**Outline**

Access the Python Development environment and follow the tutorial to gain an initial exposure to a programming language. Begin to develop an familiarity with basic programming concepts.

**Objectives**

* Use correct terminology to describe programming concepts;
* Describe the types of data that computers can process and store (e.g., numbers, text);
* Explain the difference between constants and variables used in programming;
* Use variables, expressions, and assignment statements to store and manipulate numbers and text in a program

**Materials**

* Python3 Development Environment at: //repl.it/
* Python Tutorial at: <http://www.letslearnpython.com/learn/>

**Accessing the Python3 Web IDE Environment**

Accessing the IDE

* Go to: <https://repl.it/>
* Select Python3
* Sign-up / Create an account
* Make sure you can remember your account information for the rest of the course.

Using the IDE

* Use the black area like a calculator to try simple statements or commands
* Use the white area to create programs with multiple statements

**Accessing the Tutorial**

Accessing the Tutorial

* Go to: <http://www.letslearnpython.com/learn/>
* Read up to “Lesson 3: Math”

**Level 1: Basic Math & Strings**

Access the Tutorial and start at “Lesson 3: Math”.

Questions

1. Complete “Lesson 3: Math – Math Basics” by typing the sample commands in the black area of the IDE.
   1. Create your own expression using 5 “+” and “-“ operators.
   2. List your expression and the result below.

5+7-90 =-78

1. Complete “Lesson 3: Math – More Operators” by typing the sample commands in the black area of the IDE.
   1. Create your own expression using 5 “\*” and “/” operators.
   2. List your expression and the result below.

6\*8/4= 12

1. Complete “Lesson 3: Math – More Division” by typing the sample commands in the black area of the IDE.
   1. Create one division expression that gives a whole number answer

8/4

* 1. And one division expression that gives a decimal number answer.

8/4/3

* 1. List your expressions and the results below.

A =2; b = 0.666~

1. Complete “Lesson 3: Math – Floats” by typing the sample commands in the black area of the IDE.
   1. Use the “round()” function for the expressions you created in question #3 above.
   2. List your “round()” expressions and the results they return below.

Round(8/4/3)=1

1. Read through “Lesson 3: Math – Comparison Operators”.
   1. Why do you think Equals is “==” instead of “=”?

*The equals is '==' instead of '=' because '=' is an assignment operator used to define variables whereas the '==' is a comparison operator used to compare variables.*

* 1. What does “=” mean?

*The '=' sign is used to define the value of a variable. For example, x = 6 and y = 'Hello'*

1. Complete “Lesson 3: Math – Practice” and “Lesson 3: Math – Practice Answers” by typing the sample commands in the black area of the IDE.
   1. Create an expression using 5 different operators that returns a “True” result
   2. And an expression using 5 different operators that returns a “False” result.
   3. List your expressions and the results returned below.
2. Complete “Lesson 4: Strings – Strings” and “Lesson 4: Strings – Examples” by typing the sample commands in the black area of the IDE.
   1. Explain why typing “apple” works and why typing apple without quotes gives an error.
   2. Also explain why “2 + 5” does not equal 7.
3. Complete “Lesson 4: Strings – Operators” by typing the sample commands in the black area of the IDE.
   1. Explain why typing “appl” + “e” works and why typing “apple” - “e” gives an error.
   2. Also explain why “Hello” \* 10 works but why “Hello” / 10 does work.
4. Complete “Lesson 4: Strings – Indexes” by typing the sample commands in the black area of the IDE.
   1. List the letters in your first name and the index for each letter in your first name.
5. Complete “Lesson 4: Strings – Indexes Examples” by typing the sample commands in the black area of the IDE.
   1. Explain why print(“Hello!”[4]) does not print “l”.
   2. What does print(“Hay, Bob!”[4]) print? For a hint try print(“Hay, Bob!”[3]) and print(“Hay, Bob!”[5])
6. Complete “Lesson 4: Strings – Rules” by typing the sample commands in the black area of the IDE.
   1. Explain why print(“Hello!”[7]) gives an error.

**Level 2: Booleans & Variables**

Access the Tutorial and start at “Lesson 5: Variables”

Questions

1. Complete “Lesson 5: Variables – Save a Value” by typing the sample commands in the black area of the IDE.
   1. What do you get if you type puppies / 3?

If puppies/3 is typed, python returns 12, which is 36/3

* 1. Why doesn’t typing kittens / 3 work?  
     Typing kittens / 3 does not work because the variable kittens is not defined.

1. Complete “Lesson 5: Variables – Assign a New Value” by typing the sample commands in the black area of the IDE.
   1. Explain how the following sequence of commands works:
      * puppies = 36
      * puppies = puppies / 6

puppies  
puppies = 36

In [12]:

puppies = puppies/6

In [13]:

puppies

Out[13]:

6.0

1. Read through “Lesson 5: Variables – Rules”.
2. Complete “Lesson 5: Variables – Math Operators” by typing the sample commands in the black area of the IDE.
   1. Explain what happens for following sequence of commands:
      * colour = “red”
      * puppies = 36
      * colour + puppies
3. Complete “Lesson 5: Variables – String Operators” by typing the sample commands in the black area of the IDE.
   1. Explain why the following commands give different results:
      * Color + day \* fishes
      * ( Color + day ) \* fishes  
        The commands give different outputs because python performs calculatin in BEDMAS form. Therefore, multiplication happens before addition in the first command but the second one performs operations inside the brackets first and then multiplication
4. Complete “Lesson 5: Variables – Indexes” by typing the sample commands in the black area of the IDE.
   1. What is the index of ‘r’ in “watermelon”?

The index of 'r' in watermelon is 4.

Write an expression using mynumber to return ‘r’  
fruit = "watermelon"

mynumber = 4

fruit[mynumber]

Out[14]:

'r'



1. Complete “Lesson 5: Variables – Assignments or Comparisons” by typing the sample commands in the black area of the IDE.
   1. What is the difference between “=” and “==”?

The "=" is used to assign a value to a variable in python whereas the "==" is used to check if a variable or expression is equal to another variable or expression.

* 1. Create your own mnemonic to remember this difference.  
     y = 7

x = 4 + 3

y == x

Out[15]:

True

1. Complete “Lesson 6: Errors – Examples” by typing the sample commands in the black area of the IDE.
   1. What doesn’t “friend” + 5 work?

The expression 'friend' + 5 does not work because integers cannot be added to string.

* 1. What is the difference between int and str?

The abbreviation int refers to an integer(number) and the abbreviation str stands for string, which is also known as text.

1. Read through “Lesson 6: Errors – Parts of an Error Message”.
   1. Is “friend” + 5 an example of:
      1. A Syntax Error?
      2. A Runtime Error?
      3. A Logic Error?

The expression "friend"+5 is an example of a syntax error because integers cannot be added to strings.

1. Read through “Lesson 6: Errors – Fixing Errors”.
   1. Use the ‘print’ command to print your first name and last name.
2. print("Raul","Singh")
3. Raul Singh
4. Complete “Lesson 7: Booleans – Types of Data” by typing the sample commands in the black area of the IDE.
   1. What is the value of: type(“True”)

The value of type("True") is a string because it is defined using double quotes.

* 1. What is the value of: type( True )

The value of type(True) is bool because a boolean has either the value True or False.

* 1. Why is the result different?

The result is different because in the first command, the input is in form of a string whereas in the second command the input is just True, which is a bool. If the input was true instead of True, then it would have raised a Name Error.

1. Complete “Lesson 7: Booleans – What Is A Boolean” by typing the sample commands in the black area of the IDE.
   1. Why do you think that having a Boolean data type is important in computer programming?

Having a boolean data type is important because it acts like a switch for certain functions.

1. Complete “Lesson 7: Booleans – Trying Out Booleans” by typing the sample commands in the black area of the IDE.
   1. Why do you think that there is no Maybe” Boolean data value in computer programming?

There is no Maybe data value in computer programming because everything must be either True or False and a computer can only check if something is True or False. If the computer calculates to see if something is True or False, there can no maybe because it will only figure out if something is True or False.

**Level 3: Lists & Logic**

Access the Tutorial and start at “Lesson 7: Booleans”

Questions

1. Complete “Lesson 7: Booleans – AND Comparisons” by typing the sample commands in the black area of the IDE.
   1. Try the following Python statements and record the results.
      1. True and True
      2. True and False
      3. False and True
      4. False and False
2. **True** **and** **True**
3. Out[17]:
4. True
5. In [18]:
6. **True** **and** **False**
7. Out[18]:
8. False
9. In [19]:
10. **False** **and** **True**
11. Out[19]:
12. False
13. In [20]:
14. **False** **and** **False**
15. Out[20]:
16. False
    1. Explain if there are any other combinations of True / False.

There are no other combinations of True/False. Because booleans can only have two values, there can only be 4 combinations of True or False.

* 1. Explain how the AND operator is similar to a math operator and how it is different.

1. Complete “Lesson 7: Booleans – OR Comparisons” by typing the sample commands in the black area of the IDE.
   1. Try the following Python statements and record the results.
      1. True or True
      2. True or False
      3. False or True
      4. False or False
2. **True** **or** **True**
3. Out[21]:
4. True
5. In [22]:
6. **True** **or** **False**
7. Out[22]:
8. True
9. In [23]:
10. **False** **or** **True**
11. Out[23]:
12. True
13. In [24]:
14. **False** **or** **False**
15. Out[24]:
16. False
    1. Explain how the OR operator is similar to the AND operator and how it is different.

*The or operator is similar to the and operator because it is used between two values to check for something. It is different from the AND operator because AND requires both inputs to be True to return True but the OR operator returns True even if one input is True.*

1. Complete “Lesson 7: Booleans – NOT Comparisons” by typing the sample commands in the black area of the IDE.
   1. Try the following Python statements and record the results.
      1. not (True or True)
      2. not (True or False)
      3. not (False or True)
      4. not (False or False)
2. **not** (**True** **or** **True**)
3. Out[25]:
4. False
5. In [26]:
6. **not** (**True** **or** **False**)
7. Out[26]:
8. False
9. In [27]:
10. **not** (**False** **or** **True**)
11. Out[27]:
12. False
13. In [28]:
14. **not** (**False** **or** **False**)
15. Out[28]:
16. True
    1. Explain how the combination of the NOT & OR operators is similar to the AND operator by itself and how it is different.

The combination of the ***NOT & OR*** operators is similar to the ***AND*** operator becuase they return the same output. The combination is different from using NOT & AND because then the output will be inverted

1. Complete “Lesson 7: Booleans – Expressions” by typing the sample commands in the black area of the IDE.
   1. Explain why the following two Python statements give different results.
      1. not (True or True)
      2. not True or True

These statement give different results because the first one is returning the opposite of (True or True), which when simplified becomes ***not True***, which returns False. The second command is asking if either ***Not True*** or ***True*** are true. One of the booleans is True, so the output returned is True.

**not** (**True** **or** **True**)

Out[29]:

False

In [30]:

**not** **True** **or** **True**

Out[30]:

True

* 1. Explain why the following two Python statements give the same results.
     1. not (True and True)
     2. not True and True

These statements give the same output because in the first statement, ***(True and True)*** is True but there is a ***not*** in front, which returns False. In the second statement, ***not True and True*** is false because the and operator requires both inputs to be True to return true, but in this statement, the input ***not True*** is ***False***.

**not** (**True** **and** **True**)

Out[31]:

False

In [32]:

**not** **True** **and** **True**

Out[32]:

False

1. Complete “Lesson 7: Booleans – Practice” by typing the sample commands in the black area of the IDE.
   1. Create three more practice expressions similar to those in the tutorial.
2. name = "Genius"
3. name == 'Genius'
4. Out[33]:
5. True
6. In [34]:
7. name == 'Genius' **and** 4==1
8. Out[34]:
9. False
10. In [35]:
11. name == name+'s'
12. Out[35]:
13. False
    1. Provide the results for your practice expressions
14. Complete “Lesson 8: Lists – A Collection of Objects” by typing the sample commands in the black area of the IDE.
    1. Create a list of your favorite sports teams.
    2. Assign your list to a variable.
    3. Confirm that your variable and your list are the same.
15. ['Raptors','Maple Leafs','Blue Jays']
16. Out[36]:
17. ['Raptors', 'Maple Leafs', 'Blue Jays']
18. In [37]:
19. teams = ['Raptors','Maple Leafs','Blue Jays']
20. In [38]:
21. ['Raptors','Maple Leafs','Blue Jays'] == teams
22. Out[38]:
23. True
24. Complete “Lesson 8: Lists – List Indexes” by typing the sample commands in the black area of the IDE.
    1. What is the list index of the last team in your list of favorite sports teams.
    2. In the tutorial, the error produced by typing “fruit[3]” is an example of:
       1. A Syntax Error?
       2. A Runtime Error?
       3. A Logic Error?

*It's actually an Index Error.*

In [39]:

fruit = ["apple", "banana", "grape"]

fruit[3]

---------------------------------------------------------------------------

IndexError Traceback (most recent call last)

<ipython-input-39-cb84ff869027> in <module>()

**1** fruit = ["apple", "banana", "grape"]

----> 2 fruit[3]

IndexError: list index out of range

1. Complete “Lesson 8: Lists – Practice” and “Lesson 8: Lists – Practice Answers” by typing the sample commands in the black area of the IDE.
2. fruit = ["apple", "banana", "grape"]
3. fruit[3]
4. ---------------------------------------------------------------------------
5. IndexError Traceback (most recent call last)
6. <ipython-input-39-cb84ff869027> in <module>()
7. **1** fruit = ["apple", "banana", "grape"]
8. ----> 2 fruit[3]
9. IndexError: list index out of range

NOTE: Starting with Lesson 9 you should use the WHITE area of the IDE for entering example code with multiple statements.

1. Complete “Lesson 9: Logic – Making Decisions” by typing the sample commands in the white area of the IDE.
   1. Modify the tutorial code to print “Hi Alfred!” based on a decision using numbers
2. mynumber = 5
3. **if** mynumber == 5:
4. print("Hi Alfred!")
5. Hi Alfred!
6. Complete “Lesson 9: Logic – Adding A Choice” by typing the sample commands in the white area of the IDE.
   1. Modify the tutorial code to print your first name or your last name based on a choice (using “else”).
7. math=**False**
8. first\_name='Raul'
9. last\_name='Singh'
10. **if** math:
11. print(first\_name)
12. **else**:
13. print(last\_name)

Raul

1. Complete “Lesson 9: Logic – Adding Many Choices” and “Lesson 9: Logic – Practice” by typing the sample commands in the white area of the IDE.
   1. Modify the tutorial code and “elif” statements to make a choice using at least 4 of your friends names.
2. friend\_no=3
3. friend\_names=['Saj','Krish','Parmveer','Tyler']
4. **if** friend\_no == 1:
5. print('Hello',friend\_names[0])
6. **elif** friend\_no == 2:
7. print('Hello',friend\_names[1])
8. **elif** friend\_no == 3:
9. print('Hello',friend\_names[2])
10. **elif** friend\_no == 4:
11. print('Hello',friend\_names[3])
12. Hello Parmveer