Programming in Python

Python is a widely-used programming language that is available on most modern operating systems and computers. On the Pi, Python is a great way to write software and games, and to control the General Purpose Input/Output (GPIO) pins. Your First Python Program “Hello, World!” is a short program that people often write as their first program in a new environment or language. It is a test program, and only displays the message “Hello, World!” to the user. In Python, you can write this program with one line of code, and it is a good way of introducing you to the integrated development environment (IDE) and how to run Python programs.

Python Basics

Note: If you have not yet done so, it’s important that you change your keyboard layout as follows:
Menu > Personalize > Mouse and Keyboard Settings > Keyboard > Keyboard Layout > English (US) >
OK > OK

Activity 1: Run a simple Python command

1. Click the Raspbian **Menu** button, point to **Programming**, and then click **Python 3 (IDLE)**.

2. Type **print("Hello, World!")** and then press **Enter**. You can use apostrophes (‘) instead of quotation marks (“) if you wish.

The first thing to note is that the window is titled “Python Shell”. The Python Shell works like the Linux terminal – it runs commands that you type here when you press the Enter key.

Python programs (or “scripts”) are text files that contain all of the commands that you want to run. You can use any text editor to create these files, but there is one built-in to the Python packages on the Pi.

Activity 2: Create a Python script

1. In the Python Shell, click **File** and then click **New File**.

2. Type **print("Hello, World")**

3. Click **File** and then click **Save As**. Save your file with the name **hello.py**.

4. Click **Run** and then click **Run Module**. The results of your program are shown in the Python Shell.

Instead of running your program within the Python IDE, you can also run it from a Linux terminal.

Activity 3: Run a Python script from a Linux terminal

1. Click the Raspbian **Menu**, then click **Accessories**, then click **Terminal**.

2. Type **python3 hello.py**.

You can also create a desktop shortcut, but it's a little more complicated than a Windows shortcut. But it's a good way to show you how important little details are, and you get to see what's under the hood! It's not that bad. Programmers must learn to look under the hood. A shortcut is a short text file that creates an icon on your desktop and describes what happens when the user clicks it.

Activity 4: Create a desktop shortcut for your Python script

1. Click the Raspbian **Menu** button, then click **Accessories**, and then click **Text Editor** to open the Leafpad editor.

2. Type the following text into the document exactly as you see it including blank spaces. Note that the letter after = in line four is a lower case L. There are three blank spaces in the fourth line. Small details matter, so be careful!

[Desktop Entry]
Type=Application
Name=My Hello World Script
Exec=lxterminal –e "python3 /home/pi/hello.py"
Terminal=False
Categories=None

3. Click **File**, and then click **Save As**.

4. Save the file to your desktop with the name **hello.desktop**.

A desktop icon for your hello script should appear. If you double-click the desktop icon, you should see a terminal window open and display the text *Hello, World!* However, the terminal window closes as soon as the program completes. While you are developing your programs, it can be useful to keep the program running (and keep the terminal window open) until the user presses a key.

Activity 5: Cause your program to pause so you can view it

1. Add the following command as the second and last line in your hello.py file (careful! This goes in your **.py** file, NOT your .desktop file!): **input("Press ENTER to close this window.")**

2. Click **File** and then click **Save**.

3. Run the program again by double-clicking the desktop icon and you should now be able to view the output from the program. Press **ENTER** to close the window.

Comments in Python

As your Python scripts become more complicated, you can put in explanations of what parts of your code do. This is to help you in case you need to a change a program that you wrote a long time ago. It also helps if somebody else has to understand your code.

Activity 6: Add a comment to your Hello World! script

1. On a new line at the top of the Python script, type **#This is my Hello World! program.**
2. Run your Python script again (using any of the three methods we learned) and you'll see that its behavior has not changed. The comment is only for documentation purposes.

Performing Basic Arithmetic

You have seen the function print() that displays information to the user. From a Python .py file, you need to use this to show the result of arithmetic calculations. But from the Python Shell, you can type the expression and the result is automatically shown.

Activity 7: Perform arithmetic in the Python shell

1. In the Python Shell, type the following expression and then press Enter: **5 + 4**
2. In the Python Shell, type the following expression and then press Enter: **8 - 3**
3. In the Python Shell, type the following expression and then press Enter: **6 * 7**
4. In the Python Shell, type the following expression and then press Enter: **9 / 2**
5. In the Python Shell, type the following expression and then press Enter: **9 // 2**

Note that each expression is calculated and the result is displayed. *What is the difference between the operations in steps 4 and 5?*

Using Variables in Python

Variables are one of the key concepts you must master in programming. A variable is an area of the computer's memory that you give a name to. You can store information in these variables and access it later. To create a variable, you must specify a name, followed by an equals sign, and then the value it should contain.

Activity 8: Create and display two variables

1. In the Python Shell, type this command: **number_1 = 8 + 7**
2. In the Python Shell, type this command: **number_2 = 6 * 3**
3. In the Python Shell, type this command: **print(number_1, number_2)**

These commands created two variables, named number_1 and number_2 and displayed them with the print command. In the variables, we stored two values, which are the results of arithmetic operations: 15 and 2. Think of it like this:

number_1

15

number_2

18

Variable names (like number_1 and number_2 above) must start with a letter (a–z, or A–Z). The remaining characters in the name can be letters, numbers, or an underscore (_). You cannot use punctuation marks or other special characters in the names of variables, and there are several words (such as “print”) that you cannot use as a variable name because they are already used for something else.

In Python, variables can hold any type of information. Numbers and strings are both commonly used.

Activity 9: Create and display a variable that stores a string

1. In the Python Shell, type this command: **string_1 = “This is fun!”**
2. In the Python Shell, type this command: **print(string_1)**

These commands created one variable, named string_1, and displayed it with the print command. In the variable, we stored a string value, which is a series (or string) or characters. Think of it like this:

string_1

This is fun!

Well, that really was fun! That’s a very basic introduction to computer programming in Python. I hope you enjoyed it! The things you learned here are foundations to build upon if you want to create more complex programs.

First Principles

- *Encapsulation* – if you want to print in Python, you must use the **print** function. This function controls how you can display information, and you must use it properly to send output to the screen. You don’t have direct access to the computer’s display. The print command controls how you can produce screen output.
- *Abstraction* – when you create a variable in Python, the value you store in the variable is stored in memory. You only need to give the variable a name and a value. The details are hidden from you. You don’t see how many bytes of

memory the variable requires, and you don't need to know the memory addresses that are used. Those are details that you really don't need to know.

- *Information Hiding* – when you create a string in Python, it is actually multiple characters. Each character requires one byte of memory. How those bytes are arranged is hidden from you. All you do is specify the characters in the string, and Python takes care of the details about how to make sure the characters are stored in the right order.

GenCyber Cybersecurity Concepts:

- *Defense in Depth*: allow multiple layers or levels of security in your programs so if one layer fails, another layer of security is already in place to stop the attack. Ex: company data is secured by a firewall, passwords and encryption.
- *Confidentiality*: information is not disclosed to unauthorized users. Ex: medical information of a patient at a hospital, bank account information.
- *Integrity*: system has not been modified or destroyed in an unauthorized manner. Ex: computer system is virus-free and uncompromised.
- *Availability*: information is accessible and usable upon demand. Ex: Student grades can be viewed by student and principal but modified by the teacher.
- *Think Like an Adversary*: Always be prepared to think ahead and be defensive. Ex: virus and malware attacks and prevention
- *Keep It Simple*: Try to keep your programs simple and easy to follow. Ex: Complex coding is not easy to understand and can have too many bugs.

Lab Exercise #1

Write a Python program that asks a user for his/her name and age. The program should then tell the user how many years until he/she can vote. Assume the user is under 18 years of age. To finish this lab, you'll need to know a few more things about Python:

1. To ask a user for input and store the user's input in a variable, use a command like this:

   my_variable = input("Enter something.")

   This command asks the user to enter something and stores whatever he/she types in a variable named my_variable.

2. When you receive input from a user, it is stored as a string, even if they type a number. If you want to perform calculations on the input (assuming they type a number), you must convert from string to a number. Here's how you can convert a variable from string to integer:

my_variable = int(my_variable)

3. To perform a calculation using a variable and a number, you can use a command like this:

   difference = my_variable – 15

   This command takes the number stored in my_variable, subtracts 15 from it, and stores the result in the variable named difference.

4. To display output that contains literal strings and variable values, you can use a command like this:

   print(“The difference is”, difference)

   This command displays whatever you put between the quotation marks, followed by the contents of the variable named difference. You can print as many items like this as you wish. Just separate them by commas inside the parentheses.

Notes about programming style:

1. You should include comments in your code to explain what it does. It’s also a good idea to include your name.
2. When asking for input, display a helpful and meaningful message.
3. Use meaningful variable names! If you are asking for someone’s age, use a variable named age or something similar.
4. Make sure your output looks good and is understandable to the user. Spell your words correctly and include blank spaces and punctuation where they belong.

Check your project for completion:

1. Does your program ask for both required pieces of information (name and age) and store each in a variable?
2. Does your program correctly calculate how many years until voting age?
3. Does your program clearly display the users name and the number of years until voting age?
4. Does your program use meaningful variable names?
5. Does your program have helpful and descriptive comments?
6. Run your program three different ways: from the Python IDE, from a Linux terminal, and from a desktop icon.
7. Show your program to your team leader to check it out!

Lab Exercise #2

Write a simple Python calculator. The program should ask the user for two numbers and display the sum, difference, product, and quotient of the numbers. A session might look like this:

Enter your name: GenCyber

Enter your first number: 14

Enter your second number: 7

The product of 14 and 7 is 98.

The sum of 14 and 7 is 21.

The difference of 14 and 7 is 7.

The quotient of 14 and 7 is 2.

What GenCyber concepts did you demonstrate: _____

Check your project for completion:

1. Does your program ask for two numbers and store each in a variable?
2. Does your program correctly calculate the product, sum, difference, and quotient of the two numbers?
3. Does your program clearly display the two numbers along with the product, sum, difference, and quotient?
4. Does your program use meaningful variable names?
5. Does your program have helpful and descriptive comments?
6. Run your program three different ways: from the Python IDE, from a Linux terminal, and from a desktop icon.
7. Show your program to your team leader to check it out!

Learning More

Want to learn more about Python? There are lots of free resources out there. Python is the most popular language for colleges in introductory programming courses. A great site for learning more about Python is:

https://www.python.org/

Here you can download Python, learn all about the language, and discuss Python with other programmers.

Also, remember that Google is your friend! If you want to know how to do something in Python, usually a quick Google search will find all the answers you need. For example, if you want to know how to do loops (repetition) in Python, just do a Google search for *Python Programming Loops* and you'll find lots of answers!

Enjoy your programming! This lesson has given you a start. You can build on this foundation bit by bit and before you know it, you'll be writing really cool Python programs!

Simple drawing with Turtle

“Turtle” is a python feature like a drawing board, which lets you command a turtle to draw all over it! You can use functions like turtle.forward(...) and turtle.left(...) which can move the turtle around.

Before you can use turtle, you must import it. We recommend playing around with it in the interactive interpreter first, as there is an extra bit of work required to make it work from files. Just go to your terminal and type:

```
import turtle
turtle.forward(25)
turtle.left(30)
```

The turtle.forward(...) function tells the turtle to move forward by the given distance. turtle.left(...) takes a number of degrees which you want to rotate to the left. There is also turtle.backward(...) and turtle.right(...), too.

Want to start over? You can type turtle.reset() to clear the drawing that your turtle has made so far. The standard turtle is just a triangle. That’s no fun! Let’s make it a turtle instead with the turtle.shape() command:

```
turtle.shape("turtle")
```

If you put the commands into a file, you might have recognized that the turtle window vanishes after the turtle finished its movement. (That is because Python exits when your turtle has finished moving. Since the turtle window belongs to Python, it goes away as well.) To prevent that, just put turtle.exitonclick() at the bottom of your file. Now the window stays open until you click on it:

```
import turtle
turtle.shape("turtle")
turtle.forward(100)
turtle.exitonclick()
```

Drawing a square

You’re not always expected to know the answer immediately. Learn by trial and error! Experiment, see what python does when you tell it different things, what gives beautiful (although sometimes unexpected) results and what gives errors. If you want to keep playing with something you learned that creates interesting results, that’s OK too. Don’t hesitate to try and fail and learn from it!

Lab Exercise: Draw a square.

Hint: For a square, you will probably need a right angle, which is 90 degrees. Type the following code into a file and save it as myTurtle.py.

```
turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
```

If you want to get creative, you can modify your shape with the turtle.pensize(...) turtle.shape(…) and turtle.color(...) functions. How do you use these functions? Before you can use a function, you need to know its signature (for example what to put between the parentheses and what those things mean.) You can modify the color with turtle.color(colorstring). colorstring values include but are not limited to “red,” “green,” and “blue.”

Creating Turtle Objects:

```
import turtle           # Allows us to use turtles
wn = turtle.Screen()    # Creates a playground for turtles

anup = turtle.Turtle()  # Create a turtle object,assign to your name
anup.color("hotpink")
anup.pensize(5)
anup.shape(“turtle”)

anup.forward(50)        # Tell anup to move forward by 50 units
anup.left(90)           # Tell anup to turn by 90 degrees
anup.forward(30)        # Complete the second side of a rectangle

wn.mainloop()           # Wait for user to close window
```

Exercise: On your own

Can you draw a rectangle?
Can you draw a triangle?
Can you draw a star?