

Programming Project #1

Assignment Overview

This assignment involves coding and testing of a program based on the “Hello World” program from the first lab, and then using the **handin** program to hand it in.

You can turn in programs using **handin** as often as you want. Try turning in the “Hello World” program (or the completed project) during lab so if there is a problem, the TA can help you solve it. During lab you should create the project so that if you need help, you are still in lab with the TA available. Also, submit the program (even if it isn’t done, you can always do it later) to make sure that you are enabled for handin. Complete the project and hand it in again when done.

The basic design of the first programs that you construct in this class consists of a prompt for information, receiving information, processing that information and finally displaying the results.

This assignment is worth 10 points (1% of course grade), and must be completed before 11:59 PM on Monday, September 12th. Projects are typically due on Mondays.

Background

This programming project will use the `raw_input` and `print` functions along with some simple mathematics for conversion. The important part of the project is to learn the skills needed to access the class web site to download a project description, create a new program in Python and finally to hand it in.

The national debt is a hot topic this year, both because it continues to set new records and because this is the beginning of the presidential election season. Just how big is the debt? It is a very big number, and its size makes it difficult to comprehend. Let's write a program that tries to help us understand this big number.

Program Specifications

Your program will prompt for two numbers:

- The size of the national debt in dollars. It is a number that is hard to track, but there are various web sites that are only too happy to give you a number. Try <http://www.usdebtclock.org> (slow) or http://www.brillig.com/debt_clock (faster).
- The denomination of U.S. currency bills you want to use. You can go up to \$100,000 but in reality \$100 is the largest, common U.S. bill (see http://en.wikipedia.org/wiki/Large_denominations_of_United_States_currency)

Your program will print out two numbers:

- The height in **miles** of your chosen bill denomination required to equal the debt number that was input.
- The calculated distance in miles translated to a multiple of the average distance to the Earth's moon.

Useful Facts

Some useful facts:

- The thickness of every U.S. denomination bill is 0.0043 inches (<http://www.trivia-library.com/a/money-in-the-us-history-of-the-dollar-bill.htm>)
- The average distance to the moon is 238,857 miles ([http://en.wikipedia.org/wiki/Lunar_distance_\(astronomy\)](http://en.wikipedia.org/wiki/Lunar_distance_(astronomy)))
- The word **string** in computer science has a special meaning. It means "a sequence of characters", usually delimited by quotes. We'll see it used often in our work. It differs from a number: "12" is different from 12.

Deliverables

proj01.py -- your source code solution (remember to include your section, the date, project number and comments).

1. Please be sure to use the **specified file name**, i.e. "proj01.py"
2. Save a copy of your file in your CSE account disk space (H drive on CSE computers).
3. You will electronically submit a copy of the file using the "handin" program:
<http://www.cse.msu.edu/handin>

Assignment Notes:

To input the numbers it is necessary to use the `raw_input` function. The `raw_input` function takes a string, a sequence of characters between quotes, as a prompt to print to the user. It then waits until the user types a response, terminated by the user typing the Enter key. A string, again as a sequence of characters, is returned.

The returned string must be converted to a number. In this assignment we are strictly working with floating point numbers, a string is converted to a float using the `float` function. The `float` function takes as an argument a single string and returns integer number the string represents. A typical interaction would be something like:

```
num_str = raw_input('Please enter a number: ')
my_float = float(num_str)
```

`print` is a command that will print on the output window any combination of variables, values and strings. Each item to be printed must be separated from other items by a comma. All the items will be printed together, followed by a new line. For example:

```
bills_float = 3.14159
print 'The number ',bills_float,' times two is ', bills_float*2
```

This command has 4 items to print: a string ('The number '), the value in the variable `bills_float` (3.14159) as a string, another string (' times two is ') and the result of an expression (3.14159 * 2). Thus the statement will print:

```
The number 3.14159 times two is 6.28318
```

Look at the program `numberInput.py` in the `proj01` directory as an example of using `raw_input`, `print` and `float` (also `int` for integer numbers).

Once converted to numbers, the operations on these numbers are, respectively: `+` (sum), `-` (difference), `*` (product), `/` (division) and `%` (remainder). The last two deserve special comment.

In Python, if an integer is divided by another integer, the result is an integer. Thus the result of `6/4` is `1` (not `1.5`). That is, the `/` operation results in the integer **quotient**. The result of `6%4` is the integer remainder of the division, thus `2` (6 divided by 4 is 1 with a remainder of 2). *Play around with the quotient and remainder operators in the Python shell.*

To clarify the problem specifications, we provide at the end of this document a snapshot of interaction with the already written program.

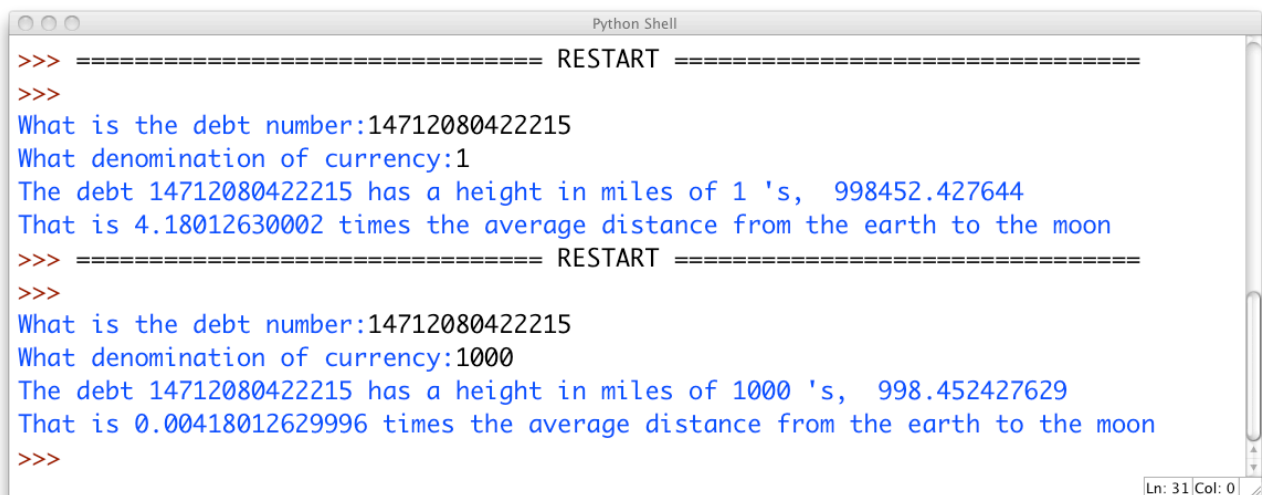
Getting Started

1. Using IDLE create a new program.
2. If you are in a CSE lab, select the H: drive as the location to store your file
3. Save the name of the project: `proj01.py`
4. Using the example from `numberInput.py`, write the code. Track down any errors
5. Run the program
6. Use the handin web site to hand in the program (to make sure you can do it)
7. Edit the program
8. Now you enter a cycle of edit-run to incrementally develop your program.
9. Use handin to hand in your final version (only last one counts!!)

Questions for you to consider (not hand in)

1. What happens when you try to divide by zero when you run your program (for example, enter a 0 for the bill denomination)?
2. What happens when you enter a letter instead of a number at the prompt?

Sample Interaction



```
Python Shell
>>> ===== RESTART =====
>>>
What is the debt number:14712080422215
What denomination of currency:1
The debt 14712080422215 has a height in miles of 1 's, 998452.427644
That is 4.18012630002 times the average distance from the earth to the moon
>>> ===== RESTART =====
>>>
What is the debt number:14712080422215
What denomination of currency:1000
The debt 14712080422215 has a height in miles of 1000 's, 998.452427629
That is 0.00418012629996 times the average distance from the earth to the moon
>>>
```

Ln: 31 Col: 0