type 연산자
is 연산자
type(a) is int >>> T or F

import math

문자열
S = 'abcdefghijkl'
S[0] = 'a'
S[-1] = 'l'
S[1] = 'b'

슬라이드 : [start:end:step]
길이 : len()
문자열 분리 : split()
>>> s = 'I am gum'
>>> k = s.split()
>>> k
>>> ['I' , 'am' , 'gum'] // 리스트 형식
>>> 'haha#you#are#me.'.split('#') // split 안에 문자열로 분해한다.
>>> ['haha' , 'you' , 'are' , 'me']
문자열 결합 : join()
>>> ' '.join(k) // 앞에 문자를 사이에 대입하여 문자열 합침
>>> 'I am gum'

리스트에 항목 추가
변수.append('추가할 항목') // 리스트 맨 뒤에 항목을 추가
변수.insert(인덱스, '추가할 항목') // 인덱스 위치에 항목을 추가, 인덱스가 한계를 넘으면 맨 뒤에 추가함

리스트에 항목 삭제
del 변수[인덱스] // 인덱스 번호에 있는 항목을 제거
변수.remove('특정한 값') // 리스트에서 '특정한 값'인 항목을 삭제한다.
변수.pop(인덱스) // 인덱스에 있는 항목을 반환하고 리스트에서 삭제한다. 인덱스가 없으면 맨 뒤에 항목을 삭제

리스트 병합
변수1.extend(변수2) // 변수1에 변수1 + 변수2 리스트로 재입력이 된다.
+ 연산자 사용

리스트 내에 항목 찾기
변수.index('찾을 항목') // 찾을 항목에 해당하는 인덱스를 반환한다.

리스트 내에 항목 포함유무 확인하기
'확인할 항목' in 변수 // True or False 의 결과를 반환한다.

리스트 내 특정값의 개수를 확인

변수.count('항목') // 변수 리스트 내에 '항목'이 몇 개 들어있는지 확인

리스트를 문자열로 합치기
''.join(변수)

정렬
sort() 메서드 : 리스트 자체의 내부 정렬
sorted() 함수 : 복사본을 만들어 정렬을 하고 반환, 원본은 정렬 안됨.
reverse=True를 추가하면 내림차순으로 정렬

항목 개수 반환
len(변수) // 개수를 반환

** 리스트를 대입연산자(=)로 할당할 경우 리스트 객체를 공유한다. **
** 리스트 객체를 공유하지 않고 할당하는 경우는 copy() 매서드를 사용한다. **

a = [1,2,3]
b = a.copy() // 복사
c = list(a) // 복사
d = a[:] // 복사
e = a // 객체 공유 할당

튜플

튜플 언태킹
```
>>> t = (1,2,3)
>>> a, b, c = t
>>> a
1
>>> b
2
```

swap 코드 활용
```
>>> a, b = 1, 2
>>> a
1
>>> a, b = b, a
>>> a
2
```

딕셔너리

키 값은 변경이 불가능한 값들로만 대입이 가능하다.

값 출력

get 매서드
>>> 변수.get('키 값', '등록된 값이 아닌 경우에 출력하는 값')

항목 추가하기
변수['추가할 키'] = '추가할 값'

항목 결합/갱신
변수1.update(변수2) // 변수 2를 변수 1에 결합

항목 삭제
del 변수[인덱스] // 인덱스의 키와 값을 삭제
변수.clear() // 변수의 딕셔너리가 초기화

딕셔너리의 키 존재 확인
'찾고 싶은 키 값' in 변수 // T or F

모든 키 얻기
변수.keys()

모든 값 얻기
변수.values()

모든 키-값 얻기
변수.items()

** 리스트를 대입연산자(=)로 할당할 경우 리스트 객체를 공유한다. **
** 리스트 객체를 공유하지 않고 할당하는 경우는 copy() 매서드를 사용한다. **

셋

셋의 항목은 변경이 불가능한 값들만 가능하다.

특정 값이 들어있는지 확인
'찾을 값' in 변수 // T or F

교집합 구하기
{} & {}
{}.intersection({})

합집합 구하기
{} | {}
{}.union({})

차집합 구하기
{} - {}
{}.difference({})

대칭 차집합 구하기
{} ^ {} // 교집합을 뺀 값
{}.symmetric_difference({})

부분집합 물어보기
셋1.issubset(셋2) // 셋1이 셋2에 포함되냐
셋1.issuperset(셋2) // 셋2가 셋1에 포함되냐
< , > , = 로 부분집합 기호 표시

주석 달기
#

라인 유지하기
\

조건문
```
if 조건식:
  문장1
  문장2
  …
elif 조건식:
  문장1
  문장2
  …
else:
  문장4
  문장5
```

None, 정수 0, o.o, “”. [], (). {}, set() 모두 False로 판별이 된다.

```
age = int(input('나이 입력하세요: '))
if age <= 13:
  print('어린이')
elif 13 < age <= 18:
  print('청소년')
else:
  print('성인')
```

랜덤 숫자 출력
```
import random
```

random.randrange(시작,끝) // 범위에 있는 숫자 1개 출력
random.randint(시작,끝)

반복문

while문

while 조건식:
  문장
  문장
  …
  문장

프로그램 종료
import sys
sys.exit()

---
for in 구문

for 변수 in 순회 가능한 객체 혹은 범위:
  문장
  문장
  …
  문장

d = {'apple' : '사과', 'orange' : '오렌지'}
for k, v in d,items():
  print(k, v)
>>> apple 사과
    orange 오렌지
---
zip()
여러 시퀀스를 병합한 순회 가능한 객체를 생성
길이가 다를 경우 가장 짧은 시퀀스를 기준으로 병합
>>> name = {'haha','gege'}
>>> age = {'16','22','55'}
>>> zip(name, age)
[('haha',16) , ('gege',22)]

range()
range(start, end, step) / start에서 end-1까지의 정수 생성, step이 있으면 간격대로 숫자 생성

for I in range(1, n+1):

break문

continue문 // 반복문의 조건문으로 넘어감

대문자로 바꾸는 함수
변수.upper()

컴프리헨션
하나 이상의 순회 가능한 객체로부터 파이썬의 자료 구조를 간단하게 만드는 방법
파이썬을 더욱 파이썬스럽게 사용하는 방법

변수 = 표현식 for 항목 in 순환 가능한 객체 또는 범위
예)
str_numbers = [str(x) for x in range(1,11)] // 1~10을 문자열로 변환하여 리스트 생성

변수 = 표현식 for 항목 in 순환 가능한 객체 또는 범위 if 조건
예)
even_numbers = [x for x in range(1, 11) if x % 2 == 0]

딕셔너리 컴프리헨션
s = '문장 문장 문장 ...'
counts = {letter : s.count(letter) for letter in s} // 이는 중복된 문자도 세는 오류가 있다.
counts = {letter : s.count(letter) for letter in set(s)} // 중복된 문자를 제거한다.

셋 컴프리헨션
변수 = {x for x in range(1,8)}

제너레이터 컴프리헨션
()로 감싸는 것은 튜플 컴프리헨션이 아니라 제너레이터 컴프리헨션이 된다.

함수

함수의 정의
```
def 함수이름(매개변수1, 매개변수2):
    문장1
    문장2
    return 문장3
```

함수 호출
변수 = 함수이름(변수1, 변수2)

가변 매개변수
def 함수이름(*args):
    print(args)

함수이름(1, 3, 'haha', True) // 튜플로서 매개변수에 전달됨

키워드 매개변수
def 함수이름(**kwargs):
    print(kwargs)

함수이름(name='haha' , age=20) // 딕셔너리로 매개변수에 전달이 됨

매개변수를 사용하는 순서는
일반 매개변수 - 디폴트 매개변수 - 가변 매개변수 - 키워드 매개변수 의 순서대로 나열되어야 한다.
가변이랑 키워드는 한 함수에 하나씩만 사용 가능하다.

리스트 spread

리스트를 가변 매개변수로 넘길 때 튜플로 되면서 리스트 전체가 하나의 튜플로 되는 오류가 발생
이를 막기 위해 리스트를 가변 매개변수로 넘길 때 풀어서 전달하는 과정임

def 함수이름(*args):
    문장
    return 변수
ls = [10,20,30,40]
r2 = 함수이름(*ls) // *ls : 리스트 spread , ls으로 값을 넘기면 오류 발생

딕셔너리 spread

키워드 매개변수가 있을 때, 딕셔너리를 넘기려고 하면 딕셔너리 spread를 사용하여 오류를 막는다.

...
d2 = 함수이름(**ds)
...

---
docstring

함수의 설명을 출력하는 방법

def 함수이름(매개변수):
    '설명설명...'
    문장들

>>> print(함수이름.__doc__)
설명설명

클로저

외부함수가 내부함수에 의해서 기억이 되는 경우

람다함수

함수 자체를 함수에서 쓰려고 할 때, 매개변수로 함수를 넘기는 과정에서 미리 선언한 함수를 넘기는 것이 아니라 넘기는 과정에서 한줄로 선언하면서 바로 넘기는 과정

변수1 = 함수이름1(변수2, 함수이름2)
변수1 = 함수이름1(변수2, 람다 word: 함수2의 내용)

데코레이터

@함수이름1
함수이름2() // 함수이름2를 선언하는게 아니라 함수이름1의 내용이 출력되고 함수이름2는 내부함수로 넘어간다.

제너레이터

전제 시퀀스를 한번에 출력하지 않고, 순차적으로 생성하는 객체를 만들어 사용
range 함수도 제너레이터에 속함

네임스페이스와 스코프

네임스페이스 = 지역변수와 전역변수 느낌으로, 지역에서 먼저 변수이름을 찾게 되고, 없으면 한단계 위로 올라가거나 전역변수에서 이름을 찾게된다.

지역변수에서 global을 붙이면 전역변수의 같은 이름의 변수를 연동시켜줌.

locals() 함수 : 로컬 네임스페이스의 내용이 담긴 딕셔너리 반환
globals() : 전역 네임스페이스의 내용이 담긴 딕셔너리 반환, 내용이 많다.

변수의 이름을 설정할 때
__<이름>__처럼 언더바 2개를 연속으로 앞뒤로 사용하는 방법은 추천하지 않는다
안되는 것은 아닌데, 다른 예약어, 함수와 겹칠 수 있기 때문에 추천하지 않는다는 것이다.

커맨드 라인 인자

콘솔창에서 입력한 변수의 값을 이용하여 값을 만든 방법

import sys
sys.argv[콘솔창에서 입력하는 순서]

모듈 또는 import문

```
if __name__ == '__main__':
```

이 파일이 메인 파일 일 때 라는 뜻이다.

import 불러올다른파일이름또는모듈이름
변수1 = 불러올다른파일이름또는모듈이름.그파일안에있는함수의이름()

다른 이름으로 import하는 경우

import report as wr // report라는 파일을 wr로 부르기로 한다.

방법1

```
import report

description = report.get_description()
```

방법2

```
from report import get_description

description = get_description()
```

방법3

```
import report as wr

description = wr.get_description()
```

random 안에 choice는 리스트 안에 하나를 골라서 반환해 주는 함수이다.

패키지

game 디렉터리

computer.py

```
from random import choice

def choose_computer():
return choice(['가위','바위','보'])
```

```
if __name__ == '__main__':
for _ in range(10):
print(choose_computer())
```

play.py

```
count = {'win':0 , 'lose':0, 'draw':0}

def run(p1, p2):
if p1 == p2:
result = 'draw'
elif p1 == '가위':
result = 'win' if p2 == '보' else 'lose'
elif p1 == '바위':
result = 'win' if p2 == '가위' else 'lose'
else:
result = 'win' if p2 == '바위' else 'lose'
count[result] += 1
return result

def print_result():
print('Result'.center(40, '#'))
print('Win:', count['win'],',Lose:',count['lose'],',Draw:',count['draw'])
print('#'*40)


if __name__ == '__main__':
print(run('가위','보'))
print(run('가위','바위'))
print(run('가위','가위'))
print(run('바위','가위'))
print(run('보','가위'))
print_result()
```

user.py

```
def choose_user():
error_message = "잘못된 입력입니다. 1~3 사이의 숫자로 입력해주세요.\n"
choices = ['가위','바위','보']

while True:
for i,v in enumerate(choices):
print(i+1 , v)
try: # 오류가 발생되지 않으면 실행
index = int(input('선택>'))-1
if 0 <= index < len(choices):
return choices[index]
print(error_message)
except: # 오류가 일어나면 실행
print(error_message)
continue

if __name__ == '__main__':
result = choose_user()
print(result)
```

veny 디렉토리

main,py

```
from game import user, computer, play

while 1:
you = user.choose_user()
com = computer.choose_computer()
result = play.run(you, com)
print('You:', you, ', Computer:', com, ',Result:', result)
answer = input('게임을 더 하시겠습니까?(y/n)').lower() # 소문자로 변환
if answer.startswith('n'): # 처음 시작하는 단어가n일 경우
```

```
break

play.print_result()
```

또는

```
import game.user
import game.play
import game.computer


while 1:
you = game.user.choose_user()
com = game.computer.choose_computer()
result = game.play.run(you, com)
print('You:', you, ', Computer:', com, ',Result:', result)
answer = input('게임을 더 하시겠습니까?(y/n)').lower() # 소문자로 변환
if answer.startswith('n'): # 처음 시작하는 단어가n일 경우
break

game.play.print_result()
```

또는

```
from game.user import choose_user
import game.play
import game.computer


while 1:
  you = choose_user()
  com = game.computer.choose_computer()
  result = game.play.run(you, com)
  print('You:', you, ', Computer:', com, ',Result:', result)
  answer = input('게임을 더 하시겠습니까?(y/n)').lower() # 소문자로 변환
  if answer.startswith('n'): # 처음 시작하는 단어가n일 경우
    break

game.play.print_result()
```

__init__.py 파일의 __all__ 변수가 있어야 * 참조를 했을 때 오류가 안남

__init__.py

```
__all__ = ['user', 'computer', 'play']
```

표준 라이브러리

해당 프로그래밍 언어에서 기본적으로 제공되는 라이브러리를 말한다.

collections 라이브러리

setdefault()

defaultdict(‘값’) // 만약 값이 없는 경우에는 값을 디폴드 값으로 가진다.

counter() // 입력된 문자열이나 정수들의 입력횟수를 출력해준다.

orderedDict() // 입력된 값을 순서대로 출력해주는 함수

deque // stack(FILO) 과 qucue(FIFO)를 섞은 함수

itertools 라이브러리

chain() // 리스트나 튜플에 묶여 출력할시 한번에 다 나올 때 사용하면 각각 하나씩 출력할 수 있게 나눠준다.
cycle() // 어떤 집단의 값들을 순환하여 계속 반환되게 해준다.
accumulate(리스트, 함수) // 리스트의 값들을 차례로 함수 계산을 해준다. 디폴트는 더해주는 것이다.
[1,2,3,4] 이면 1, 3, 6, 10으로 차례로 출력이 된다.
combinations

pprint 라이브러리

pprint() // ,로 길게 이어진 출력문에서 ,단위로 줄바꿈을 해주면서 출력을 해준다.

서드파티 라이브러리
언어 자체에서 제공해 주는 것이 아닌 다른 회사나 개인 개발자가 작성하여 제공해주는 라이브러리를 말한다.

로또 번호를 만드는 코드

```
from itertools import combinations
from random import choice

numbers = list(range(1,46))
lotto_numbers = list(combinations(numbers,6))
true_lotto_number = choice(lotto_numbers)
print(true_lotto_number)
```

객체
숫자에서 모듈, 데이터, 코드 등등 까지 모든 것을 객체로 취급

클래스
객체를 생성하기 위한 설계도

__init__(self) 메서드
클래스로 객체를 생성할 때 객체의 초기화를 위한 메서드

__init__() 메서드를 이용하여 객체 속성값 초기화

```
class Person:
  def __init__(self, name):
    self.name = name

person = Person('임꺽정')
print(person.name)
```

메서드

객체를 명사와 비교한다면 메서드는 동사와 비교될 수 있음
객체가 할 수 있는 행동(기능)

메서드 안에서 메서드를 선언하는 예제

```
class Person:
  def __init__(self, name, age):
```

```
    self.name = name
    self.age = age

  def is_adult(self):
    return True if self.age > 19 else False

  def eat(self):
    print('냠냠' if self.is_adult() else '오물오물')

baby = Person('김아가', 3)
adult = Person('김어른', 30)
baby.eat()
adult.eat()
```

인스턴스 메서드의 첫 번째 인자로 self를 항상 포함해야 함

인스턴스 메서드

클래스 메서드

클래스 자체를 객체로 보고 클래스에 대해 작동하는 메서드
인스턴스 메서드와 달리 첫 번째 매개변수의 이름을 관례적으로 cls라는 이름을 사용하며 여기에 클래스가 전달
상속의 의미가 있음
@classmethod

정적 메서드

클래스에 의해 정의되는 네임스페이스에 들어 있는 함수
정적 메서드는 클래스에 의해 호출되기 때문에 첫 번째 매개변수로 self, cls가 없음
@staticmethod

```
import time

class Date:
def __init__(self, year, month, day):
self.year = year
self.month = month
self.day = day

@staticmethod
def now():
t = time.localtime() # 시스템의 현재 시간을 가져옴
return Date(t.tm_year, t.tm_mon, t.tm_mday)

@staticmethod
def tomorrow():
t = time.localtime(time.time() + 86400) # 하루는86400초
return Date(t.tm_year, t.tm_mon, t.tm_mday)

a = Date(1967, 4, 7)
b = Date.now() # 객체를 사용할 필요 없이 클래스 그대로 가져온다.
c = Date.tomorrow()

print(a.year, a.month, a.day)
print(b.year, b.month, b.day)
print(c.year, c.month, c.day)
```

정적 속성(멤버 변수)

클래스에 정의된 변수로 객체 인스턴스로 접근하지 않고 클래스를 통해 접근
모든 객체 인스턴스가 데이터를 공유

```
class Account:
num_accounts = 0 # 정적 속성(정적 멤버 변수)

def __init__(self, name, balance):
self.name = name
self.balance = balance
Account.num_accounts += 1

def deposit(self, amt):
self.balance += amt

def withdraw(self, amt):
self.balance -= amt

account1 = Account('홍길동', 1000)
print(account1.num_accounts)
print(Account.num_accounts)

account2 = Account('임꺽정', 2000)
print(account1.num_accounts)
print(account2.num_accounts)
print(Account.num_accounts)
```

객체의 작동 방식

각 데이터 타입의 작동 방식은 해당 타입이 구현하고 있는 특수 메서드들의 집합에 따라 결정
특수 메서드들을 구현하여 객체의 작동 방식을 변경할 수 있음

연산자 오버로딩

같은 이름으로 다양한 타입의 객체가 연산이 되도록 구현하는 것을 말함
파이썬은 특수 메서드들을 오버로딩하는 방식으로 연산자 오버로딩을 구현
__이름__(매개변수 ...) : 특수 메서드

객체의 문자열 표현

```
    def __str__(self):
        return '%s의 잔고: %d' % (self.name, self.balance)
```

객체 비교와 순서 매기기

```
    def __bool__(self):
        return True if self.balance > 0 else False
```

```
    def __lt__(self, other):
        return self.balance < other.balance
```

타입검사

속성접근

호출 가능 인터페이스
__call__ 메서드를 제공하여 객체가 함수처럼 호출 가능하도록 할 수 있음

```
import math

class Distance:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __call__(self, x, y):
        return math.sqrt((self.x - x) ** 2 + (self.y - y) ** 2)

distance = Distance(0, 0)
print(distance(4,3))
print(distance(8,7))
```

private 네임 맹글링

파이썬에는 완전하지 않지만 외부에서 사용자가 직접 접근할 수 없도록 속성이나 메서드 이름 앞에 두 개의 언더스코어(__)를 붙여 이름을 지을 수 있다.
__<이름> 형태의 속성이나 메서드는 객체를 사용하는 외부에서 객체의 해당 이름으로 직접 접근할 수 없어 이름을 사용자로부터 감추는 기능을 하게 됨.
맹글링 : 이름을 다른 형태로 바꾸는 것을 말함

getter, setter, 프로퍼티

프로그래머는 private 속성값을 읽고 쓰기 위해 getter 메서드나 setter 메서드를 이용
파이썬에는 이보단 프로퍼티를 사용

getter의 경우 @property 데코레이터를 사용
setter의 경우 @<property 데코레이터가 붙은 메서드명<프로퍼티 사용

상속

기존 클래스의 작동 방식을 특수화하거나 변경함으로써 새로운 클래스를 만드는 매커니즘
원본 클래스는 기반 클래스, 상위 클래스, 부모 클래스 라고 함
상위 클래스가 없는 경우라도 모든 클래스는 object로부터 상속을 받으며, object는 모든 클래스의 조상 클래스

매서드 오버라이드
상위 클래스의 메서드를 새롭게 재정의 하는 것을 매서드 오버라이드라고 함
다형성을 구현할 수 있음

super() 함수
super() 는 자식 클래스에서 부모 클래스의 메서드에 접근할 때 사용
코드 재사용

컴포지션

상속은 일반적인 is a 관계이다. (슈퍼카는 자동차이다.)
상속을 한다는 것은 상위 클래스를 더 구체화하는 과정

컴포지션은 has a 관계이다. (자동차는 엔진이 있다.)
객체는 다른 객체를 포함할 수 있다.
포함하는 관계는 컴포지션으로 구현할 수 있다.

텍스트 데이터

아스키 코드
1960년대 정의 되었으며 모든 언어를 표현하기엔 용량이 부족하여 완성시킬수가 없었음.

유니코드
전 세계 언어의 문자를 정의하기 위한 국제 표준 코드
수학 및 기타 분야 기호도 포함
사용방법 : \u____ (16진수 4자리) 또는 \U________(16진수 8자리)

unicodedata
유니코드 식별자와 이름으로 검색할 수 잇는 함수 제공
lookup() : 대/소문자를 구분하지 않는 인자를 취하고, 유니코드 문자를 반환
name() : 인자로 유니코드 문자를 취하고 대문자 이름을 반환

UTF-8 인코딩과 디코딩
1바이트 : 아스키코드
2바이트 : 키릴 문자를 제외한 대부분의 파생된 라틴어
3바이트 : 기본 다국어 평면의 나머지
4바이트 : 아시아 언어 및 기호를 포함한 나머지
파이썬, 리눅스, HTML의 표준 텍스트 인코딩

인코딩과 디코딩 실습

```
place = 'caf\u00e9'
print(place)
type(place)

place_bytes = place.encode('utf-8')
print(place_bytes)
```

```
type(place_bytes)

print(place_bytes.decode('utf-8'))
# place_bytes.decode('ascii')
print(place_bytes.decode('latin-1'))
print(place_bytes.decode('windows-1252'))
```

서식지정자
%s 문자열
%d 10진 정수
%x 16진 정수
%o 8진 정수
%f 10진 부동소수점수
%e 지수로 나타낸 부동소수점수
%g 10진 부동소수점수 혹은 지수로 나타낸 부동소수점수
%% 리터럴%

포맷
```
print('{n}, {f}, {s}'.format(n=42, f=7.03, s='string'))
```

정규표현식
특정한 패턴의 문자열의 집합을 표현하는 형식
re.match(pattern, source) : source가 pattern과 일치하는 지 확인
re.search(pattern, source) : source에서 pattern과 일치하는 첫 번째 문자열 리턴
re.findall(pattern, source) : source에서 pattern과 일치하는 모든 문자열 리턴
re.split(pattern, source) : pattern에 일치하는 문자열로 source를 나눔
re.sub(pattern, replacer, source) : source에서 pattern과 과 일치하는 문자열을 replacer로 치환

정규표현식 특수 문자
```
import re

source = '123a'
m = re.match(r'\d\d\d\w',source)

if m:
    print(m.group())
```

❖ 정규표현식 특수 문자

특수문자	설명	사용예
₩d	숫자(0 ~ 9)	r'₩d+' : 숫자가 1번 이상 연달아 나타나는 패턴
₩D	비숫자(0 ~ 9)제외	r'₩Da' : 숫자가 아닌 문자 뒤에 a가 나타나는 패턴
₩w	알파벳문자(a ~ z, A ~ Z)	r'₩w₩d' : 알파벳문자 하나, 숫자 하나로 이루어진 패턴(a0 등)
₩W	비알파벳문자	r'₩W₩d' : 알파벳이 아닌 문자 다음에 숫자가 나타나는 패턴
₩s	공백문자	r'₩s₩w₩w₩w₩s' : 공백 문자 사이에 알파벳 문자가 3개 나타나는 패턴
₩S	비공백문자	r'₩Sabc' : 공백 문자가 아닌 문자 다음에 abc가 나타나는 패턴
₩b	단어경계(₩w ₩W 또는 ₩W와 ₩w 사이의 경계)	r'₩babc₩b' : 단어 경계 사이에 abc가 나타나는 패턴
₩B	비단어경계	r'₩Babc₩B' : 단어 경계가 아닌 문자 사이에 abc가 나타나는 패턴

정규표현식 패턴 지정자

❖ 정규표현식 패턴 지정자

특수문자	설명	사용예
abc	리터럴 abc	r'abc' : abc와 일치하는 패턴
(*expr*)	*expr*. 정규표현식	r'(abc)' : abc와 일치하는 패턴
$expr_1$|$expr_2$	$expr_1$ 또는 $expr_2$	r'ABC|abc': ABC 또는 abc
.	₩n을 제외한 임의의 문자	r'You.' : You 다음에 임의의 문자 하나가 오는 패턴
^	문자열의 시작	r'^You' : source가 You로 시작할 경우 일치
$	문자열의 끝	r'₩d$': source가 숫자로 끝날 경우 일치
prev?	*prev*가 0또는 1회	r'a?bc' : bc, abc
*prev**	0회 이상의 최대 *prev*	r'a*bc' : bc, abc, aabc, aaabc, ... (강의에서 설명)
*prev**?	0회 이상의 최소 *prev*	r'a*?bc' : bc, abc, aabc, aaabc, ...(강의에서 설명)
prev+	1회 이상의 최대 *prev*	r'a+bc' : abc, aabc, aaabc, ... (강의에서 설명)
prev+?	1회 이상의 최소 *prev*	r'a+bc' : abc, aabc, aaabc, ... (강의에서 설명)
prev{m}	m회의 *prev*	r'₩d{3}' : 숫자가 3번 나오는 패턴
prev{m, n}	m ~ n 회의 최대 *prev*	r'₩d{8, 11}' : 숫자가 8 ~ 11번 나타나는 패턴
prev{m, n}?	m ~ n 회의 최소 *prev*	r'₩d{8, 11}?' : 숫자가 8 ~ 11번 나타나는 패턴
[abc]	a 또는 b 또는 c (a|b|c 와 같음)	r'[₩d₩w]?' : 숫자 또는 알파벳이 0번 또는 1번 나타나는 패턴
[a-z]	a ~ z 중의 하나	r'[a-zA-Z]' : a ~ z 또는 A ~ Z 중 알파벳 하나
[^abc]	a 또는 b 또는 c를 제외한 패턴	r'[^012]₩d': 0 또는 1 또는 2가 아닌 문자 다음에 숫자가 나오는 패턴

```
import re

source = 'ABCDE'
m = re.search('^AB',source)

if m:
    print(m.group())
```

AB

```
import re

source = 'hi "1234" "5678" '
m = re.search(r'".*"',source)

if m:
    print(m.group())
```

“1234” “5678”

```
import re

source = 'hi "1234" "5678" '
m = re.search(r'".*?"',source)

if m:
    print(m.group())
```

“1234”

❖ 정규표현식 패턴 지정자

특수문자	설명	사용예
prev(?=*next*)	뒤에 *next*가 오면 *prev*	예제를 통해 설명
prev(?!*next*)	뒤에 *next*가 오지 않으면 *prev*	
(?<=*prev*)*next*	전에 *prev*가 오면 *next*	
(?<!*prev*)*next*	전에 *prev*가 오지 않으면 *next*	

❖ 정규표현식 실습

```
# match() 실습
>>> import re
>>> source = '''I wish I may, I wish might
... Have a dish of fish tonight.'''
>>> re.findall(r'wish', source)
>>> re.findall(r'wish|fish', source)
>>> re.findall(r'^wish', source)
>>> re.findall(r'^I wish', source)
>>> re.findall(r'fish$', source)
>>> re.findall(r'fish tonight\.$', source)
>>> re.findall(r'[wf]ish', source)
>>> re.findall(r'[wsh]+', source)
>>> re.findall(r'ght\W', source)
>>> re.findall(r'I (?=wish)', source)
>>> re.findall(r'(?<=I) wish', source)
>>> re.findall(r'\bfish', source)
```

이진데이터

바이트와 바이트 배열

바이트는 튜플처럼 불변한다.
바이트 배열은 바이트의 리스트처럼 변경가능하다.

struct 모듈
이진 데이터를 파이썬 데이터 구조로 바꾸거나 파이썬 데이터 구조를 이진 데이터로 변환할 수 있음

전체 주석 = ctrl + /

struct.unpack()

비트연산

❖ 파이썬은 C언어와 유사한 비트단위 정수 연산을 제공

a = 5 (이진수로 0b0101), b = 1, 이진수로 0b0001)로 가정

연산자	설명	예제	10진수 결과	2진수 결과
&	AND	a & b	1	0b0001
|	OR	a | b	5	0b0101
^	배타적exclusive OR	a ^ b	4	0b0100
~	NOT(0 → 1, 1 → 0)	~a	-6	정수에 따라 이진 표현이 다름
<<	비트 왼쪽 이동	a << 1	10	0b1010
>>	비트 오른쪽 이동	a >> 1	2	0b0010

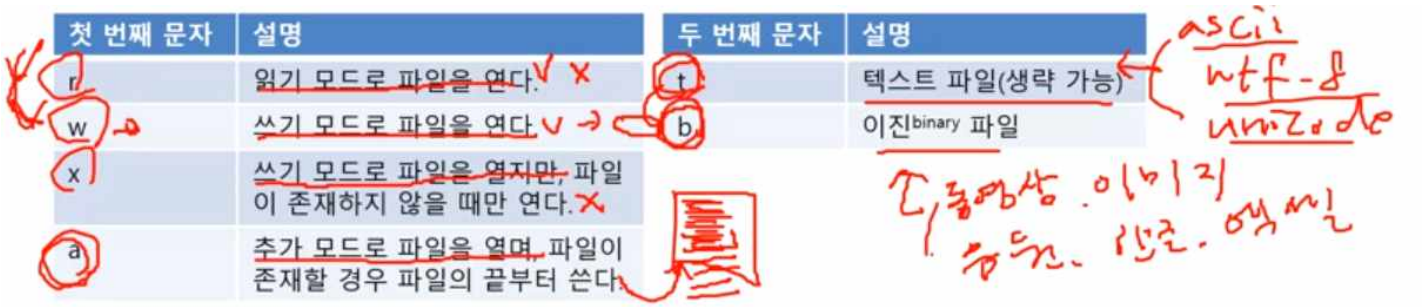
파일 입출력과 파일 읽기 쓰시

변수 = open('경로', '모드')

모드

첫 번째 문자	설명
r	읽기 모드로 파일을 연다.
w	쓰기 모드로 파일을 연다.
x	쓰기 모드로 파일을 열지만, 파일이 존재하지 않을 때만 연다.
a	추가 모드로 파일을 열며, 파일이 존재할 경우 파일의 끝부터 쓴다.

두 번째 문자	설명
t	텍스트 파일(생략 가능)
b	이진binary 파일



변수.close()

파일을 열고 쓰고 저장하기

```
poem = '''
sdkf ksd
sdfa faf afaf af
sdfaewf esfafe
staen wlerhfakd ewjflajfae efjslife
'''

print(len(poem))

fout = open('test1.txt', 'wt')
result = fout.write(poem)
print(result) # 쓴 양의 총 바이트 수
fout.close()
```

```
poem = '''
sdkf ksd
sdfa faf afaf af
sdfaewf esfafe
staen wlerhfakd ewjflajfae efjslife
'''

print(len(poem))

fout = open('test2.txt', 'wt')
print(poem, file=fout, sep='', end='') # 출력을 파일에 진행하고, 각 단어 사이의 공백을 넣지 않고, 출력하고 나서 아무것도 넣지 않겠다.
# print(poem)

fout.close()
```

파일 읽기

```
fin = open('test1.txt','rt')
poem = fin.read()
fin.close()
print(len(poem))
print(poem)
```

파일을 일정크기로 읽기 실습

```
poem = ''
size = 100
fin = open('test1.txt','rt')
while 1:
    chunk = fin.read(size)
    if not chunk:
        break
    poem += chunk
fin.close()
print(len(poem))
print(poem)
```

readlines()메서드

```
fin = open('test1.txt','rt')
lines = fin.readlines() # 리스트 형식으로 각 줄이 저장이된다.
fin.close()

print('{} lines read'.format(len(lines)))
print(lines)

for line in lines:
    print(line, end='')
```

이진 파일 쓰기

```
bdata = bytes(range(0, 256))
print(len(bdata))
fout = open('binary.dat','wb')
```

```
result = fout.write(bdata)
fout.close()
```

이진 파일 읽기

```
fin = open('binary.dat','rb')
bdata = fin.read()
fin.close()

print(len(bdata))
print(bdata)# 이진데이터로 표현이 되어있는 것을 볼 수 있다.
```

with 구문
구문 내의 내용이 실행되고 나면 자동으로 close() 메서드가 진행되어 공간을 정리해준다.

파일 위치 다루기 : tell(), seek()
tell() 메서드 : 파일의 시작 위치로부터 현재 오프셋을 반환
seek() 메서드 : 다른 바이트 오프셋으로 위치를 이동

```
fin = open('binary.dat', 'rb')
pos = fin.tell()
print(pos)
pos = fin.seek(255) # 파일의 마지막에서1바이트 전으로 이동
print(pos)
bdata = fin.read()
print(len(bdata))
print(bdata[0])

fin.seek(100)
bdata = fin.read()
print(len(bdata)) # 256-100 = 156이 반환
print(bdata) # 156개의 이진데이터를 반환
fin.close()
```

seek() 메서드에서 os 기능을 이용해서 위치를 처음, 끝, 현재 위치를 반환할 수 있다.

구조화된 텍스트 파일

csv (엑셀)
콤마(,)로 데이터가 구분된 구조화괸 텍스트 파일로 스프레드시트와 데이터베이스의 교환형식으로 자주 사용
표준 csv 모듈을 사용하는 것이 좋음

```
import csv

people = [
['홍길동',20] , ['임꺽정',30] , ['유관순',17] , ['전우치',25]]

with open('people.csv', 'wt', newline='', encoding='utf-8') as fout: # newline 을 없다고 하지 않으면 데이터간
간격이 한줄씩 띄어져있다.
csvout = csv.writer(fout)
for row in people:
csvout.writerow(row)
```

csv 파일 읽기

```
import csv

with open('people.csv', 'rt', encoding='utf-8') as fin:
csvin = csv.reader(fin)
people = [row for row in csvin]

print(people)
print(sum([int(row[1]) for row in people]) / len(people))
```

csv 파일을 딕셔너리로 읽기

```
import csv

with open('people.csv', 'rt', encoding='utf-8') as fin:
csvin = csv.DictReader(fin, fieldnames=['name','age'])
people = [row for row in csvin]

print(people)
print(sum(int(row['age']) for row in people) / len(people))
```

csv 파일을 딕셔너리로 쓰기

```
import csv

people = [
        {'name':'홍길동', 'age':20},
        {'name':'임꺽정', 'age':30},
        {'name':'유관순', 'age':17},
        {'name':'전우치', 'age':25}
]

with open('people2.csv', 'wt', newline='', encoding='utf-8') as fout:
    csvout = csv.DictWriter(fout, ['name','age'])
    csvout.writeheader()
    csvout.writerows(people)
```

헤더가 있는 csv 딕셔너리 파일 읽기

```
import csv

with open('people2.csv', 'rt', encoding='utf-8') as fin:
    csvin = csv.DictReader(fin)
    people = [row for row in csvin]

print(people)
print(sum(int(row['age']) for row in people) / len(people))
```

XML

데이터를 구분하기 위해 태그를 사용

계층적인 구조이다. 마크업 형식

공백, 들여쓰기는 상관이 없다. 무시

조상 부모 자식 자손 요소 등으로 표현, 트리구조라고 한다.

JSON

자바스크립트의 객체의 표기법이지만, 현재는 데이터를 교환하는 아주 인기 있는 형식이 되었음
프로그램간 데이터를 교환할 때 파이썬과 궁합이 잘 맞음.

Json에서 파이썬 : 디코딩 - load(), loads()
파이썬에서 Json : 인코딩 - dump(), dumps()

```
{
        "breakfast" : {
                "hours" : "7-11",
                "items" : {
                        "breakfast burritos" : "$6.00",
                        "pancakes" : "$4.00"
                }
        },
        "lunch" : {
                "hours" : "11-3",
                "items" : {
                        "hamburger" : "$5.00"
                }
        },
        "dinner" : {
                "hours" : "3-10",
                "items" : {
                        "spaghetti" : "$8.00"
                }
        }
}
```

JSON에서 파이썬으로 디코딩

```
import json

with open('menu.json', 'rt', encoding='utf-8') as file:
menu = json.load(file)

print('menu: %s' % menu)

breakfast = menu['breakfast']
print('breakfast: %s' % breakfast)

hours = breakfast['hours']
items = breakfast['items']
print('hours: %s' % hours)
print('items: %s' % items)

for k, v in items.items():
print('%s: %s' % (k, v))
```

파이썬에서 json으로 인코딩

```
import json

people = [
{'name':'홍길동', 'age':20},
{'name':'임꺽정', 'age':30}
]

people_json = json.dumps(people)
print('1)', type(people_json))
print('2)', people_json)
print('3)', json.dumps(people, ensure_ascii=False))

with open('people.json', 'wt', newline='',encoding='utf-8') as file:
json.dump(people, file, ensure_ascii=False)
```

---

pip
PyPI 저장소로부터 파이썬 패키지를 받아 설치하는 패키지 관리 도구

PyPI
파이썬의 서드파티 오픈소스 패키지들을 위한 저장소

cmd에서 진행

버전 확인
pip —version

패키지 설지
pip install --upgrade pip
pip install requests

설치되어있는 목록확인
pip list

설치되어있는 것중에 업데이트가 가능한 패키지 확인
pip list —outdated

설치된 패키지 삭제(예시: requests)
pip uninstall requests
y

설치된 패키지의 상세정보를 알아볼 때
pip show 상세정보를확인할패키지

검색
pip search 검색할패키지이름

---

virtualenv

virtualenv 설치
pip install virtualenv

virtualenv 버전확인
virtualenv --version

프로젝트 디렉터리를 만들고 프로젝트 디렉터리로 이동
mkdir envex
cd envex

virtualenv 생성
virtualenv venv

특정 python버전으로 virtualenv 생성
virtualenv venv —python=python3.8

웹

```
from urllib.request import urlopen

url = 'https://en.wikipedia.org/wiki/Python_(programming_language)'

conn = urlopen(url)
print(conn)  # HTTPResponse

data = conn.read()  # 웹 페이지로부터 바이트 코드로 데이터를 읽어옴
print(data)
print(data.decode('utf-8'))

print(conn.status)  # HTTP 응답코드 출력(다음 페이지 설명)
# 2XX : 성공
# 3XX : 리다리렉트
# 4XX : 클라이언트에서 에러
# 5XX : 서버에서 에러

print(conn.getheader('Content-Type'))  # Content-Type 정보 출력

for key, value in conn.getheaders():
print(key, value)  # 헤더 정보들을 출력
```

이미지 크롤링

```
from urllib.request import urlopen
import re

def get_data(url):
conn = urlopen(url)
if 200 <= conn.status < 300:
return conn.read().decode('utf-8')
else:
return None
def get_img_urls(html_data):
pattern = re.compile(r'\"imageUrl\":\"(.*?)\"')
return pattern.findall(data)


# url 입력
url = input('Input url : ')

data = get_data(url)
result = get_img_urls(data)

# 이미지url 출력
for img in result:
print(img)
```

웹 서버 구동하기

cmd에서 작동

파일 생성
- mkdir 생성할파일이름
- mkdir 생성할파일이름\생성파일의생성할하위폴더이름
- cd 위치할파일이름
  를 한후
  mkdir 생성할파일이름

css문서 : 시각적인 내용들을 정해주는 문서
- notepad css문서를저장할 경로\css파일이름.css

작성내용
-
```
h1 {  // html의 h1과 매칭이 되는 것이다.
  font-size: 1.5em;
  text-align: center;  // 가운데 정렬
}

img {
  display: block;  // 이미지를 블록으로 정렬
  margin: 1em auto; // 위아래 1, 좌우는 자동 정렬
}
```

이미지 저장
- 이미지를 images 파일에 저장

index.html 문서 작성 : 서버가 가장 먼저 여는 문서
- notepad index.html

내용
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>The 2st Simple Web Page</title>
    <link rel="stylesheet" href="/assets/styles/main.css">
  </head>
  <body>
    <h1>Simple Web Page</h1>
    <img src="/assets/images/image.jpg" alt="NO image">
  </body>
</html>
```

```
<!DOCTYPE html>
```

```
<html>
  <head> // 크게 두가지로 나뉜다
    <meta> // 메타 코드의 정보를 담고 있다.
    <meta charset=“”> // 어떤 언어로 인코딩을 할지 정한다.
    <title> 웹의 이름을 적는다.</title>
    <link> // 디자인(css)를 적용하기 위해 만들고, link뒤에 오는 것들은 속성이라고 한다.
    <link rel="stylesheet" href="css의 위치“>
  </head>
  <body> // 크게 두가지로 나뉜다
    <h1>문서의 제목</h1> // 숫자는 단계를 뜻하며 1~6단계가 있다.
    <img> // 이미지 정보를 보여준다.
    <img src=“이미지의 위치” alt=“만약 이미지가 로딩이 안될 경우 출력되는 언어”>
  </body>
</html>
```

서버를 호출

- python -m http.server

index - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>The 2st Simple Web Page</title>
    <link rel="stylesheet" href="/assets/styles/main.css">
  </head>
  <body>
    <h1>Simple Web Page</h1>
    <img src="/assets/images/image.jpg" alt="NO image">
  </body>
</html>
```

명령 프롬프트 - python -m http.server

```
Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. All rights reserved.

C:₩Users₩User>mkdir webserver-ex

C:₩Users₩User>cd webserver-ex

C:₩Users₩User₩webserver-ex>mkdir assets

C:₩Users₩User₩webserver-ex>mkdir assets₩images

C:₩Users₩User₩webserver-ex>mkdir assets₩styles

C:₩Users₩User₩webserver-ex>explorer

C:₩Users₩User₩webserver-ex>notepad assets₩styles₩main.css

C:₩Users₩User₩webserver-ex>notepad index.html

C:₩Users₩User₩webserver-ex>python -m http.server
Serving HTTP on :: port 8000 (http://[::]:8000/) ...
```

- 서버
웹서버1
서버프로그램들1
서버프로그램들2
서버프로그램들3

ip
- 000.000.000.000
- 127.0.0.1 // 내 자신의 컴퓨터 자체의 주소
- http://127.0.0.1:8000
- localhost:8000

- DNS

포트번호
http - 80
https - 443

내 컴퓨터의 ip주소를 보는 방법
- ipconfig

192.168.0.4

웹 프레임워크
- 특정 종류의 프로그램을 개발하기 위한 골격, 개발을 더 쉽고 빠르게 할수 있도록 해줌

마이크로 프레임워크 - 구성요소를 가볍게 다룰수 있음

풀 스택 프레임워크 - 모든 요소를 다룰수 있음

Flask
- 파이썬의 대표적인 웹 프레임워크 중에 하나

웹 API

REST
- url에 대항하는 특정 자원을 HTTP 메서드를 통해 처리
- HTTP 표준 프로토콜에 따르는 모든 플랫폼에서 사용가능
- 여러 가지 서비스 디자인에서 생길수 있는 문제를 최소화
- 서버와 클라이언트의 역할 분리
- 하지만 표준이 존재하지 않고 사용할 수 있는 메서드가 제한되어 있다.

strip(문자) 메서드 : 양쪽의 있는 문자를 삭제시켜준다.

도로명 주소 크롤링

```
from requests import get
from pprint import pprint
import json

api_url = 'https://business.juso.go.kr/addrlink/addrLinkApiJsonp.do'
key = 'devU01TX0FVVEgyMDIzMDkxNjE0NDc1MzExNDEwODU='

def search_address(keyword, page, limit):
params = {
'keyword': keyword, 'currentPage': page, 'countPerPage': limit,
'confmKey': key, 'resultType': 'Json'}

res = get(api_url, params=params)

# print(res.text)
results = json.loads(res.text.strip('(').strip(')'))['results']

# pprint(results)
total = results['common']['totalCount']

return results['juso'], total

if __name__ == '__main__':
keyword = input('검색어 입력: ')
page = input('검색 페이지(기본1): ')
limit = input('페이지당 검색 수(기본10): ')

page = int(page) if page else 1
limit = int(limit) if limit else 10

addresses, total = search_address(keyword, page, limit)

print('{}개의 결과'.format(total))
for i, addr in enumerate(addresses):
num = limit * (page - 1) + i + 1
print('{:3d}) {}'.format(num, addr['roadAddr']))
```

웹 크롤러

- 월드와이드 웹을 탐색하며 자동으로 원하는 데이터를 수집

BeautifulSoup

- HTML 데이터를 분석하여 파이썬에서 사용하기 쉬운 형태로 가공

링크 따오기

requests, beatifulsoup install 하기

```
import sys
import requests
from bs4 import BeautifulSoup as soup

def get_links(url):
result = requests.get(url)
page = result.text
doc = soup(page, 'html.parser')
links = [element.get('href') for element in doc.find_all('a')]
```

```
return links

if __name__ == '__main__':
for url in sys.argv[1:]:
print('Links in', url)
for num, link in enumerate(get_links(url), start=1):
print(num, link)
print()
```

이미지 크롤링

requests, beatifulsoup install 하기

```
import re
import requests
from bs4 import BeautifulSoup as soup

def get_google_image_urls(keyword):
    ptn = re.compile(r'^http?://')
    base_url = 'https://www.google.com/search'
    params = {'q': keyword,'tbm': 'isch'}
    res = requests.get(base_url, params=params)
    doc = soup(res.text, 'html.parser')
    return [e.get('src') for e in doc.find_all('img')]

def download_image(url, filename):
    try:
        res = requests.get(url)
        if res.status_code != 200:
            return ''
        type = res.headers['content-Type']
        if type == 'image/jpeg':
            ext = '.jpg'
        elif type == 'image/png':
            ext = '.png'
        elif type == 'image/gif':
            ext = '.gif'
        else:
            return ''

        filename = '{}{}'.format(filename, ext)

        with open(filename, 'wb') as f:
            f.write(res.content)
        return filename
    except:
        return ''


if __name__ == '__main__':
    keywords = ['강아지','고양이']
    urls = []
    for keyword in keywords:
        urls = [*urls, *get_google_image_urls(keyword)]

    for i, url in enumerate(urls, start=1):
```

```
        filename = download_image(url, '{:03d}'.format(i))
        if filename:
            print('download {} --> {}'.format(url, filename))
```

파일과 데이터

파일 생성
- open() 함수를 이용

파일 존재 여부 확인
- os.path.exists() 함수는 파일 또는 디렉터리 존재 여부를 체크

파일 타입 확인
- os.path.isfile(filepath) 함수 : 인수로 넘긴 filepath가 피일인지 체크
- os.path.isdir(filepath) 함수 : 인수로 넘긴 filepath가 디렉터리인지 체크
- os.path.isabs(filepath) 함수 : 인수로 넘긴 filepath가 절대경로인지 체크
    상대 경로 : 헌재 작업 디렉터리를 기준으로 위치를 표현
    절대 경로 : 루트 디렉터리를 기준으로 파일의 위치를 표현

파일 복사하기
- shutil.copy('<source file>', '<destination file>')

파일 이름 바꾸기
- os.rename('<oid filename>','<new filename>')

파일 연결하기
- os.link() : 하드 링크 확인
- os.symlink() : 심벌릭 링크 생성(바로가기 파일) / 관리자 권한이 필요함
- os.path.lslink() : 해당 파일이 심벌릭 링크인지 확인

파일 권한 변경
- 파일에 대해서 소유자, 그룹, 사용자에게 읽기, 쓰기, 실행 권한을 설정하는 것
- os.chmod() 함수를 이용해 퍼미션을 바꿀 수 있음
    windows 운영체제에서는 완전히 동작하지 않음

절대 경로 얻기
-os.path.abspath() 함수 사용

심벌릭 링그 얻기
- os.path.realpath() 함수 사용

파일 삭제
- os.remove() 함수 사용

- 파일에 대한 쓰기권한이 없으면 삭제시 PermissionError 예외 발생

디렉터리 생성
- os.mkdir() 함수 사용

디렉터리 삭제
- rmdir() 함수 사용

디렉터리 내 파일 목록 나열
- os.listdir() 함수 사용

작업 디렉터리 변경
- os.chdir()

일치하는 파일 나열
- glob.glob() 함수 사용

프로그램과 프로세스

프로그램
- 어떤 작업을 수행하기 위해 실행 할 수 있는 파일

프로세스
- 프로그램이 실행된 상태

subprocess 모듈
- 존재하는 다른 프로그램을 실행시킬 수 있음.
- 쉘에서 프로그램을 실행하여 생성된 결과를 얻고 싶은 경우 grtoutput() 함수를 이용
- 프로그램의 종료 상태를 표시하려면 getstatusoutput() 함수를 사용
  리턴값은 (종료상태, 실행 결과)로, 종료상태가 0이면 정상적인 종료를 의미

```
import subprocess

result = subprocess.getoutput('dir')
print('dir'.center(80, '='))
print(result)


result = subprocess.getoutput('dir venv')
print('dir venv'.center(80, '='))
print(result)


result = subprocess.getstatusoutput('dir')
print('dir'.center(80, '='))
print('status =', result[0])
print('result: ')
print(result[1], end='\n\n')


result = subprocess.getstatusoutput('dir unknown')
```

```
print('dir unknown'.center(80, '='))
print('status =', result[0])
print('result: ')
print(result[1], end='\n\n')
```

multprocessing 모듈
- 파이썬 함수를 별도의 프로세스로 실행하거나 한 프로그램에서 독립적인 여러 프로세스를 실행 할 수 있음.

crowl() 함수 : 크롤링하는 함수

실행되고 있는 프로세스를 종료하고자 하면 terminate() 메서드를 사용

```
from multiprocessing import Process
import os
import time
import random

def do_this(what, count=1):
whoami(what, count)

def whoami(what, count):
loop = 1
while loop <= count:
sleep = random.randint(0,5)
print('%d) Process %d says: %s, sleep: %d sec' % (loop, os.getpid(), what, sleep) )
time.sleep(sleep)
loop += 1

if __name__ == '__main__':
do_this("i'm the main process")

process_list = [Process(target=do_this, args=("i'm Function %d " % i , 5)) for i in range(1,5)]
for p in process_list:
p.start()

time.sleep(10)

for p in process_list:
p.terminate()
```

달력과 시간

datetime 모듈은 여러 메서드를 가진 4개의 주요 객체를 정의
- date : 년월일
- time : 시분초 마이크로초
- datetime : 날짜와 시간
- timedelta : 날짜 또는 시간의 간격

```
from datetime import date, time, timedelta

# halloween = date(2020, 10, 31)
# print(halloween) # class는 객체
# print(halloween.isoformat()) # class는str
# print(halloween.year)
# print(halloween.month)
```

```
# print(halloween.day)

# today = date.today()
# one_day = timedelta(days=1)
# tomorrow = today + one_day
# yesterday = today - one_day
#
# print(tomorrow) # 내일 날짜
# print(today + one_day*17) # 17일 뒤 날짜
# print(today + one_day*30) # 30일 뒤 날짜
# print(yesterday) # 어제 날짜

# noon = time(12, 30, 0)
# print(noon)
# print(noon.hour)
# print(noon.minute)
# print(noon.second)
# print(noon.microsecond)

some_day = datetime(2020, 6, 9, 5, 10, 0)
print(some_day)
print(some_day.isoformat())

now = datetime.now()
print(now)
print(now.year)
print(now.month)
print(now.day)
print(now.hour)
print(now.minute)
print(now.second)
print(now.microsecond)

noon = time(12)
this_day = date.today()

# date와time 객체를datetime 객체로 묶기
noon_today = datetime.combine(this_day, noon)

print(noon_today)
print(noon_today.date())
print(noon_today.time())
```

time 모듈

```
import time

now = time.ctime() # 현재 시점을 에포치로부터 지난 시간으로 변환
print(now)
print(time.ctime(now)) # 에포치를 이용하여 시간을 문자열로 표현
print(time.localtime(now)) # 시스템의 표준 시간대로 출력(struct_time 객체 반환)
print(time.gmtime(now)) # UTC(협정 세계시) 시간으로 출력(struct_time 객체 반환)
tm = time.localtime(now) # struct_time을 에포지 초로 변환
print(time.mktime(tm)) # 밀리세컨드 단위는 무시됨
```

strftime()

- 날짜와 시간을 문자열로 변환할 수 있음.

strptime 매서드

문자열을 날짜와 시간으로 바꾸기 위해서 사용

로케일

- 지역에 따라 각자 다른 문화(언어, 시간, 날짜 등)을 갖고 있는데, 프로그램의 국제화는 사용자로 하여금 프로그램 수행시 로케일이란 것에 의해 지역에 맞는 환경을 선택할 수 있도록 해줌

```
import locale
from datetime import date

halloween = date(2020, 10, 31)
for lang_country in ['ko_kr', 'en_us', 'fr_fr', 'de_de', 'is_is']:
    # locale.setlocale(locale.LC_ALL, lang_country)  # 모든 로케일 설정
    locale.setlocale(locale.LC_TIME, lang_country)  # 시간 설정
    locale.setlocale(locale.LC_CTYPE, lang_country)  # 문자 유형 설정
    print(lang_country, halloween.strftime('%A, %B %d))
```