CS 235 Lab 1 report

Sections

Notes from Lab1

- **The benefits of using virtual environments to develop:**
    - Wo55vrking in distinct environments with their own distinct pacakages and python installation
    - Using virtual environments, we can use different versions of a package.
        - This is useful as major versions of software are not ussually backwards compatible.
    - We can generate a file with all the package requirements easier.
- **We learnt how to set up a virtual environment using conda- a package and virtual enviroment manager:**

> Creation of an enviroment:

```
conda create -n environmentName python=python_version
```

> Activation of an environment:

```
conda activate environmentName
```

> Install packages to a virtual enviroment

```
conda install -c channelName package1 package2
```

> Deactivation of an environment:

```
conda deactivate environmentName
```

- **Miscellaneous**
    - Git is a widely used version controll system.

        - Github is a hosting platform for repository for collabrative projects.
    - Markdown is a markup langauge most commonly used to write README.md files git up repositories.
    - The Breakpoint() function in python is useful for debugging; especially with the help of vs codes variable

---

Reflection

What I learnt:

- I learnt about how and why one would want to employ a virtual environment to develop code in.
- I learnt that conda is a system that allows the creation and management of such virtual environments.
    - It does this by partitioning a portion of the drive and reserving it for virtualization when the enviroment is active

---

Hands on tasks:

1. Create a virtual environment with Conda.

- Encountered an HTTP conncection failure after downloading conda and trying to set up a virtual environment.

```
Collecting package metadata (current_repodata.json): failed

CondaHTTPError: HTTP 000 CONNECTION FAILED for url <https://conda.ar
Elapsed: -
```

After some googling, I realized this was attributed to conda's installer not copying some dll files from the Scripts to the Library\bin folder.
After manually copying them over, I was able to create the environment as such:

```
PS C:\Users\rukun\Documents\University\Year2Semester 2\Compsci 230\Labs> conda create -n cs235 python=3.9 pip
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.12.0
  latest version: 4.13.0

Please update conda by running

    $ conda update -n base -c defaults conda
```

2. Installing a package

- I installed pyauto gui into the environment with no trouble:

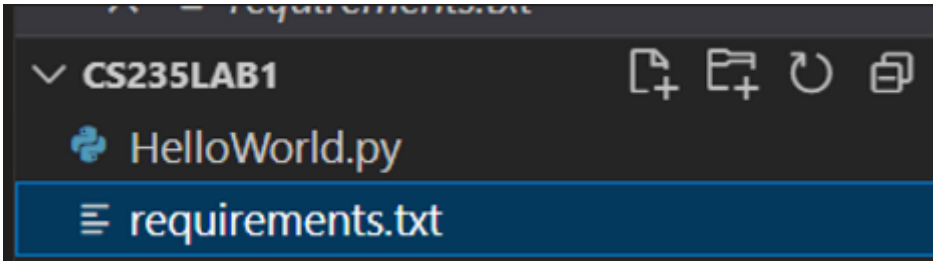
```
(cs235) C:\Users\rukun\Workspaces\Python\cs235Lab1>conda install -c conda-forge pyautogui
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

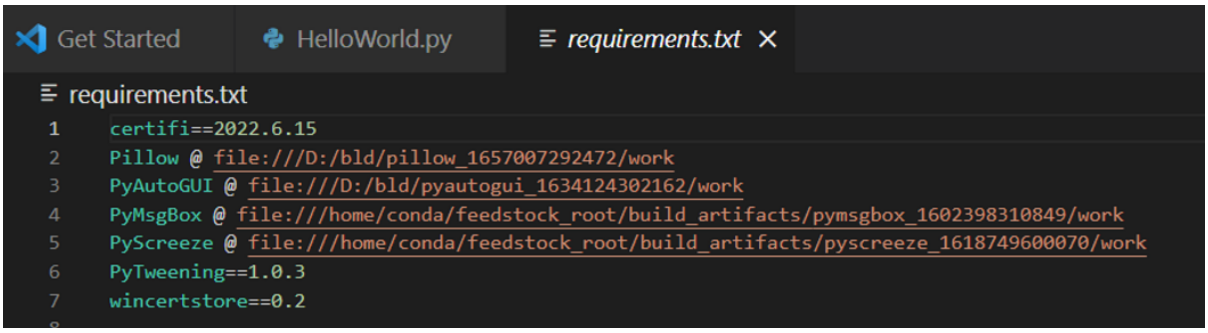
3. Generate requirements.txt file

- Used the command to generate my file with all the necessary packages:

```
pip freeze requirements.txt
```

- All the packages in the virual enviroment were added to the file.





4. Debugging questions and exercises:

- **Question 1:** What is the probability of winning a game?

  Answer:

```
def pickANumber():

return random.randint(1, 100)
```

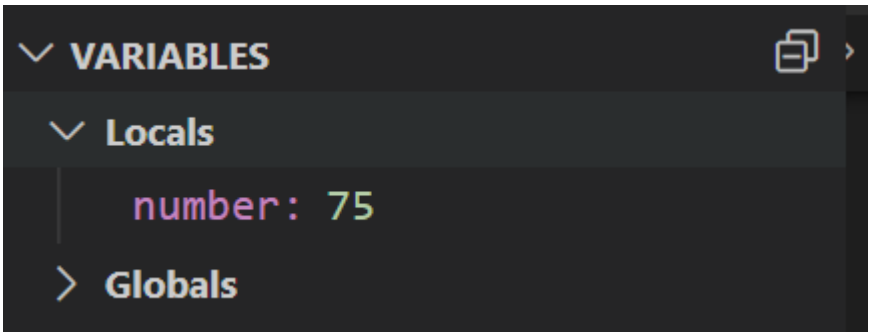
```
The probability of winning a game is 1/100 or 0.01 as the pickANumber
function, which generates a random number, chooses a number between 1
and 100. This would lead to a 100 possibilities with the chance of us
choosing the random number correctly being 1/100.
```

- **Question 2:** Using the debugging feature, WITHOUT changing any code, it is possible to win every game. What line did you insert the breakpoint at?

```
def computer_pick_number():
 print(PICK_NUMBER)
 number = pickANumber()
 breakpoint()
 return number
```

I added a breakpoint() right after the program generates its random number and before it is returned to the main game function.

By doing this I was able to view what value for returned to number and guess the same number to guarantee my win every time.



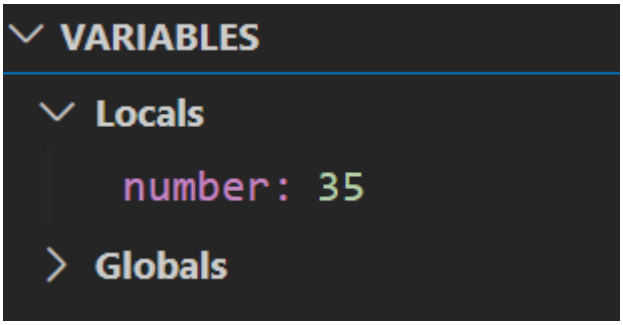

```
Enter a number between 1 and 100: 75
Wow, lucky guess!!
```

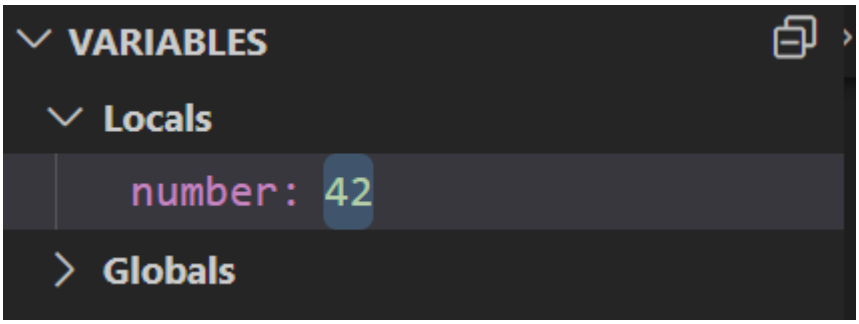
- **Question 3:** Using the debugging feature only, is it possible the user can win every game by guessing "42"?

  By using the same breakpoint():

```
def computer_pick_number():
print(PICK_NUMBER)
number = pickANumber()
breakpoint()
return number
```

I am able to edit the value of number before it is returned to the game function.






```
Let's play!
I'm thinking of a number... got it.
Now your turn to guess!
Enter a number between 1 and 100: 42
Wow, lucky guess!!
Would you like to play again? (Type 'y' for yes or  'n' for no.): n
Game over!
You won 1 out of 1 games.
That's improbable! Are you cheating??
```

```
I was able to overwrite the random number to 42 due the breakpoint()
stopping the code just before the number was returned.
Hence  we can always win by guessing 42, by overwriting the random
number we have to guess.
```

4. My Github Account
