교육 몽골에 전파
http://edu.donga.com/?p=article&ps=view&at_no=20180618112210898837
-----
대전교육청, 4차 산업혁명 대비 소프트웨어 교육 몽골에 전파
http://www.daejonilbo.com/news/newsitem.asp?pk_no=1322088
-----
대전교육청, 4차 산업혁명 대비 소프트웨어 교육 몽골에 전파
http://www.daejeontoday.com/news/articleView.html?idxno=500470
-----
```