

# HackerFrogs Afterschool

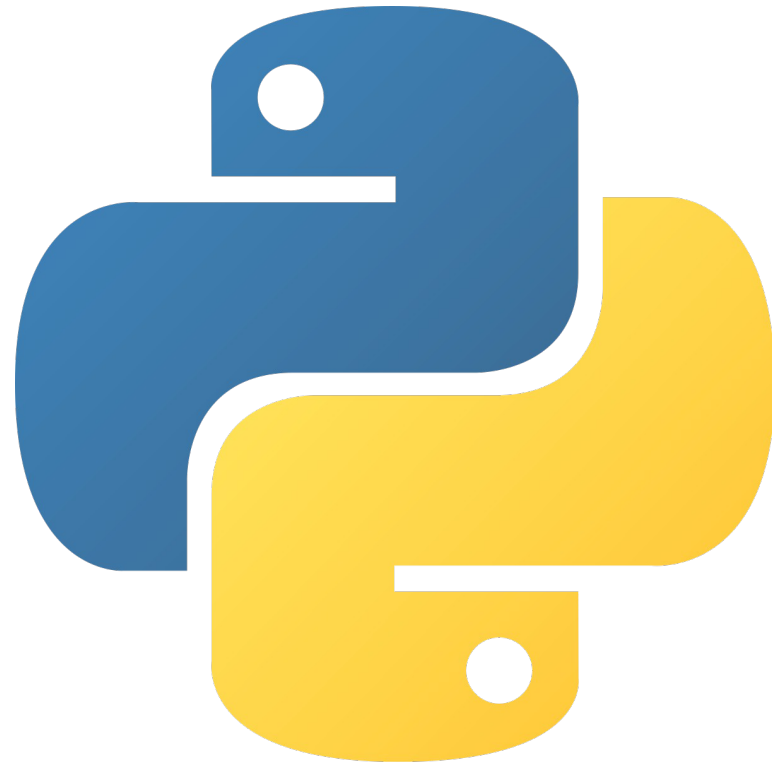
## Python Programming Basics: Part 1

Class:  
Programming (Python)

Workshop Number:  
AS-PRO-PY-01

Document Version:  
1.5

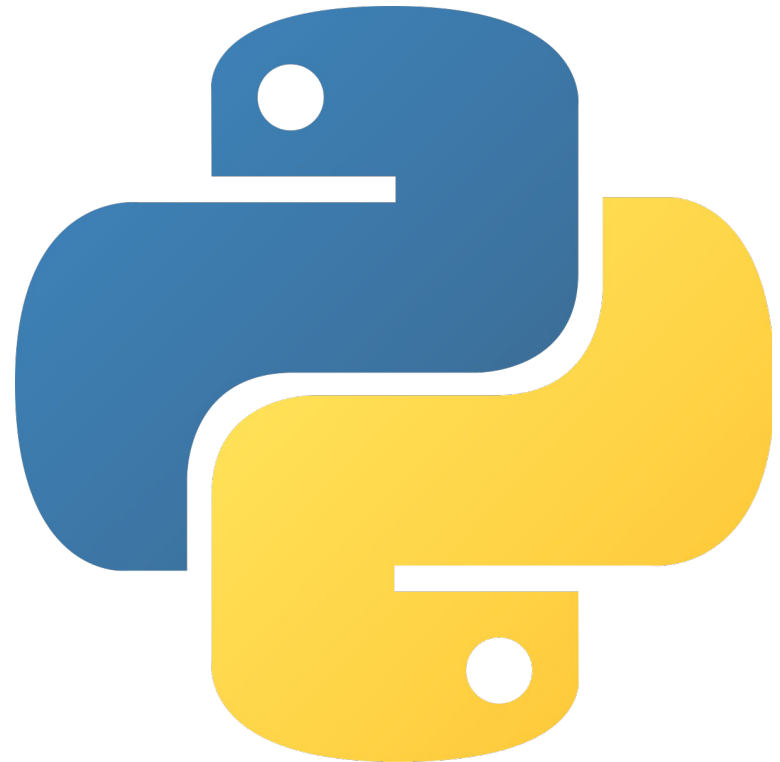
Special Requirements:  
None



# Python Programming Intro

This is the first  
workshop to intro  
Python programming.

Let us begin!



# Part 1: Hello, World!



Programmers learning a language usually write a “Hello, World!” program as their first exercise.

# Hello, World!



```
Hello, World_
```

To complete our **Hello, World!** exercise, we'll go over the Python print function first.

# The Print Function

```
>>> print("This is how you print text in Python!")  
This is how you print text in Python!
```

Virtually all programming languages have some method of printing text to the screen (often referred to as the console). In Python, the **print** function is the primary method of doing so.

# The Print Function

```
>>> print("This is how you print text in Python!")  
This is how you print text in Python!
```

To use the **print** function, we use the function name, **print**, then a pair of parentheses, then encase what we want printed, in either single or double quotes, between the parentheses.

# The Print Function

```
>>> variable = "A variable could be a lot of different things!"  
>>> print(variable)  
A variable could be a lot of different things!
```

Another common use of the print function is to print the values of variables to the console. In this case, we simply input the name of the variable we want printed between the parentheses.

# The Print Function

```
print "Hello, world!"
```

```
File "main.py", line 2
```

```
    print "Hello, world!"
```

```
        ^
```

```
SyntaxError: Missing parentheses in call to 'print'. Did you mean print("Hello, world!")?
```

The syntax for the Print function requires parentheses to work properly. If omitted, Python will display an error message when run.



# The Print Function

```
print('Printed using single quotes.')
```

```
print("Printed using double quotes.")
```

```
Printed using single quotes.  
Printed using double quotes.
```

Also, if we print text, we have to use either use single quotes or double to wrap what we want to print.

# Let's Learn with Learnpython.org

For a more hand-on and interactive session, we'll be using the Learnpython.org website. Please navigate to the following URL:

[https://learnpython.org/en/Hello,\\_World!](https://learnpython.org/en/Hello,_World!)

# Part 1 – Hello World Quiz - Q1

When we use the print function to print text, should we use single quotes or double quotes?

- A) You should only use double quotes ( " " )
- B) You should only use single quotes ( ' ' )
- C) You can use either single or double quotes ( ' ' , " " )
- D) Neither, you should use backticks ( ` ` )

# Part 1 – Hello World Quiz - Q1

When we use the print function to print text, should we use single quotes or double quotes?

- A) You should only use double quotes ( " " )
- B) You should only use single quotes ( ' ' )
- C) You can use either single or double quotes ( ' ' , " " )
- D) Neither, you should use backticks ( ` ` )

# Part 1 – Hello World Quiz - Q2

What happens if you use the Print function without parentheses?

- A) The Print function will still work
- B) An error message appears at runtime
- C) Parentheses only affect whether or not there are parentheses in the output.
- D) None of the above

# Part 2: Variables

Virtually all programming languages use variables, which are storage locations associated with a specific name, as well as a value. Take the following as an example:

```
my_name = 'shyhat'
```

# Variables

Wherever we refer to **my\_name** in the code, **shyhat** will be inserted there. So, if we have the following code:

```
my_name = 'shyhat'  
print(my_name)
```

The output should be **shyhat**.

# Variables

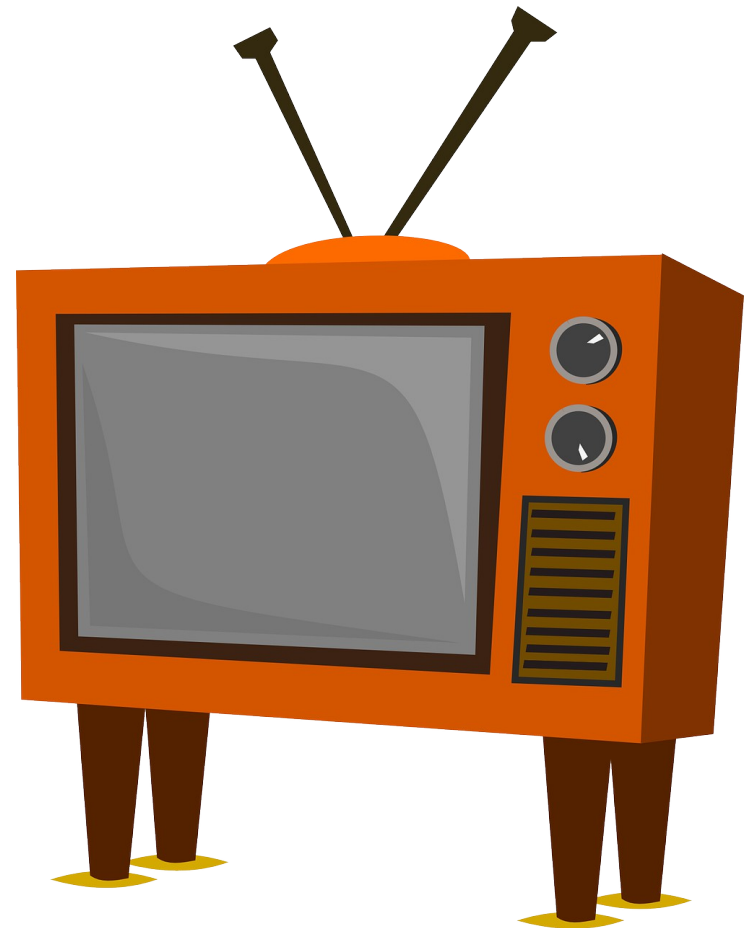
The real benefit of using variables in programming is that a variables can be referenced multiple times in the same code, saving a lot of time and accounting while writing code.

In addition, variables can change value after they've been defined.



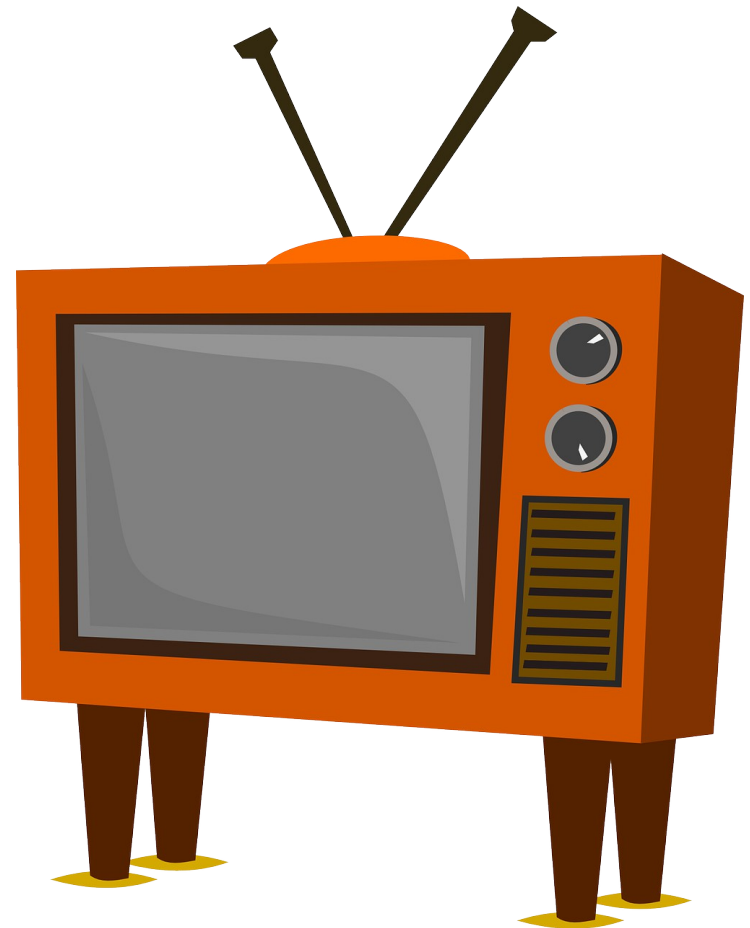
# Variables

For example, say we have an online store program where we sell televisions.



# Variables

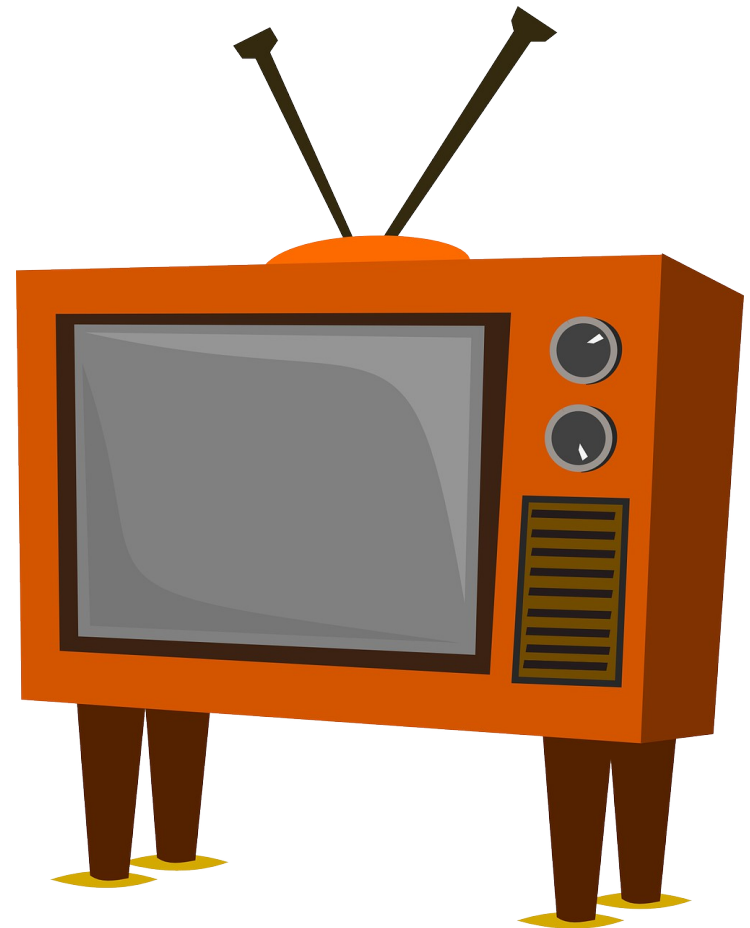
The program has two variables, **tv\_price**, which is the price of a television, and **current\_total**, which keeps track of the total price of the user's shopping cart.



# Variables

The two variables are set to 199.99 and 0, respectively. However, if a TV is purchased, then the **total\_price** variable increases by a value equal to the **tv\_price** variable.

See the following:



# Variables

```
tv_price = 199.99  
current_total = 0
```

```
current_total = current_total + tv_price
```

```
print(current_total)
```

The output should be **199.99**

# How to name Variables (1)

```
a = "We are Hackerfrogs!"
```

```
HF_slogan = "We are HackerFrogs!"
```

First, variable names should be as short as possible, while still describing its value.

# How to name Variables (1)

```
a = "We are Hackersfrogs!"
```

```
HF_slogan = "We are HackerFrogs!"
```

Letters like **a** and **x** are sometimes okay variable names, but most of the time variable names should be descriptive.

# How to name Variables (2)

```
class = "Economics"
```

```
True = "Pineapple pizza is the best"
```

There is a list of keywords used in Python which cannot be used as variable names, otherwise your program won't run.

# How to name Variables (2)

And, assert, break, class, continue, else,  
except, False, finally, import, return,  
True, with

Here's a short list of keywords that could be  
mistakenly used as variable names.



# How to name Variables (2)

```
my_class = "Economics"
```

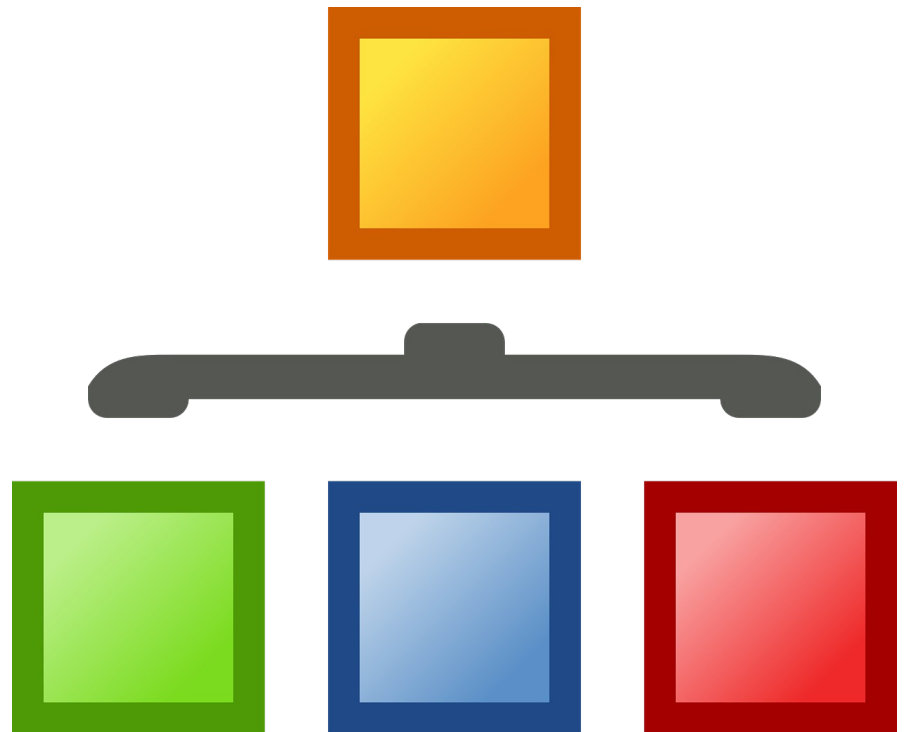
```
my_True = "Pineapple pizza is the best"
```

To get around this restriction, we can prepend any name that we suspect could be keyword with "my\_".

# Data Types

Data types are attributes of data which tell the compiler or interpreter how to use the data.

The data types covered in this workshop are as follows:



# Data Types

## **Strings (str)**

Non-numerical characters, or a combination of numerical and non-numerical characters. E.g, “Mark Twain”, “P@ss\Word123”, “carrot”

## **Integers (int)**

Whole numbers. E.g., 1, 255, 1337

## **Floating-Point Numbers (float)**

Numbers that are accurate to several decimal places. E.g., 0.5 or 12.758

# Data Types



The reason why data types are important is because computers do not inherently know how to interpret data.

# Data Types



For example, if we have a program which calculates the average of school grades (adds all grades together, then divides the sum by the number of assignments) --

# Data Types



the program wouldn't work if it was given words instead of numbers for the grades.

# Data Types

Error: Grades cannot contain non-numeric input.

Because there are data types, we can have the program check that all of the grades are numbers, and if there is a non-number in the input, we can inform the user that the program only accepts numbers for input, and prompt the user to input another value.

# Data Types – The type function

```
id_string = type("HackerFrogs")  
id_int = type(1)  
id_float = type(3.14)  
print(id_string, id_int, id_float)
```

```
<class 'str'> <class 'int'> <class 'float'>
```

There's a handy function (type) in Python that can be used to check what data type a given object is



# The Input Function – Users Type It In

```
fav_fruit = input("What is your favorite fruit? ")  
print("Your favorite fruit is " + fav_fruit)
```

```
What is your favorite fruit?  
pineapple  
Your favorite fruit is pineapple
```

The last useful function we'll learn is the **Input** function, which can turn user input into a variable

# The Input Function – Users Type It In

```
fav_fruit = input("What is your favorite fruit? ")  
print("Your favorite fruit is " + fav_fruit)
```

```
What is your favorite fruit?  
pineapple  
Your favorite fruit is pineapple
```

This is a typical example of prompting user input, and it can be used to assign variables.

# Variables and Types Exercise

Let's cover the next few exercises on the  
Learnpython.org website:

[https://www.learnpython.org/en/Variables\\_and\\_Types](https://www.learnpython.org/en/Variables_and_Types)

## Part 2 – Variables and Types Quiz (Q1)

Which of the following about naming variables is true?

- A) Python keywords (e.g., class, def, True, False, None) should never be used as variable names
- B) Use single-letter variable names (x, y, a, b) as much as possible
- C) Variables that contain number should be named in UPPERCASE letters, and variables with text should be named in lowercase letters.
- D) None of the above

## Part 2 – Variables and Types Quiz (Q1)

Which of the following about naming variables is true?

A) Python keywords (e.g., class, def, True, False, None) should never be used as variable names

B) Use single-letter variable names (x, y, a, b) as much as possible

C) Variables that contain number should be named in UPPERCASE letters, and variables with text should be named in lowercase letters.

D) None of the above

## Part 2 – Variables and Types Quiz (Q2)

Which of the following is not a Python data type?

- A) Strings – letters or combinations of letters and number that are encased in single or double quotes
- B) Integers – whole numbers, such as 1, 1337, and 256. If put between quotes, they become strings.
- C) Floats (floating point numbers) – numbers with decimal points, such as 0.0, 5.285, and 3.16.
- D) HackerFrogs – folks who like hacking and live streams

## Part 2 – Variables and Types Quiz (Q2)

Which of the following is not a Python data type?

- A) Strings – letters or combinations of letters and number that are encased in single or double quotes
- B) Integers – whole numbers, such as 1, 1337, and 256. If put between quotes, they become strings.
- C) Floats (floating point numbers) – numbers with decimal points, such as 0.0, 5.285, and 3.16.
- D) HackerFrogs – folks who like hacking and live streams

# Workshop Review Exercise

Before finishing today's workshop, let's learn how to run Python on picoCTF:

[https://play.picoctf.org/practice/challenge/250?  
originalEvent=69&page=1&search=](https://play.picoctf.org/practice/challenge/250?originalEvent=69&page=1&search=)




# Summary



Let's review the programming concepts we learned in this workshop:

# Hello, World!



Hello, World\_

We wrote the **Hello, World!** program in Python, which is the first program students write for any computer language.

# The Print Function

```
>>> print("This is how you print text in Python!")  
This is how you print text in Python!
```

We learned how to use the Python print function, which prints information to the console and is common to almost all programs written in Python.

# Variables

```
>>> name = "HackerFrogs"  
>>> greetings = "Hello, "  
>>> print(greetings + name)  
Hello, HackerFrogs
```

We learned about programming variables, named storage locations which hold specific values, which can be referenced multiple times in a program.

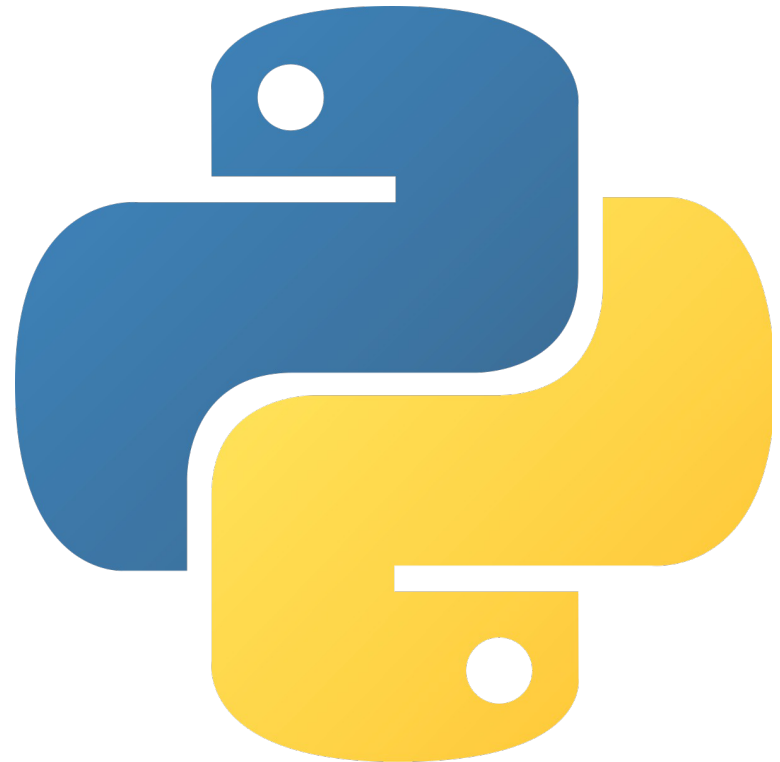
# Data Types

```
>>> pi = 3.14
>>> number = 1337
>>> name = "shyhat"
>>> print(type(pi),type(number),type(name))
(<class 'float'>, <class 'int'>, <class 'str'>)
```

We learned about Data Types, including **strings** (non-numeric text), **integers** (whole numbers), and **floats** (numbers which include decimal places).

# What's Next?

In the next HackerFrogs Afterschool programming workshop, we'll continue learning Python with the [learnpython.org](https://learnpython.org) website.



# Next Workshop's Topics

- Python Lists
- List methods
- Basic Arithmetic Operations

# Extra Credit

Looking for more study material on this workshop's topics?

See this video's description for links to supplemental documents and exercises!





# Until Next Time, HackerFrogs!

