



'2021 멋쟁이사자처럼'

Python

INDEX

1- Python언어의 특징

2- Python 요소들

3- Django에서의 쓰임



Python 특징

* 파이썬은 문법이 쉬워 빠르게 배울 수 있다

* 파이썬은 개발 속도가 빠르다

* 무료 소프트웨어(오픈소스)이다.

* 파이썬 프로그램안에 다른 언어로 쓰인 프로그램을 포함 할 수 있다.

※ 오픈 소스(Open Source)란 저작권자가 소스 코드를 공개하여 누구나 별다른 제한 없이 자유롭게 사용 · 복제 · 배포 · 수정할 수 있는 소프트웨어이다.

Python 특징

Python의 응용분야

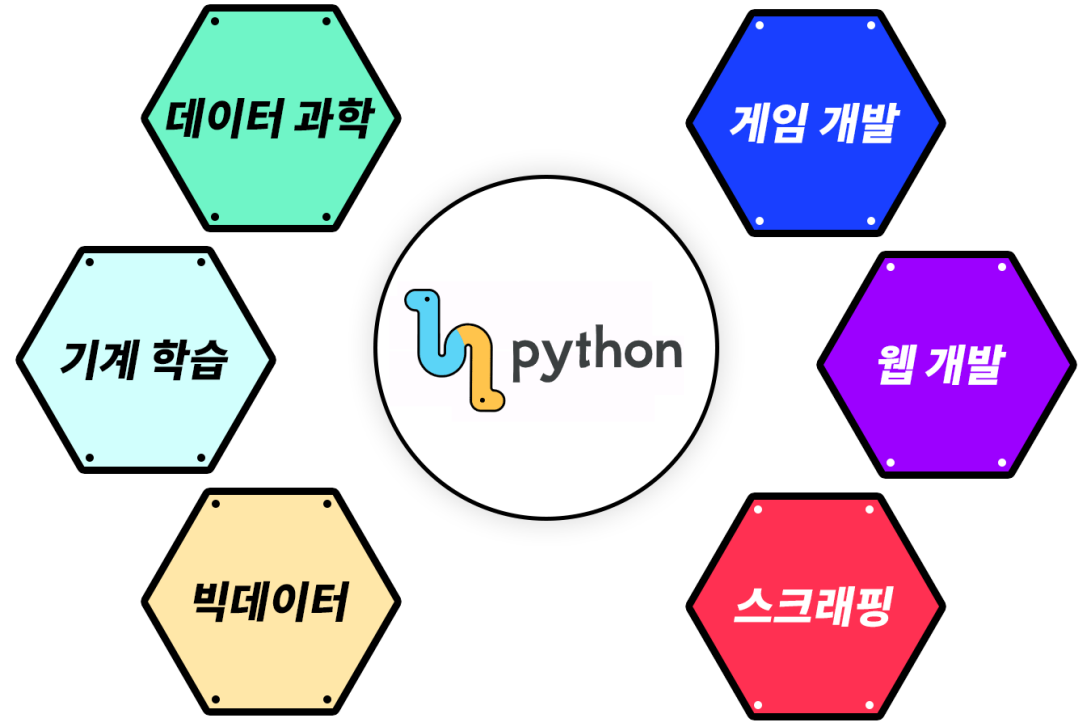
웹 프로그래밍

C/C++와의 결합

수치 연산 프로그래밍 (Numpy모듈)

데이터 분석(Pandas모듈)

사물 인터넷(라즈베리파이)



Python 요소들

변수

제어문 (if,elif,else,while,for)

함수

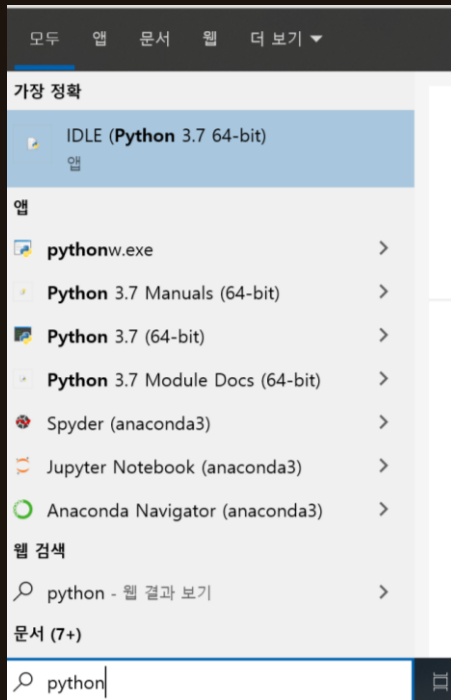
클래스

모듈, 패키지, 라이브러리

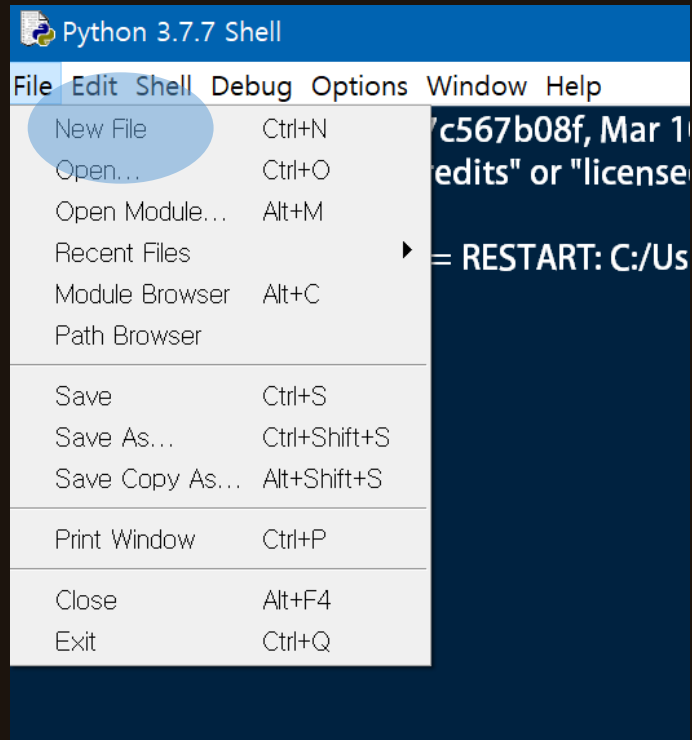
<https://wikidocs.net/33> : 점프투파이썬 요약본 => 또 다른 요소들에 대한 설명은 여기에서!

Python 사용하기

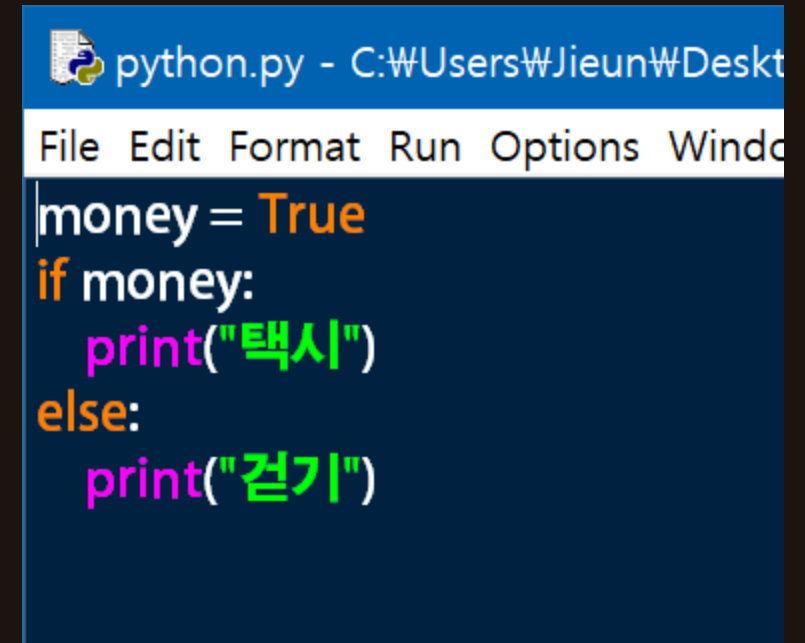
1. Python IDLE 실행



2. New File 선택



3. 코드 작성



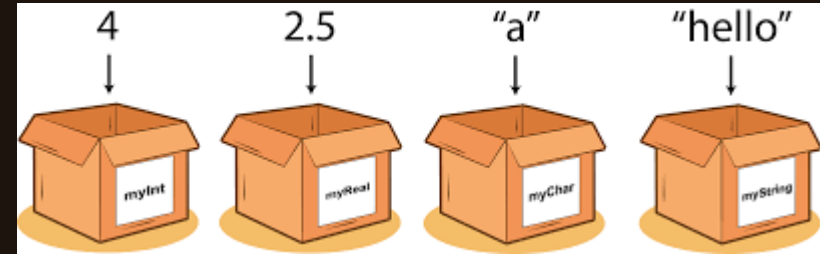
<https://wikidocs.net/33> : 점프투파이썬 요약본 => 또 다른 요소들에 대한 설명은 여기에서!

Python 요소들

변수

: 값을 저장하는 공간

```
a = 1  
b = "python"  
c = [1,2,3]
```



“변수 이름 = 변수에 저장할 값”

리스트 ; 간단히 말해서 숫자나 문자의 모음. 하나의 리스트 안에 0개 이상의 요소를 넣을 수 있다.
문자나 숫자 구분없이 하나의 리스트에 포함 할 수 있음.

Python 요소들

Django에서의 쓰임

```
settings.py  home.html  models.py  admin.py  views.py >
post > views.py > ...
1  from django.shortcuts import render
2  from .models import Blog
3  # Create your views here.
4
5  def home(request):
6      blogs = Blog.objects
7      return render(request, 'home.html', {'blogs': blogs})
```

1) 변수 사용 예시
Blogs라는 변수에 값을 넣는다.

2) 리스트의 사용 예시

```
settings.py X
Blog > settings.py > ...
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'post.apps.PostConfig',
41  ]
42
```


Python 요소들

제어문 If

: 조건을 판단 후 해당 조건에 부합하는 코드를 실행한다.

```
money = True
if money:
    print("택시를 타고 가라")
else:
    print("걸어 가라")
```

출력 : 택시를 타고 가라

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

+) Bool 자료형 : True / False의 값

a = True / b = False

* 첫 문자를 항상 대문자로 사용해야 함.

Python 요소들

조건문

: 참과 거짓을 판단하는 문장 => True or False로 출력되는 문장

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

출력 : 걸어가라

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

+) 조건문에서 a값이 3과 같은지 확인 > “if a==3:”

Python 요소들

조건문

: 다양한 조건을 판단하는 elif

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

출력: 택시를 타고가라

```
If <조건문>:
    <수행할 문장1>
    <수행할 문장2>
elif <조건문>:
    <수행할 문장1>
    <수행할 문장2>
elif <조건문>:
    <수행할 문장1>
    <수행할 문장2>
else:
    <수행할 문장1>
    <수행할 문장2>
...
```

Python 요소들

반복문 While

: 조건문이 참인 동안에 반복한다.

```
num = 1
while num <= 100:
    print(num)
    num = num + 1
```

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

Python 요소들

반복문

무한 루프 : 조건이 무한히 True값을 가져 반복되는 문장

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

+) 수행하고자하는 문장이 얼마나 반복될지 모를 때 주로 사용합니다.

Python 요소들

반복문

무한루프에서 빠져나오는 방법

```
while True:
```

```
    수행할 문장1
```

```
    수행할 문장2
```

```
    if 조건문:
```

```
        break
```

```
    수행할 문장3
```

Python 요소들

반복문 for

: 첫번째 요소부터 마지막 요소까지 차례로 변수에 대입되어 문장 수행.

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

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

Python 요소들

반복문

range함수를 이용한 for문

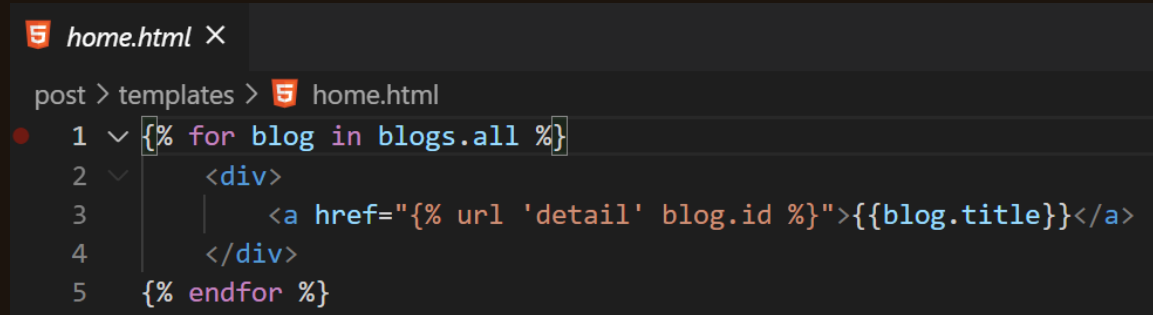
```
add = 0
for i in range(1, 11):
    add = add + i
print(add)
```

55

range 함수 : 숫자 리스트를 자동으로 만들어 준다.
=> 반복문에서 range()는 첫번째 숫자부터 마지막-1숫자까지 반복한다.

Python 요소들

Django에서의 쓰임



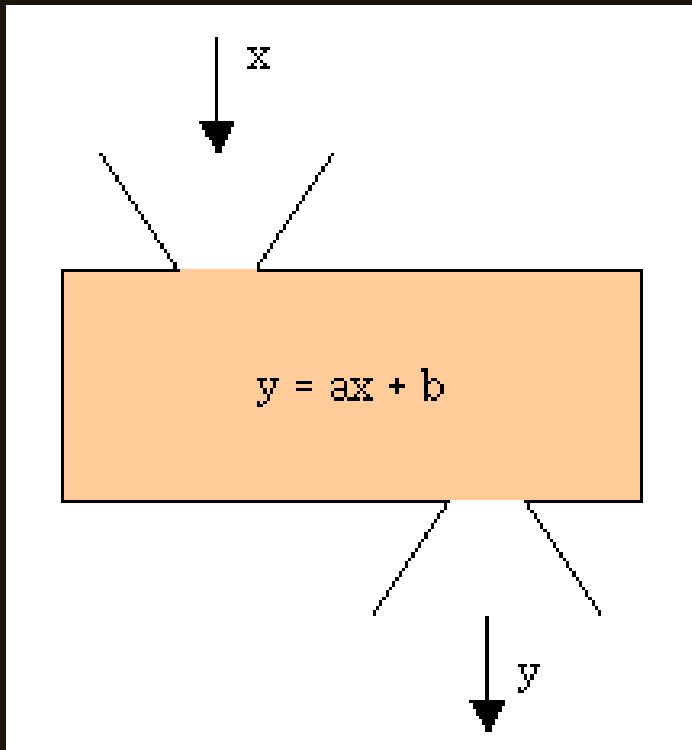
```
home.html ×  
post > templates > home.html  
1 {% for blog in blogs.all %}  
2     <div>  
3         <a href="{% url 'detail' blog.id %}">{{blog.title}}</a>  
4     </div>  
5 {% endfor %}
```

for문을 사용하여 blogs에 들어있는 blog 게시물 각각을 반환하면서 blog의 개수만큼 반복한다.

Python 요소들

함수

: 수학적 함수와 동일하게 어떤 입력값을 주었을 때 함수에 따른 결과 값을 돌려주는 것.



```
def f1(x):  
    a = 3  
    b = 5  
    y = a * x + b  
    return y # y 값을 반환한다
```

```
c = f1(10)  
print(c)  
35
```

Python 요소들

함수의 기본 형태

```
def 함수명(매개변수):  
    <수행할 문장1>  
    <수행할 문장2>
```

```
Ex) def lion(x):  
    print(x)  
    return x
```

함수 사용하기

함수 호출

함수반환 값 받을 변수 = 함수명(인수)

Ex) a = lion(3)

출력 : 3

매개변수 : 함수에 입력으로 전달된 값을 받는 변수

Python 요소들

Django에서의 쓰임

```
<> home.html <> detail.html models.py admin.py views.py X
post > views.py > ...
1  from django.shortcuts import render, get_object_or_404
2  from .models import Blog
3  # Create your views here.
4
5  def home(request):
6      blogs = Blog.objects
7      return render(request, 'home.html', {'blogs': blogs})
8
9  def detail(request, blog_id):
10     blog_detail = get_object_or_404(Blog, pk=blog_id)
11     return render(request, 'detail.html', {'blog_detail': blog_detail})
```

def 함수명(매개변수):
 <수행할 문장1>
 <수행할 문장2>

함수 home과 detail

detail 함수의 매개변수 : request, blog_id

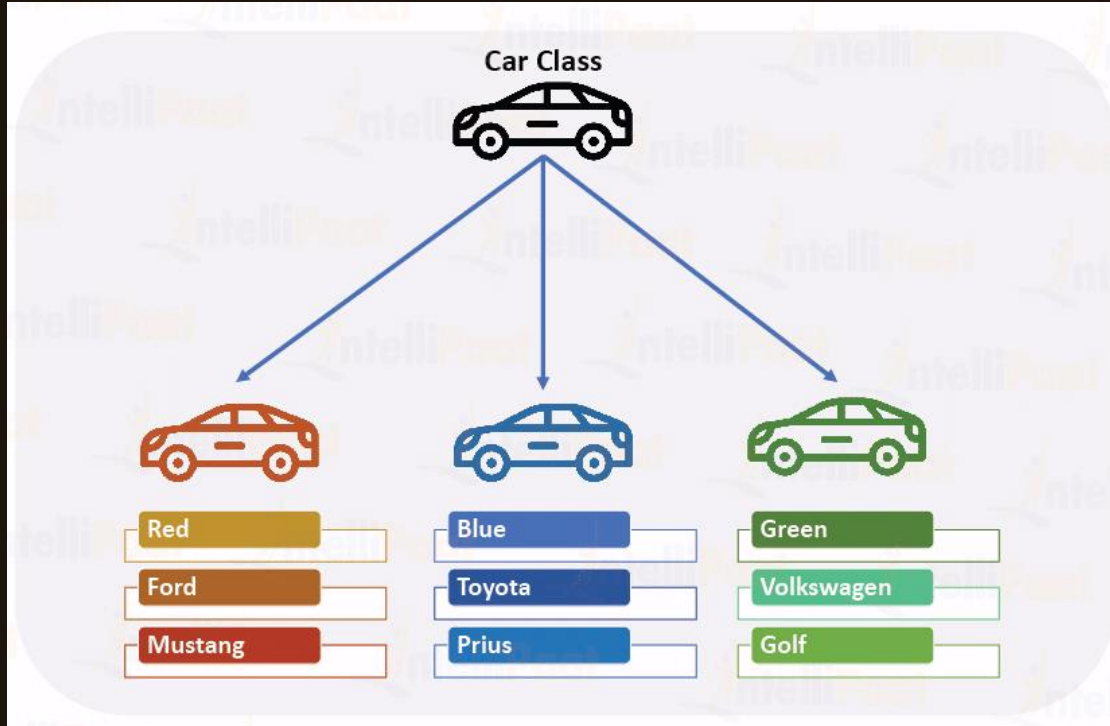
반환값 : render함수(템플릿을 불러오는 기능 ; 예시에선 home.html과 detail.html을 불러온다.)

- views.py는 웹의 매개변수인 request를 꼭 써준다.

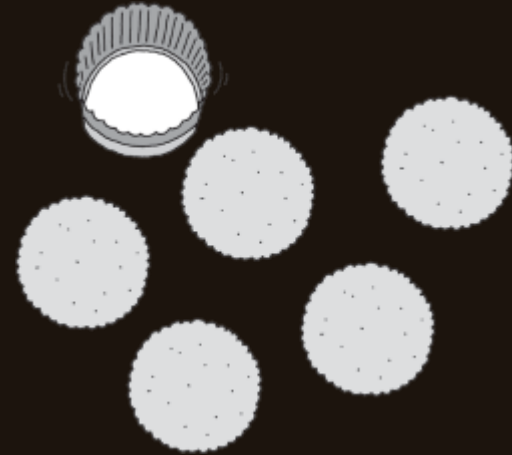
=> 위 이슈에 대한 자세한 답변 : <https://pythonq.com/so/python/1710902>

Python 요소들

클래스와 객체



속성이 아직 정의되지 않은 차 -> 클래스
클래스를 기반으로 속성이 정의된 차들 -> 객체



과자 '틀' -> 클래스
과자 틀에 의해서 만들어진 과자 -> 객체

Python 요소들

클래스와 객체

객체는 클래스로 만들며 1개의 클래스는 무수히 많은 객체를 만들어 낼 수 있다.

```
class Cookie:  
    pass
```

```
a = Cookie()  
b = Cookie()
```

클래스 선언 방법_

```
class 클래스명:  
    속성  
    메서드
```

객체 선언 방법_

```
객체명 = 클래스명()
```

* pass: 실행할 코드가 없다는 것을 의미함.

Python 요소들

클래스 선언

```
class Amazon:
```

```
    strength = 20
```

```
    energy = 15
```

```
    def attack(self):
```

```
        return 'Jab!!!'
```

```
    def exercise(self):
```

```
        self.strength += 2
```

클래스 선언 방법

class 클래스명:
 속성

메서드(행동):

“ 클래스를 만들기 위해서는 객체가 가질 **성질(속성)**과 하는 **행동**을 정의 ”

self : 자기자신 즉 객체를 가리킨다.

클래스를 만드는 경우 메서드를 정의할 때 반드시 self를 쓴다고 생각하면 좋아요.

Python 요소들

클래스와 객체

클래스 생성

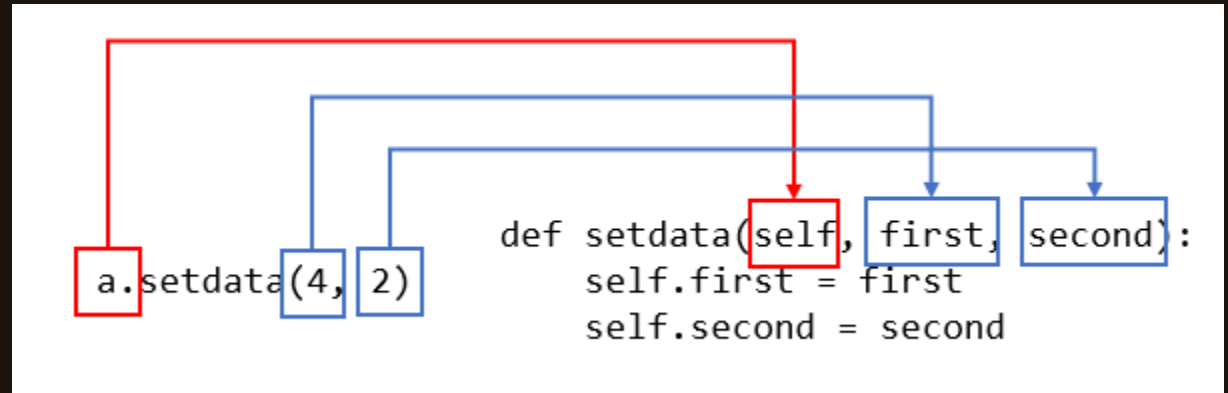
```
class Cal:
    def __init__(self, first, second):
        self.first = first
        self.second = second
    def setdata(self, first, second):
        self.first = first
        self.second = second
    def add(self):
        result = self.first + self.second
        return result
```

객체 생성

```
a = Cal(4, 2)
```

객체 a의 메서드 호출

```
a.add()
출력 : 6
```



Python 요소들

클래스와 객체

생성자

```
class Cal:
    def __init__(self, first, second):
        self.first = first
        self.second = second
    def setdata(self, first, second):
        self.first = first
        self.second = second
    def add(self):
        result = self.first + self.second
        return result
```

0 a = FourCal(4, 2) X a = FourCal()

생성자(Constructor)란 객체가 생성될 때 자동으로 호출되는 메서드를 의미한다.
매개변수에 값이 전달되지 않으면 생성 오류가 남.

Python 요소들

Django에서의 쓰임

```
settings.py  home.html  models.py X  views.py  urls.py
post > models.py > ...
1  from django.db import models
2
3  # Create your models here.
4  class Blog(models.Model):
5      title = models.CharField(max_length=100)
6      pub_date = models.DateTimeField('date published')
7      body = models.TextField()
8
```

Class 클래스명(상속):
변수(속성) = 값

```
.py  home.html  detail.html X  models.py
post > templates > detail.html > h4
1  <h1>{{blog_detail.title}}</h1>
2  <h3>{{blog_detail.body}}</h3>
3  <h4>{{blog_detail.pub_date}}</h4>
```

blog._detail 객체의 변수에
직접 접근

```
17  def create_blog(request):
18      blog = Blog()
19      blog.title = request.GET['title']
20      blog.body = request.GET['body']
21      blog.pub_date = timezone.datetime.now()
22      blog.save()
23      return redirect('/detail/'+str(blog.id))
```

객체 만들기
객체명 = 클래스명()

Python 요소들

모듈

'함수나 변수 또는 클래스를 모아 놓은 파일'

다른 파이썬 프로그램에서 불러와 사용할 수 있는 파이썬 파일

mod1.py

```
def add(a, b):  
    return a + b  
def sub(a, b):  
    return a-b
```

main.py

```
import mod1  
print(mod1.add(3, 4))
```

출력:7

방법1
: 모듈 전체 불러오기

main.py

```
from mod1 import add  
print(add(3, 4))
```

출력:7

방법2
: 모듈 내부 함수만 불러오기

Python 요소들

모듈 내부 모든 함수 가져오고 싶을 때

`from 모듈명 import *`

`from mod1 import *`

models.py 모듈에서 Blog 클래스를 가져옴.

```
from .models import Blog
from django.utils import timezone
```

Python 요소들

패키지

도트(.)를 사용하여 파이썬 모듈을 계층적(디렉터리 구조)으로 관리할 수 있게 해준다.

```
game/  
  __init__.py  
  sound/  
    __init__.py  
    echo.py  
    wav.py  
  graphic/  
    __init__.py  
    screen.py  
    render.py  
  play/  
    __init__.py  
    run.py  
    test.py
```

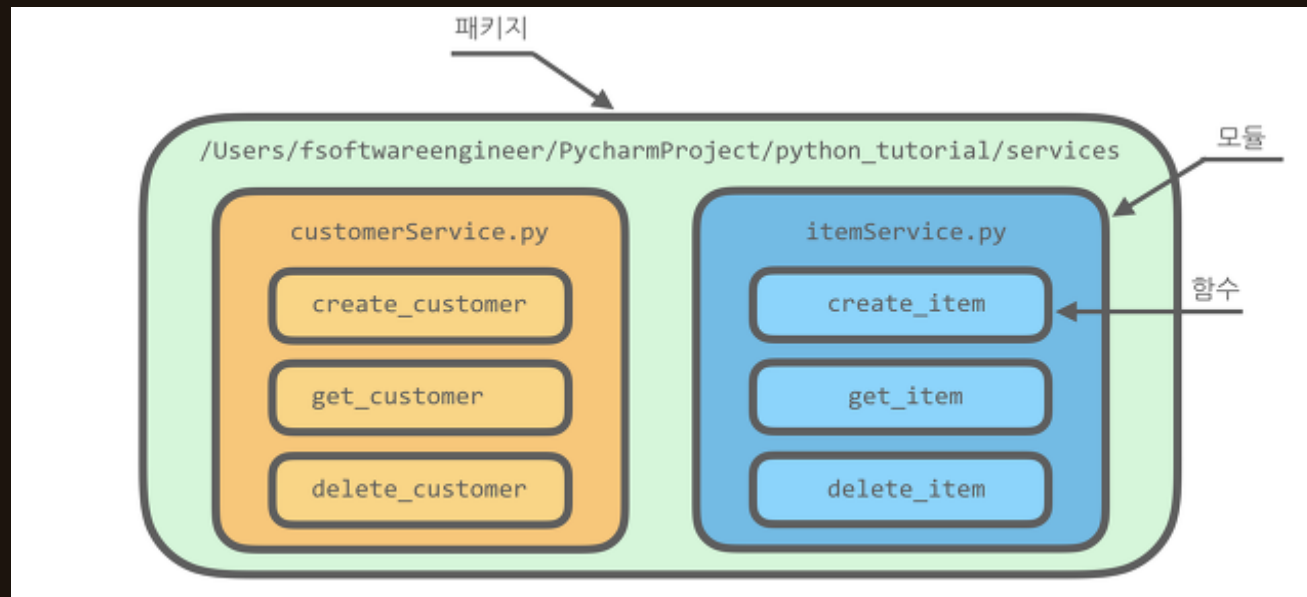
```
import game.sound.echo
```

직접 명령프롬프트로 현 위치를 조정하지 않아도
(도트)를 이용하여 모듈에 쉽게 접근가능하게 해준다.

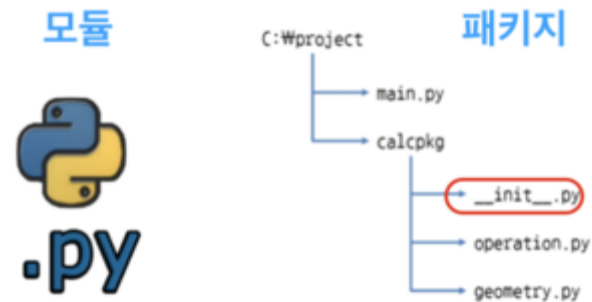
```
settings.py  home.html  views.py  X  
post > views.py > ...  
1  from django.shortcuts import render  
2  
3  # Create your views here.  
4  
5  def home(request):  
6      return render(request, 'home.html')
```

Python 요소들

Python 모듈과 패키지의 관계



- 모듈 : .py
- 패키지 : 폴더, 디렉토리



실습

Django의 views.py에서


python 개념 사용

* `blog_id == 3`인 게시물을 비밀 게시물로 만들기

views.py

```
def detail(request, blog_id):  
    if blog_id==3:  
        return render(request, 'three.html')  
    blog_detail = get_object_or_404(Blog, pk=blog_id)  
    return render(request, 'detail.html', {'blog_detail':blog_detail})
```

three.html

post > templates >  three.html

1 세번째게시물은 보이지않습니다.

blog_id

127.0.0.1:8000/admin/post/blog/1,

수고하셨습니다 😊

수고하셨습니다 😊