09 프로그래밍

9. 프로그래밍

- **꼭 필요한 문법 설명(1) : 함수**
  - 함수 = 약속
  - 아톰 에디터에서 파란색 부분
  - print() : 괄호 안의 것을 프롬프트에서 보여주라는 “함수”

- **함수 만들어보기**

```
>>> def plus_thirty(x):
...     y = x+30
...     return y
...
>>> plus_thirty(20)
50
```

  - 헷갈리는 이유 : 우리는 y=x+30 정도면 함수라고 하는데, 파이썬 함수는 이름을 붙여줘야 함
  - plus_twenty(20)을 해보자
  - plust_thirty를 이용하여 a라는 변수에 저장해보자, 그리고 print() 함수로 출력하자

- **함수의 구성 요소**
  - def : 필수 / define의 약자
  - 함수 이름() : 필수 / 괄호 안에는 인자가 들어가도 되고 안 들어가도 되지만, 들어가는 게 일반적
  - return : 선택 / 들어가는 게 일반적 / return 되는 게 있을 경우 변수에 저장 가능

9. 프로그래밍

- **꼭 필요한 문법 설명(2) : import**
  - 남이 만든 모듈을 불러와서 내 것처럼 사용하기

```
1  from django.shortcuts import render
2
3  # Create your views here.
4
5  def list(request):
6      return render(request, 'list.html')
```

  - 장고 개발자들이 django폴더의 shortcuts.py에 render 라는 함수를 만들어 둔 것임.
  - 찾아보기

- **import 연습**
  - import random
  - random.randint(1,100)

  - from random import randint
  - randint(1,100)

9. 프로그래밍

- **꼭 필요한 문법 설명(3) : 선언**
  - 프로그래밍 언어에서 =은 비교가 아니라, 선언.
  - a = 50 → 이제부터 a를 50과 같은 것으로 취급하겠다.
  - b = ‘blog’→ 문자를 선언할 때는 따옴표 혹은 쌍따옴표 필수

  - 변수

  - 비교는 ==
  - 50==49
  - a==49
  - a==50

9. 프로그래밍

- **프로그램을 만드는 원리**
  - 위에서 아래로 읽는다.

```
10  print(1)
11  print(2)
12  print(3)
13  print(4)
14  print(5)
```

```
(base) C:₩Users₩송규호₩pyworks>python plus.py
1
2
3
4
5
```

  - 단, 함수는 호출되기 전까지는 작동하지 않는다. → abccc.py 실행해보기
  - 함수는 호출 전에 정의되어 있어야 한다. 적어도 import는 되어 있어야 한다.

```
1   def calculate(a,b):
2       c = a*b+a/b-a**b+b**a
3       return c
4
5   def calculate2(a,b):
6       c = a*b+a*2000
7       return c
8
9
10  print(1)
11  print(2)
12  print(3)
13  print(4)
14  print(5)
15  print(calculate(2,1))
```

```
(base) C:₩Users₩송규호₩pyworks>python abccc.py
1
2
3
4
5
3.0
```

9. 프로그래밍

- **프로그램을 만드는 원리**
  - 재사용성

I. 함수를 만들지 않고 한 프로그램에서 바로 처리

abccc.py

```
1  a = 1*2+1/2-1**2+2**1
2  print(a)
```

– 한 프로그램 내에서도 다시 이용 안 할 거다~ 싶은 코드 (재사용성 X)

II. 함수를 만들고 그 프로그램에서 바로 처리

– 이 프로그램 내에서는 지속적으로 사용할 것 같은 코드 (재사용성 △)

abccc.py

```
1  def calculate(a,b):
2      c = a*b+a/b-a**b+b**a
3      return c
4
5  print(calculate(1,2))
6  print(calculate(2,3))
7  print(calculate(3,4))
8  print(calculate(4,5))
9  print(calculate(5,6))
10 print(calculate(6,7))
11 print(calculate(7,8))
12 print(calculate(8,9))
13 print(calculate(9,10))
```

abccc.py

```
1  def calculate(a,b):
2      c = a*b+a/b-a**b+b**a
3      return c
4
5  def calculate2(a,b):
6      c = a*b+a*2000
7      return c
8
9
10
11 print(calculate(1,2))
12 print(calculate(2,3))
13
14 print(calculate2(1,2))
15 print(calculate2(2,3))
```

- **프로그램을 만드는 원리**
  - 재사용성

III. 함수를 만들어 모듈화하고 다른 프로그램에서 import

abccc.py

```
1  def calculate(a,b):
2      c = a*b+a/b-a**b+b**a
3      return c
4
5  def calculate2(a,b):
6      c = a*b+a*2000
7      return c
```

- 다른 목적의 프로그램을 만들 때도 사용할 것 같다 싶은 코드 (재사용성 ○ 범용성 ○)
- 다음번엔 calculate 함수를 만들지 않고도, import 만으로도 프로그램 구성 가능
- 자주 사용할수록 생산성 증대!

plus.py

```
1  from abccc import calculate, calculate2
2
3  print(calculate(1,2))
4  print(calculate(2,3))
5
6  print(calculate2(1,2))
7  print(calculate2(2,3))
```

9. 프로그래밍

- **프로그램을 만드는 원리**
  - 함수 안의 함수

plus.py

```
1  from abccc import calculate, calculate2
2
3  print(calculate2(calculate(1,2), calculate(1,2)))
```

```
(base) C:₩Users₩송규호₩pyworks>python plus.py
7012.25
```

  - 숫자 바꿔보기

plus.py

```
1  from abccc import calculate, calculate2
2
3  print(calculate2(calculate(1,2), calculate(10,5)))
```

9. 프로그래밍

- **프로그램을 만드는 원리**
  - 함수 안의 함수

plus.py

```
1  from abccc import calculate, calculate2
2
3  print(calculate2(calculate(1,2), calculate(10,5)))
```

  - 복잡하니까 하나하나 차근차근 변수로 선언하기

plus.py

```
1  from abccc import calculate, calculate2
2
3  a = calculate(1,2)
4  b = calculate(10,5)
5  c = calculate2(a,b)
6
7  print(c)
```

  - 결과는 똑같다

```
(base) C:₩Users₩송규호₩pyworks>python plus.py
33836869.5
```

9. 프로그래밍

- **함수가 좋은 이유**
  - 그 함수가 얼마나 복잡하게 설계되었든 상관 없이, 함수 사용법만 알면 되기 때문

  - randint 사용법을 잊어버림.
  - randint(100) → 오류 발생
    1) 구글링 : How to use randint / randint document / how to get random number in python
    2) 오류내역 복붙 : randint() missing 1 required positional argument: 'b'
    3) 직접 파이썬 파일 탐구

```
import random
for x in range(10):
  print random.randint(1,101)
```

```
217     def randint(self, a, b):
218         """Return random integer in range [a, b], including both end points.
219         """
220
221         return self.randrange(a, b+1)
```

5 results found for 'randint'

randint

10 장고 이해하기

10. 장고 이해하기

- **장고 코드 최소한의 이해**
  - ycc/urls.py

urls.py

```
1
2  from django.contrib import admin
3  from django.urls import path, include
4
5  urlpatterns = [
6      path('admin/', admin.site.urls),
7      path('blog/', include('blog.urls')),
8  ]
```

  - ‘blog/’ → ‘ycc/’

10. 장고 이해하기

- **장고 코드 최소한의 이해**
  - blog/urls.py

urls.py — ycc | urls.py — blog

```python
from django.urls import path
from . import views

app_name = 'blog'

urlpatterns = [
    path('list/', views.list),
    path('new/', views.new),
]
```

  - ‘list/’ → ‘ppp/’

10. 장고 이해하기

- **장고 코드 최소한의 이해**
  - blog/urls.py

urls.py — ycc | urls.py — blog

```python
from django.urls import path
from . import views

app_name = 'blog'

urlpatterns = [
    path('list/', views.list),
    path('new/', views.new),
]
```

  - views.list → views.qwe

```
  File "C:₩Users₩송규호₩pyworks₩django_ex₩ycc₩blog₩urls.py", line 7, in <module>
    path('list/', views.qwe),
AttributeError: module 'blog.views' has no attribute 'qwe'
```

  - 에러가 : blog.views 라는 모듈에 qwe 라는 게 없다 → 만들어주면 되지!
    → views.py 로 ㄱㄱ

10. 장고 이해하기

- **장고 코드 최소한의 이해**
  - blog/views.py

```python
from django.shortcuts import render

# Create your views here.

def list(request):
    return render(request, 'list.html')
```

  - def list → def qwe

```python
def qwe(request):
```

  - ‘list.html’ → ‘iu.html’
  - ycc/list.html → ycc/iu.html

10. 장고 이해하기

- **복습 문제**
  1) 상위 ycc 폴더로 이동하세요.
  2) Community라는 이름의 app을 만드세요.
  3) 127.0.0.1:8000/community/write/ 라는 url을 만드세요.
  4) 위 url을 입력했을 때 “hello naver” 라는 글자가 쓰여 있는 hello.html을 만들고 연결하세요.

- **응용 문제**
  5) hello naver를 클릭했을 때, 네이버 홈으로 이동하는 링크를 거세요.
  6) 저번에 만들었던 list.html을 살짝 변경하여, ‘good!!’를 클릭하면 ‘127.0.0.1:8000/community/write/’ 주소로 이동하는 링크를 거세요.

  ※ 1주차에 했던 1장 〈HTML 이해〉를 참고

10. 장고 이해하기

- **복습 문제 답**

1) community 라는 이름의 앱 만들기

Anaconda Prompt

```
(base) C:₩Users₩송규호₩pyworks₩django_ex₩ycc3>django-admin startapp community
```

2) settings.py 에서 앱 등록하기

settings.py

```
27
28  ALLOWED_HOSTS = []
29
30
31  # Application definition
32
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      'blog',
41      'community',
42  ]
```

3) urls.py에서, 'community/' 라는 url 만들기

```
16  from django.contrib import admin
17  from django.urls import path, include
18
19  urlpatterns = [
20      path('admin/', admin.site.urls),
21      path('blog/', include('blog.urls')),
22      path('community/', include('community.urls')),
23  ]
```

10. 장고 이해하기

- **복습 문제 답**

4) community/urls.py 새로 만들고 기존 url에 'write/' 추가하기

urls.py

```python
from django.urls import path
from . import views

app_name = 'community'

urlpatterns = [
    path('write/', views.write),
]
```

5) views.py에서 html과 연결하기

views.py

```python
from django.shortcuts import render

# Create your views here.

def write(request):
    return render(request, 'hello.html')

```

6) 최상위 ycc 폴더에 hello.html 만들기

hello.html

```
hello naver
```

10. 장고 이해하기

- **응용 문제 답**

7) ycc/list.html에서 〈a〉〈/a〉태그 추가하기

list.html

```
1   <a href = "http://127.0.0.1:8000/community/write">good!!</a>
```

8) ycc/hello.html 에서 〈a〉〈/a〉태그 추가하기

hello.html

```
1   <a href = "https://www.naver.com">hello naver</a>
```

10. 장고 이해하기

- **의문**
  - http://127.0.0.1:8000/blog/list/
  - http://127.0.0.1:8000/community/write/
  - 왜 한꺼번에 url 이름이 두 개나 바뀔까? 너무 비효율적인 것 아닌가?

- **맞습니다**
  - 마치 네이버 지식인 홈 → 네이버 웹툰 홈으로 링크가 걸린 격.
  - 네이버 지식인 홈 → 네이버 지식인 글쓰기로 링크가 걸리는 게 일반적.

- **일반적인 방법**
  - http://127.0.0.1:8000/blog/list/
  - http://127.0.0.1:8000/blog/new/
  - http://127.0.0.1:8000/blog/edit/
  - 한 앱에서의 연결성이 좋아야 한다.
  - 방금 전에 만든 community는 머리에서 싹 잊고, blog 앱에만 집중해 봅시다.

10. 장고 이해하기

- **자동으로 url 만들기**
  - blog/urls.py 에서 'post/<int:pk>/' url 만들기

urls.py

```
1  from django.urls import path
2  from . import views
3
4  app_name = 'blog'
5
6  urlpatterns = [
7      path('list/', views.list),
8      path('post/<int:pk>/', views.post),
9  ]
```

  - views.py 에서 post 함수 만들기 → post.html 만들고 아무 말이나 쓰기

views.py

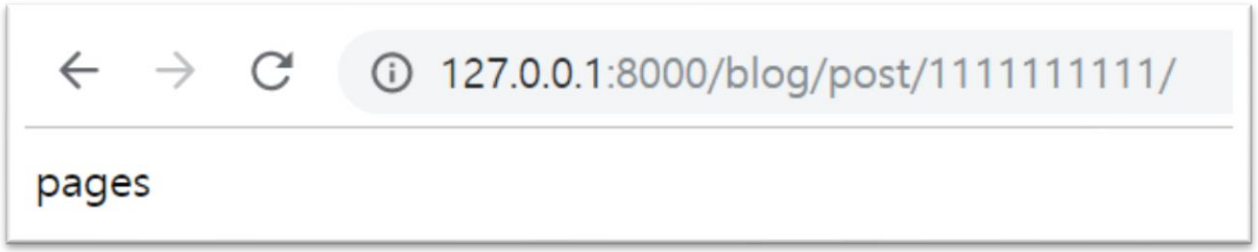
```
1  from django.shortcuts import render
2
3  # Create your views here.
4
5  def list(request):
6      return render(request, 'list.html')
7
8  def post(request, pk):
9      return render(request, 'post.html')
```

post.html

```
1  pages
```

10. 장고 이해하기

- **자동으로 url 만들기**
  - 127.0.0.1:8000/blog/post/숫자/



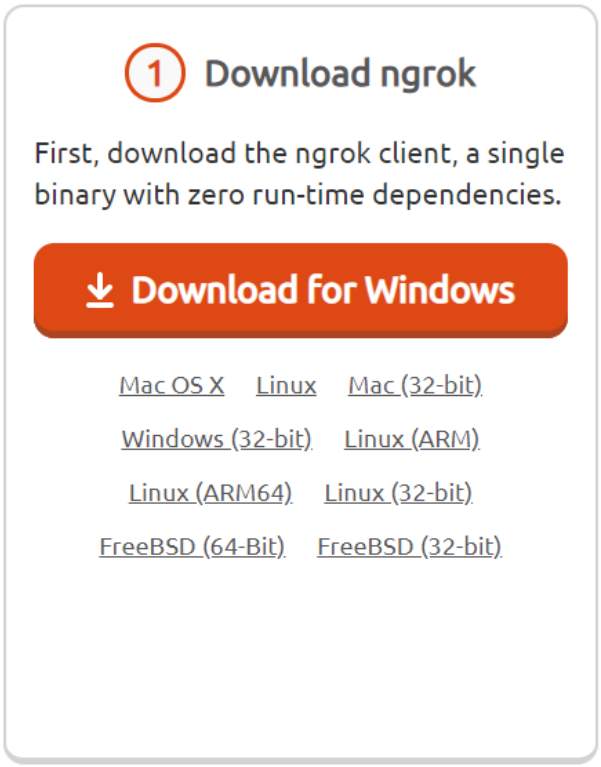
  - 127.0.0.1:8000/blog/post/문자/

  - 〈int:pk〉 → 〈pk〉

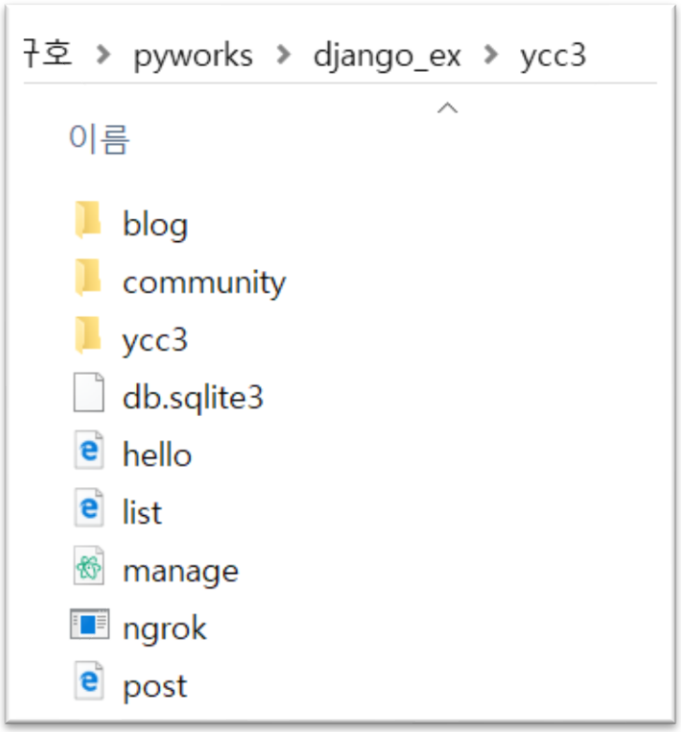
11 Ngrok 서버

11. NGROK 서버

- Ngrok 다운로드
  - https://ngrok.com/download → 다운로드 → 압축풀기



  - manage.py 가 있는 폴더로 ngrok.exe 옮기기



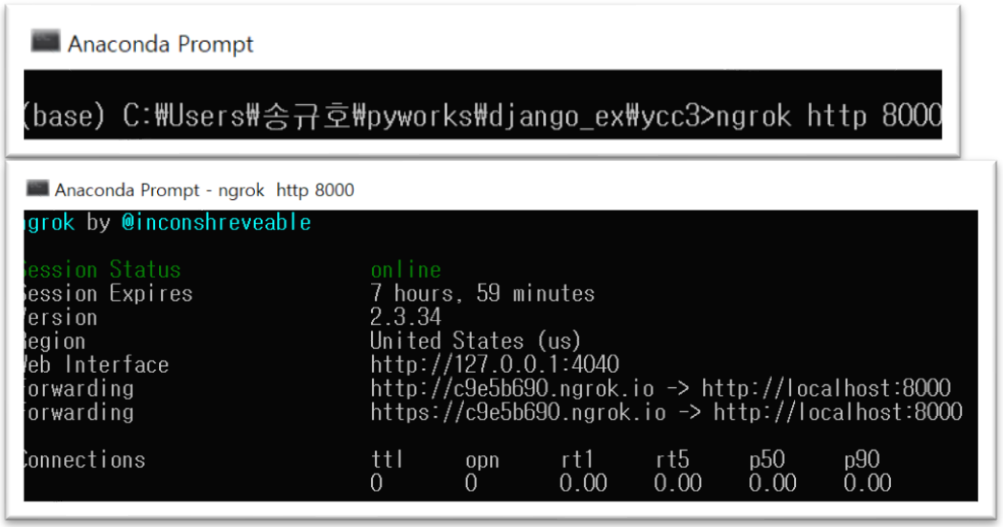
11. NGROK 서버

- **settings.py**
  - ALLOWED_HOSTS = ['*'] // * : 모든 호스트를 다 인정하겠다는 뜻

settings.py

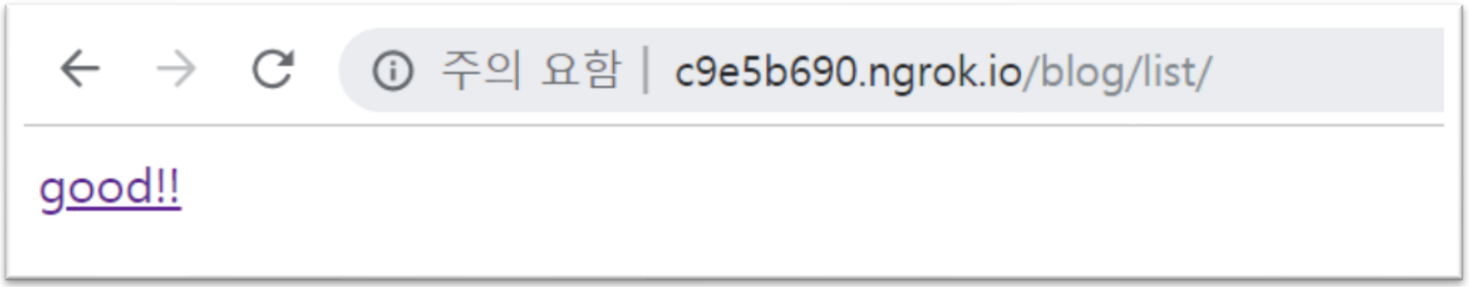
```
27
28  ALLOWED_HOSTS = ['*']
```

- **Ngrok 실행**
  - 아나콘다 프롬프트 하나 더 실행 → ngrok.exe가 있는 폴더로 이동
  - ngrok http 8000



11. NGROK 서버

- http://c9e5b690.ngrok.io/blog/list



- **스마트폰에서 들어가보기!!**
  - 주소복사 → 카톡 자기에게 보내기 → 스마트폰 카톡으로 링크 클릭

- **외부에서 접속할 수 있는 길을 임시로 뚫어줍니다.**

여기까지 튜토리얼이었고요

- 이제부터 진짜입니다.
- 지금까지 한 건, 진짜 기계적인 내용. 더 응용할 것도 없어요.
- 여러 프로젝트를 만들다 보면 100번은 반복합니다
- 당연히 익숙해진다~

- 다음주부터는 아마도 좀 어렵다.
- 시험기간은 언제부터…?