Python: Basic Syntax

Alan Wang

17.3 Multitier Application Architecture

- Web-based applications are often multitier applications that divide functionality into separate tiers. Although tiers can be located on the same computer, the tiers of web-based applications typically reside on separate computers.
- The bottom tier (also called the data tier or the information tier) maintains the application's data.
- The middle tier implements business logic, controller logic and presentation logic to control interactions between the application's clients and its data.
  - Business logic in the middle tier enforces business rules and ensures that data is reliable before the server application updates the database or presents the data to users. Business rules dictate how clients can and cannot access application data, and how applications process data.
- The top tier, or client tier, is the application's user interface. In response to user actions, the client tier interacts with the middle tier to make requests and to retrieve data from the information tier. The client tier then displays the data retrieved for the user. The client tier never directly interacts with the information tier.

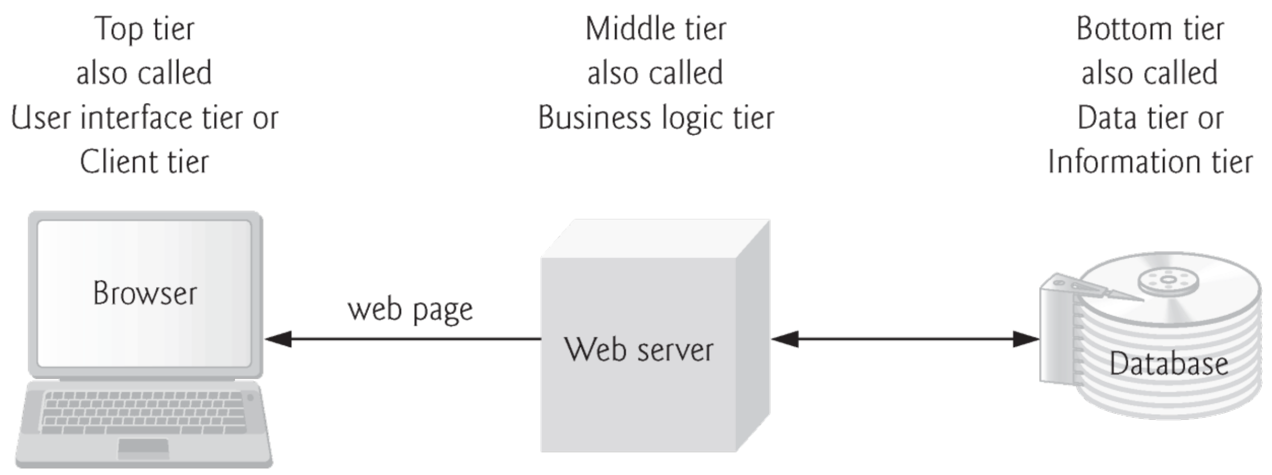


Fig. 17.3 | Three-tier architecture.

17.3 Multitier Application Architecture

- Top tier/client tier/presentation tier
  - Static or dynamically generated content rendered by the browser (**front-end**), e.g., HTML, CSS, JavaScript. jQuery
- Logic tier
  - An application server for dynamic content processing and generation (**middleware**), e.g, Python, PHP, Java EE, ASP.NET
- Data tier
  - A DBMS that manages and provides access to the data (**back-end**), e.g., MySql, Sql Server, Oracle, DB2

17.4 Client-Side Scripting versus Server-Side Scripting

- Client-side scripting can
  - validate user input
  - interact with the browser,
  - enhance web pages
  - add client/server communication between a browser and a web server.
- Limitations of client-side scripting
  - The browser or scripting host must support the scripting language and capabilities
  - Sensitive information, such as passwords or other personally identifiable data, cannot be stored or validated on the client
  - Placing large amounts of JavaScript on the client can cause security issues

17.4 Client-Side Scripting versus Server-Side Scripting

- Server-side scripting languages have a wider range of programmatic capabilities than their client-side equivalents
- Limitations of server-side scripting
  - Increased demand for computing resources on the server
  - Increased network traffic
  - Dependent on a network connection

17.5 Accessing Web Servers

- To request documents from web servers, users must know the hostnames on which the web server software resides.
- Users can request documents from local web servers or remote web servers.
- Local web servers can be accessed through the name `localhost`—a hostname that references the local machine and normally translates to the IP address `127.0.0.1` (also known as the **loopback address**)
- A remote web server can be accessed by a domain name (which is translated to an IP address by Domain Name Servers) or an IP address

17.6.2 Running MAMP

- MAMP bundles the Apache Web server, MySql database server, and PHP into a single package
- Make sure to change the port numbers to: Web Server HTTP (80), and MySQL Database Server (3306). If there is any port conflicts, you can change to another port number (e.g., web server being 8080 and database server being 3307).

Introduction to Python

Python

- Python is a popular general-purpose programming language
- Dynamically typed and garbage-collected
- Supports structured (or procedural), object-oriented, and functional programming
- Has great extensibility with thousands of modules supporting a wide range of functions
- Uses whitespace indentation, rather than {} or keywords to delimit code blocks

Python Server-side Web Development

- Pros:
  - Readability and ease of use
  - Extensible libraries and frameworks including the use of machine learning and AI
  - Increased productivity
  - Large community and support

- Cons:
  - Slow in performance as an interpretive language
  - Lack of parallel execution that impairs scalability
  - High memory consumption
  - Not suitable for client-side scripting

Comments

- Single-line comments
  - Begin with

    #
- Multi-line comments are not officially allowed

Python Language References

- https://www.w3schools.com/python/default.asp
- http://harrywang.me/python (Created by Dr. Harry Wang)

Variables

Variable naming rules:

1. It can be only one word.
2. It can use only letters, numbers, and the underscore (`_`) character.
3. It can't begin with a number.

Example:

```
>>> first_name = 'Harry'
>>> first_name
'Harry'
```

Global Variables

- Global variables are the variables created outside of a function
- Given a variable created inside a function with the same name as a global variable, this variable will be local, and can only be used inside the function.
- Use the keyword, global, to declare a global variable inside a function

```
x = "awesome"

def myfunc():
  x = "fantastic"
  print("Python is " + x)

myfunc()  # Python is fantastic

print("Python is " + x) # Python is awesome
```

```
x = "awesome"

def myfunc():
  global x     # value assignment is not allowed
  x = "fantastic"
  print("Python is " + x)

myfunc()       # Python is fantastic

print("Python is " + x) # Python is fantastic
```

Data Types

Data Type	Examples
Integers	`-2, -1, 0, 1, 2, 3, 4, 5`
Floating-point numbers	`-1.25, -1.0, --0.5, 0.0, 0.5, 1.0, 1.25`
Strings	`'a', 'aa', 'aaa', 'Hello!', '11 cats'`

Constants

- Constants are declared in the same way as variables
- The convention is to use all capital letters in a constant name, e.g., PI=3.14

Lamda Functions

This function:

```
>>> def add(x, y):
        return x + y

>>> add(5, 3)
8
```

Is equivalent to the *lambda* function:

```
>>> add = lambda x, y: x + y
>>> add(5, 3)
8
```

Strings

- Strings in Python are surrounded by either single quotation marks or double quotation marks.
- You can assign a multiline string to a variable by using three quotes

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
```

- Strings are arrays
- String length:
- Loop through a string
- Check for a substring

```
a = "Hello, World!"
print(a[1]). # e
```

```
for x in "banana":
  print(x)
```

```
a = "Hello, World!"
print(len(a)) # 13
```

```
a = "Hello, World!"
print(”World” in a) # True
```

Slicing Strings

spam = “Hello World!”

```
>>> spam[0:5]
'Hello'
```

```
>>> spam[:5]
'Hello'
```

```
>>> spam[6:]
'world!'
```

```
>>> spam[6:-1]
'world'
```

```
>>> spam[:-1]
'Hello world'
```

Other String Methods

- upper(), lower(), isdecimal(), isspace(), join(), split(), strip()
- String formatting

  f-strings (Python 3.6+)

```
>>> name = 'Stephen Curry'
>>> born = 1988
>>> print(f'{name} is born in {born}.')
Stephen Curry is born in 1988.
```

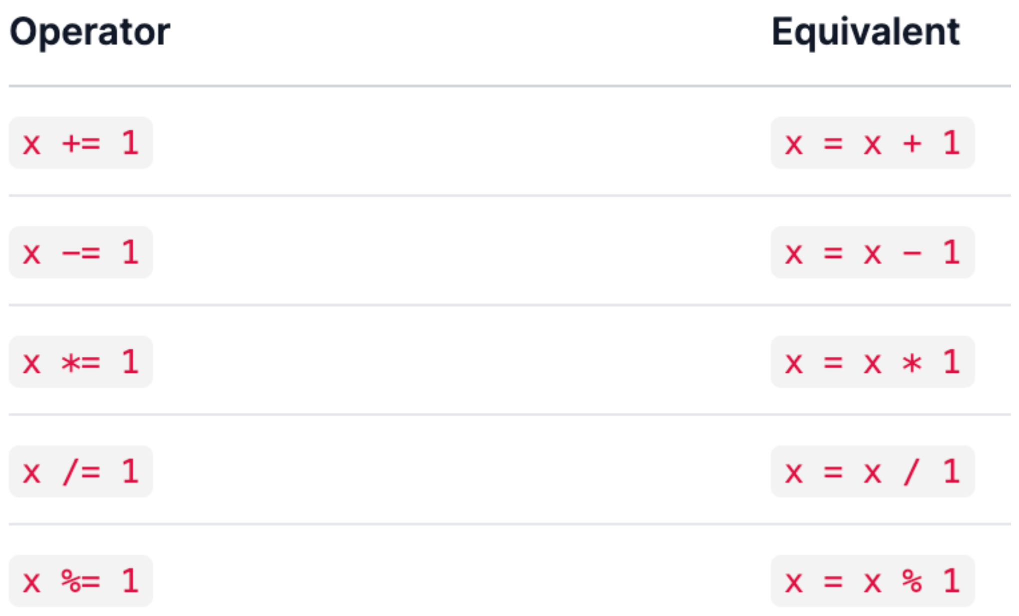
```
>>> pi = 3.1415926
>>> print(f'pi with two decimal places is {pi:.2f}')
pi with two decimal places is 3.14
```

Arithmetic Operators

From **Highest** to **Lowest** precedence:

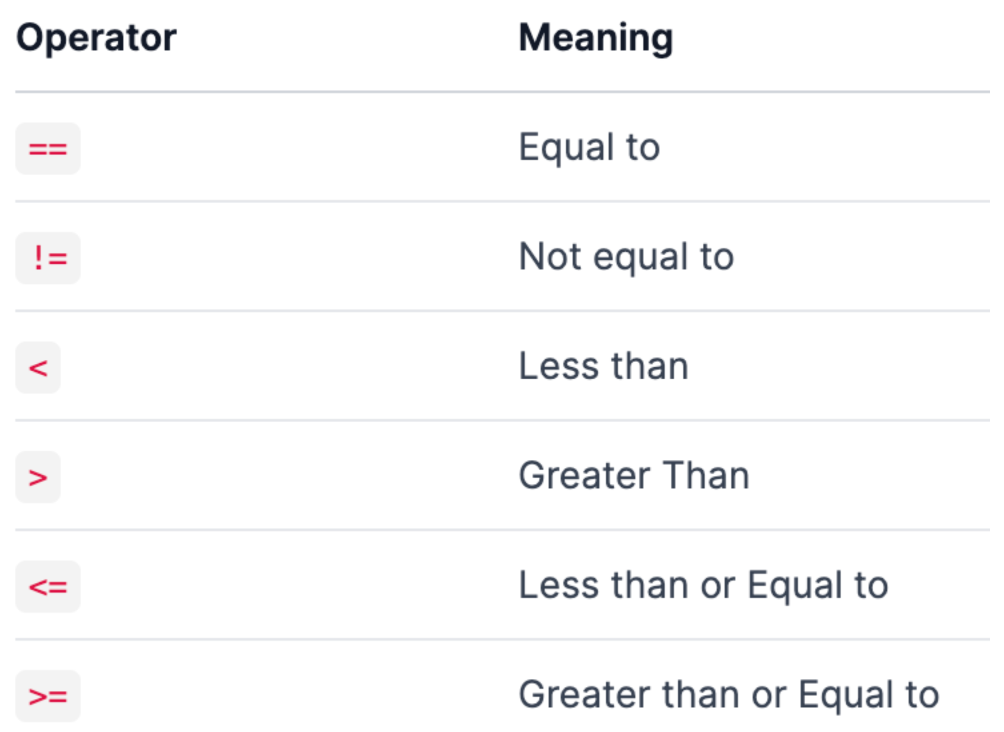
Operators	Operation	Example
**	Exponent	`2 ** 3 = 8`
%	Modulus/Remainder	`22 % 8 = 6`
//	Integer division	`22 // 8 = 2`
/	Division	`22 / 8 = 2.75`
*	Multiplication	`3 * 3 = 9`
-	Subtraction	`5 - 2 = 3`
+	Addition	`2 + 2 = 4`

Augmented Assignment Operators



Operator	Equivalent
`x += 1`	`x = x + 1`
`x -= 1`	`x = x - 1`
`x *= 1`	`x = x * 1`
`x /= 1`	`x = x / 1`
`x %= 1`	`x = x % 1`

Comparison Operators



Operator	Meaning
`==`	Equal to
`!=`	Not equal to
`<`	Less than
`>`	Greater Than
`<=`	Less than or Equal to
`>=`	Greater than or Equal to

Logical Operators

Operator	Name	Example
not	NOT	not b
and	AND	a and b
or	OR	a or b

The is operator checks for object identity. In other words, x is y evaluates to True only when x and y evaluate to the same object. The is operator has an opposite, the is not operator.

```
>>> x = []
>>> y = []
>>> x is x
True
>>> x is not x
False
>>> x is y
False
>>> x is not y
True
```

Reference: realpython.com

Control Structures

`if...elif...else`

```
credit_score = 600
student = 'no'
if credit_score >= 700:
    print('card approved')
elif student == 'yes':
    print('student card approved')
else:
    print('application declined')
```

Looping Statements

`break` Statements

If the execution reaches a `break` statement, it immediately exits the while loop's clause:

`continue` Statements

When the program execution reaches a continue statement, the program execution immediately jumps back to the start of the loop.

`for` Loops and the `range()` Function

```
>>> for i in range(0, 10, 2):
>>>     print(i)
0
2
4
6
8
```

while Loop Statements

```
a = 0
while a < 5:
    print('Hello, world.')
    a = a + 1
```

Lists, Dictionaries, and Sets

List comprehension

```
>>> a = [1, 3, 5, 7, 9, 11]

>>> [i - 1 for i in a]
[0, 2, 4, 6, 8, 10]
```

Dict comprehension

```
>>> c = {'name': 'Pooka', 'age': 5}
>>> {v: k for k, v in c.items()}
{'Pooka': 'name', 5: 'age'}
```

Set comprehension

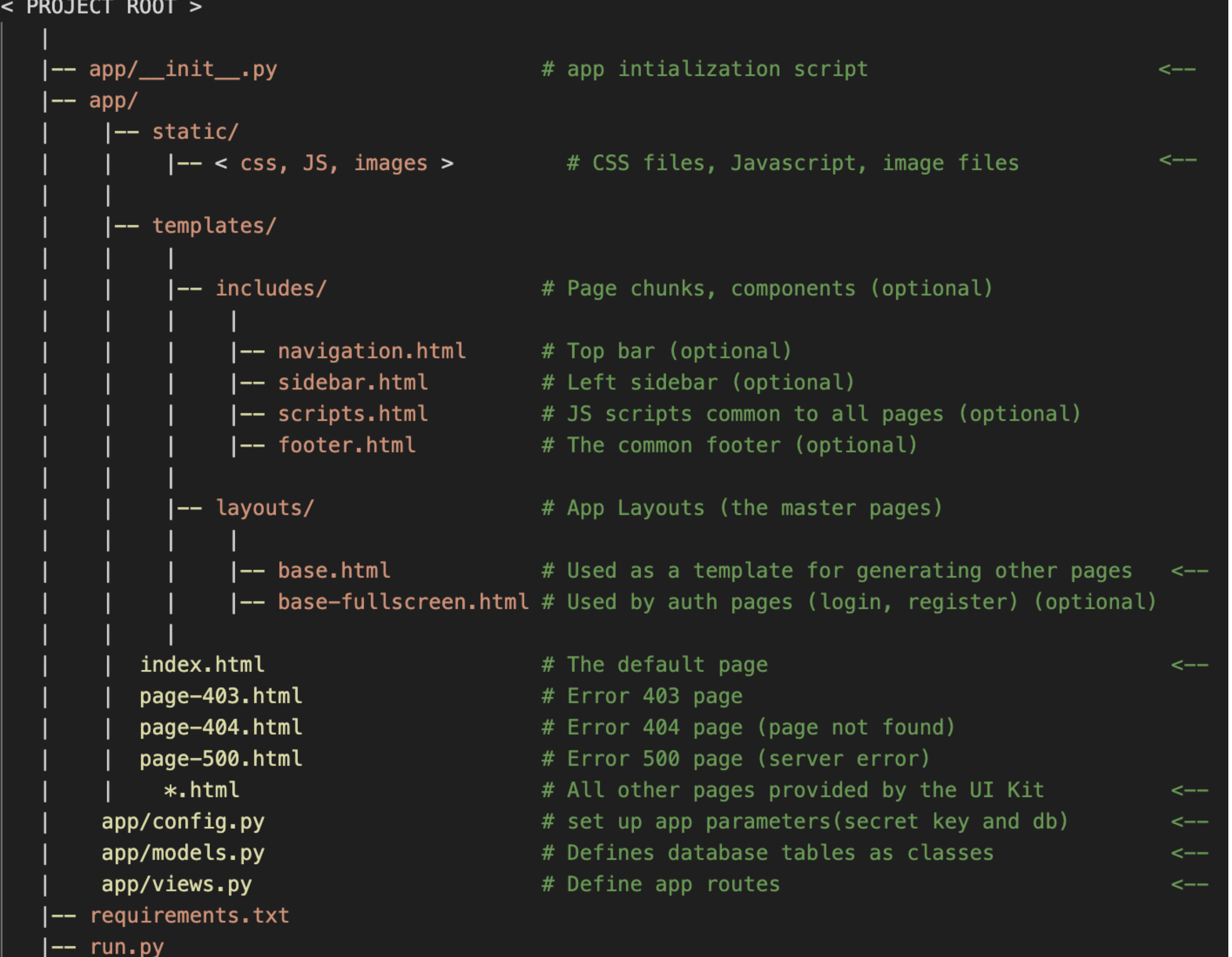
```
>>> b = {"abc", "def"}
>>> {s.upper() for s in b}
{"ABC", "DEF"}
```

Python Web Development

Python Web Development Frameworks

- Django, Flask, and FastAPI among others
- Compared to other alternatives, Flask is simple, flexible, and easy to learn

A Flask Project Structure



```
< PROJECT ROOT >
   |
   |-- app/__init__.py                   # app intialization script                        <--
   |-- app/
   |    |-- static/
   |    |    |-- < css, JS, images >       # CSS files, Javascript, image files            <--
   |    |
   |    |-- templates/
   |    |    |
   |    |    |-- includes/               # Page chunks, components (optional)
   |    |    |    |
   |    |    |    |-- navigation.html    # Top bar (optional)
   |    |    |    |-- sidebar.html       # Left sidebar (optional)
   |    |    |    |-- scripts.html       # JS scripts common to all pages (optional)
   |    |    |    |-- footer.html        # The common footer (optional)
   |    |    |
   |    |    |-- layouts/                # App Layouts (the master pages)
   |    |    |    |
   |    |    |    |-- base.html          # Used as a template for generating other pages   <--
   |    |    |    |-- base-fullscreen.html # Used by auth pages (login, register) (optional)
   |    |    |
   |    |  index.html                    # The default page                                <--
   |    |  page-403.html                 # Error 403 page
   |    |  page-404.html                 # Error 404 page (page not found)
   |    |  page-500.html                 # Error 500 page (server error)
   |    |    *.html                      # All other pages provided by the UI Kit          <--
   |   app/config.py                     # set up app parameters(secret key and db)        <--
   |   app/models.py                     # Defines database tables as classes              <--
   |   app/views.py                      # Define app routes                               <--
   |-- requirements.txt
   |-- run.py
```

Steps

1. Download the Flask Project Template from Canvas and extract the files to a local folder (e.g., MISY350);
2. Install the required Python version and libraries:

   Inside the local Flask project folder, type the following:

```
python –version        #check if python is installed
pip install flask_sqlalchemy
                          #this package must be installed globally
 virtualenv env
              #if command not found, run “pip install virtualenv”
```

Steps (cont'd)

```
(mac) source env/bin/activate
(Windows) .\env\Scripts\activate
pip install -r requirements.txt
(Mac) export FLASK_APP=run.py
(Windows) set FLASK_APP=run.py
(Mac) export FLASK_ENV=development
(Windows) set FLASK_ENV=development
flask run --host=0.0.0.0 --port=5000
```

3. Open a browser and browse to http://localhost:5000

Frontend Development

- Adopt a Bootstrap template
  - Startbootstrap.com (uncheck vue, angular, and pro)
  - Bootstrapmade.com (free download)
  - Mdbootstrap.com
  - Create your own using a software (layoutit.com/build or Bootstrap Studio)
- Open index.html, identify the main content element (M) and the rest of the page structure (R)
- Replace the elements in app/templates/layouts/base.html with R following the example

Frontend Development (Cont'd)

- Replace the elements in app/templates/index.html with M following the example
- Copy the assets (css, js, images) to app/static and update the references to them in base.html and index.html

Backend Development

- Update the database server information (e.g., database server hostname, port number, database name) in app/config.py
- Define a new class for each database table in app/models.py following the example
- Define website routes in app/views.py