

# Cisco Networking Academy

Education, technical training and mentorship

Juan Carlos Domínguez Ayala

Instructor CEC-EPN

02/2022



# *Python*

## Introduction to Basic Programming with Python

Juan Carlos Domínguez Ayala

UPS sede Quito

12/2020



# Introduction



# Welcome



# Verify Your PC is Ready

- You should have already completed the **Lab - PC Setup for Workshop**.
- Your PC should include the following:
  - Python and the required modules are installed and verified
  - Postman installed
- Verify that your PC is ready to complete the tasks of the workshop.

# Code and Communities of Practice



Code



# Power of Code

*Coding is a huge  
base...to build off  
of...to go in the  
direction [you] want to.*

*Chris Bosh  
NBA All-Star*

*Everybody... should learn how  
to program a computer...  
because it teaches you how to  
think.*

*Steve Jobs  
Founder, Apple*

*[Coding] is the closest  
thing we have to a  
superpower.*

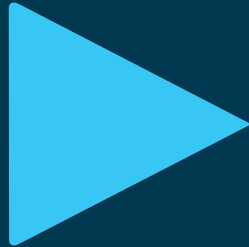
*Drew Houston  
Dropbox Creator*

*Great coders are today's  
rock stars.*

*will.i.am  
Black Eye Peas Creator*



# The Power of Code



# Communities of Practice



# Communities of Practice (CoPs)

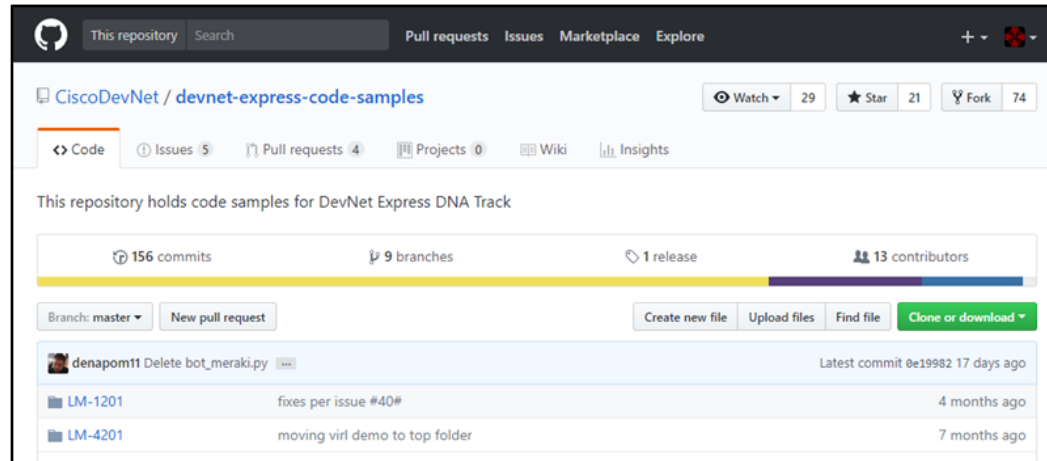
*Communities of practice are groups of people who share a concern or a passion for something they do and learn how to do it better as they interact regularly.*

*Jean Lave & Etienne Wenger*



# CoPs for Programmers - GitHub

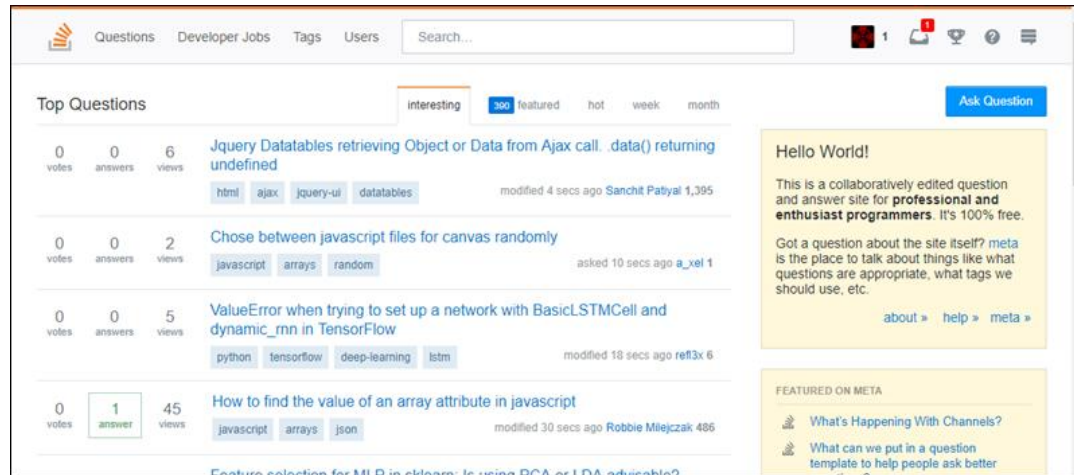
GitHub is the open source software version control system started by Linus Torvalds, the creator of Linux.



<https://github.com/>

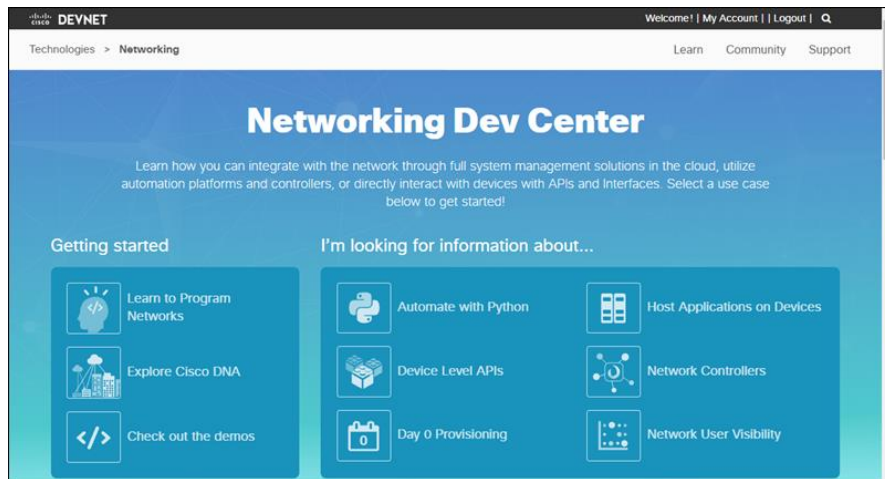
# CoPs for Programmers - Stack Overflow

Stack Overflow maintains a library of detailed answers to every question about programming.



# CoPs for Programmers - Cisco DevNet

Cisco DevNet offers support to developers and programmers who want to build Cisco-enable applications or use Cisco APIs to enhance and manage their networks



<https://developer.cisco.com>

# Python Basics



# Python Interpreter





# Start Python

## Windows

```
C:\> python  
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit  
(Intel)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>>
```

## Mac or Linux

```
$ python3  
Python 3.5.2 (default, Aug 18 2017, 17:48:00)  
[GCC 5.4.0 20160609] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>>
```

# Use Interactive Interpreter as a Calculator

```
$ python3
Python 3.5.2 (default, Aug 18 2017, 17:48:00)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+3
5
>>> 10-4
6
>>> 2*4
8
>>> 20/5
4
>>> 3**2
9
```

# Use Interpreter to print Hello World

- Strings can be enclosed with single quotes or double quotes.
- To remove the single quotes in the output, use the print command.

```
>>> "Hello World!"  
'Hello World!'  
>>> 'Hello World!'  
'Hello World!'  
>>> print("Hello World!")  
Hello World!
```

# Quit the Interpreter and Start IDLE

- Python includes the Integrated Development Environment (IDLE)
- Windows - open IDLE from the Start menu
- Mac or Linux - open IDLE from the command line.

## Windows

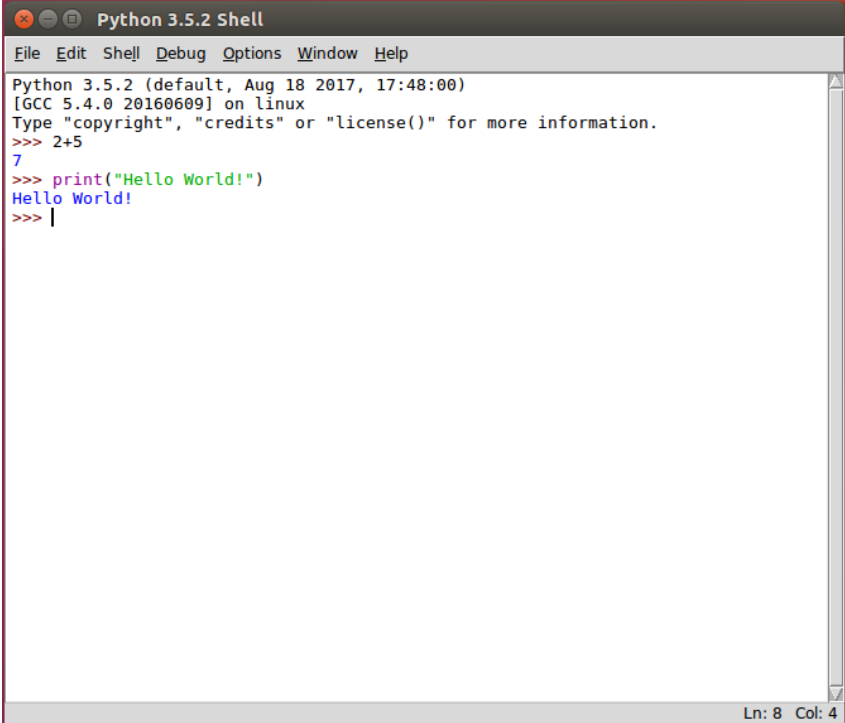
**Start > Python 3.x > IDLE (Python 3.x 32-bit).**

## Mac or Linux

```
>>> "Hello World!"  
'Hello World!'  
>>> 'Hello World!'  
'Hello World!'  
>>> quit()  
$ idle3
```

# IDLE Benefits

- Provides color coding
- Includes a text editor for writing programs
- Quickly save and run programs



The screenshot shows a window titled "Python 3.5.2 Shell". It has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the following content:

```
Python 3.5.2 (default, Aug 18 2017, 17:48:00)
[GCC 5.4.0 20160609] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 2+5
7
>>> print("Hello World!")
Hello World!
>>> |
```

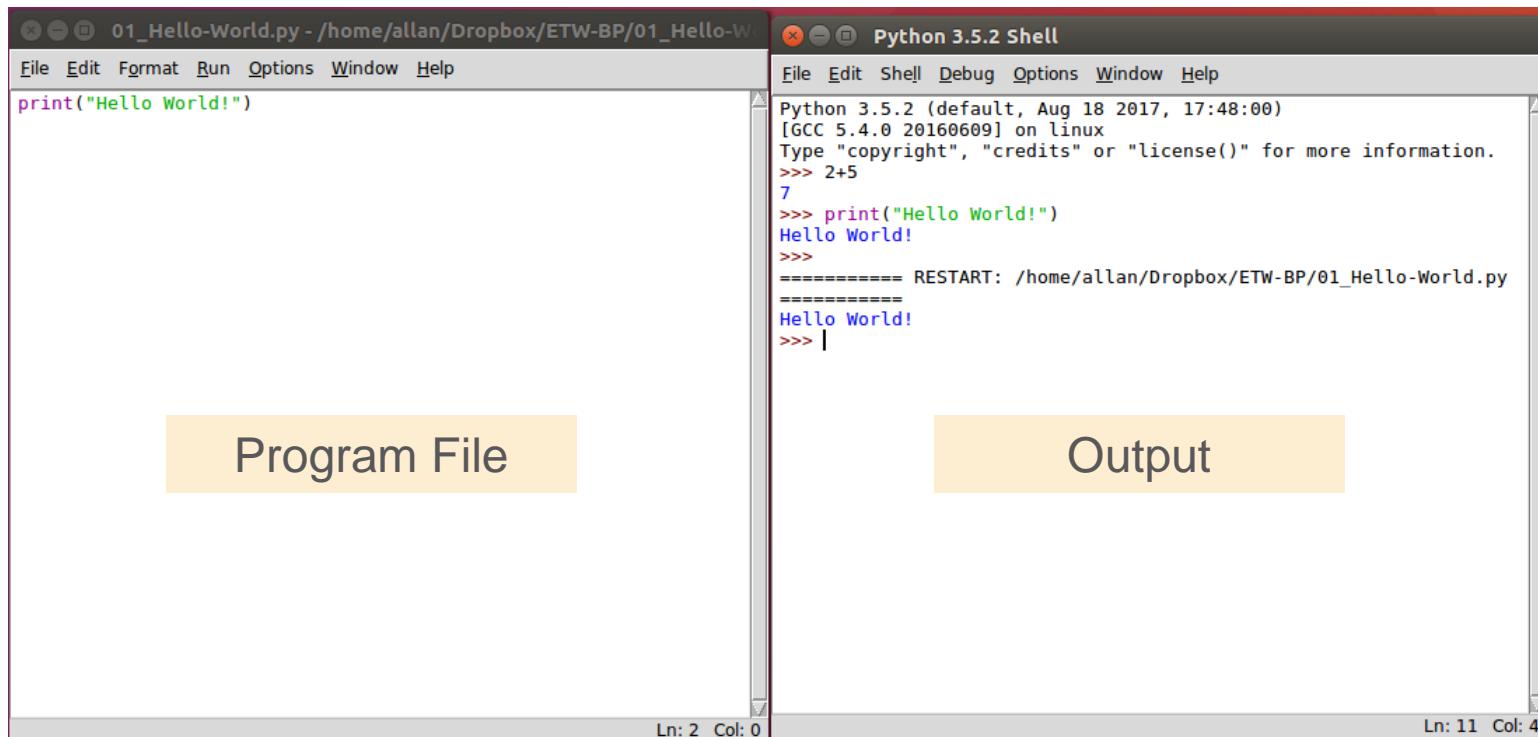
The status bar at the bottom right indicates "Ln: 8 Col: 4".

# Activity - Write, Save, and Run Your First Program

1. In IDLE, click **File > New File (Ctrl+N)** to open an Untitled script file.
2. Save the file as **01\_hello-world.py** in your project directory.
3. Enter the following in the script:  

```
print("Hello World!")
```
4. Save the script; click **File > Save (Ctrl+S)**
5. Run the script; click **Run > Run Module (F5)**

# First Program and Output



The image shows a side-by-side comparison of a Python program file and its execution output. On the left, a window titled '01\_Hello-World.py' displays the source code: `print("Hello World!")`. Below this window is a yellow label 'Program File'. On the right, a window titled 'Python 3.5.2 Shell' shows the execution process. It includes the Python version, GCC version, and a series of prompts and outputs: `>>> 2+5` resulting in `7`, and `>>> print("Hello World!")` resulting in `Hello World!`. It also shows a restart of the file. Below this window is a yellow label 'Output'. The status bars at the bottom indicate 'Ln: 2 Col: 0' for the file and 'Ln: 11 Col: 4' for the shell.

```
01_Hello-World.py - /home/allan/Dropbox/ETW-BP/01_Hello-W
File Edit Format Run Options Window Help
print("Hello World!")

Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (default, Aug 18 2017, 17:48:00)
[GCC 5.4.0 20160609] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 2+5
7
>>> print("Hello World!")
Hello World!
>>>
===== RESTART: /home/allan/Dropbox/ETW-BP/01_Hello-World.py
=====
Hello World!
>>> |
```

Program File

Output

Ln: 2 Col: 0

Ln: 11 Col: 4

# Data Types, Variables, and Conversions





# Basic Data Types

- The four basic data types we will use are:
  - Integer
  - Float
  - String
  - Boolean
- Use the `type()` command to determine the data type.

```
>>> type(98)
<class 'int'>
>>> type(98.6)
<class 'float'>
>>> type("Hi!")
<class 'str'>
>>> type(True)
<class 'bool'>
```

# Boolean Comparison Operators

Operator	Meaning
>	Greater than
<	Less than
==	Equal to
!=	Not equal to
>=	Greater than or equal to
<=	Less than or equal to

```
>>> 1<2
True
>>> 1>2
False
>>> 1==1
True
>>> 1!=1
False
>>> 1>=1
True
>>> 1<=1
True
```

# Creating and Using a Variable

- Use a single equal sign to assign a value to a variable.
- A variable can then be called for other operations.

```
>>> x=3
>>> x*5
15
>>> "Cisco"*x
'CiscoCiscoCisco'
```

# Concatenate Multiple String Variables

- Concatenation is the process of combining multiple strings.

```
>>> str1="Cisco"  
>>> str2="Networking"  
>>> str3="Academy"  
>>> space=" "  
>>> print(str1+space+str2+space+str3)  
Cisco Networking Academy  
>>>
```

# Converting Data Types

- Concatenation does not work for different data types.

```
>>> x=3
>>> print("This value of X is " + x)
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    print("This value of X is " + x)
TypeError: Can't convert 'int' object to str implicitly
```

# Converting Data Types

- Use the **str()** command to convert the data type to a string.

```
>>> x=3
>>> print("The value of x is " + x)
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    print("This value of X is " + x)
TypeError: Can't convert 'int' object to str implicitly
>>> print("The value of x is " + str(x))
The value of x is 3
>>>
```

# Converting Data Types

- The type for the variable `x` is still an integer.

```
>>> x=3
>>> print("The value of x is " + x)
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    print("This value of X is " + x)
TypeError: Can't convert 'int' object to str
implicitly
>>> print("The value of x is " + str(x))
The value of x is 3
>>> type(x)
<class 'int'>
```

# Converting Data Types

- To convert the data type, reassign the variable to the new data type.

```
>>> x=3
>>> print("The value of x is " + x)
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    print("This value of X is " + x)
TypeError: Can't convert 'int' object to str implicitly
>>> print("The value of x is " + str(x))
The value of x is 3
>>> type(x)
<class 'int'>
>>> x=str(x)
>>> type(x)
<class 'str'>
```



# Converting Data Types

- Use “**{:.2f}**”.format to display a float to two decimal places.
- Change the **2** to increase or decrease decimal places.

```
>>> pi = 22/7
>>> print(pi)
3.142857142857143
>>> print("{:.2f}".format(pi))
3.14
>>>
```

# Lists and Dictionaries



# Lists

- A list is an ordered list of items.

- Create a list using the brackets `[ ]` and enclosing each item in the list with quotes.
- Use the **type()** command to verify the data type.
- Use the **len()** command return the number of items in a list.
- Call the list variable name to display it's contents.

```
>>> hostnames=["R1","R2","R3","S1","S2"]
>>> type(hostnames)
<class 'list'>
>>> len(hostnames)
5
>>> hostnames
['R1', 'R2', 'R3', 'S1', 'S2']
```

# Lists

- Use the index to refer to an item and manipulate the list

- The first item in a list is indexed as zero, the second is indexed as one, and so on.
- The last item can be referenced with index **[-1]**
- Replace an item by assigning a new value to the index.
- Use the **del()** command to remove an item from a list.

```
>>> hostnames=["R1","R2","R3","S1","S2"]
>>> type(hostnames)
<class 'list'>
>>> len(hostnames)
5
>>> hostnames
['R1', 'R2', 'R3', 'S1', 'S2']
>>> hostnames[0]
'R1'
>>> hostnames[-1]
'S2'
>>> hostnames[0]="RTR1"
>>> hostnames
['RTR1', 'R2', 'R3', 'S1', 'S2']
>>> del hostnames[3]
>>> hostnames
['RTR1', 'R2', 'R3', 'S2']
>>>
```

# Dictionaries

- A list of unordered key/value pairs
  - Create a dictionary using the braces { }
  - Each dictionary entry includes a key and a value.
  - Separate key and values with a colon.
  - Use quotes for keys and values that are strings.

```
>>> ipAddress =  
{ "R1": "10.1.1.1", "R2": "10.2.2.1", "R3": "10.3.3  
.1" }  
>>> type(ipAddress)  
<class 'dict'>
```

# Dictionaries

- Use the key to refer to an entry

- The key is enclosed with brackets [ ].
- Keys that are strings can be referenced using single or double quotes.
- Add a key/value pair by setting the new key equal to a value.
- Use **key in dictionary** command to verify if a key exist in the dictionary

```
>>> ipAddress =  
{ "R1": "10.1.1.1", "R2": "10.2.2.1", "R3": "10.3.3.1" }  
>>> type(ipAddress)  
<class 'dict'>  
>>> ipAddress  
{ 'R1': '10.1.1.1', 'R2': '10.2.2.1', 'R3': '10.3.3.1' }  
>>> ipAddress['R1']  
'10.1.1.1'  
>>> ipAddress["S1"]="10.1.1.10"  
>>> ipAddress  
{ 'R1': '10.1.1.1', 'R2': '10.2.2.1', 'R3': '10.3.3.1', 'S1': '10.1.1.10' }  
>>> "R3" in ipAddress  
True  
>>>
```

# Activity - Troubleshoot List and Dictionary Code

1. Open `02_list-dicts.py`.
2. Run the code.
3. Troubleshoot the code until the script runs without errors.
4. What errors did you fix in the script?

# User Input





# The Input Function

- The **input()** function provides a way to get information from the user.

```
>>> firstName = input("What is your first  
name? ")  
What is your first name? Bob  
>>> print("Hello " + firstName + "!")  
Hello Bob!  
>>>
```

# Activity - Create a Script to Collect Personal Information

1. Open a blank script file and save it in your GitHub project directory as **03\_personal-info.py**.
2. Create a script that asks for four pieces of information such as: first name, last name, location, and age.
3. Create a variable for a space: **space = " "**
4. Add a print statement that combines all the information in one sentence.
5. Run the script and troubleshoot any errors.

# If Functions and Loops



# If/Else Function

- Open a blank script and save it as **04\_if-vlan.py**.
- Create a simple **if** function that compares two values and prints the results.
- Run the script and troubleshoot any errors.
- Change the values to test the **else** print statement.

```
nativeVLAN = 1
dataVLAN = 100
if nativeVLAN == dataVLAN:
    print("The native VLAN and the data VLAN
are the same.")
else:
    print("This native VLAN and the data VLAN
are different.")
```

# If/Elif/Else Function

- Open a blank script and save it as **05\_if-acl.py**.
- Create a more complex **if** function that takes user input and includes an **elif** loop.
- Note that the input needs to be converted to an integer.

```
aclNum = int(input("What is the IPv4 ACL  
number? "))  
if aclNum >= 1 and aclNum <= 99:  
    print("This is a standard IPv4 ACL.")  
elif aclNum >=100 and aclNum <= 199:  
    print("This is a extended IPv4 ACL.")  
else:  
    print("This is not a standard or extended  
IPv4 ACL.")
```

# For Loop

- A for loop iterates through items in a list, dictionary, or other sequenced data type.
- The variable name “item” is arbitrary and can be anything the programmer chooses.

```
>>> devices=["R1","R2","R3","S1","S2"]  
>>> for item in devices:  
    print(item)
```

```
R1  
R2  
R3  
S1  
S2  
>>>
```

# For Loop with Embedded If

- Using an If loop inside the For loop

```
>>> for item in devices:  
        if "R" in item:  
            print(item)
```

```
R1  
R2  
R3  
>>>
```

# Use a For Loop to Create a New List

- Create an empty list called switches.
- Iterate through the devices list to create the switch list.

```
>>> switches=[]  
>>> for item in devices:  
    if "S" in item:  
        switches.append(item)  
  
>>> switches  
['S1', 'S2']  
>>>
```



# Create a While Loop

- Open a blank script and save it as **06\_while-loop.py**.
- Create a program with a while loop that counts to a user's supplied number.
  - Convert the string to an integer:  
**x = int(x)**
  - Set a variable to start the count:  
**y = 1**
  - **While y <= x**, print the value of y and increment y by 1.

```
x=input("Enter a number to count to: ")
x=int(x)
y=1
while y<=x:
    print(y)
    y=y+1
```

# Modify the While Loop to Use Break

- Modify the while loop to use a Boolean check and break to stop the loop.
  - Replace **while y<=x** with **while True**
  - Add an if function to break the loop when **y>x**.

```
x=input("Enter a number to count to: ")
x=int(x)
y=1
while True:
    print(y)
    y=y+1
    if y>x:
        break
```

```
while True:
    x=input("Enter a number to count to: ")
    if x == 'q' or x == 'quit':
        break
x=int(x)
y=1
while True:
    print(y)
    y=y+1
    if y>x:
        break
```

# Use a While Loop to Check for User Quit

- Add another while loop to the beginning of the script which will check for a quit command.
- Add an if function to the while loop to check for 'q' or 'quit'.

```
while True:
    x=input("Enter a number to count to: ")
    if x == 'q' or x == 'quit':
        break

    x=int(x)
    y=1
    while True:
        print(y)
        y=y+1
        if y>x:
            break
```

