# National Coding Symposium
# Python Activity Lesson Plan

Designed by Stephanie Ludi for Coding Symposium 2022
Based on excerpts from w3schools.com
February 2022

| | |
|---|---|
| Audience: | Students interested in more advanced coding concepts.<br>Intermediate/advanced typing, editing, formatting skills needed.<br>Designed with High school and college age students in mind |

## Learning targets (what do I want the students to learn?)

| | |
|---|---|
| Objective #1 | **What is the Python coding language and what is it used for?** |
| Objective #2 | **How to code variables in Python** |
| Objective #3 | **How to code if,else and logic using Python** |

## Materials needed (What I need to teach the lesson?)

Technology:

- Mac with TextEdit or Windows with Notepad++
- Python installed on your computer

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have Python installed on your computer, then you can download it for free from the following website: https://www.python.org/

**Note: For computers on a school network, you should get assistance from your IT staff.**

**ECC High School Readiness Checklist sections referenced for this lesson:**

| | |
|---|---|
| Grade band: | Grades 5 - 8 |
| ECC area: Assistive Technology | I can type at a minimum of 45 words per minute with 80% accuracy (five words per grade-- 45 by 9th, 50 by 10th, etc.).<br>I can create, edit, save, and share a variety of formats such as Microsoft Word, Google Docs, Google Sheets, Excel, PDF, etc.<br>I can use NVDA, JAWS, or Chromevox to complete word processing projects.<br>I can use commonly used keyboard commands for reading, editing, and navigation. |
| Level: (Required, Novice, Accomplished or Proficient) | Required Skills |

# Lesson 1 Welcome to Python

Python is a popular programming language that has been around since the early 1990's. It has gained popularity in the last 10 years and is now commonly recommended as one of the first programming languages to learn. Python 3 is the latest version of the Python language. This tutorial will follow Python 3.

## 1.1 What is Python used for?

- web development (server-side)
- software development
- mathematics-focused software

## 1.2 What can you do with Python?

- Python can be used on a server to create web applications.
- Python can be used to create simple games and process images.

- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle large sets of data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

## 1.3 Advantages to Python

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.).
- Python has a simple syntax similar to the English language, which makes it easier to learn than other programming languages.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. You do not need to compile it before you run the program, as is common in other programming languages.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

## 1.4 Points to Keep in Mind as We Go Through this Tutorial

In this tutorial Python will be written in a text editor (such as Notepad++ on Windows or TextEdit on MacOS). It is possible to write Python in an Integrated Development Environment, such as Microsoft VSCode (with the Python extension), Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

## 1.5 Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

## 1.6 On to the activity

As mentioned earlier, Python is an interpreted programming language, this means that as a programmer you write Python files and save them with the extension .py from your text editor. Then you put those files into the python interpreter to be executed.

In your text editor (e.g. Notepad++ or TextEdit), type the following code:

```
print("Hello, World!")
```

then save the file as helloworld.py

Next, open your command line (e.g. Terminal or Command Line), navigate to the directory where you saved your file, and run the Python file by entering:

```
C:\Users\Your Name>python helloworld.py
```

You start the command with python to indicate that you are going to your file through the Python interpreter.

Then you enter helloworld.py as it is the name of your python file that you want to run.

The output is the traditional first output for a first program written in a programming language – Hello, World!

High Five! You have written your first Python program.

# Lesson 2 Syntax

## 2.1 Indention

During the overview in Lesson 1, we mentioned that Indentation is used to define blocks of code to help group parts together that need to be (e.g. use with If-else, for or while). Indentation refers to the spaces at the beginning of a code line. Many other programming languages use brackets to show the start and end.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code. A block is relative as it can be one or more lines that go together.

Let's take the following example, which you can enter into a file called example1.py

```
if 5 > 2:
  print("Five is greater than two!")
```

The line of code that has the print command has two spaces at the beginning of the line. This indicates that the print statement is run if 5 >2 is true.

The number of spaces is up to you, but it needs to be at least 1 and it needs to be consistent within the same block of code.

Take the following example:

```
if 5 > 2:
 print("Five is greater than two!")
        print("Five is still greater than two!")
```

The first print statement is indented one space and the second print statement is indented 8 spaces. If you try to run this program, Python will give you an error.

Try it and see for yourself. Now fix it so that the two print statements have the same number of spaces for their indentation and rerun the program.

## 2.2 What about variables?

In programs you are often processing data, including that entered by the users. That is where variables come in as they act like a bucket that the value of something is kept and that can be changed or presented as needed by the program.

In Python, a variable is created when you assign a value to it. In other languages a special command is often needed to create a variable. Not so in Python.

Here are two examples of what variables can look like in a program:

```
x = 5
y = "Hello, World!"
```

If you saved these two lines to a Python program and ran it, nothing would happen. Yes, the number 5 is assigned to x and Hello, World is assigned to y but that is it. Nothing is output. To have that program do something that you as the programmer can see, use that print command again.

In your text editor, enter the following:

```
x = 5
y = "Hello, World!"
print(x)
print(y)
```

Save this file to example2.py and then run it. What happens?

If the output was the number 5 followed by Hello, World! Then you can now see that your program output those values for you to see.

Go back and edit the program to add another variable and then output it in the same way.

If you have programmed in other languages before you may notice something else different. In most other programming languages, you need to say that a variable is an integer or a string of characters. Not in Python. No formal type declaration is needed in Python.

As you use more variables, you will name them with more descriptive names than x and y.  When you name your variables, you should keep in mind that they are case sensitive so Python will treat a variable with a lowercase x as a separate variable that is an uppercase X.

## 2.3 Variables can Hold Multiple Things

In the previous example, each variable contained one value.  Such as:

```
x=5
```

Sometimes a variable needs to contain multiple things.

There are different ways of doing that in Python: sets, lists, and tuples to name a few.

This is one example of a set of items:

```
thisset = {"bike", "boat", "house", "car"}
```

This is a set of items where there are not repetitions.

The following example is of a list:

```
listx = ["apple", "windows", "linux"]
```

Lists can be changed and the items are in an order.   You will see this in action in a later tutorial.

# Lesson 3 – Loops and Logic

There will be times when you need your code to behave differently based on the value of something or the need for repetition (as examples).  Programming languages have mechanisms to handle those situations.  We will explore some in Lessons 3 and 4.

## 3.1. If..Else and Logic

Python supports the usual logical conditions from mathematics:

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

These logical conditions are used in if..else statements and in loops.

Using the following code as an example:

```
a = 5
b = 10
if b > a:
  print("b is greater than a")
  print("b is definitely a bigger number than a")
```

Enter this code and save it as if1.py

The code block associated with the if statement (in this case two print statements) only executes if b is greater than a.  Is it?

Try out the program and see if in fact both print statements are output, each on their own line.

## 3.2. Elif

Sometimes you need to have more than one comparison.  Elif lets you compare another thing if the first if was not true.  For example, enter the following code:

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

Note that for the logical expression of a equals b that there are two equal signs.  This is common in several programming languages.

Save this example as if2.py and try it out to confirm what the output is.  Then, try different values before rerunning the program to see how the values affect what gets printed.

## 3.3. Nested Ifs

Sometimes you need to have if statements inside of other if statements.

```
x = 20

if x > 10:
  print("Above ten,")
  if x > 15:
    print("and also above 15!")
  else:
    print("but not above 15.")
```

You can save this example as if3.py

In this example, there are two new things.  One is an else by itself.  The else is basically shorthand for any other conditions if the earlier ones are not true.  You can have this without there being any nesting.

The main item being showcased with this example is that once the first if is true there is another if inside that is tests the values.   Go back and change the values to predict what the output is.  Note that unless the text is written to make sense, the text that is output may not make logical sense.  For example, the text may not match the logic or you might talk about your favorite pet.

# Lesson 4 – For Loops and User Input

In this lesson, we will cover for loops and basic user input so that you can add some interaction to your programs.

## 4.1 For Loops

For loops allow you to repeat a block of code statements over a set of items like characters in a string, a set of items, a dictionary of items, etc.  The following examples will show the for loop in action.

### 4.1.1 Example 1

Enter the following code as its own program:

```
food = ["pizza", "milkshake", "candy", "hot dog"]
for x in food:
  print(x)
```

Save the file as example1.py

In the example above, the for loop will run the code associated with it (in this case the print statement), for each item in the list of fruit.

### 4.1.2 Example 2

Enter the following code as its own program:

```
for x in "pizza":
  print(x)
```

Save this program to example2.py

Try to predict what the output it. Then run the program and compare the actual output with your prediction. Were you correct?

You may have thought that the word pizza would be output all on the same line, but the for loop is iterating through the string one character at a time. Because of that x holds one character at a time and thus prints it one character per line at a time.

### 4.1.3 Break statement

You can use a break statement to stop the code being run in the for loop, yet the program can continue to run from any code after the for loop.

Example 3

Enter the following code:

```
food = ["pizza", "milkshake", "candy", "hot dog"]
for x in food:
  if x == "milkshake":
    break
  print(x)
```

Save the program as example3.py

What happens? What gets printed? Notice that once the break statement is hit by the interpreter, it stops executing the for loop. So the third is never visited in the code and is not output to the screen.

### 4.1.4 What about iterating a specific number of times?

You can also do that with the For loop. Take the following example:

Enter the following code:

```
for x in range(6):
  print(x)
```

Same the program as example4.py

Predict what you think will happen before you run the program. This one is a tricky one.

In this short example, the for loop basically runs the code (in this case the print statement) 6 times and so it prints out the number of which iteration it is on each run through the code and stops at the 6th iteration.

While people tend to start counting at 1, computers start at 0. So instead of the output being 1-6, the computer outputs 0 through 5.

If you want to force the program to count within a specific range of numbers you can do that. Take the example:

```
for x in range(1, 5):
  print(x)
```

In this example you use two different numbers with range where the first one is the starting number and the second number is the ending number. You can enter this in its own program to test this out.

## 4.2 Basic User Input

So far we have coded specific values for variables. Most programs require input from the user or through a calculation or some other ways of gathering data values.

There is a slight variation in how basic user input is called for in a program between different versions of Python. We are using the new function call.

Example 1:

Enter the following code:

```
firstname = input("Enter your first name:")
print("Your first name is " + firstname)
```

Save this program as input1.py

Run this program and notice that you are being prompted to enter your first name. The program will wait for you to enter something.

Once you do it will then print it to the screen. The first line of code uses the input function call that displays the prompt and then assigns the value input to the variable called firstname.

The second line then prints out a string that says Your first name is and the text that you entered is then appended to the end.

This is a simple way of getting input, and there are many others that you can learn about as you continue to learn more about Python.

From this very simple input method, let’s revisit a prior example:

Let’s say that you are asked to enter a number and depending on the number, a statement will be output to the screen.

Let’s open a new Python file and name it final.py

I’ll start you out with the input:

```
number1 = input("Enter a number between 1 and 10. ")
number2 = input("Enter another number between 1 and 10. ")
```

Then you need to add the logic and what the output will be for each case.

```
if number1 < number2:
  print("The second number is larger than the first")
if number2 < number1:
  print("The second number is smaller than the first")
if number2 == number1:
  print("The second number is smaller than the first")
```

Save your file and run it.  If it does not work as you expect, check the logic or that you didn’t confuse the variables.

This set of tutorials scratches the surface of what Python can do.  I hope you learned some Python concepts and continue to explore the possibilities out there.