

# CODE CAMP

## Day 001: Getting Started With Python



## Create your account

Username:

Email:

Password:

Password (again):

☒ I agree to the [Terms and Conditions](#) and the [Privacy and Cookies Policy](#), and confirm that I am at least 13 years old.

[Register](#)

We promise not to spam or pass your details on to anyone else.

## .001 Create Your Python Anywhere Account

Visit [PythonAnywhere.com](https://PythonAnywhere.com) and create your free account. Don't forget to verify your email address with the email sent by the site. Once you create your account Python Anywhere will give you a quick tour. Complete the tour.

# Dashboard

Welcome, [frameworkTelevision](#)CPU Usage: 1% used – 1.11s of 100s. Resets in 23 hours, 45 minutes [More Info](#)[Upgrade Account](#)

File storage: 0% full – 100.0 KB of your 512.0 MB quota

## Recent Consoles

[+](#) 5 [-](#)

You have no recent consoles.

## New Console

\$ Bash

&gt;&gt;&gt; Python ▾

[More...](#)

## Recent Files

[+](#) 5 [-](#)

You have no recently edited files.

[+ Open another file](#)[Browse files](#)

## Recent Notebooks

[+](#) 5 [-](#)

Your account does not support Jupyter Notebooks. [Upgrade your account](#) to get access!

## All Web apps

You don't have any web apps.

[Open Web tab](#)

## .002 Create a new Bash Console

The bash console provides a window in a Linux operating system environment. This is where we'll create and execute Python code. Click the button that says **\$ Bash**.



## .002 Create a new Bash Console

If you've done everything correctly, your screen should appear similar to the image above. This is called a prompt. Instead of clicking to interact with the computer, we'll use the keyboard. Type the command **ls** and press enter.



## .002 Create a new Bash Console

If you've done everything correctly, your screen should appear similar to the image above. This is called a prompt. Instead of clicking to interact with the computer, we'll use the keyboard. Type the command **ls** and press enter.

```
18:20 ~ $ ls  
README.txt  
18:22 ~ $
```

## .002 Create a new Bash Console

The **ls** command lists all the content in the directory that you are in. Right now there is one file called README.txt in the directory.

Issue the command **pwd** and press return.



```
18:20 ~ $ ls
README.txt
18:22 ~ $ pwd
/home/frameworkTelevision
18:24 ~ $
```

## .002 Create a new Bash Console

The **pwd** command shows the path to the directory you're currently in. You're currently in your home directory. You'll notice the name of the user account you created earlier.



We're ready to start coding. Issue the command **nano** and press enter.





### .003 Writing Your First Lines of Code


This is the **nano** application. It's a text editor. Compared to the Windows or Mac applications you use it's very primitive. We'll use it to write code and save it.



We'll then execute the code within our Bash shell. Be careful to enter the code exactly like the example. If you make an error, just update your code with the correct version.



 Get Help  
 Exit

 Write Out  
 Read File

 Where Is  
 Replace

 Cut Text  
 Uncut Text

 Justify  
 To Spell

 Cur Pos  
 Go To Line

 Prev Page  
 Next Page



```
print("My name is Mark Lassoﬀ")  
print("Welcome to Python!")
```

### .003 Writing Your First Lines of Code

Carefully type the code on the left into nano. Substitute your name for Mark's in the code.

Once you've typed the code hit CTRL-O on your keyboard to write out (save) the file. This will save it to the virtual shell you created-- not to your local computer.

Once you hit CTRL-O notice the bottom of your console.

When prompted use the file name **greeting.py**. When the save operation is complete you'll notice that nano introduces color syntax highlighting to your code.

```
print("My name is Mark Lassoff")  
print("Welcome to Python!")
```

### .003 Writing Your First Lines of Code

Syntax highlighting is designed to make code more readable.

Hit CTRL-X to exit the nano text editor.

[ Wrote 4 lines ]

^G Get Help  
^X Exit

^O Write Out  
^R Read File

^W Where Is  
^\_ Replace

^K Cut Text  
^U Uncut Text

^J Justify  
^T To Linter

^C Cur Pos  
^\_ Go To Line

^Y Prev Page  
^V Next Page

^M-\ First Line  
^M-/ Last Line



```
18:41 ~ $ ls
README.txt  greeting.py
18:41 ~ $
```

## .004 Executing Your First Program

Now that you've exited nano, you've been returned to the Bash terminal. Issue the **ls** command you'll observe that your directory now includes the `greeting.py` file that you just create.

You'll run the program with the **python** command. The format is:

**python <filename>**

So, in this case, the complete command is **python greeting.py**. (For Python code always use the **.py** extension.) You will see the result of your program appear in the bash shell.



Bash console 15153805

[Share with others](#)

```
18:41 ~ $ ls
README.txt  greeting.py
18:41 ~ $ python greeting.py
My name is Mark Lassoﬀ
Welcome to Python!
18:45 ~ $
```

## .004 Executing Your First Program

Congratulations! You're written and executed your first Python program. *Take a picture of yourself with your first successful program and upload it to the class Slack discussion! Congratulations.*

```
print("My name is Mark Lassoﬀ")  
print("Welcome to Python!")
```

## .005 Analyzing the Code

The **print** function in Python does exactly like it sounds like it does. It prints out content. In this case the content printed out-- which is always contained in parentheses-- are strings of characters which are always enclosed in quotes. The general format is:

```
print("XXXXXXXXX")
```

Where XXX represents the text that you want to output.

## .006 Variables

Let's make this more interesting and add some variables to our code. Variables are place holders for the storage of text, values and objects. We'll get to objects later, but let's deal with text and values first.

Return to the Dashboard by clicking the three line (hamburger) menu in the upper-right hand corner of your screen and choose **Dashboard** from the dropdown.

Once the Dashboard appears Click the button that says **\$ Bash** to create a new Bash console. (You have a limited number in the free edition of Python Anywhere).

```
name = "Mark Lassoff"  
age = 46  
petName = "Gertie"  
petType = "cat"  
print("Hello, my name is " + name)  
print("I am " + str(age) + " years old.")  
print("I have a " + petType + " named " +  
petName + ".")
```

## .006 Variables

Type **nano** to enter the nano text editor and carefully key in the code on the left.

Again, please feel free to substitute your own information or make up fictional information.

Be very careful with the quotes and plus signs as putting them in the wrong place will cause your code to error.

When done typing the code hit **CTRL-O** to Write Out (save) the code. Use the filename **vars.py**.

Exit **nano** with CTRL-X.



```
19:07 ~ $ ls
README.txt  greeting.py  vars.py
19:07 ~ $ python vars.py
Hello, my name is Mark Lassoﬀ
I am 46 years old.
I have a cat named Gertie.
19:07 ~ $
```

## .006 Variables

You have returned to your shell. Issue the **ls** command to list the files in your directory. You should see **vars.py** which you just saved.

Issue the command **python vars.py** to run your program.

Examine the output carefully.

If you see an error return to your code by typing **nano vars.py**, correct any errors and save the result. Exit nano and test again. Repeat until you have corrected all problems and see your output correctly. If you need help visit the Slack discussion.



```
name = "Mark Lassoff"  
age = 46  
petName = "Gertie"  
petType = "cat"  
print("Hello, my name is " + name)  
print("I am " + str(age) + " years old.")  
print("I have a " + petType + " named " +  
petName + ".")
```

## .007 Analyzing the Code

The first four lines of code assign values to the variables **name**, **age**, **petName**, and **petType**. We would say, “Name is assigned the value ‘Mark Lassoff’.” or “age is assigned the value 46.”

In the **print** statements we recall the variable values. Notice that the variables in the **print** statements are not in quotes.

### Answer in Slack

What do you think would happen if the variable values were inside quotes? Why?

```
name = "Mark Lassoff"
age = 46
petName = "Gertie"
petType = "cat"
print("Hello, my name is " + name)
print("I am " + str(age) + " years old.")
print("I have a " + petType + " named " +
petName + ".")
```

## .007 Analyzing the Code

Notice that the variable **age** inside the **print** statement is inside another function, **str()**. Because we stored a number inside **age**, not a **string** of characters, we need to convert the value to a **string** to output it. **print** only works with **string** values.

```
band1 = "Journey"  
band2 = "REO Speedwagon"  
band3 = "Queen"  
band4 = "The Cure"  
band5 = "Doobie Brothers"
```

## .008 Activity

Create a new file in **nano** and set your five favorite bands or musical acts to variables as demonstrated in the code on the left.



```
19:19 ~ $ ls
README.txt  bands.py  greeting.py  vars.py
19:19 ~ $ python bands.py
My Favorite Bands
1: Journey
2: REO Speedwagon
3: Queen
4: The Cure
5: Doobie Brothers
19:19 ~ $
```

## .008 Activity

Write a series of **print** statements, so that your output looks *exactly* like the output on the left. You must do so without putting the band's names directly in the **print** statements. Use the variables you just declared. Also make sure the numbering appears as in the example.