**Error 1.**

**Description:**

The bug shown below at line 7 in main.py file. When attempting to append the planets list we are currently missing a parenthesis in order to close the append call. This is a syntax error. We are unable to call the append until the parenthesis is closed.

**Type:**

Syntax

**Bug Fix Description:**

To fix this I added a closing parenthesis to the end of the append function at line 11.

**Screenshot of the fixed code:**

**A screen shot of a computer program

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

A screenshot of a computer

Description automatically generated

**Error 2.**

**Description:**

The bug shown below is at line 41 in the main.py file. This is due to the Indentation at line 41 not matching the outer loop. It looks like the line t\_total += dt was spaced incorrectly. This will call an error, since it will not make sense to the flow of the program for when we need to add dt to the the t\_total variable.

**Type:**

Syntax

**Bug Fix Description:**

To fix this I want the t\_total to be updated each time the for loop, loops through all of the planets. So I modified the indentation to be outside the for loop but inside the while loop. Which is four spaces or one tab from the left.

**Screenshot of the fixed code:**

**A screenshot of a computer program

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

A screenshot of a computer

Description automatically generated

**Error 3:**

**Description:**

The bug is coming from the planet.py file when we import the Planet class. When looking inside the planet.py file, we can see that we are having an issue with the method force at line 16. The error is due to the keyword arguments or positional arguments being placed after the default argument in the force method. Python 3 requires positional arguments to come before all arguments that have a default argument. Not having the correct position of the arguments, makes the method un-callable and the code will not run when attempting to import the Planet class.

**Type:**

Syntax

**Bug Fix Description:**

To fix this I swapped places with the r and planet argument in the force method. Now as shown in the screenshot below, the default argument comes after the keyword argument when defining the method.

**Screenshot of the fixed code:**

**A black screen with white text

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

A screenshot of a computer program

Description automatically generated

**Error 4:**

**Description:**

The bug is coming from the planet.py file when we import the Planet class. When looking inside the planet.py file, we can see that we are having an issue with the method force at line 16. The error is due to the semicolon not being placed at the end of the force method. Not having the semicolon, makes the method un-callable and the code will not run when attempting to import the Planet class.

**Type:**

Syntax

**Bug Fix Description:**

To fix this, I added a semicolon to the end of the line at line 16 in the planet.py file for the force method. This makes the force method callable.

**Screenshot of the fixed code:**

**A black background with white text

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

A screenshot of a computer

Description automatically generated

**Error 5:**

**Description:**

This error comes from line 6 in the main.py file. Specifically, we are seeing a type error when calling the strip and split methods on the line variable. This error is not as straightforward as the others. To see why we are having this error we have to trace it back a bit further. We create a line when ever we call readlines on the object f. We create f using the open method on the planets.txt file. It looks like we are opening the txt file in binary mode. This is shown by using the ‘rb’ with in the open method. This is causing the error, the functions being used are expecting a string, but we are opening up the txt file in binary mode.

**Type:**

Semantic

**Bug Fix Description:**

This is an easy fix. I changed the rb to r. We are now reading in reading mode, and the default is to read in the file as a string.

**Screenshot of the fixed code:**

**A black background with white text

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

**A screenshot of a computer

Description automatically generated**

**Error 6**

**Description:**

This error comes from line 7 in the main.py file. We are trying to append a list called planets, but it seems we do not have a list called planets to append too (the program is prematurely appending to a list that has not been initialized yet). If we do not have a list that has been initialized, the program will not run, since we are trying to append an object that does not exist.

**Type:**

Semantic

**Bug Fix Description:**

To fix this bug I initialized an empty list called planets, and inserted this list before the for loop that loops through the planets.txt file.

**Screenshot of the fixed code:**

**A screen shot of a computer program

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

**A screenshot of a computer

Description automatically generated**

**Error 7**

**Description:**

This error is coming from line 9 in the main.py file. When we try to initialize a planet object using the float of the planet\_data, we are receiving an error saying that the Planet class takes no arguments. This sounds like an error with the Planet class itself. With further inspection of the Planet class, it looks like the init constructor was set up incorrectly. Instead of using a constructor, the class created a python method called init. This means that the class doesn’t take in any arguments, just has a bunch of methods, and does initialize any attributes. This is a major issue for the code to work.

**Type:**

Syntax

**Bug Fix Description:**

To fix this code, I changed the init method to an init constructor.

**Screenshot of the fixed code:**

**A screen shot of a computer

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

**A screenshot of a computer

Description automatically generated**

**Error 8**

**Description:**

This error is coming from line 14 in the main.py file. We are receiving a value error when attempting to take the float of 1.9898e+30. At first glance, this seems like the python module is doing something incorrect, but at further inspection, we can see the zero in the ‘30’ is not a zero but actually an uppercase alphabetically character O. This will cause an error when trying to take the float of the string. Python is not able to turn a string of an letter O to zero using int or float. If we don’t fix this, the Planet class will not be able to run due to the error being raised.

**Type:**

Input Error

**Bug Fix Description:**

To fix this error, we will need to create a try except block that calls the except when we see a ValueError. I inputted the original planets.append(Planet(….)) in the try block, and came up with a method to handle any cases where a letter O is placed instead of the expected zero, which is placed in the except ValueError block. To fix this, I needed to fix the one value in the list of items that was failing in the planet data, specifically, the 1.9890e+30 item. To do this, I first initialized an empty list called corrected\_data. I then create a for loop to loop through each item in planet data of the list that failed in the try block. I then created another try except block, that attempted to take the float of the value in the planet\_data list. If the float method worked, then I would append the value to the corrected\_data list, if not then we have identified the specific element that is causing the value error.

Now that I have the string that is failing, I need to replace the string with a correct string with zeros instead of O's. Strings are immutable, so we are going to have to create a new string to add to the corrected list instead of modifying the string in place. To do this, I created an empty string called corrected\_value. I then looped through the indices of the value using the range and len function. If the value[index] is equal to the capital O then I would add the zero to the corrected value string, otherwise I would add character at the index to the corrected value string. After looping through all characters, I would then append the corrected\_data list with the corrected value.

Finally, I would use the planets.append(Planet(…)) to create the Planet object but this time using the corrected\_data list that has all O’s replaced with zeros.

**Screenshot of the fixed code:**

**A screen shot of a computer program

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

A screen shot of a computer

Description automatically generated

**Error 9:**

**Description:**

The next error is coming from line 53 in the main.py file when we called the force method. We can then see that the error is coming from the force method in the planet.py file on line 17 when we attempt to return G\*planet \*planet/r\*\*2. It looks like we are incorrectly calling the class variable G. Since we are incorrectly calling the class variable G, the program will be unable to run. We are calling a variable called G, not the class variable G.

**Type:**

Semantic

**Bug Fix Description:**

To fix this line of code I added Planet and used the dot operator to call the class variable G on line 17.

**Screenshot of the fixed code:**

**A computer screen with white text

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

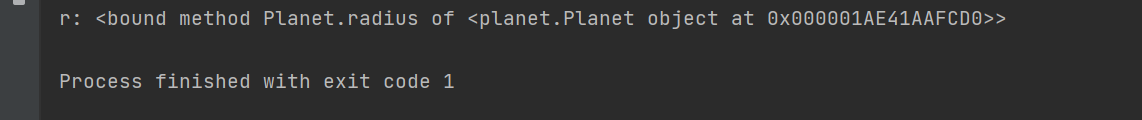
**A screenshot of a computer program

Description automatically generated**

**Error 10**

**Description:**

We can see that the error is happening in main.py at line 53 when we call the force method. When looking at the force method in the planet.py at line 17, we can see that the error is happening at line 17. It looks like we are having some kind of error when attempting to square the r variable. With further inspection of the main.py file, we can see that the r variable is created by calling the radius method on the sun. This is where the error is coming from. With a quick print, I can see that the radius being created at the moment is not a number but instead an method object . We can not square a method object, the syntax to call this method is currently incorrect. Right now the program is treating the method like an instance attribute. To call a method, we need open and closed parenthesis, and any arguments that the method requires.



**Type:**

Syntax

**Bug Fix Description:**

To fix this error. All I needed to do was correct the syntax of calling the radius method. On line 51 in the main.py file. I added an open and closed parenthesis to call of the radius method, and added the argument planet, as that is required in the method shown in the planet.py file.

**Screenshot of the fixed code:**

**A black background with white text

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

**A black background with blue and red text

Description automatically generated**

**Error 11:**

**Description:**

On line 61 in the main.py we are seeing a type error. When attempting to calculate the planet.vx attribute. The error says that we can’t multiply sequence by non-int type of float. It means that either the ax or dt attribute variables are non int. Upon closer inspection, I can see that dt is initialized as a string.

**Type:**

Semantic

**Bug Fix Description:**

To fix the error, I changed the dt variable on line 43 in the main.py to reference 25000.00, a float, instead of the string ‘25000.0’.

**Screenshot of the fixed code:**

**A screenshot of a computer

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

A black background with blue and red text

Description automatically generated

**Error 12:**

**Description:**

This next error is in the init constructor. We it looks like the px argument is being mapped to both self.px and self.py.

**Type:**

Logic

**Bug Fix Description:**

To fix this, I mapped the self.py to the py argument inside the init constructor.

**Screenshot of the fixed code:**

**A black screen with white text

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

N/A

**Error 13:**

**Description:**

**A black background with white text

Description automatically generated**

Another error is the logic used to calculate the force in the force method. The current code is calculate the force incorrectly. It looks like we are taking the square of the planet mass, multiplying by the constant G and dividing by the square of the radius. To calculate the force between to objects, we need to multiply by the current planets mass and the mass of the object we are calculating the force between. For this code, we are using the sun as the caller to estimate the force, so one of the planet.m’s needs to be changed so that it references the current instance of the mass for the object that is calling it.

**Type:**

Logic

**Bug Fix Description:**

To Fix this, I modified one of the planet.m’s to self.m.

**Screenshot of the fixed code:**

**A screenshot of a computer

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

N/A

**Error 14:**

**Description:**

The final error in the code is that we are missing the sun list from our final output when we loop through the planets and print out each planets px,py,vx,vy and mass with 4 decimals in scientific notation. This is due to us deleting the sun item from the planets list earlier on in our code. The code will run fine with out adding this back into the planets list, but in order to match desired output, we will need to add the sun list back to the list of planets.

**Type:**

Logic

**Bug Fix Description:**

To fix this, I inserted the sun list back into the planets list, right before the for loop that is used to print out each component with 4 decimals in scientific notation. I made sure to insert at the same index that was used when to delete the item from the planets list(3).

**Screenshot of the fixed code:**

**A screen shot of a computer

Description automatically generated**

**Screenshot of each exception caused by the bug/error:**

N/A

**Final output:**

A screenshot of a computer program

Description automatically generated