**Name : ID:**

**Lab # 2: Getting the data inputs using PYTHON**

**OBJECTIVES**

In this lab you will learn:

- Python variable

- Data Casting

- User input

- Representing and operating on numbers

**Creating Variables**

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

**Task 1**

**Example**

**x = 5  
y = "John"  
print(x)  
print(y)**

**Task 2**

Variables do not need to be declared with any particular type, and can even change type after they have been set.

**x = 4       # x is of type int  
x = "Grams" # x is now of type str  
print(x)**

**Task 3**

**Get the Type**

You can get the data type of a variable with the type() function.

**Example**

**x = 5  
y = "John"  
print(type(x))  
print(type(y))**

**Task 4**

**Casting**

If you want to specify the data type of a variable, this can be done with casting.

**x = str(3)    # x will be '3'  
y = int(3)    # y will be 3  
z = float(3)  # z will be 3.0**

**print(x, type(x))  
print(y, type(y))**

**print(z, type(z))**

**Task 5**

**Single or Double Quotes**

String variables can be declared either by using single or double quotes:

**Example**

**x = "John"  
# is the same as  
x = 'John'**

**Task 6**

**Example**

**x = "Python"  
y = "is"  
z = "awesome"  
print(x, y, z)**

You can also use the + operator to output multiple variables:

**Example**

**x = "Python "  
y = "is "  
z = "awesome"  
print(x + y + z)**

Notice the space character after "Python " and "is ", without them the result would be "Pythonisawesome".

For numbers, the + character works as a mathematical operator:

In the print() function, when you try to combine a string and a number with the + operator, Python will give you an error:

**Task 7**

**Example**

**x = 5  
y = "John"  
print(x + y)**

**PYTHON DATA TYPES**

**Built-in Data Types**

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

|  |  |
| --- | --- |
| Text Type: | str |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple, range |
| Mapping Type: | dict |
| Set Types: | set, frozenset |
| Boolean Type: | bool |
| Binary Types: | bytes, bytearray, memoryview |
| None Type: | NoneType |

**OBTAINING USER DATA**

The most general means of obtaining information from the user is the **input()** function. When Python executes this command, it will wait for the user to enter a string of characters until the user hits the Enter key.

These characters are then assigned as a string to a variable. Usually, some form of user prompt will be required (i.e., the question posed to the user). This can be accomplished via a separate print statement or as an argument to the function. Below is a simple example which asks the user for their name and then prints it back out.

**Task 8**

**print( "What is your name?" )**

**n = input()**

**print( "Hello", n )**

Alternately, this can be shortened with the following:

**n = input("What is your name? ")**

**print( "Hello", n )**

**Task 9**

It is important to remember that this function **[input( )]** always returns a string variable. If the entered data is numeric, it must be turned into either a float or integer.

This can be accomplished via the **float( )** and **int( )** functions. For example

**W = float(input("What is your weight in kilograms ?"))**

# one kilogram is approximately 2.2 pounds

**P= W\*2.2**

**print("You weight is ",P,"Pounds." )**

**Task 10**

Let’s consider how we might create a simple **Ohm’s law calculator**. Before we start coding, we must define exactly what we wish the program to do and create a logical outline. This outline is not written in python but rather a simplified form of English which shows the steps required to solve the problem. One line of pseudo code might correspond to one line of Python. Alternately, it might correspond to many lines of Python. Pseudo code is not tied to a specific language.

In our example, we shall use Ohm’s law in the form **V=I\*R**. Here’s the pseudo code:

1. Give the user directions.

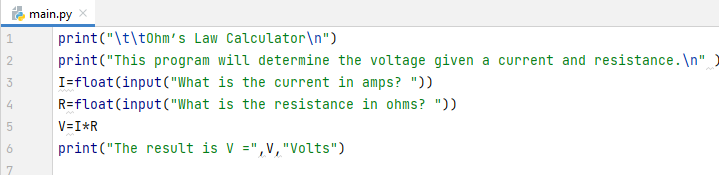
2. Ask the user for current in amps.

3. Ask the user for the resistance in ohms.

4. Compute the voltage from V=I\*R.

5. Print out the resulting voltage in volts.

Step one might be very short or very detailed. It all depends on the complexity of the program. Note that in steps two, three and five, we have specified the units. This is important. The user should not assume that it’s OK to enter a current in milliamps, for example. It is worth noting that input( ) can deal with an exponent so a value such as 1.2 milliamps can be entered as 1.2e-3 So here’s the Python code based on this pseudo code.

****

* The \t and \n sequences are used to enter a Tab and a Newline, respectively. This makes the print out look a little nicer.
* Enter this program, and run it.
* Try the values 2 amps and 10 ohms. The result should be 20 volts.
* Also try some large and small values, for example, test the program with a current of 3 milliamps (3.0e−3) and 5 k ohms (5.0e3).
* The result for this should be 15 volts.
* **It is important to always test your code!** Never assume that it runs properly without exhaustive testing.

**LAB ASSIGNMENT 1**

Based on Task 10, create two new programs using Ohm’s law.

1. The first version should ask for current and voltage to determine and print the resistance. (Test the first program using 4 amps and 20 volts.)
2. The second should ask for voltage and resistance to determine and print the current.

(Test the second program with 12 volts and 5 k ohms.)