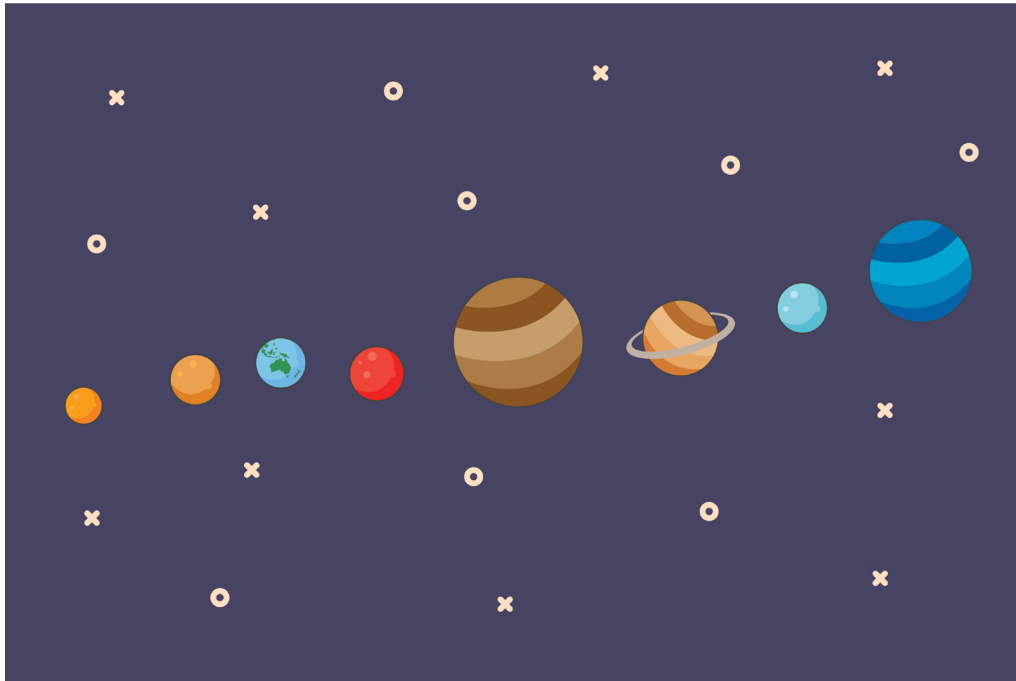


MOONHACK

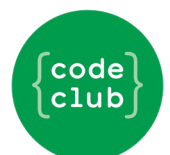
— MOONHACK 2018 PYTHON PROJECT —



INTRODUCTION

Earth isn't the only planet with a moon. Many planets have moons. In our solar system, some planets have a lot of moons (Jupiter has a whopping 63 moons!), some don't have very many, and some have none at all!

In this project, you will write a program that will display the names of every moon for any planet in our solar system.

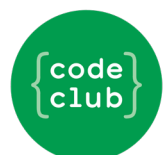
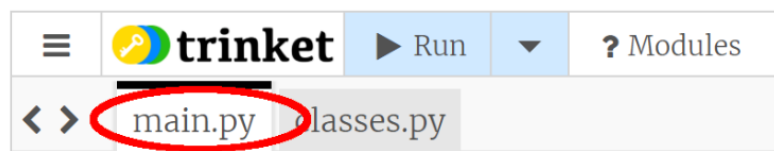


STEP 1: IMPORT THE STARTER CODE

Let's take a look at what has already been provided to us.

Activity Checklist

- Open the Moonhack 2018 Python project online [here](#) or at bit.ly/mh-python-18
- You might notice two tabs. One is called `main.py`, the other is called `classes.py`. We're going to be programming in `main.py`, but if you have a look at "classes.py" you'll see a lot of code that has already created the objects that we're going to use.

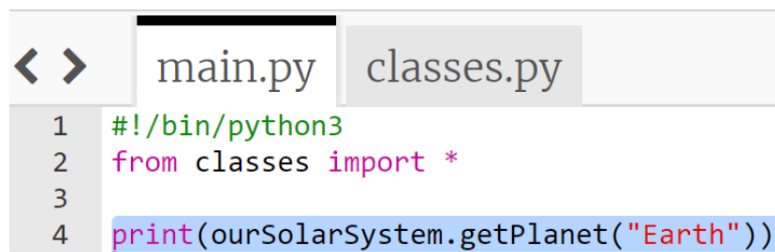


STEP 2: LISTING A PLANET'S INFORMATION

In this step we are going to display all of the information we have about a single planet.

Activity Checklist

- To display the information of a single planet, we can use the “getplanet()” method of the `ourSolarSystem` object. First, we’re going to display the information we have on Earth (because we’re rather fond of Earth!). Add the highlighted line of code below to your `main.py` file.



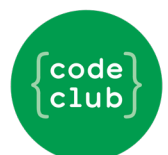
```
< > main.py classes.py
1  #!/bin/python3
2  from classes import *
3
4  print(ourSolarSystem.getPlanet("Earth"))
```

- After adding this code, click the `Run` button. What information do you see?

Challenge: The Other Planets

Now that you have the information from Earth, can you change your code to provide information about another planet in our solar system?

**** Don't forget to use a capital letter when choosing a planet! ****



STEP 3: INTERACTING WITH THE USER

In this step we are going to ask the user to choose a planet for which they want to know the moons of.

Activity Checklist

- To allow the user to type in their content we are going to use an input statement. **Put this statement ABOVE the line of code we just wrote.**
- Also, use a hash symbol (#) in front of the print statement (now on line 6) to comment out the line (meaning that python will just ignore it).

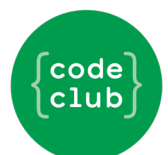
```
3
4 input("Type the name of a planet:")
5
6 #print(ourSolarSystem.getPlanet("Earth"))
```

- Next, we are going to need to store the information the user types into a variable called `targetPlanet` so we can use it later. This code will go on the same line as what we just typed in.

```
4 targetPlanet = input("Type the name of a planet:")
5
6 #print(ourSolarSystem.getPlanet("Earth"))
```

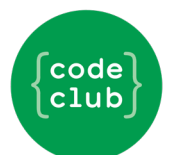
- Next, we are going to make sure our code is working the way it should. We are going to ask our program to print what the user typed in.

```
3
4 targetPlanet = input("Type the name of a planet:")
5 print(targetPlanet)
6
7
8 #print(ourSolarSystem.getPlanet("Earth"))
```



- Run your program. Type a planet after it asks you on the right side of the screen and hit enter. Do you see your selection appear? If not, check your code for any errors before moving on.
- Next, we are going to comment the code we just typed because in our final program we do not want the `targetPlanet` variable to appear in the program.

```
4 targetPlanet = input("Type the name of a planet:")  
5  
6 #print(targetPlanet)  
7  
8 #print(ourSolarSystem.getPlanet("Earth"))
```



STEP 4: USING THE USER INPUT

In this step we are going to use what the user has typed to get information about a planet.

Activity Checklist

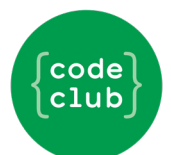
- Remove the hashtag (Uncomment) the second print statement and replace “Earth” with `targetPlanet`.

```
4 targetPlanet = input("Type the name of a planet:")
5
6 #print(targetPlanet)
7
8 print(ourSolarSystem.getPlanet(targetPlanet))
```

- Run your code. Type your planet and hit enter. Do you see the information for the planet you're after? If not, check your code for any errors before moving on.
- What happens when the user types in the name of something other than a planet?

Congratulations!

You now have a fully functioning application to inform others about the planets and the moons in our solar system! In the next steps, we will add code and other features to make your application more user-friendly.



STEP 5: VALIDATING USER INPUT

If the user spells the name of a planet incorrectly, or inputs something that isn't a planet, the program doesn't return a planet object, but instead returns 'None'. We can check if we're getting a planet object, or a 'None' object.

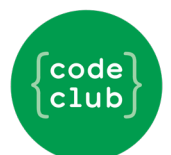
Activity Checklist

- First, we need to assign the result of the `getPlanet` method to a variable. We'll call this variable `myPlanet`. Replace the print method, with a variable assignment.

```
4 targetPlanet = input("Type the name of a planet:")
5
6 #print(targetPlanet)
7
8 myPlanet = ourSolarSystem.getPlanet(targetPlanet)
```

- Test your code by running your program. If done properly, nothing should happen. If errors appear, be sure to debug them before moving forward.
- Next, we are going to test whether or not the user typed in an actual planet.

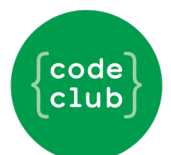
```
8 myPlanet = ourSolarSystem.getPlanet(targetPlanet)
9
10 if myPlanet is not None:
11     print(myPlanet)
```



- Test your code! Run the program several times. Try putting in both real and incorrect planets. What happens? If you put in a name that isn't a planet, the program doesn't print anything. That's not very user friendly! We should tell the user they've made a mistake. Add an else statement to your if statement that tells the user they haven't selected a correct planet.

```
8 myPlanet = ourSolarSystem.getPlanet(targetPlanet)
9
10 if myPlanet is not None:
11     print(myPlanet)
12 else:
13     print(targetPlanet + " is not a planet!")
```

- Test your code again. Does it tell the user their input is not a planet?



STEP 6: LIST THE MOONS

This whole time we've been outputting everything in the planet object, but we really only want to know the moons. Let's fix this!

Activity Checklist

- The planet object has a method "get_moons()" which returns a list of the planets moons. Use that method on the `myPlanet` object.

```
10 ▾ if myPlanet is not None:
11     print(myPlanet.get_moons())
12 ▾ else:
13     print(targetPlanet + " is not a planet!")
```

- Now your code prints just a list of the planets. What happens if you choose a planet with no moons (like Mercury)? Displaying just a set of empty brackets isn't very user friendly. Let's fix that! We'll start by assigning the list of moons to a variable.

```
10 ▾ if myPlanet is not None:
11     moons = myPlanet.get_moons()
12 ▾ else:
13     print(targetPlanet + " is not a planet!")
```

- Next we want to test if the planet has any moons, by testing if the moons list is bigger than zero. To do this we will use the len function (which is short for length).

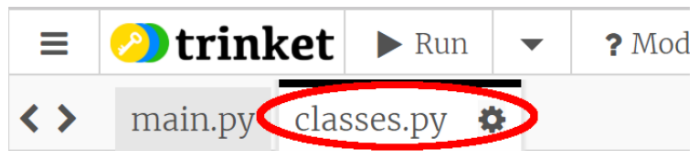
```
10 ▾ if myPlanet is not None:
11     moons = myPlanet.get_moons()
12 ▾     if len(moons) > 0:
13         print(moons)
14 ▾     else:
15         print(targetPlanet + " has no moons!")
16 ▾ else:
17     print(targetPlanet + " is not a planet!")
```

STEP 7: WHAT ABOUT PLUTO?!

In 2006 the International Astronomical Union decided to demote Pluto from a planet, to a dwarf planet. Were you sad to see Pluto leave? If so, complete step 7 to bring Pluto back!

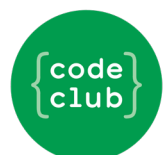
Activity Checklist

- First, we need to know how all of the other planets were added in the classes file. Click on the `classes.py` tab.



- Have a look at how the other planets and their moons are added. Start at line 51. Let's look at the code for three planets in particular, Venus (0 moons), Earth (1 moon), and Mars (2 moons). What do they have in common? What's different?

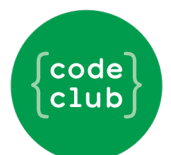
```
51 ourSolarSystem.addPlanet(planet("Venus",[]))
52 ourSolarSystem.addPlanet(planet("Earth",["Moon"]))
53 ourSolarSystem.addPlanet(planet("Mars",[
54     "Deimos",
55     "Phobos"
56 ]))
```



■ Create 3 new lines of code at the bottom of the page (after Neptune) for Pluto. This new code should will bring Pluto and its moon Charon back into our Solar System!

```
214 ourSolarSystem.addPlanet(planet("Neptune",[
215     "Naiad",
216     "Thalassa",
217     "Despina",
218     "Galatea",
219     "Larissa",
220     "Proteus",
221     "Triton",
222     "Nereid",
223     "Halimede",
224     "Sao",
225     "Laomedeia",
226     "Psamathe",
227     "Neso"
228 ]))
229
230 Add code here for Pluto!
231
```

■ After you have added your code for Pluto, click **Run** and type Pluto into your program. What do you see?



CONGRATULATIONS!

You have completed the Moonhack Python project of 2018! Celebrate by high-fiving everyone in the room :)

Now that this project is done there are so many other things you can do.

Extra Activities

- Have a go at the Moonhack Scratch project.
- Create your own “Moon” themed project in Scratch, Python or HTML/CSS!
- Improve this project by completing the challenges for this activity;

Challenge: Your Own Planet!

Now that you added Pluto back into the solar system, can you update the classes.py file to include a brand new planet in the solar system named after you?

Challenge: Case Sensitive

What happens if you forget to put a capital letter for the planet you want to know more about? Python requires us to type in EXACTLY what it expects. However, let's modify our code to fix this.

Use the “string”.capitalize() function to turn what the user types into a new variable. Need a hint? Try Googling “python string capitalize”.

```
4 targetPlanet = input("Type the name of a planet:")
5 capitalPlanet =
6 #print(targetPlanet)
7
8 myPlanet = ourSolarSystem.getPlanet(targetPlanet)
```

Add your new code here