Python

Mount St. Mary’s College

December 13, 2013

Mihai Mihailescu

Python is a well-known programming language that has been growing in popularity since its conception. It was developed by Guido Van Rossum during the 1990s and ultimately first released on February 20, 1991 to the public. Development and implementation of Python began in the Dutch research institute Centrum Wiskunde & Informatica (CWI) during the 1989. The institute was focused on research in computer science and mathematics. It is here that the programming language ALGOL 68 was created.

Guido Van Rossum began working at CWI in 1982 as a programmer in the ABC group. ABC is a programming language aimed at beginners as a replacement for the then popular BASIC language. After his ABC group was terminated, Rossum began work with the Amoeba group. Amoeba was a language that was meant to create a unifying experience for networks. Its purpose was to make a network of computers appear as a single machine for the user to access. After the creator of the language Andrew Tanenbaum accepted a professorship, the group was terminated and Rossum was sent to the new CWI Multimedia Group. Rossum states that ABC was his key inspiration for developing Python, Amoeba was his motivation, and the multimedia group helped Python grow.

Python was created because Rossum felt there was a need for a higher-level language in the Amoeba projects that he was working on. For Rossum, the task of creating system administration utilities took too long and using a Bourne shell with Amoeba would not. The Bourne shell is a program that was made to read the lines inputted by the user in the command line. Ultimately, the job of Python was to facilitate the Bourne shell into working with C. Development of Python relied on proper syntax and Rossum preferred the Algol 60, Pascal, Algol 68, and ABC syntax because it was what he had worked with the most. Python was meant to improve ABC. The next matter of business was the name of the new programming language. After much deliberation Rossum decided that Python, from Monty Python’s Flying Circus, would be the name.

The first piece of code implemented in Python was a parse generator called “pgen.” Python was always intended to be open-source and after its release on February 20, 1991 it was clear that Python was popular. To keep up with demand, Rossum used a CVS (concurrent versions system) to keep track of all changes made to the language as it developed. After gaining traction in the community, Rossum began working at CNRI. He established his own team and together furthered the development of Python. They built a mobile agent system, upgraded community infrastructure like mailing lists and CVS servers. At this stage Python 1.5.2 was the most stable and commonly used language. Jython was later released as a way to implement Python in and for Java. In October 1996, Microsoft shipped Microsoft Merchant Server 1.0, which was an e-commerce solution for businesses. Python offered Microsoft reliability and ease-of-use that other languages could not offer.

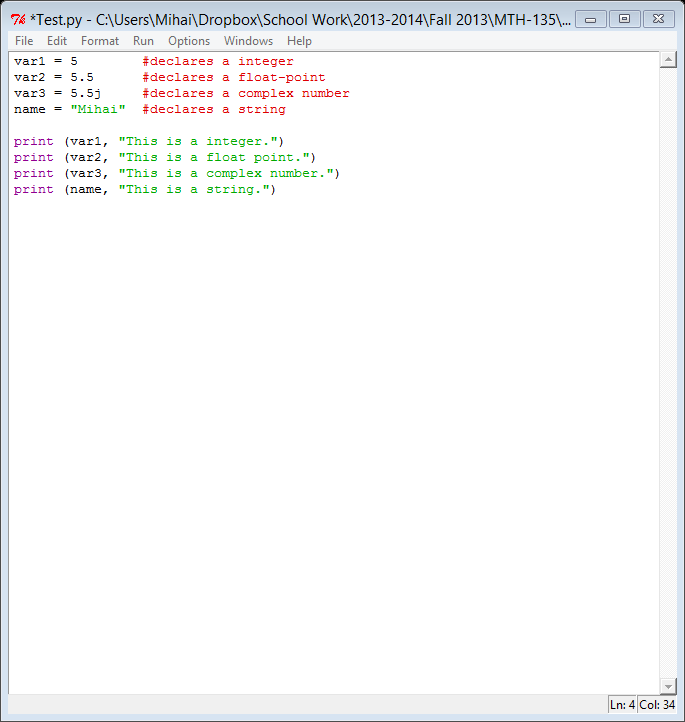
The next step for Python was to begin implementation and integration with other languages. The first was Jython, a Python implementation in Java. Eventually it was also ported to the .NET infrastructure. Python began being used in other parts of CNRI and combined with the new ports allowed more programmers to get on board. A major step for python was the creation of the Python Cryptography Toolkit, a library of cryptologic algorithms that Python users can use.

Development of Python was also faced with legal problems. Since Rossum moved locations while developing Python, certain licenses were not compatible. They had trouble with compatibility with the GNU General Public License, which intended to keep the programing languages open-source. Eventually CNRI was able to modify the licenses so that the language can be defined and stay open-source. Ultimately in 2001, the Python Software Foundation was formed as a nonprofit organization. Python is now on version 3.3.3 and tutorials can be found on www.python.org.

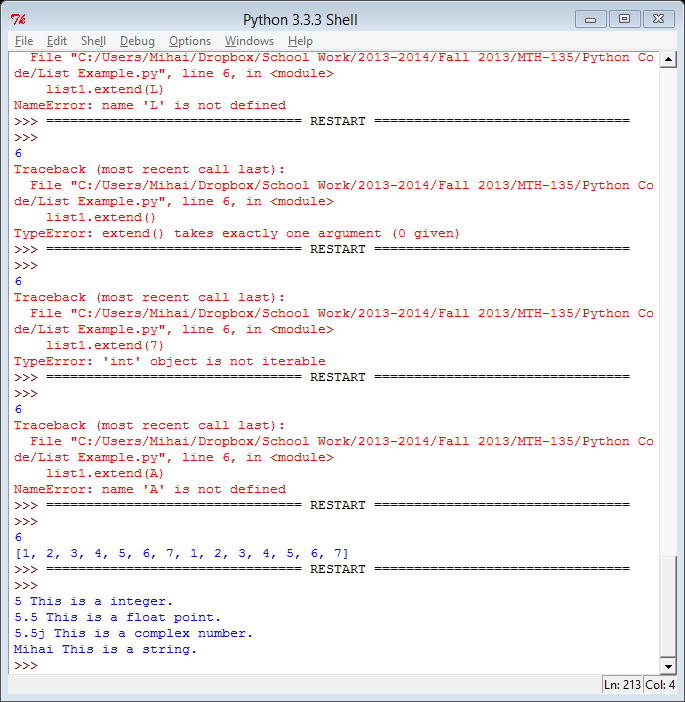
Python is an interpreted language, meaning that it does not need to be compiled into code. A python program ends with “.py”. An example of a Python program name is “research.py”. Navigating to [www.python.org](http://www.python.org) it is possible to download the Python IDLE. IDLE is the Integrated Development Environment made for programmers. Programming python code is quite simple and IDLE is not necessary. Having a simple text-editing software like Notepad is enough to begin programming, albeit reading the code gets difficult. Since Python is considered a scripting language also, it uses modules to carry out tasks. We will be working with modules instead of single-line execution as is offered by IDLE.

When writing code, to distinguish between objects names Python allows identifiers to use A to Z, a to z, underscores, and digits. Python is a case sensitive language, for example “count” and “Count” would be two completely different variables.

When working with variables, Python declares memory location when a value is assigned to a variable. Assigning a number value to a variable means that the variable is being assigned one of the three numeric types: integer, floating point number, and complex number. An integer is any number that does not have a decimal component, a float point number is one that has a decimal component, and a complex number has two real numbers and one imaginary number. The following code explains the difference.



Notice that print is being used in the second part of the code. In the first part we are assigning values to var1, var2, var3, and name is being assigned a string. Strings will be discussed later. The print statement simply takes what is in the parentheses and outputs them to the user. When ran, the code above will output the following:



In Python there are words that cannot be assigned values, for example print. If we try to assign a value to print we would the following error “TypeError: ‘int’ object is not callable. Along with print are many other variables that cannot be assigned values, these are called reserved words. All the reserved words in Python are lowercase. Here is a table with all the reserved words in Python.

|  |  |  |
| --- | --- | --- |
| and | exec | not |
| assert | finally | or |
| break | for | pass |
| class | from | print |
| continue | global | raise |
| def | if | return |
| del | import | try |
| elif | in | while |
| else | is | with |
| except | lambda | yield |

**Data Types**

Data is stored onto the memory, and in Python the date can be stored as five types: numbers, strong, list, tuple dictionary. Each data type has its own rules and its own subtypes.

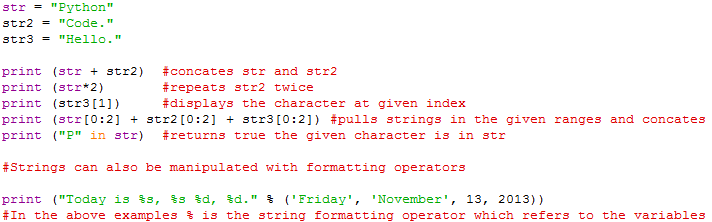
The number data types were described earlier in detail when discussing variables. To reiterate, there are three numerical types that are found in Python 3.3 integer, float, and complex. When assigning a numerical value to a object, the object becomes a number object. A simple numerical assignment would be var1 = 5. This means that the memory address that var1 has been assigned holds the numerical value of 5. To delete the assigned value of a variable, we use the statement **del**. An example would be

“del var1.”

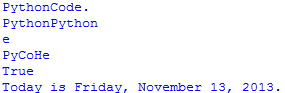
|  |  |  |
| --- | --- | --- |
| **int** | **Float** | **complex** |
| **5** | **5.5** | **5.5j** |
| **-5** | **-5.5** | **-5.5j** |
| **12345** | **-1234.5** | **-1234.5j** |

**Examples of int, float, and complex.**

Strings in Python are defined as “a contiguous set of characters in quotation marks.” When assigning strings, the characters must be contained by quotation marks. Python allows the use of either single quotation or double quotation. However, it must be noted that using the same quotation three times will create a comment block until another set of the same three quotations are included. Comment blocks will not display to the user once the code is ran. Strings can be manipulated in the following ways.



The output of this code will be:



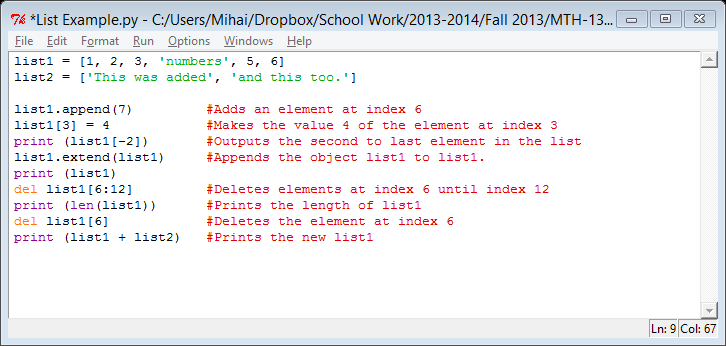
It is important to understand how the computer looks at strings. Each character has an index number from 0-9. The first character in a string will be at index 0 and the next character will be index 1 and so on. For example:

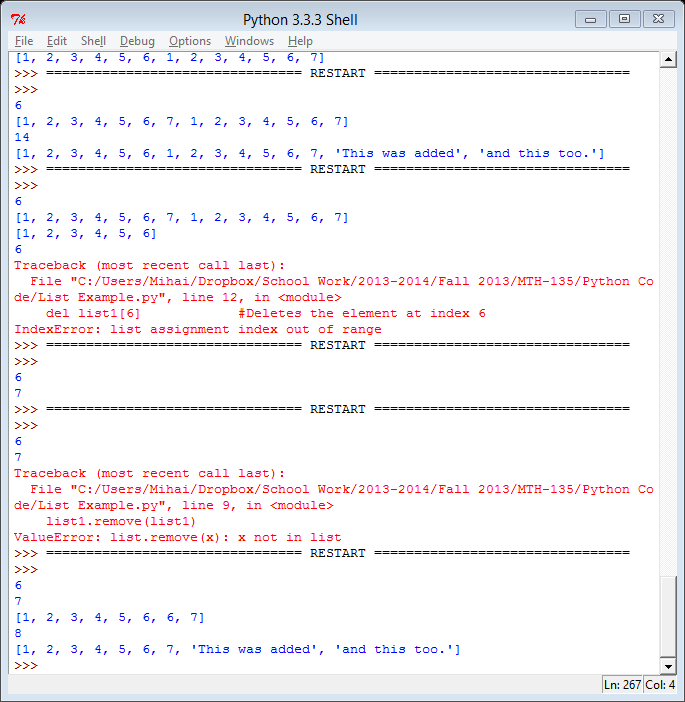
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Characters | P | Y | T | H | O | N |
| Index | 0 | 1 | 2 | 3 | 4 | 5 |

Refer back to the previous piece of code, **print (str3[1])**. This statement is telling the computer to print on the screen the character at index 1. We know that str3 has the string Hello and the index of 1 has the character “e” because index 0 is character “H.” Strings have escape characters that interpreter will run inside the string. One example is \n which will create a new line. Here is a quick example of the process.

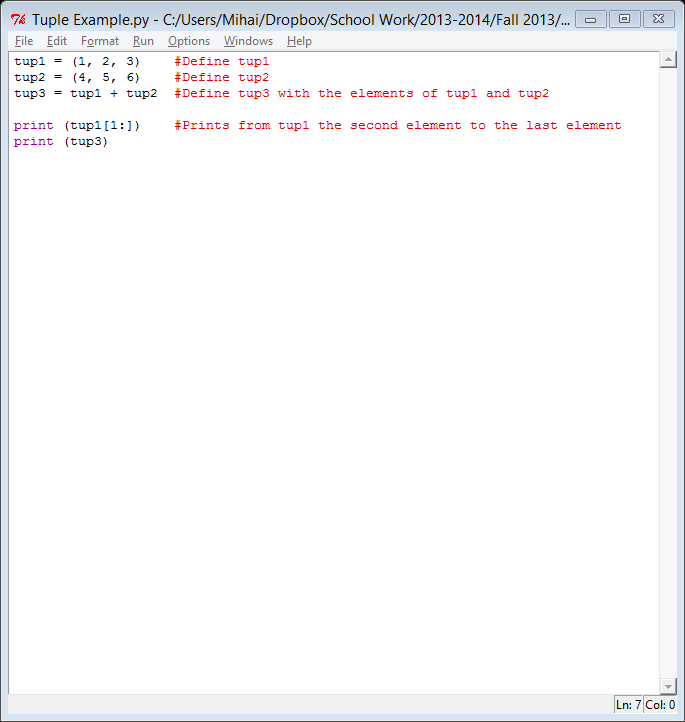


Lists are incredibly useful and versatile in Python. Lists have values in between brackets and are all separated by commas. Lists can have both strings and numbers. Lists also have indices, which like strings, start at 0. The first item in the list would be at index 0, the second at index 1, third at index 2 and so on. A basic list would be **list1 = [1, 2, 3, ‘numbers’, 5, 6].** To access any values in the list the the use of the slicing brackets is indicated. To call on the number 3 in **list1** we would use **list1[2]**. To access a range of items in the list, the correct syntax would be **list1[2:4]**. This syntax will pull 3 and numbers from the list. Another important function of lists is that at any time lists can be update and its items changed. If we want to replace the string **numbers** with 4, we would simply use **list[3] = 4**. To add on the list the correct syntax would be **list1.append(7)**. This would add the value 7 to index 6. If you want try to output an index that is not part of the list, for example trying to print the value at index 7 would result in an error because the index is out of range. To remove a list element we use the **del** statement. The proper syntax for deleting an element would be **del list1[1]**. This will delete the element at index 1. Similar to string you may concatenate using the + operator, \*x to repeat an x amount of times, checking membership with the **in** statement. Additionally you can also check the length of a list by using **len(list1)**. Another option Python implemented in lists is using negative index location. Using **list1[-1]** will refer to the last element in the list, **-2** will refer to the second to last item in the list. There are many other methods of list objects implemented into python. Below is the syntax for lists and its output.





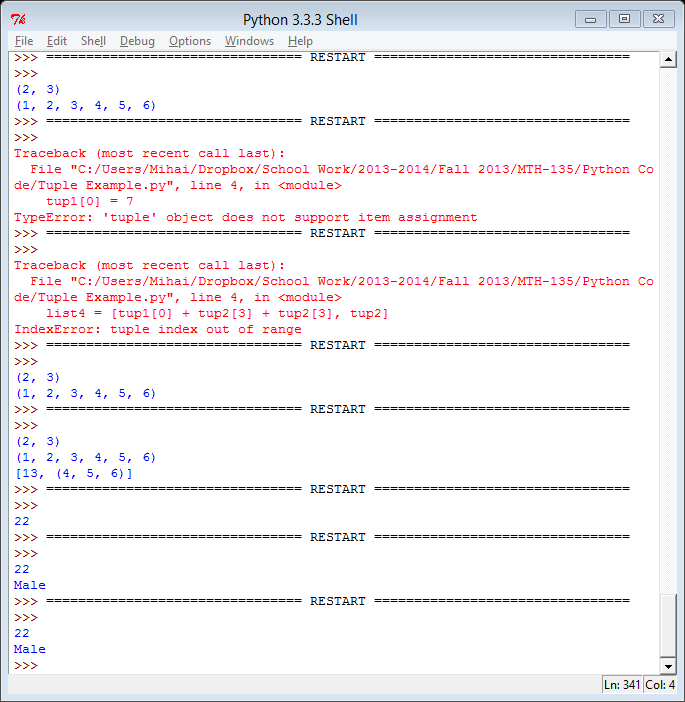
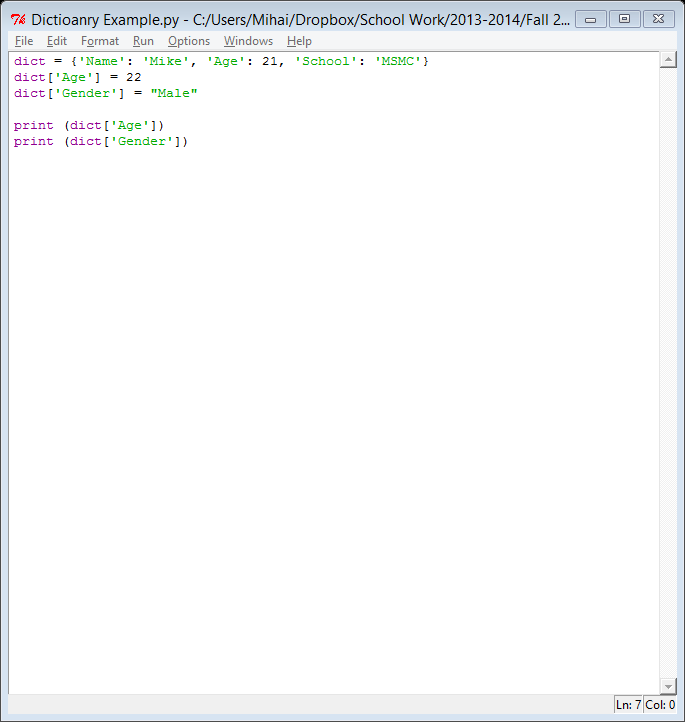
Tuples are data type in the Python language. Tuples are similar to lists in that they are separated by commas, but are surrounded by parentheses. Another important part of tuples is that the elements and the size of the tuple are immutable,or inn other words cannot be changed or updated. Tuples can be spliced, indexed, and concatated . Even though tuples cannot be changed, it is possible to take parts of exsiting tuples and combine them into a new tuple. To do this you must define a third tuple and call on the other two tuples to do what is needed. Just like lists and strings the operators +, \*x, len(), and **in**, all works for tuples. Below is a simple example of tuples and a tuple error.



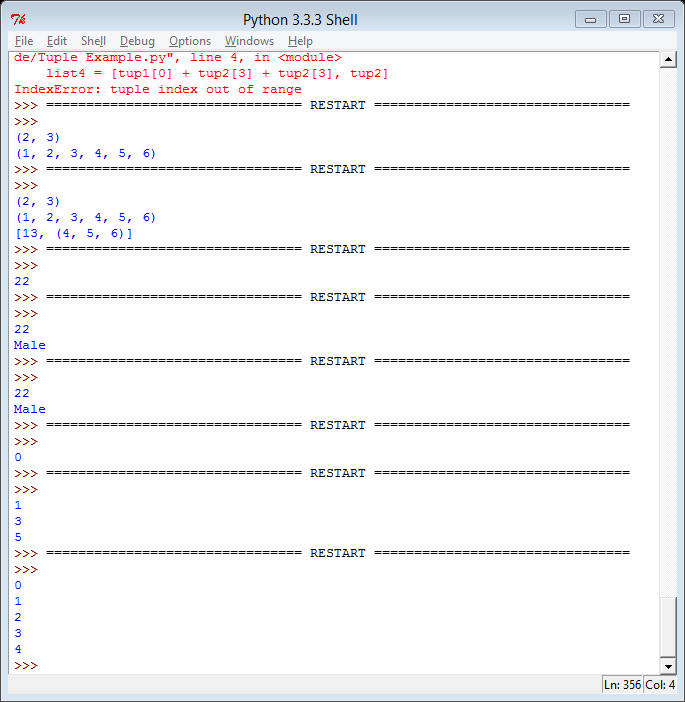
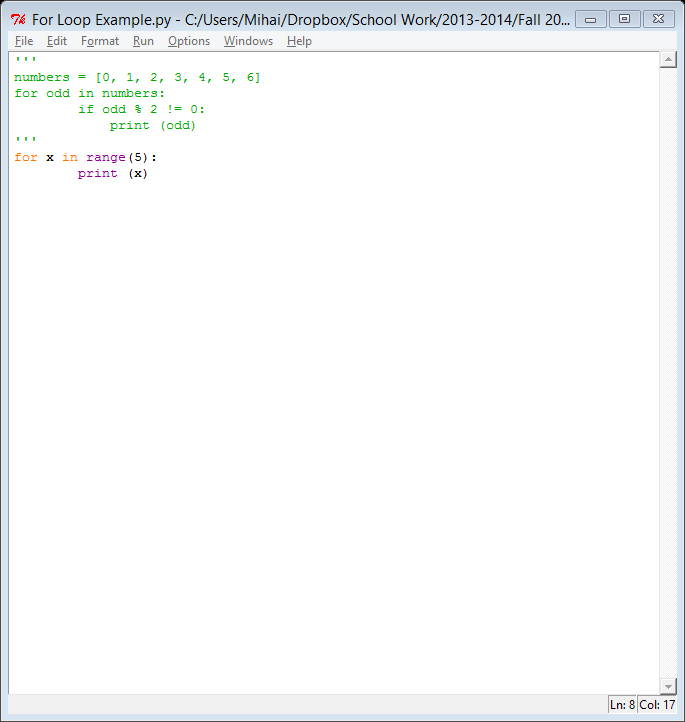


When we add the line **tup1[0] = 7** to try to change the first element in the tup1, the error message “TypeError: ‘tuple’ object does not support item assignment” appears.

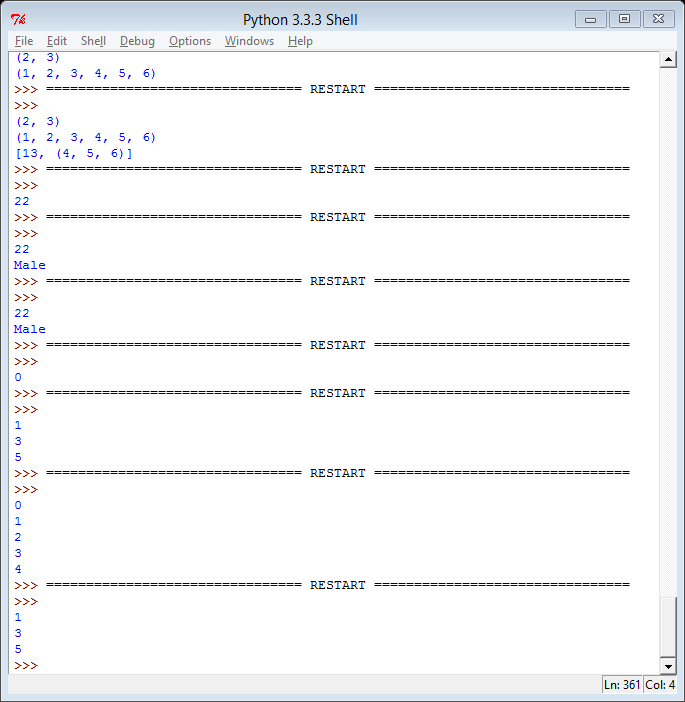
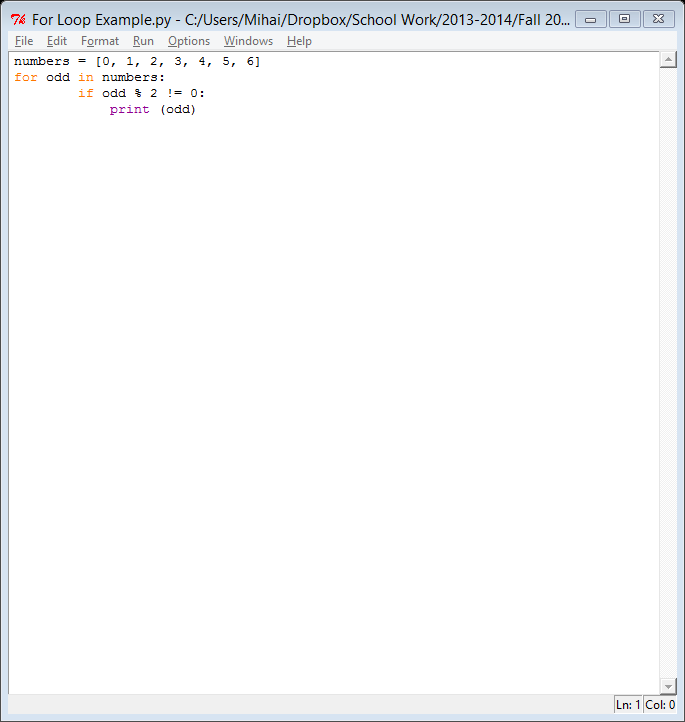
Last of the data types are dictionaries. Dictionaries are container data types that can store any data types or objects. Each element in a dictionary is a pair which is composed of a key and its value. The values are separated from the keys by a colon. Dictionaries are defined with brackets {} and entries, a key and its value, and are separated by commas. Existing keys cannot be changed, but new entry can be added. To access a dictionary entry the following syntax must be followed **dict[‘Key’];**. To change the key’s value, you must use the name of the key in quotations, **‘Key’**, surrounded by brackets and proceeded by the name of the dictionary. You may add entries to the dictionary with the following syntax, **dict[‘Gender’] = “Male”.** You can also delete entries or the entire dictionary. To remove the whole dictionary you may use the **del** statement followed by the name of the dictionary. To remove all entries but keep the dictionary you must use **dict.clear();** and to remove only an entry **del dict[‘Key’];** must be used.



Loops are common in most programming languages and Python is not any different. There are for loops, while loops, and nested loops. For loops uses set parameters and loops through the functions until the parameters are met. While loops only works if a value is true, once It becomes false then the loop stops. Nested loops are loops within loops. Python uses very specific syntax when it comes to loops. Indentation is used in Python is place for brackets or parentheses. A basic for loop looks like this **for x in y:**. This very basic for loop makes x whatever y. It is easier to think of y as a list. In lists, x becomes every value from the first index to the last. Every step of the loop the value of x changes. If y is a string, then x becomes the first character of the string and the following character at the next step of the loop. Another possibility is to use the **range()** statement. This stamen will automatically create a list of numbers starting from 0 up to the number specified in the parentheses. The basic syntax and use of a for loop and its output is written below.



It is possible to do much more with loops. In the next example, we create a list and make the loop print every odd integer in the list. To do this we must first have an understanding of the if statement. If statements will carry out operations only if the conditions of the if statement are met. The computer will think if this is the case, and then do the following. In order for the computer to figure out what number is odd, it is imperative to understand what odd means how to express it. It is much easier to output the odd numbers by telling the computer to print every number that is not even. It is much easier to do this because of the function of the % operator. This operator is called a modulus and it will divide two numbers and return the remainder. With a basic understanding of arithmetic, it is known that when a number is divided by 2 and the remainder is 0 then it is even, but when the remainder is not 0 then it is odd. Below is the final code and the output to show this logic.



Notice in the above code the indentation being used to separate the if statement from the for loop. With this syntax you can easily distinguish what a loop and what a statement does.

Python distinguishes itself from other languages by its readability. The language is very straightforward and anyone who has a small background in computer programming can understand the concepts and ideas behind the language. The designers of the language successfully made a programming language that is easy to read. The number of data types also helps improve readability. It is possible to confuse lists and tuples because the only difference between the two is mutability of lists as opposed to tuples but the straightforward application and use of the data types make Python a great language. In regards to orthogonality, Python has the ability to create very complex constructs intended to do many different things.

Python’s writability is very straightforward. There are simple statements and simple implementations that the language provides which makes the language very flexible. There are some downsides, for example, Python emphasizes on the idea of “up to” a number. It is this idea that may confuse some programmers. The very simplistic method of writing loops can be a both good and bad for a programmer. Indentation offers a very different approach to programming but this also increases the language’s writability in that it is much easier for programmers to write out blocks of code and separate them from one another in a very effective and efficient way.

Python has a very good error-checking. The way Python was developed, most errors are seen at run-time but there are some errors, like syntax errors, that are recognized after the environment saves the module. There are no real “compile-time” errors because Python does not have to be compiled. Handling the errors is not one of Python’s strong suits; if there is any kind of syntax error then the code will flat out stop and give an error. Python’s reliability as a language is seen through its high readability and easy writability. It is a simple to read and simple to write language even for someone who is not a programmer.

The cost of Python is arguably its greatest forte. Python is a free-language and tutorials are plenty. It is well documented and it is a well proliferated language. Saying anybody with a computer can code in Python for free is not an overstatement. Anybody can download the IDLE environment for free and begin coding in Python. There is a team that is constantly developing the language making it better and better. It has proven that it is a powerful language and is also reliable. The foundation of Python was to be a reliable and easy to use language.

References

afterhoursprogramming. (2013). *Pyhon*. Retrieved from <http://www.afterhoursprogramming.com/>

Python Software Foundation. (2013). *The python tutorial*. Retrieved from <http://docs.python.org/3/tutorial/index.html>

Rossum, G. V. (2009, January 13). [Web log message]. Retrieved from <http://python-history.blogspot.com/>

tutorialspoint. (2013). *Python overview*. Retrieved from <http://www.tutorialspoint.com/python/python_overview.htm>