Python 101

Pedro Silva

September 24, 2015

INTRODUCTION

In this assignment I created a program that contains four different functions. These functions are, check if a number is odd or composite, check if a number is prime or not, convert numbers to binary, and, a physics simulator. To create this program I used the Python 2.7.10 programming language.

Python was created in the 1980s by Guido van Rossum but it was only released in 1989. The main concepts of the language are:

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.

The python language has been used over the years to create many of todays most used programs, for example: Dropbox. Python also has influenced some of todays most used programming languages, for example JavaScript, Ruby and Swift.

Now a days there are three versions of python available; Python, Phython 2. , and, Python 3000 (py3k)

MAIN MENU - PROGRAM CODE

```
import time

MainMenu = True
while MainMenu:

    odd = True
    prime = True
    binary = True
    physics = True


    print "-----------------"
    print "-----Welcome-----"
    print "-------to--------"
    print "------Pyth-------"
    print "-----------------"
    print "-------By:-------"
    print "------Pedro------"
    print "-----------------"

    time.sleep(1)

    print "Programs:"
    print "         Odd Numbers[1]"
    print "         Prime Numbers[2]"
    print "         Number to Binary[3]"
    print "         Physics Simulator[4]"
    print "         Exit[ext]"
    program = raw_input()
```

These lines of code create the main menu and imports all libraries necessary to create the program.

In the first line we see the code "import time", what this does is import the "time" library. I use this library later on in the program to create pauses.

I then create a boolean called "Main-Menu" and i set it to True. I use this boolean to create a while loop for the Main Menu.

On the next four lines I set 4 booleans: "odd", "prime", "binary", "physics". I use these booleans later on the program to create multiple while loops.

Through the next few lines of code I print a title saying "Welcome to Pyth - By: Pedro"

After I print the title I use the library time for the first time. I type "time.sleep(1)". I type "time" to call the library, then i type ".sleep" to call the sleep function, then I type "(1)", what this does is it defines for how long I want the program to sleep.

After this, I print the options of what the user can do in this program.

To finish off the Main Menu part I input the line of code "program = raw_input()". What this does is it creates a variable called program with what ever choice of program the user decides to input.

```
-----------------
-----Welcome-----
-------to--------
------Pyth-------
-----------------
-------By:-------
------Pedro------
```

This is the output of the lines of code above.

```
-----------------
Programs:
         Odd Numbers[1]
         Prime Numbers[2]
         Number to Binary[3]
         Physics Simulator[4]
         Exit[ext]
```

ODD NUMBERS - PROGRAM CODE

```
if program == "1":
        print "Loading..."
        time.sleep(.7)
        print "---------------"
        print "--Odd Numbers--"
        print "---------------"
        time.sleep(1)
        while odd:
            print "Enter Number:",
            n = raw_input()
            if n == "ext":
                odd = False
                break
            n = float(n)
            if n % 2 == 1:
               print n,
               print "is an odd number."
            if n % 2 == 0:
                print n,
                print " is a composite number."
```

These lines of code is the part that runs the "Odd Numbers" program.

In the first line we see "if program == "1":", what this does is it creates a if statement that says: If the variable program(defined in the main menu) is equal to one run these lines of code.

On the next few lines of code we see almost the same thing as we had in the Main Menu. We see the print function and the time.sleep() function.

I then create a new while loop, using one of the booleans created before, specific for the "Odd Number" program. The aim with this loop is to restart the "Odd Numbers" program after it finishes.

This is the point of the code that the "Odd Numbers" program actually begins. It starts off by asking the user what number he wants to check if its odd or pair. To do this I used the "raw_input()" function.

I then check if the user input is "ext". If it is ext ill turn the odd boolean to False and Ill break the while loop. Therefore taking the user to the Main Menu. if the user input is not ext Ill continue to the rest of the "Odd Numbers" program.

To start off the "Odd Numbers" program I turn the string that I got as an user input and I make it into a float. To do this I do: "n = float(n)" where n is the name of the variable where the string is saved.

After turning the string into a float I make an If statement. In the If statement I check if when the number entered by the user is mod by 2 if the answer is 1. If the If statement comes out true I print that the Number is Odd. If the If statement comes out False I continue to another If statement. In this if statement I check if the number entered by the user mod 2 is equal to 0. If this comes out True I print that the number is composite.

```
Loading...
---------------
--Odd Numbers--
---------------
Enter Number: 5
5.0 is an odd number.
```

This is an example of what happens when you run the "Odd Numbers" program with the number 5.

PRIME NUMBERS - PROGRAM CODE

```
if program == "2":
        print "Loading..."
        time.sleep(.7)
        print "-----------------"
        print "--Prime Numbers--"
        print "-----------------"
        time.sleep(1)
        while prime:
            print "Enter Number:",
            n = raw_input()
            if not n == "2":
                if n == "ext":
                    prime = False
                    break
                n = float(n)
                if n % 2 == 1:
                    y = n / 3.0
                    while y > 0:
                        y = y - 1
                    if not y == 0:
                       print n,
                       print " is not a prime number."
                    if y == 0:
                        print n,
                        print " is a prime number."
                if n % 2 == 0:
                    print n,
                    print " is not a prime number."
            if n == "1":
                print n,
                print " is not a prime number."
            if n == "2":
                print n,
                print "is a prime number."
```

These lines of code run the "Prime Numbers" program.

The "Prime Numbers" program has many similar functions to the ones I used in the "Odd Numbers" program. We may notice I use the same concepts to start off all programs.

We only begin to see a difference when we get to the first If statement. I do an If not statement instead of the casual If statement. I do this to check if the number is not pair, because if the number is pair it means that it can be divided by 2 therefore making a non-prime number.

If the number is not pair I then divide it by 3 and I subtract 1 off the answer util the number is in between 0 and 1.99. I do this to check if when it is divided there are any remainders. If there are no remainders that means that the number can be divided by 3, therefore making a non-prime number. But, if the result is a number in between 0.01 and 0.99 that means that the number can not be divided by 3 and still give a full number as an answer, therefore making it a prime number.

This is an example of what happens when you run the "Prime Numbers" program with the number 57.

```
Loading...
-----------------
--Prime Numbers--
-----------------
Enter Number: 57
57.0  is a prime number.
```

NUMBER TO BINARY - PROGRAM CODE

```
if program == "3":
        print "Loading..."
        time.sleep(.7)
        print "------------------"
        print "-Number to Binary-"
        print "------------------"
        time.sleep(1)
        while binary:
            equ = "="
            print "Enter number to convert:"
            s = raw_input()
            if s == "ext":
                binary = False
                break

            v = int(s)

            bina = bin(v)[2:]

            print v,
            print equ,
            print bina
```

These lines of code make up the "Number to Binary" program.

Like in the other programs I start off the same way. Then I turn the sting into an integer and I create a variable using the following function: bin("variable name")[2:]

I then print the result.

```
Loading...
------------------
-Number to Binary-
------------------
Enter number to convert: 69
69 = 1000101
```

This is an example of what happens when you run the "Number to Binary" program with the number 69.

PHYSICS SIMULATOR - PROGRAM CODE

```
if program == "4":
        print "Loading..."
        time.sleep(.7)
        print "------------------"
        print "-Physics Simulator-"
        print "------------------"
        time.sleep(1)
        while physics:
            print "What height is the start(meters): " ,
            y0 = raw_input()

            if y0 == "ext":
                physics = False
                break

            print "What is the force being applied
            at the start: ",
            v0 = raw_input()

            if v0 == "ext":
                physics = False
                break

            a = -9.8
            dt = 0.01
            t = 0
            v = float(v0)
            y = float(y0)
            ymax = 0.0

            while y > -0.0000000001:
                t = t + dt
                v = v + a * dt
                y = y + v * dt

                if y > ymax:
                    ymax = y

            print "Time taken: " + str(t) + "s"
```

These lines of code make the "Physics Simulator" program.

Just like in all other programs I start off the same way.

I then ask the User to input what height is the simulation starting at. After he inputs that i check to see if he inputed "ext", If he did I go back to the main menu. If he did not input ext I ask the user to input the force applied to the object. I then once more check to see if he inputed "ext". After that I set a few variables to use in the simulation.

After that I begin a while loop that will be used to simulate. I make the while loop run until the height of the object reaches -0.000000001. I do this because then Im allowed to begin my simulation at 0 and still get a very precise conclusion.

For every instance the while loop runs I add to the time, I add to the velocity and I add to the height.

When the while loop reaches -0.000000001 I print the theoretical time taken for the object to fly up, come down and hit the ground. I print the final velocity when the object hits the ground. And, I print the max height the object reached during the simulation.

```
            print "Final velocity:" + str(v) + " m/s"
            print "Max Height: " + str(ymax) + " m"
```

```
Loading...
-------------------
-Physics Simulator-
-------------------
What height is the start(meters):  0
What is the force being applied at the start:  67
Time taken: 13.67s
Final velocity:-66.966 m/s
Max Height: 228.69572 m
```

This is an example of what happens when you run the "Physics Simulator" program with the object starting at the ground(0) and a force of 67 Newtons being applied to it.