IAB207 – Rapid Web Application Development

2019 S2

Workshop 04

Python Recap

Tutor: Name

Tutor Email

QUT Science and Engineering

WORKSHOP

a university for the real world®

Agenda

- Introduction
- Outcomes
- Last Week Recap
- Exercise 1
- Exercise 2

Workshop Participation

- http://bit.ly/2TJPhLF

Introduction

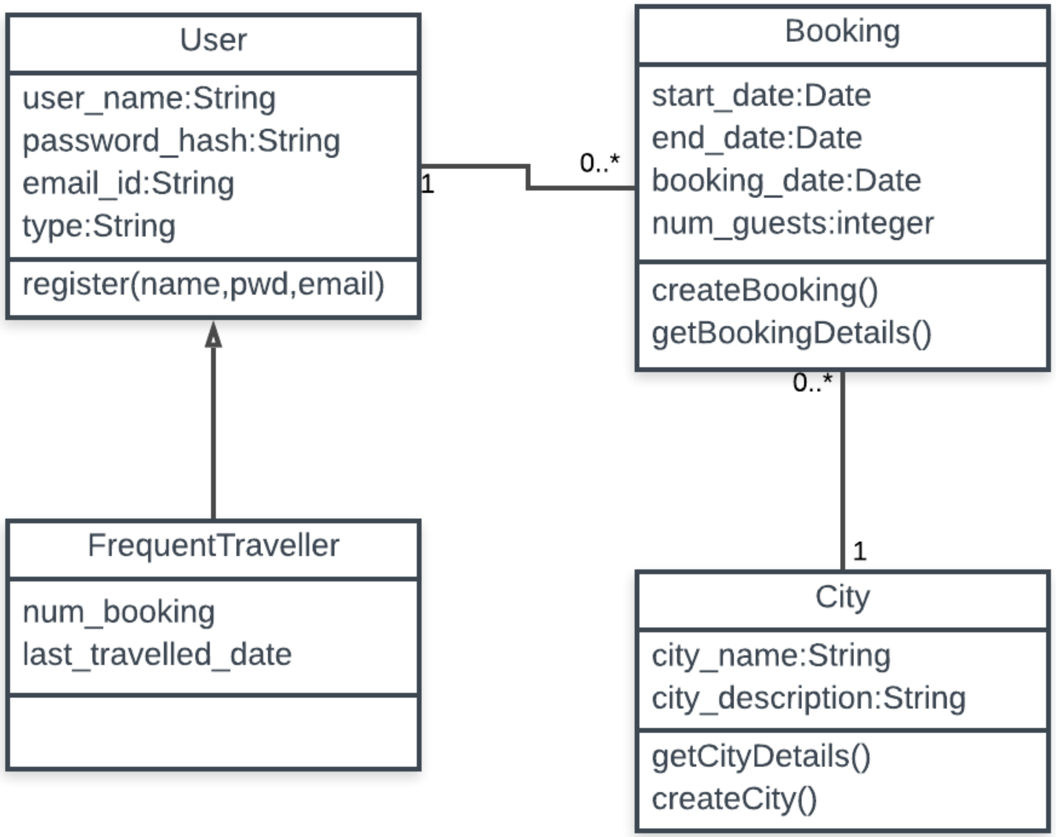
- Create a python project and a module
- Use existing python modules (datetime functions)
- Implement a set of classes with relationships
- Create a package of all classes and use the package

Outcomes

- Learn concepts related to object oriented programming – limited to help understand and use the Flask framework
- Instantiate/Create and Use classes
  - Use methods of Classes
- Creating package and modularize code

Exercise 1 (40 minutes)

- Create the following four classes in Python



Visual Studio Tutorial (10 mins)

- Refer to the video as you write your code in Visual Studio Code Editor
  - Creating files/folder
  - Running python code
  - Adding Breakpoints
  - Viewing variable

- https://www.lynda.com/Visual-Studio-tutorials/Visual-Studio-Code-Python-Developers/784291-2.html

Folder Structure (Recommended)

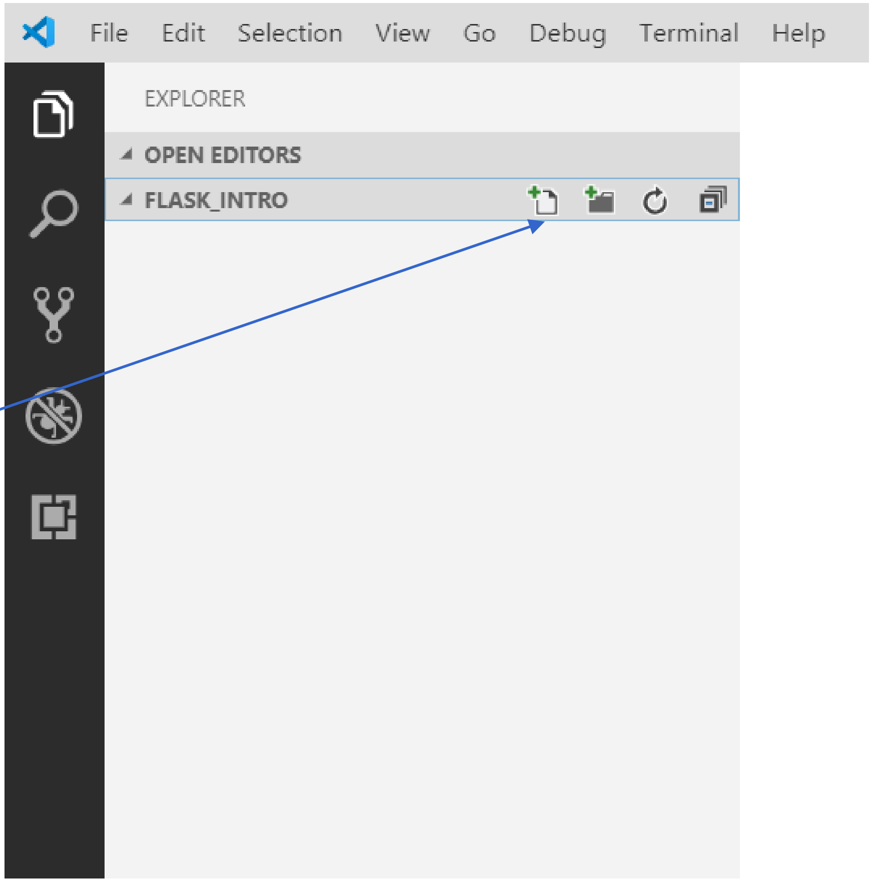
- Folder Setup
  - Create folder location on your system for IAB207
  - Create “workshop” folder
  - Create “testing” folder

Create a VS Code Project

- Open Visual Studio Code
- File-> Open Folder-> Browse Directory-'New Folder'-> week4/
  - Browse to the workshop directory

- New File travel/user.py
- This creates a folder travel and create a file user.py

- Check the Python interpreter chosen by the IDE at the bottom left corner.



Create a User Class (travel/user.py)

1. Use keyword **class**

2. Create an __init__ function that does not have any parameters
   Assign and data variable type='guest'

3. Create a register function that takes username, password and emailID.

4. Create an __repr__ function that prints all values

5. Refer to the code https://git.io/fjdrX

Create a City Class (travel/city.py)

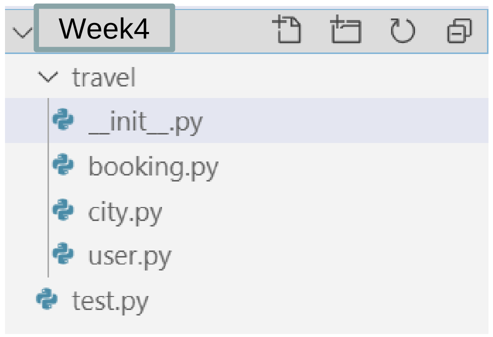
1. Use keyword **class**
2. Create an __init__ function that takes the name of the city and description.
3. Create a method that returns the details of the city
4. Create an __repr__ function that prints all values
5. https://git.io/fjdr1 (code for reference)

Create a Booking Class (travel/booking.py)

- Use keyword **class,** import datetime module
- Create an __init__ function that takes parameters (start_date, end_date,city, user)
  - Assign the num_guest=1
  - Hint: Need to import the User class as user passed as input parameter is an instance of User class
- Create an __repr__ function that prints all values
- https://git.io/fjdrQ

Test your classes

- Create a test.py in the week4 folder
  - Your folder structure should be similar to the structure shown on the slide



```
Week4
  travel
    __init__.py
    booking.py
    city.py
    user.py
  test.py
```

- Since travel is a folder and needs to be made a **package**, create a file in travel folder __init__.py

- In test.py, import the User, City and Booking class
- Hint: from <package>.<module> import <ClassName>

Test your classes

- In test.py
- Instantiate a City, User and Booking
- Print the booking to check if you are able to print the attributes of city, user and booking
- https://git.io/fjdoT

Create a FrequentTraveller

- In the user.py, create another class FrequentTraveller

- Use keyword **class,** User as a parent/base class

- Create an __init__ function that takes parameters
  - Assign the guest_type='Frequent Traveller'

- Create a register_user function that takes username, password, emailID, travellerID
  – Call the super().register (to reuse base functions)
  – Set travellerID attribute

Working with String format function

- You can work with format function in Python

```
name = "John Smith"
points = 10
str= " Hello {}, You have won {} points today !!"
print (str.format(name, points))
```

- If you change the code, what happens and why?
  - Debug the code in VS Code

```
name = "John Smith"
points = 10
str= " Hello {}, You have won {} points today !!"
str.format(name, points)
print (str)
```

Working with Dates

- **Getting the difference between two dates**
- The timedelta object represents the difference between two dates or times. To compare the difference between two date or time objects, simply subtract them

```
from datetime import datetime,date
#datetime(year, month, day)
a = datetime(2018, 11, 28)
print(a)
# datetime(year, month, day, hour, minute, second, microsecond)
b = datetime(2017, 11, 28, 23, 55, 59, 342380)
print(b)

t1 = date(year = 2018, month = 7, day = 12)
t2 = date(year = 2017, month = 12, day = 23)
diff = t1 - t2
print("Difference =", diff)
```

Working with Dates

- **Formatting Dates**
- The strftime() method is defined under the classes date, datetime and time. This method creates a formatted string from a given date, datetime or time

```
# current date and time
now = datetime.now()
t = now.strftime("%H:%M:%S")
print("time:", t)
s1 = now.strftime("%m/%d/%Y, %H:%M:%S")
# mm/dd/YY H:M:S format ... e.g. 12/26/2018, 04:34:52
print("s1:", s1)
s2 = now.strftime("%d/%m/%Y, %H:%M:%S")
# dd/mm/YY H:M:S format ... e.g. 26/12/2018, 04:34:52
print("s2:", s2)
```

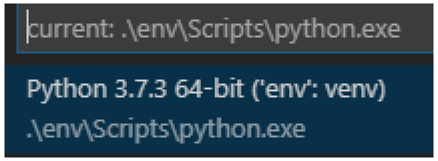
Code walk through and Questions

Install Flask

- Folder Setup (Recommended)
  - Under your workshop folder, Create “testing” folder
- In Visual studio code, ‘testing’ folder
  - Create app.py file

Install Flask (cont…)

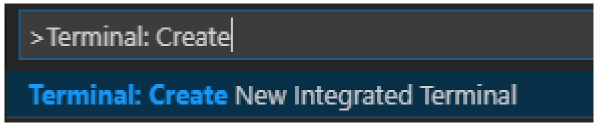
- Python Interpreter
  - Open Command Palette
    - View → Command Palette or (Ctrl+Shift+P)
    - Select **Python: Select Interpreter**



current: .\env\Scripts\python.exe

Python 3.7.3 64-bit ('env': venv)
.\env\Scripts\python.exe

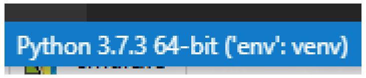
- Integrated Terminal
  - Open Command Palette
    - View → Command Palette
    - Select **Create New Integrated Terminal**



>Terminal: Create

Terminal: Create New Integrated Terminal

- Confirm environment



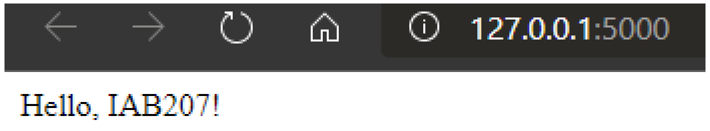
Python 3.7.3 64-bit ('env': venv)

Install Flask (cont…)

- Integrated Terminal
  - Run: **pip install flask**
  - Optional
    - Outdated pip version
      - Integrated Terminal
      - Run: **python -m pip install --upgrade pip**

Install Flask (cont…)

- Add code to “app.py”
  - https://gist.github.com/sriqut/bd51ae6f767da371df95ab6d9414d5c7
- Save file
- Integrated Terminal
  - Run: **python -m flask run**
  - Browse to: http://127.0.0.1:5000/ (default web server)
  - Ctrl+C to quit
- Congratulations! Flask is setup and working ☺



```
Hello, IAB207!
```

Thank you!