MLH localhost

Workshop

# Basic Training: Intro to Python Skills for AI, Part I

v4

1

*Using your Web Browser,*
*Open this URL:*

**http://mlhlocal.host/lhd-resources**

2

*Click on the workshop you're attending, and find:*

- Setup Instructions
- The Code Samples
- A demo project
- A Workshop FAQ
- These Workshop Slides
- More Learning Resources

MLH
MAJOR LEAGUE HACKING

## *Our Mission is to Empower Hackers.*

**65,000+**
HACKERS

**12,000+**
PROJECTS CREATED

**400+**
CITIES

***We hope you learn something awesome today!***
*Find more resources: http://mlh.io/*

## Why does this matter?

1. Coding is the new literacy.
2. There are millions of careers where you can earn a lot of money and make a difference that use Python.
3. Once you learn one programing language, you can learn any!

**How many people here have coded before?**

**What languages have you used?**

**What have you built?**

## Introduction

This workshop is the first in a series of three workshops that will teach some Python basics that you can use for many purposes, such as artificial intelligence. Artificial intelligence might sound like something from the future, but it's actually all around you:

Siri and Alexa use a type of AI called Natural Language Processing to listen to spoken words and respond to them.

## Introduction

This workshop is the first in a series of three workshops that will teach some Python basics that you can use for many purposes, such as artificial intelligence. Artificial intelligence might sound like something from the future, but it's actually all around you:

NETFLIX

Netflix uses artificial intelligence to predict what you'll like to watch. That's where the "77% match" ratings come from.

## Introduction

This workshop is the first in a series of three workshops that will teach some Python basics that you can use for many purposes, such as artificial intelligence. Artificial intelligence might sound like something from the future, but it's actually all around you:

eno® from Capital One®

Capital One's new eno tool helps you determine if someone has stolen your card number and to manage your finances using machine learning and natural language processing.

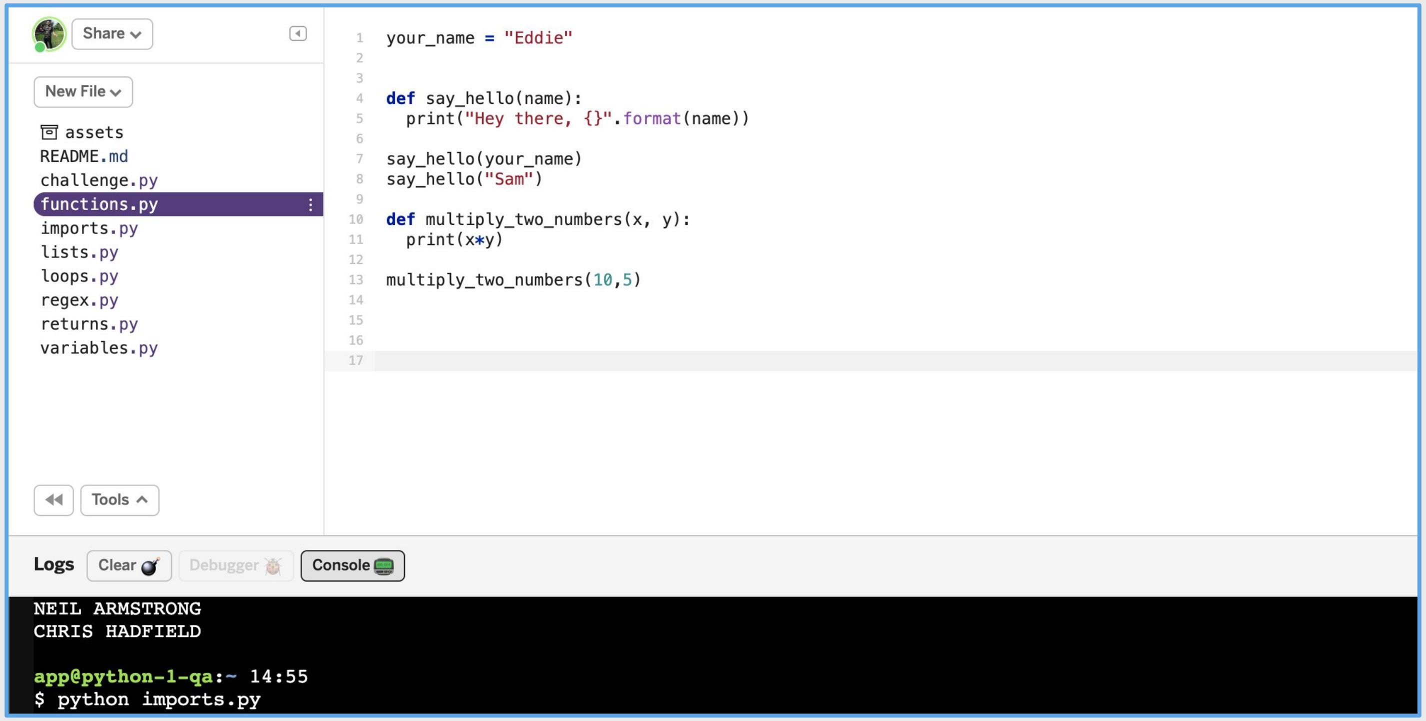
## Introduction

There are a lot of things you'll need to learn to be able to write your own artificial intelligence applications. In this series, we're going to start with the basics!

Today, you'll cover the basics of writing code in Python. In later workshops, you'll use a natural language processing library to create a bot of your favorite Twitter celebrity!

## What are you going to build today?

Today you're going to learn the fundamentals of Python by writing various pythons scripts and running them in the console.



```python
your_name = "Eddie"


def say_hello(name):
  print("Hey there, {}".format(name))

say_hello(your_name)
say_hello("Sam")

def multiply_two_numbers(x, y):
  print(x*y)

multiply_two_numbers(10,5)
```

```
NEIL ARMSTRONG
CHRIS HADFIELD

app@python-1-qa:~ 14:55
$ python imports.py
```

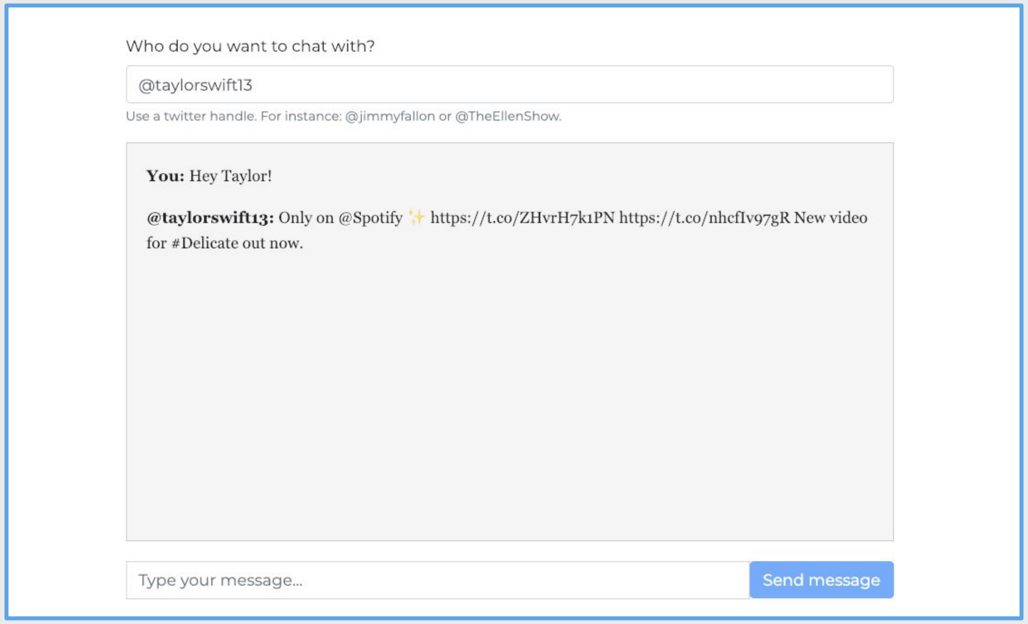
**Key Terms**

**Console:** A application that runs python files.

## What you will be able to build

You will be able to build the below project in later workshop series that uses artificial intelligence to scrape Tweets off Twitter.

**http://mlhlocal.host/glitch-bot**


Who do you want to chat with?

@taylorswift13

Use a twitter handle. For instance: @jimmyfallon or @TheEllenShow.

**You:** Hey Taylor!

**@taylorswift13:** Only on @Spotify ✨ https://t.co/ZHvrH7k1PN https://t.co/nhcfIv97gR New video for #Delicate out now.

**Key Terms**

**Web scraping**: Collecting data from a website

## How does this work?

1. When a user enters the Twitter handle of another user, that information is put into your script.
2. The script pulls some of the tweets from the Twitter user using a process called "web scraping."
3. Once your script has this information, it runs some **functions** on it. Functions are code you can use over and over again.
4. Then, a function removes the information we don't want and prints the information we do.

**Key Term**

**functions:** A few lines of code, grouped together, that your program can use as many times as it wants.

## How are you going to do it?

These are the steps you'll need to take:

1. Sign up for Glitch.
2. Copy the Code.
3. Learn the basics of Python.
4. Write a few Python functions.
5. Scrape Twitter using Python.
6. Format your information.
7. Review and next steps!

## Let's get started!

# Table of Contents

## What is Python?

Python is an interpreted, high-level coding  language.

1. Python was invented by Guido van Rossum in 1991.
2. Python can be used to make web pages, apps, and scripts.
3. One of the most common uses for Python is artificial intelligence and machine learning!
4. You're learning some Python basics today as the building blocks to go on and learn more about machine learning/AI later in your academic career.

## Python Basics

You'll learn the following basic Python tools today:

- **Variables** - variables in programming are a lot like variables in math! They store information for us.
- **Loops** - loops are a structure your program can use to perform the same task multiple times (rather than coding it out by hand every time).
- **Functions** - a structure you can use in your program to reuse bits of code throughout the program
- **Running your program** - you'll learn how to run your program by entering commands into the Glitch console

**Key Term**

**console:** Part of the Glitch coding environment where you enter commands and see output

## The best way to learn to code is to CODE, not to read about it, so let's set up Glitch!

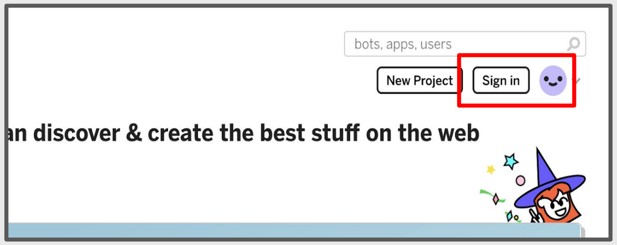
# Table of Contents

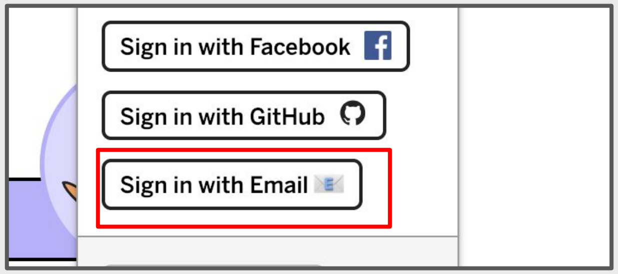
Note: You're going to need to check your email to verify sign-up. Please sign into your email now!

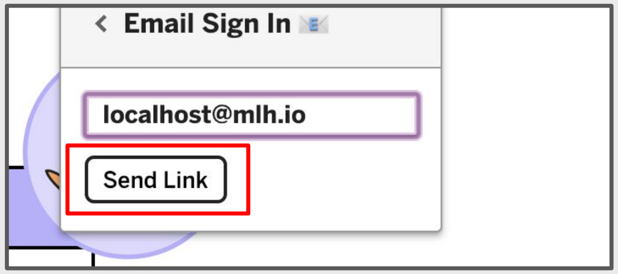
# Let's set up Glitch!

**http://mlhlocal.host/glitch**

1. Navigate to the URL above.
2. Click **Sign in**.
3. Click **Sign in with Email**.
4. Enter your email address and click **Send Link**.

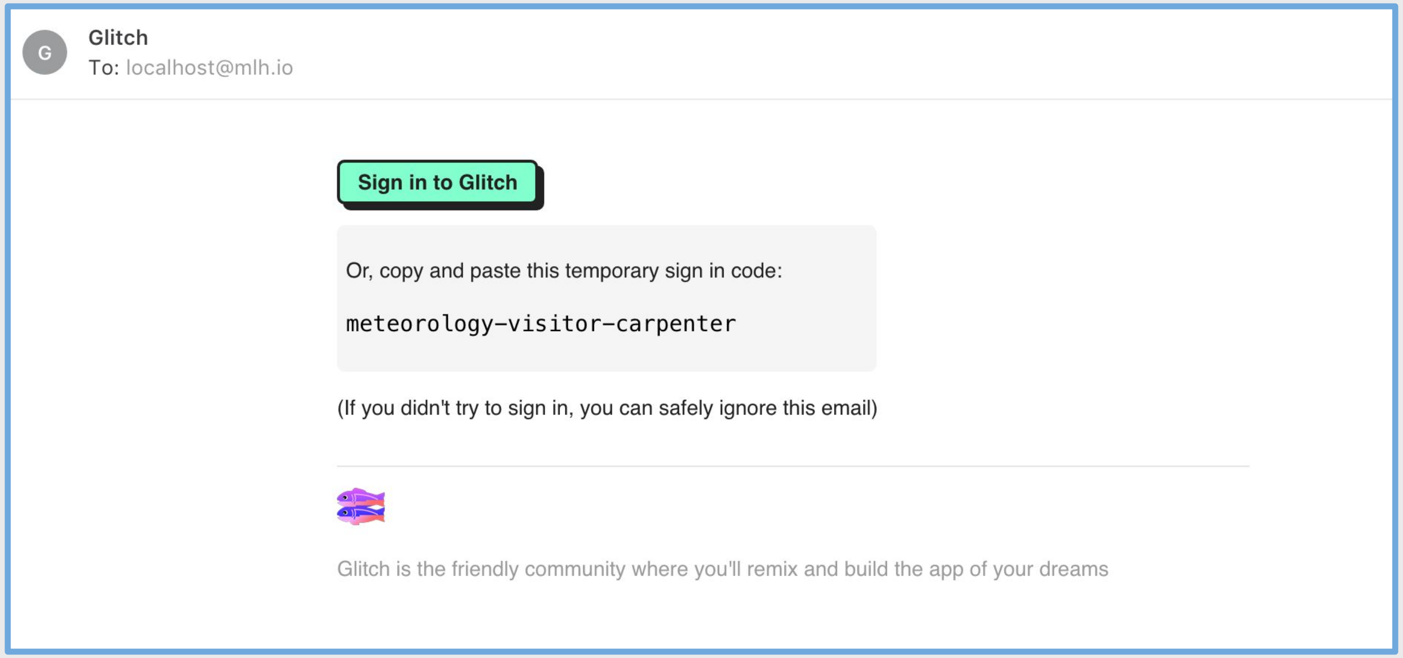






## Let's set up Glitch!

6. Go to your email and look for the email below:
7. Click **Sign in to Glitch**.



Glitch
To: localhost@mlh.io

Sign in to Glitch

Or, copy and paste this temporary sign in code:

```
meteorology-visitor-carpenter
```

(If you didn't try to sign in, you can safely ignore this email)

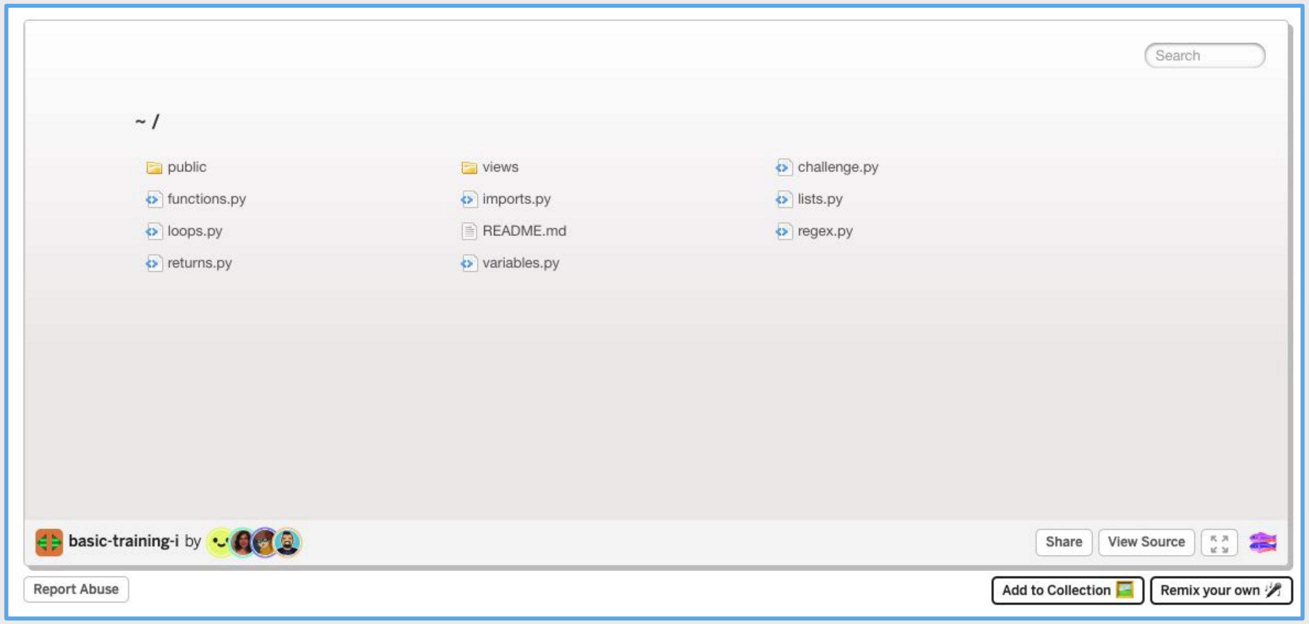
Glitch is the friendly community where you'll remix and build the app of your dreams

## Great! You're signed up. Now let's get the code!

# Get the code!

**http://mlhlocal.host/glitch-python-i**

1. Go to the URL above.
2. Click **Remix your own**, to make your own version



**You now have your own version of the project. This is the code you’ll need to complete the workshop.**

**Let's dive in.**

# Table of Contents

## Programming 101: Variables

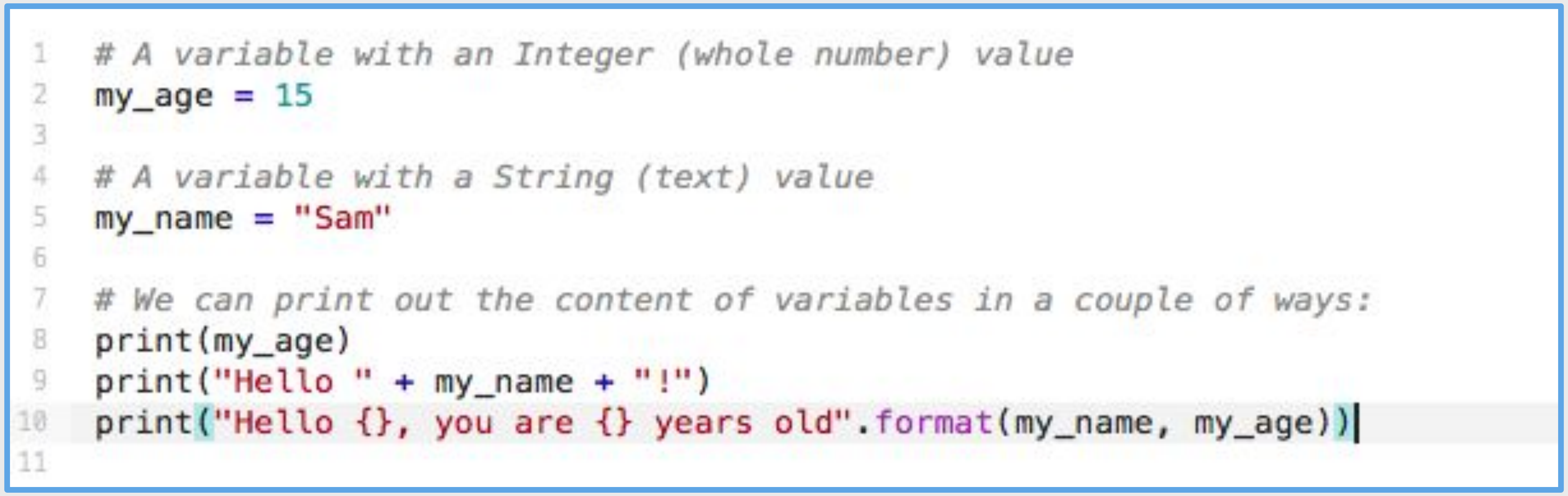
## Write Code: `variables.py`

A **variable** stores some data - like numbers or text.

We can create variables, give them values, and modify those values later.

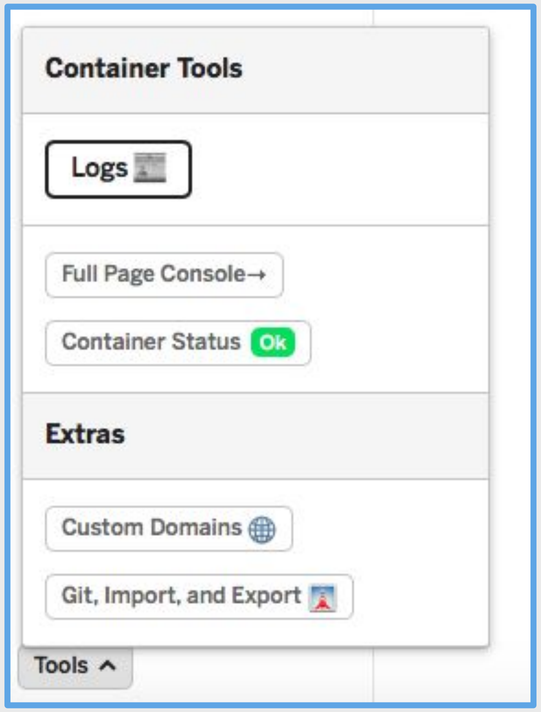
Click on **`variables.py`** in Glitch, and type

in the following code:

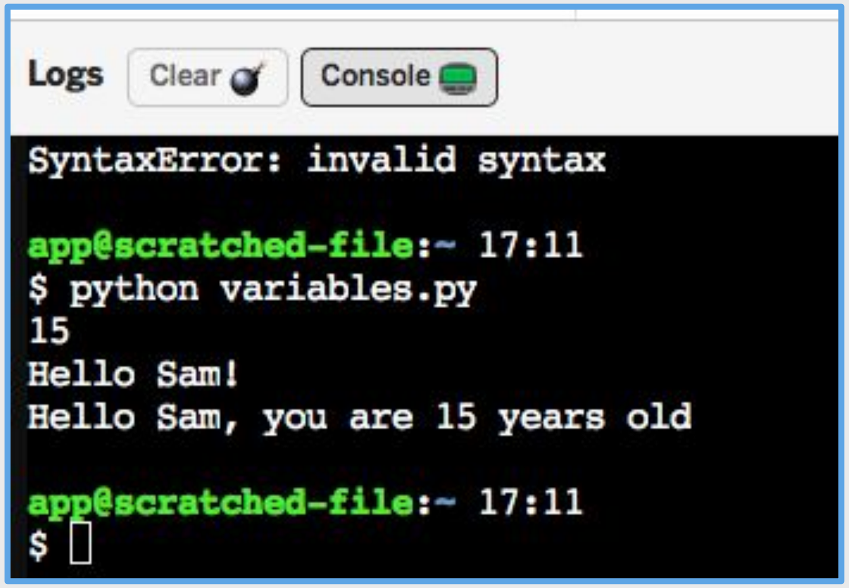


```python
# A variable with an Integer (whole number) value
my_age = 15

# A variable with a String (text) value
my_name = "Sam"

# We can print out the content of variables in a couple of ways:
print(my_age)
print("Hello " + my_name + "!")
print("Hello {}, you are {} years old".format(my_name, my_age))
```

## Running the Script



Click **Tools** at the bottom of the page, then **Logs.**



```
SyntaxError: invalid syntax

app@scratched-file:~ 17:11
$ python variables.py
15
Hello Sam!
Hello Sam, you are 15 years old

app@scratched-file:~ 17:11
$ 
```

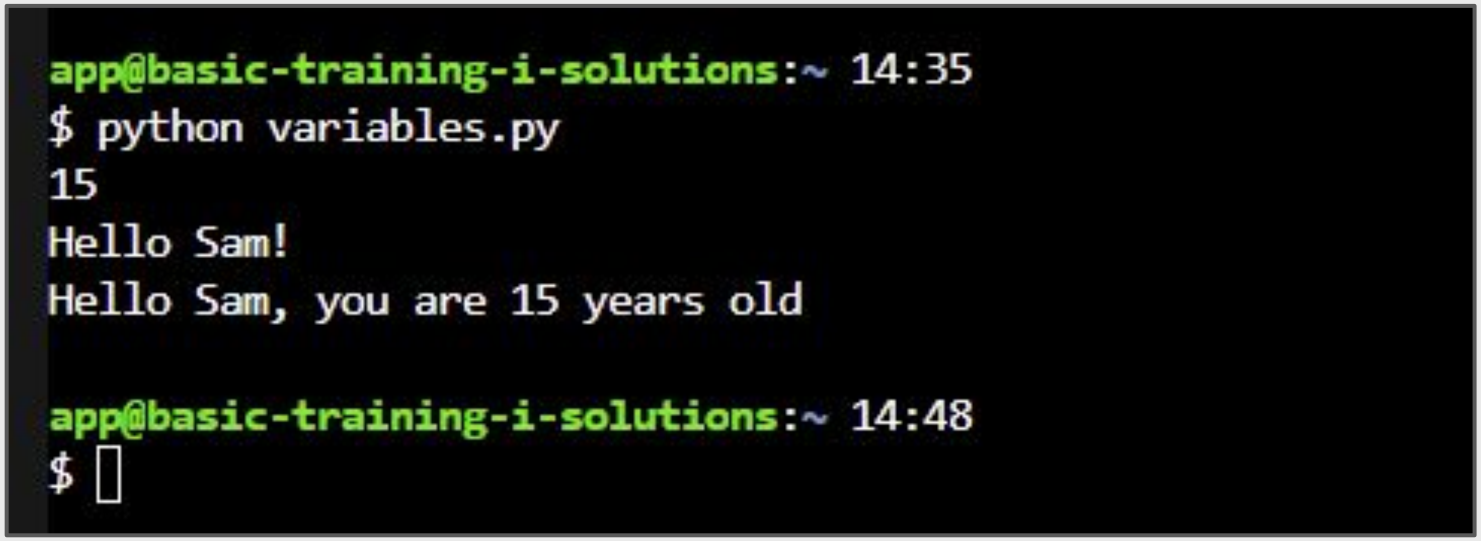
Click **Console** and type in **`python variables.py`** .

Then hit enter!

## Running the Script

The console will display anything inside the **`print()`** statements.

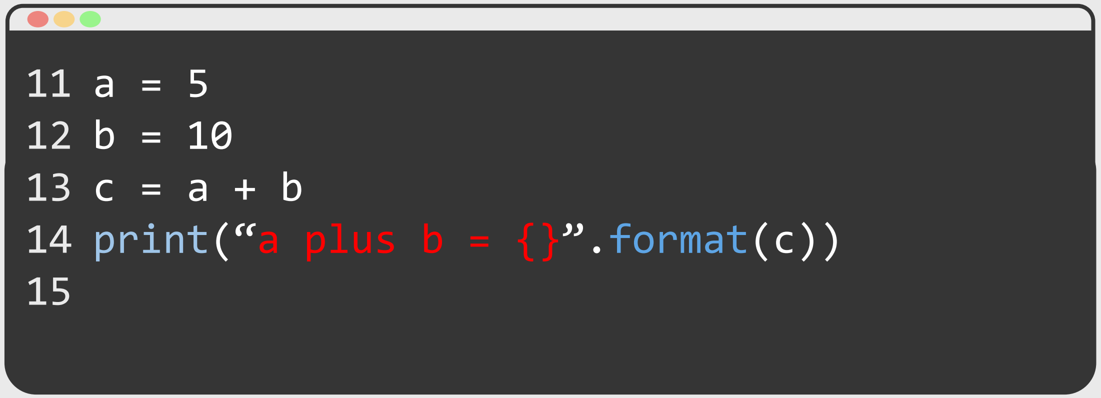
Each print statement is on a new line.



```
app@basic-training-i-solutions:~ 14:35
$ python variables.py
15
Hello Sam!
Hello Sam, you are 15 years old

app@basic-training-i-solutions:~ 14:48
$ 
```

## Write Code: Modifying Variables

Once we’ve created some variables, we can modify them - this means changing their value. Try adding the code below to **`variables.py`** and running it again:



```
a = 5
b = 10
c = a + b
print(“a plus b = {}”.format(c))

```

Experiment with different maths operators:

- Subtraction: **-**
- Division: **/**
- Multiplication: *****

## Let's talk about functions!

## Functions

A **function** allows us to re-use a block of code over and over.

You've already used a few without realising! **`print()`** and **`format()`** are Python functions.

Functions have a few features:

- Name: Functions have a name, and when we type the name, we **call** (use) the function.

- Parameters: We can give the function a variable (or multiple!) to use. When we use the **`print()`** function, everything inside the brackets () is used by the function.

## Write Code: `functions.py`

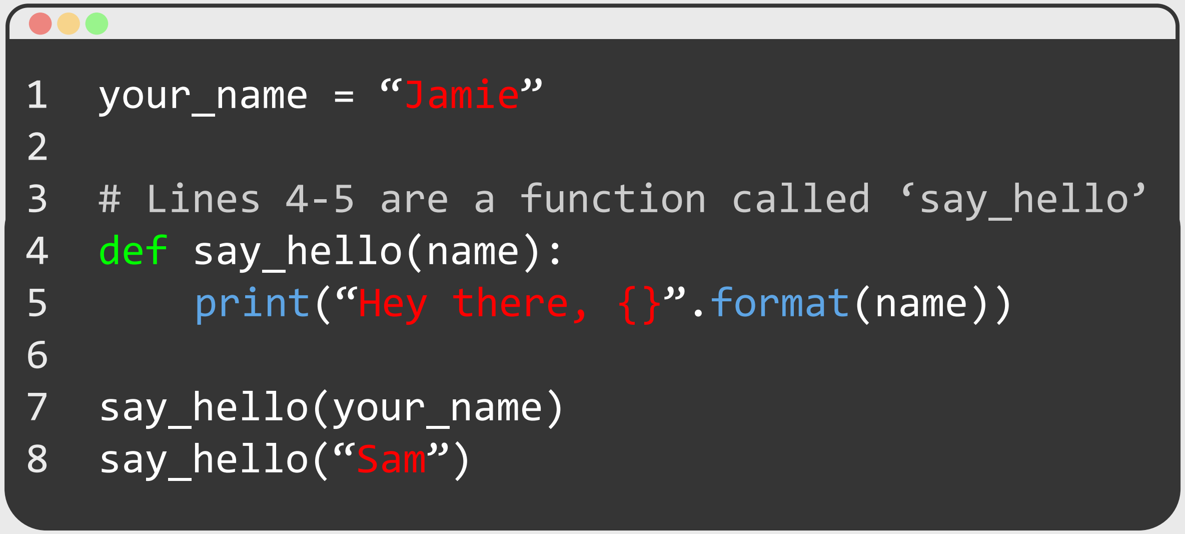
Open the file **`functions.py`** and let’s write some more code:

```
1   your_name = “Jamie”
2
3   # Lines 4-5 are a function called ‘say_hello’
4   def say_hello(name):
5       print(“Hey there, {}”.format(name))
6
7   say_hello(your_name)
8   say_hello(“Sam”)
```

Try running it by typing **`python functions.py`** into the console.

Let’s explain what’s happening...

## Code Review: `functions.py`



```
1  your_name = “Jamie”
2
3  # Lines 4-5 are a function called ‘say_hello’
4  def say_hello(name):
5      print(“Hey there, {}”.format(name))
6
7  say_hello(your_name)
8  say_hello(“Sam”)
```

- **Lines 4-5:** This is our function. We use the **def** (define) keyword to start a function. We give it a **name** (**`say_hello`**) and a **parameter** (**name**).
- Notice that line 5 is indented under line 4. Anything that comes later that is not indented is not part of the same function (line 6).

## Code Review: `functions.py`

```
1  your_name = “Jamie”
2
3  # Lines 4-5 are a function called ‘say_hello’
4  def say_hello(name):
5      print(“Hey there, {}”.format(name))
6
7  say_hello(your_name)
8  say_hello(“Sam”)
```

- **Lines 7-8:** We call the function by typing its name and giving a value to the name parameter. Whichever parameter we give the function is used! This is called **passing** the variable into the function.

## Challenge

You can pass multiple variables into a function!

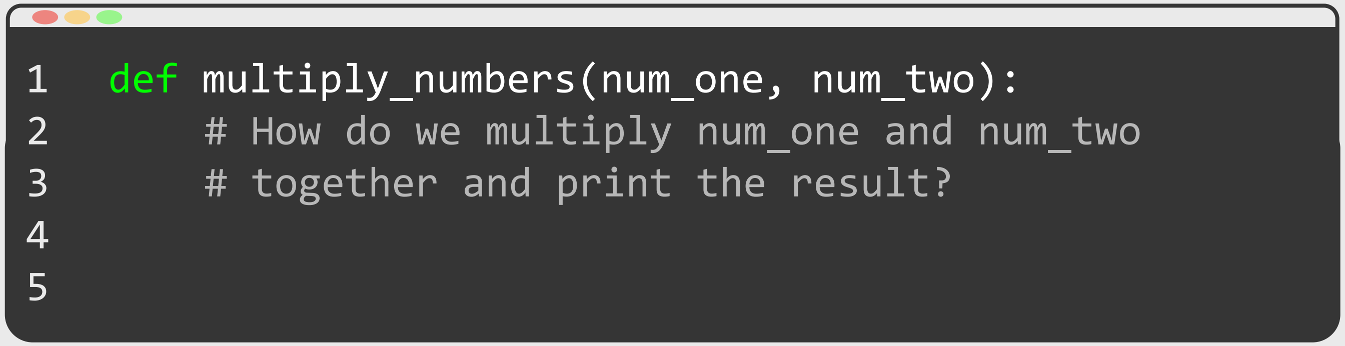
To do this, separate parameters with a comma when you create the function, like this:

```python
def my_function(parameter_1, parameter_2):
    # do things with parameter_1 and parameter_2
```

**Write a function that takes 2 parameters, multiplies them together, and prints the result.**

## Challenge

**Write a function that takes 2 parameters, multiplies them together, and prints the result.**



```python
def multiply_numbers(num_one, num_two):
    # How do we multiply num_one and num_two
    # together and print the result?


```

Here's some code to get you started.

## Don't peek! The solution is on the next slide.

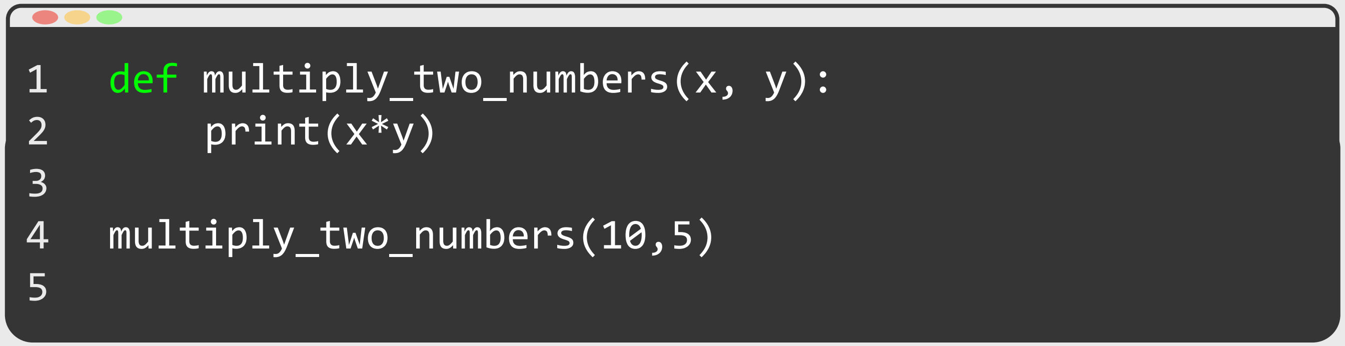
## Challenge: Solution

```python
def multiply_two_numbers(x, y):
    print(x*y)

multiply_two_numbers(10,5)
```

## Returning Values from Functions

So far, our function uses the parameters within the function.

What if we wanted to use the result somewhere else?



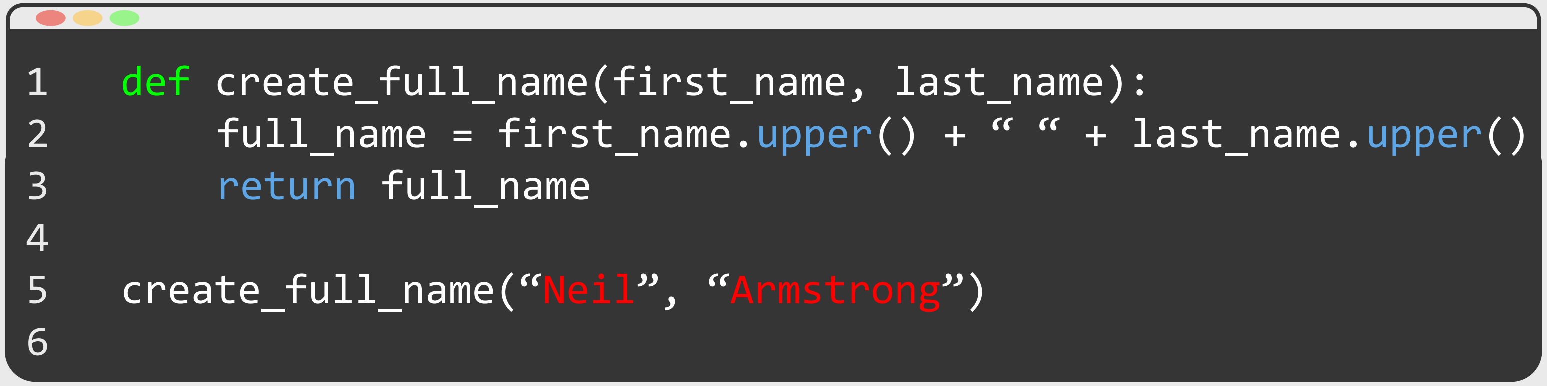
```python
def multiply_two_numbers(x, y):
    print(x*y)

multiply_two_numbers(10,5)

```

We can do this by **Returning** values from functions.

Let’s see what that looks like!

## Write Code: `returns.py`

Open up **`returns.py`** and add the following:



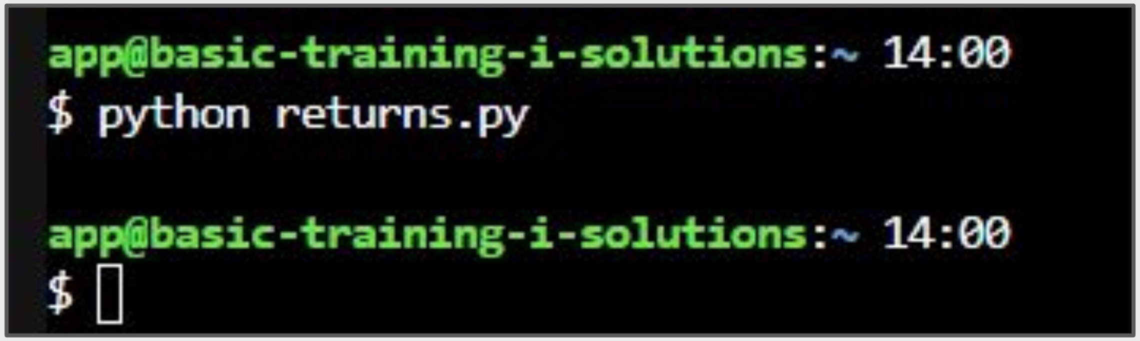
```
def create_full_name(first_name, last_name):
    full_name = first_name.upper() + “ “ + last_name.upper()
    return full_name

create_full_name(“Neil”, “Armstrong”)

```

The upper() function changes a string to all upper case (CAPITAL LETTERS!).

Run the program. What happens?

## returns.py

Nothing happened!



```
app@basic-training-i-solutions:~ 14:00
$ python returns.py

app@basic-training-i-solutions:~ 14:00
$ 
```

Maybe you expected that - we didn’t tell the function to print anything.

How can we use the **`Return`** value of our function to make it useful?

## Write Code: returns.py

Go back to the code and add a piece on **Line 5-6:**

```
1   def create_full_name(first_name, last_name):
2       full_name = first_name.upper() + “ “ + last_name.upper()
3       return full_name
4
5   result = create_full_name(“Neil”, “Armstrong”)
6   print(result)
```

Now, we have a variable called **`result`**, and we’ve set the value to be the output of our **`create_full_name()`** function! Nifty!

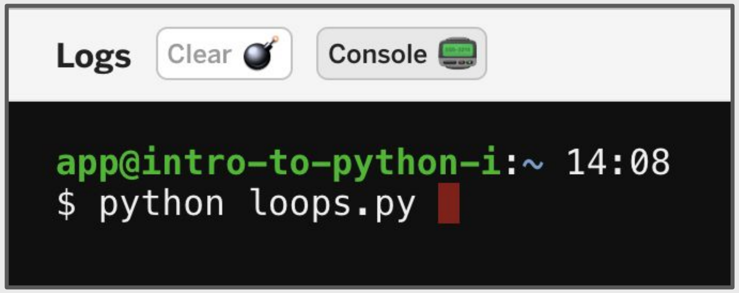
Run the program again. What happens now?

## Cool, now you know a little bit about functions!

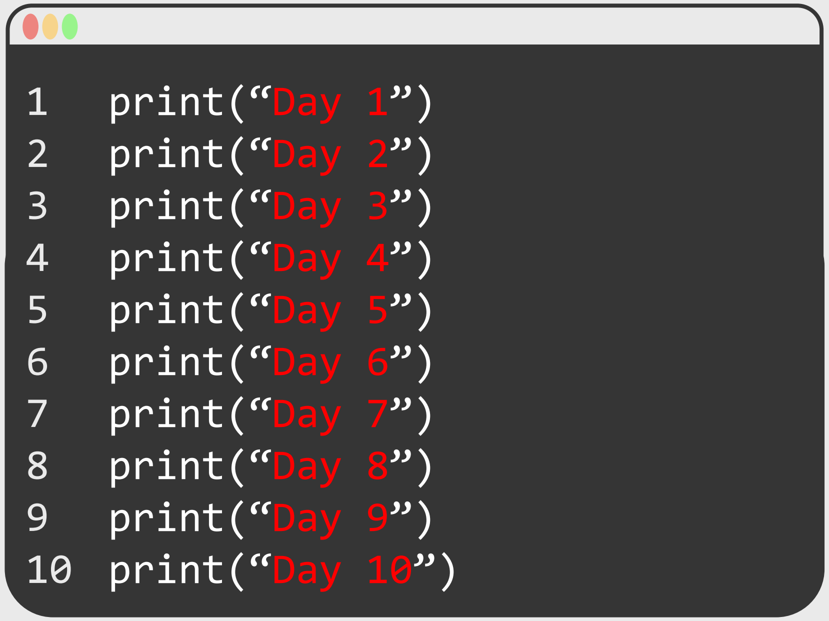
## Let's talk about loops!

## Code Review: `loops.py`

- Click on **`loops.py`.**
- What do you think this code does?
- To test it, enter
  **`python loops.py`**
  in your console!



```
app@intro-to-python-i:~ 14:08
$ python loops.py
```



```
print(“Day 1”)
print(“Day 2”)
print(“Day 3”)
print(“Day 4”)
print(“Day 5”)
print(“Day 6”)
print(“Day 7”)
print(“Day 8”)
print(“Day 9”)
print(“Day 10”)
```

## What just happened?

In the previous file, we had to write 10 separate lines of code to get the program to print Day 1, Day 2, etc. However, the only thing different on each line was the number. Imagine if we had to print 100 days! This would be awful.  Is there a better way?

## What just happened?

In the previous file, we had to write 10 separate lines of code to get the program to print Day 1, Day 2, etc. However, the only thing different on each line was the number. Imagine if we had to print 100 days! This would be awful.  Is there a better way?

**LOOPS!**

## What just happened?

In the previous file, we had to write 10 separate lines of code to get the program to print Day 1, Day 2, etc. However, the only thing different on each line was the number. Imagine if we had to print 100 days! This would be awful.  Is there a better way?

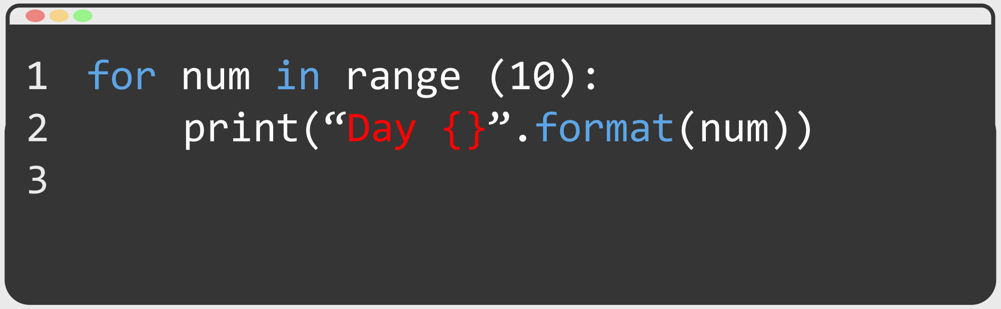
## LOOPS!

Sometimes in programming you want to do the same task over and over again. A loop is an easy way to tell your program to do that.

**Key Term**

**loop:** a way to repeat the same task many times in a program, without writing it out each time

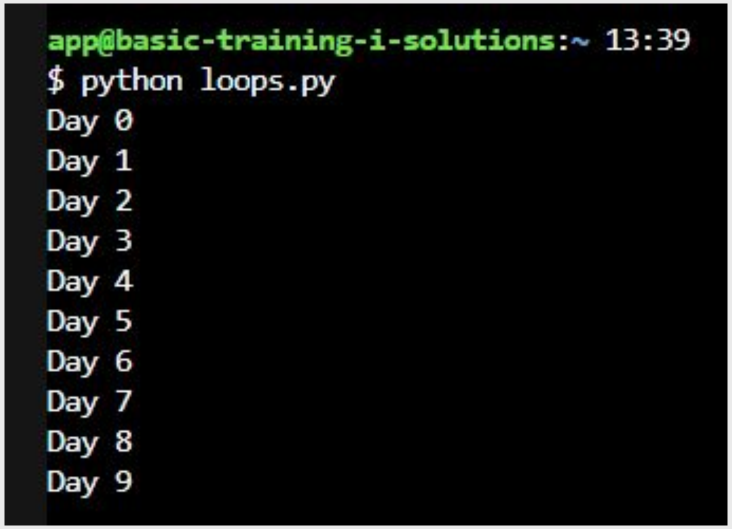
## Code Review: `loops.py`



```
for num in range (10):
    print(“Day {}”.format(num))

```

- Delete all the code in **`loops.py`**
- Add this code to **`loops.py`** and run the file.
- Before you view the output in the console, what do you think it will look like?

## What happened?

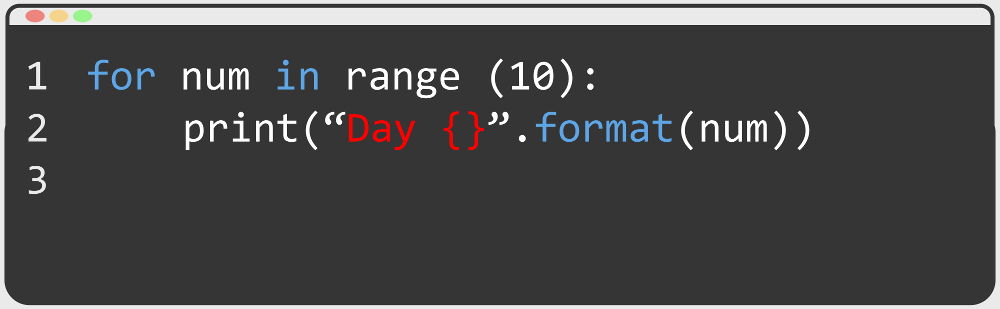


```
app@basic-training-i-solutions:~ 13:39
$ python loops.py
Day 0
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day 7
Day 8
Day 9
```

Our script printed out the ‘Day’ line 10 times! Not bad for two lines of code.

However… Notice anything wrong?

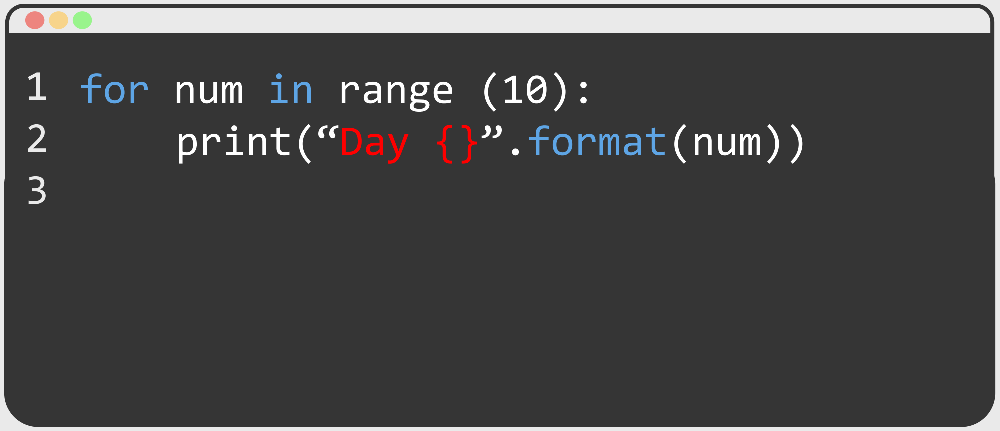
## What went wrong?



```
for num in range (10):
    print(“Day {}”.format(num))

```

- **`loops.py`** only printed up to Day 9. Why did this happen?
- When using a range loop in Python,  it prints up to but not including the number in parentheses. This is called an "off by one error," because our program did what we wanted it to except for 1 number.

## Challenge



```
for num in range (10):
    print(“Day {}”.format(num))

```

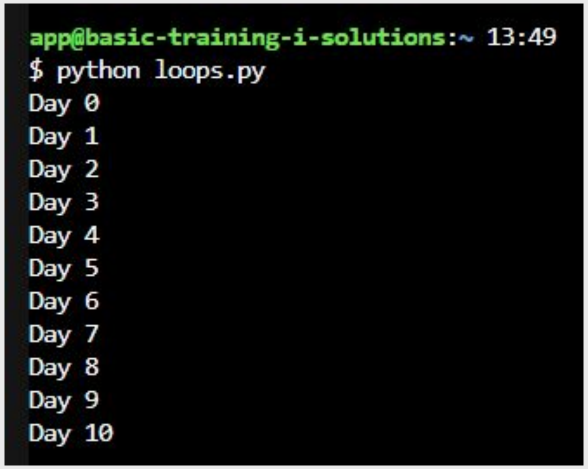
Get **`loops.py`** to print through Day 10.

## Solution

```
for num in range (11):
    print("Day {}".format(num))
```

Change (10) to (11).

Run your program again!



```
app@basic-training-i-solutions:~ 13:49
$ python loops.py
Day 0
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day 7
Day 8
Day 9
Day 10
```

**We’re making great progress!**

**Let’s move on to Lists.**

## Lists

Variables store a single value.

What if we wanted to have a list of related variables? Like a list of names, or phone numbers?

This is where **Lists** come in handy. In other programming languages, you might find them called **Arrays**.

# lists.py

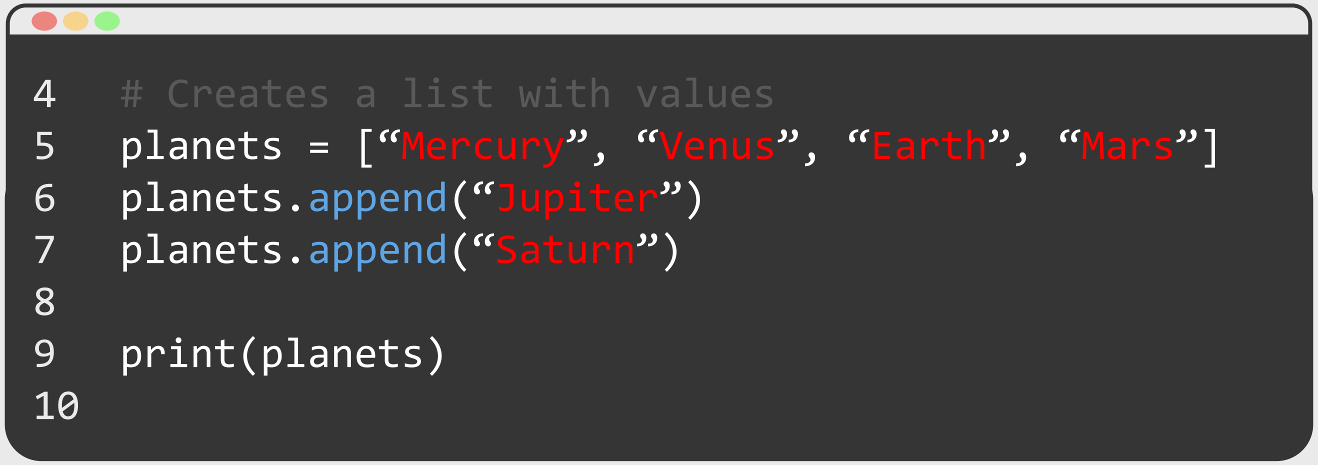
There’s a few ways we can create Lists in Python. Open up **`lists.py`** and type in this code:

```
# Creates an empty list
names = []

# Creates a list with values
planets = [“Mercury”, “Venus”, “Earth”, “Mars”]

```

Try adding some code to print **`names`** and **`planets`**. What does the output look like?

## Adding Values to Lists

Once we've created a List, we can continue adding values to it by using the **append()** function, like this:

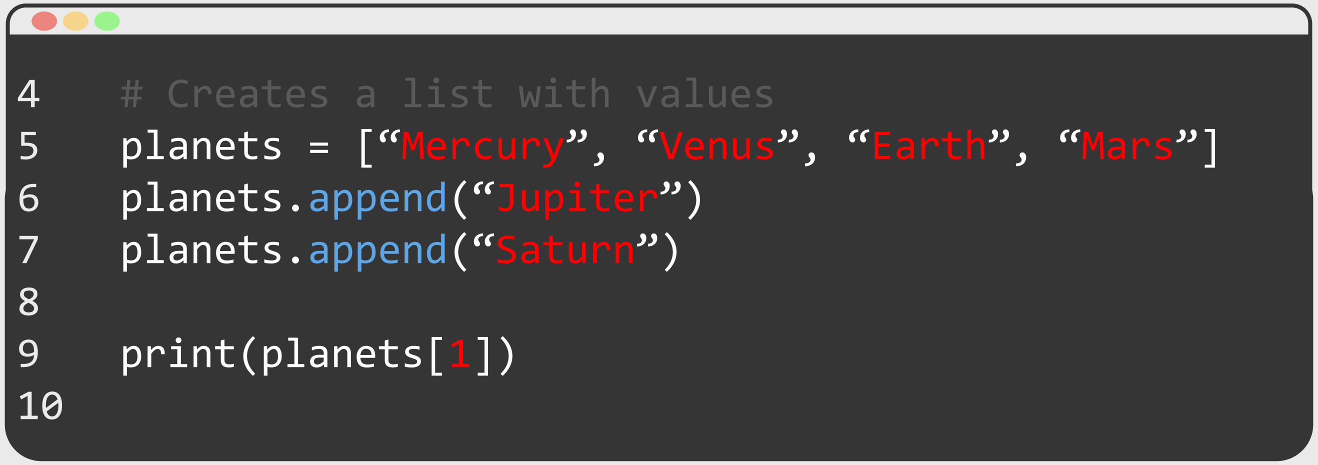


```
# Creates a list with values
planets = ["Mercury", "Venus", "Earth", "Mars"]
planets.append("Jupiter")
planets.append("Saturn")

print(planets)
```

## Retrieving Values from Lists

If we **`print()`** our list, we get all of the contents. What if we only want a single value from the list?

Thankfully for us, Python Lists maintain an **index** of their contents. This means we can access a value by its index number.

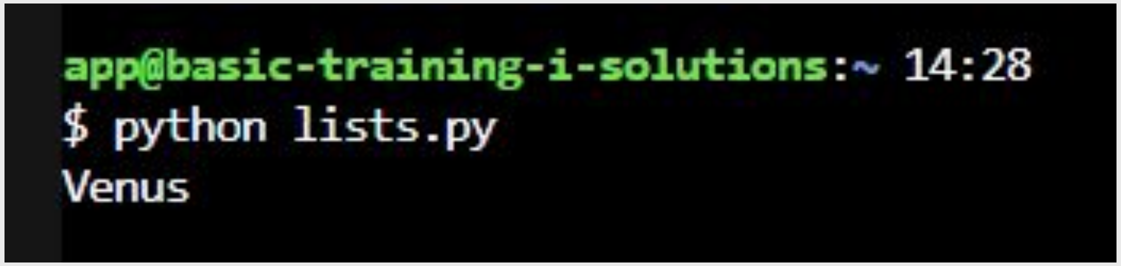


```
# Creates a list with values
planets = [“Mercury”, “Venus”, “Earth”, “Mars”]
planets.append(“Jupiter”)
planets.append(“Saturn”)

print(planets[1])

```

What will be the console output of **`planets[1]`**? Guess before you run the code!

## Retrieving Values from Lists

You might have guessed Mercury - it's the first value in the list, right?



```
app@basic-training-i-solutions:~ 14:28
$ python lists.py
Venus
```

But the console prints Venus! Why is that?

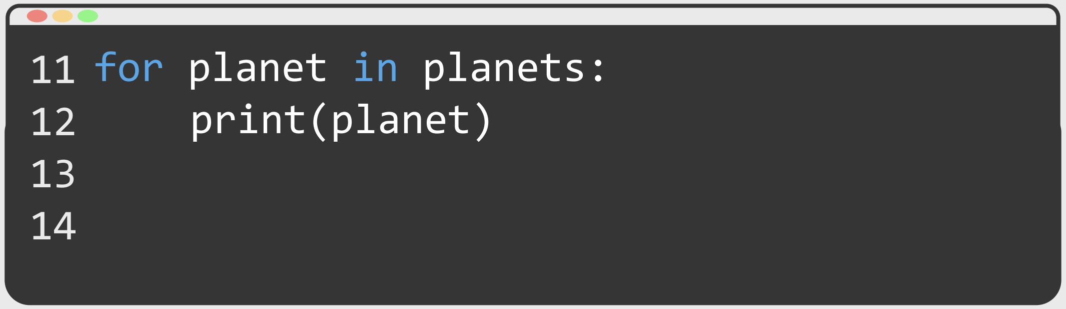
Lists start at index Zero! To a computer, 0 is a perfectly useable number.

Try printing out **`planets[0]`** instead.

## Looping through Lists

We covered Loops, and Lists, so how about... Looping over lists!

Add this loop to **`lists.py`**.



```python
for planet in planets:
    print(planet)
```

Run **`lists.py`** and check the output.

This is called **iterating** over a list.

## We’re flying through this!

## Let’s talk about Importing.

## Imports

As you've seen so far, Python has a lot of in-built functions, like **`print()`**, **`upper()`**, and **`range()`**.

Sometimes we want to extend the functionality of Python by using functions that someone else has written for a specific purpose - these are called Modules.

We can even import other Python files we've written ourselves, so we can use their functions!

## Imports

Take a look at the **returns.py** file.

By creating a Python file with some functions in it, we've created a **Module** called **Returns**.

```
def create_full_name(first_name, last_name):
    full_name = first_name.upper() + “ “ + last_name.upper()
    return full_name

result = create_full_name(“Neil”, “Armstrong”)
print(result)
```

What if we wanted to use the **create_full_name** function in another Python file? Let's do that!

## Importing Our Functions

Open up **`imports.py`** and add this:

```python
import returns

print(returns.create_full_name(“Chris”, “Hadfield”))
```

We use the **import** keyword to import modules we want to use.

Here, we’ve imported our other Python script, **`returns.py`**!

Now we can use the function we wrote inside **`returns.py`**.

All we have to do is tell Python where the function is coming from, by typing **`returns.create_full_name`** instead of just the function name.

# Importing External Modules

A useful module to have is the **Random** module. It helps us to generate random numbers.

```python
import random

print(random.randint(0,100))
```

The Random module has a function called **`randint()`** (**Rand**om **Int**eger).

We can give it two parameters: a start point and an end point.

This will generate a random whole number between these two values.

## What a whirlwind!

## We’ve learned a lot today.

## How about we put this all together?

## Challenge

So far, we've learned about Variables, Loops, Lists, Functions, Importing, and more!

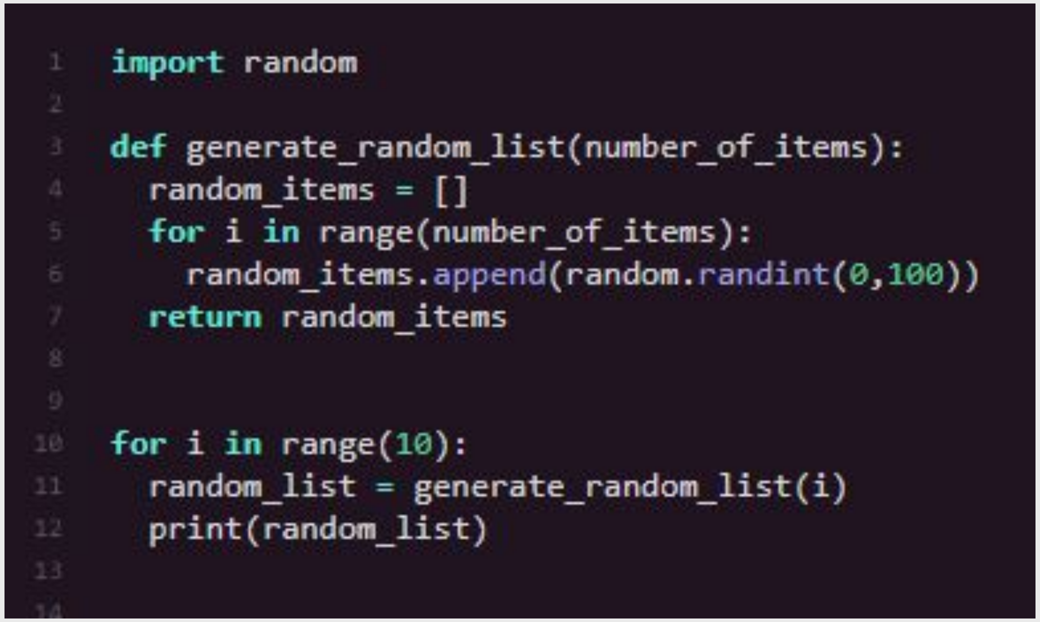
Let's write a script that combines these ideas!

Open up **`challenge.py`** and give it a try!

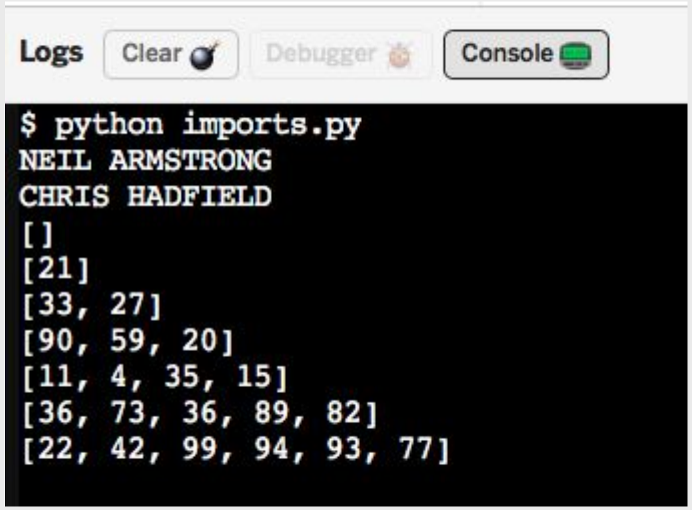
## Challenge Time!

**Find yourself a partner -  someone in the room that you don’t know!**

## Challenge Solution



```python
import random

def generate_random_list(number_of_items):
  random_items = []
  for i in range(number_of_items):
    random_items.append(random.randint(0,100))
  return random_items


for i in range(10):
  random_list = generate_random_list(i)
  print(random_list)
```



```
$ python imports.py
NEIL ARMSTRONG
CHRIS HADFIELD
[]
[21]
[33, 27]
[90, 59, 20]
[11, 4, 35, 15]
[36, 73, 36, 89, 82]
[22, 42, 99, 94, 93, 77]
```

It’s important to remember that coding problems have lots of different solutions! If yours doesn’t look like this, don’t worry.

# Table of Contents

## Let's Recap

### *Where to go from here...*

1. Python is a programming language you can use for data analysis and artificial intelligence
2. Python tools you can use include functions, lists and loops.
3. We can extend Python functionality even further by importing modules.

## What did you learn today?

We created a fun quiz to test your knowledge and see what you learned from this workshop.

**http://mlhlocal.host/quiz**

# Table of Contents

## Next Steps

***Where to go from here...***

1. Check your email for lots of resources to keep learning!
2. Edit the part of the file that prints out the timeline however you want.
3. Take the second workshop!

MLH localhost

Workshop

# Basic Training: Intro to Python Skills for AI, Part I

v.0.0.1