

INTRO TO PYTHON

E1/E2 2021



1

VARIABLES + DATA TYPES



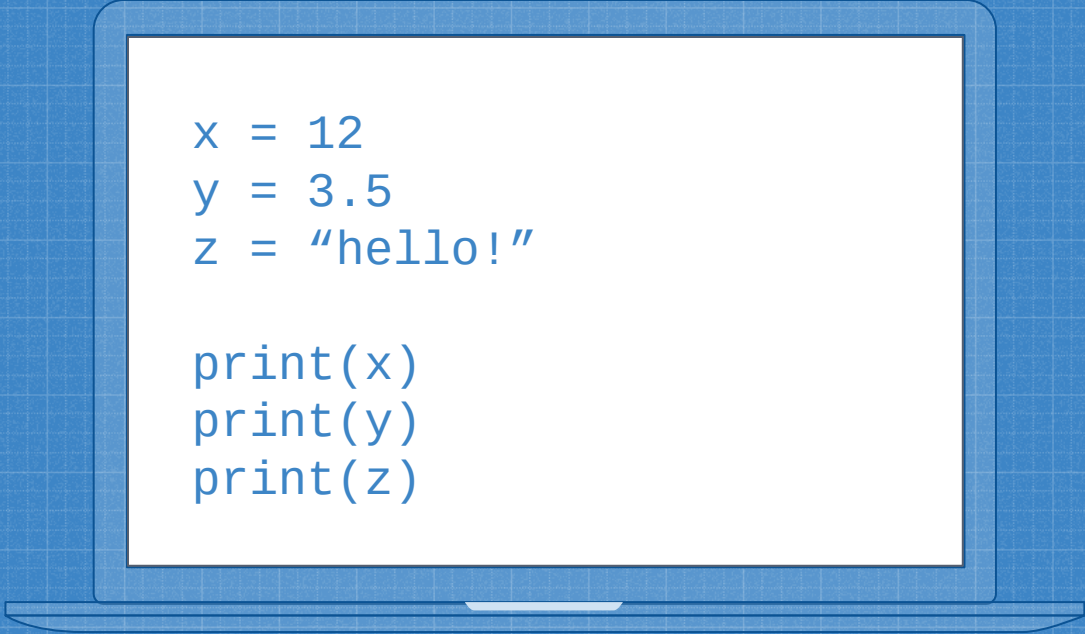
Let's start with the
very basics



VARIABLES

A variable can be thought of as a label or container for a piece of data.

To create a variable, all you have to do is **declare** it:



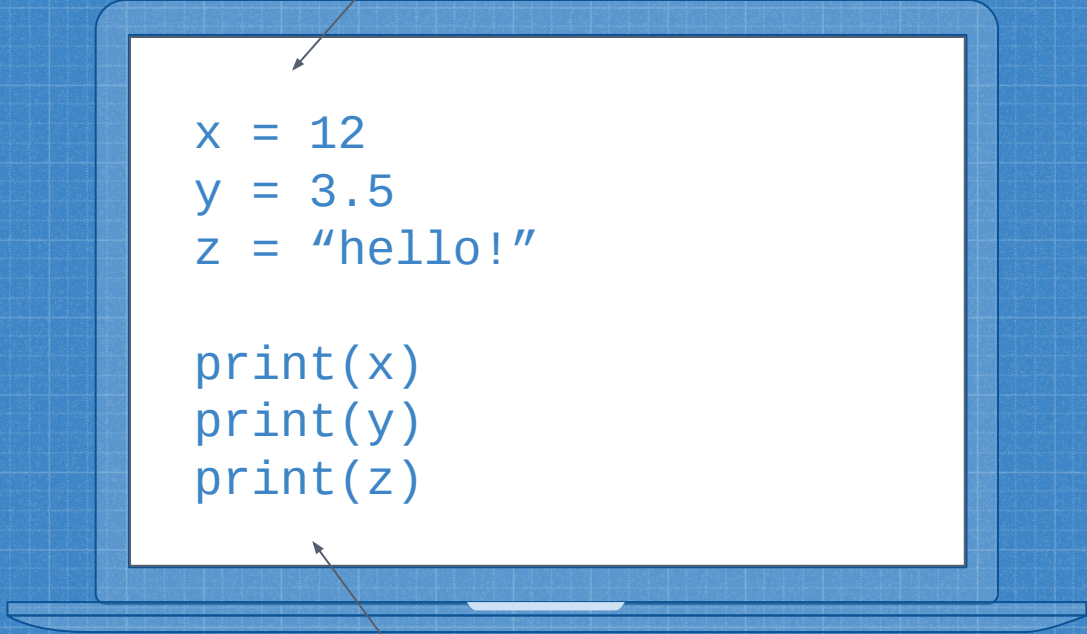
```
x = 12
y = 3.5
z = "hello!"

print(x)
print(y)
print(z)
```


VARIABLES

A variable can be thought of as a label or container for a data value.

To create a variable, all you have to do is **declare** it:



```
x = 12  
y = 3.5  
z = "hello!"
```

```
print(x)  
print(y)  
print(z)
```

These are variable
declarations

These are **print**
statements

VARIABLES

Variables can change value and even type after they have been declared!

Names of variables are also case sensitive.

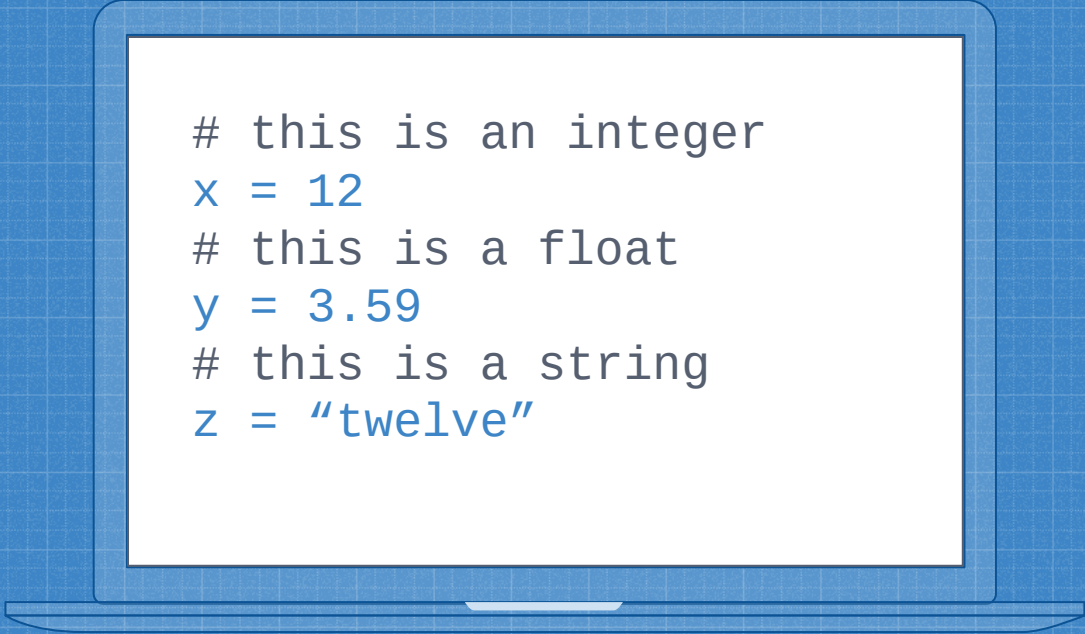
```
number = 12  
Number = 16  
number = "twelve"
```

```
print(number)  
print(Number)
```


DATA TYPES + COMMENTS

Data types let
Python know how to
treat a piece of
information.

Variables have no
set data type!

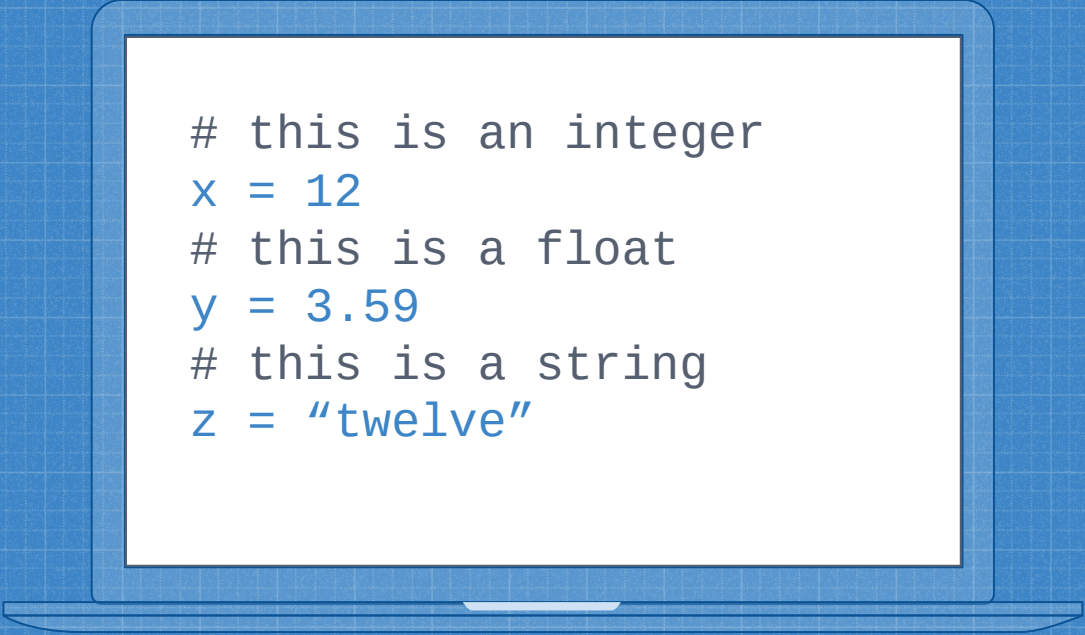


```
# this is an integer  
x = 12  
# this is a float  
y = 3.59  
# this is a string  
z = "twelve"
```


DATA TYPES + COMMENTS

Comments always start with a '#' symbol. Anything after a '#' on a line will be ignored by Python.

Use comments often to help you organize your code!

A blue laptop with a white screen. The screen displays Python code with comments. The code is as follows:

```
# this is an integer  
x = 12  
# this is a float  
y = 3.59  
# this is a string  
z = "twelve"
```

```
# this is an integer  
x = 12  
# this is a float  
y = 3.59  
# this is a string  
z = "twelve"
```


VARIABLE MATH

Once you have created a variable, you can use it as a number (or string!).

You can add, subtract, multiply, divide, and more! Python follows PEMDAS for order of operations.

```
x = 12  
y = 3.5
```

```
add = x + y  
sub = x - y  
mult = x * y  
pwr = x ** 2
```

This is x squared

VARIABLE MATH

Variables must be declared before they can be used - the order matters!

For example, this code will not work...

This won't work

```
x = x0+v0*t+0.5*a*t**2  
print("x =", x)
```

```
x0 = 0  
v0 = 5.2  
t = 10  
a = 9.8
```


VARIABLE MATH

Variables must be declared before they can be used - the order matters!

...but this code will!

```
x0 = 0
```

```
v0 = 5.2
```

```
t = 10
```

```
a = 9.8
```

```
x = x0+v0*t+0.5*a*t**2
```

```
print("x =", x)
```

This will work!



2

FUNCTIONS + MODULES

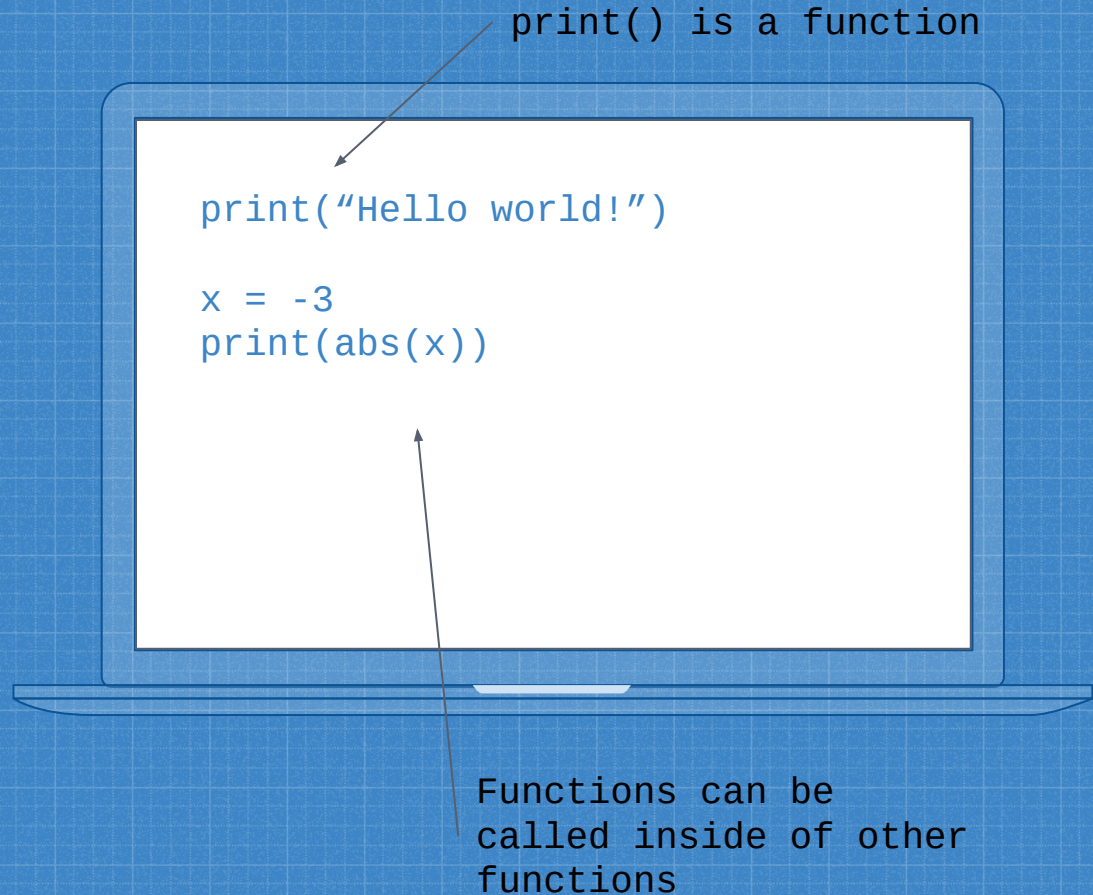


Ok, so what else can you
do with variables?

FUNCTIONS

Functions are blocks of code that can be run when called.

Many functions take at least one **argument**, which represents data that you are giving to the function.

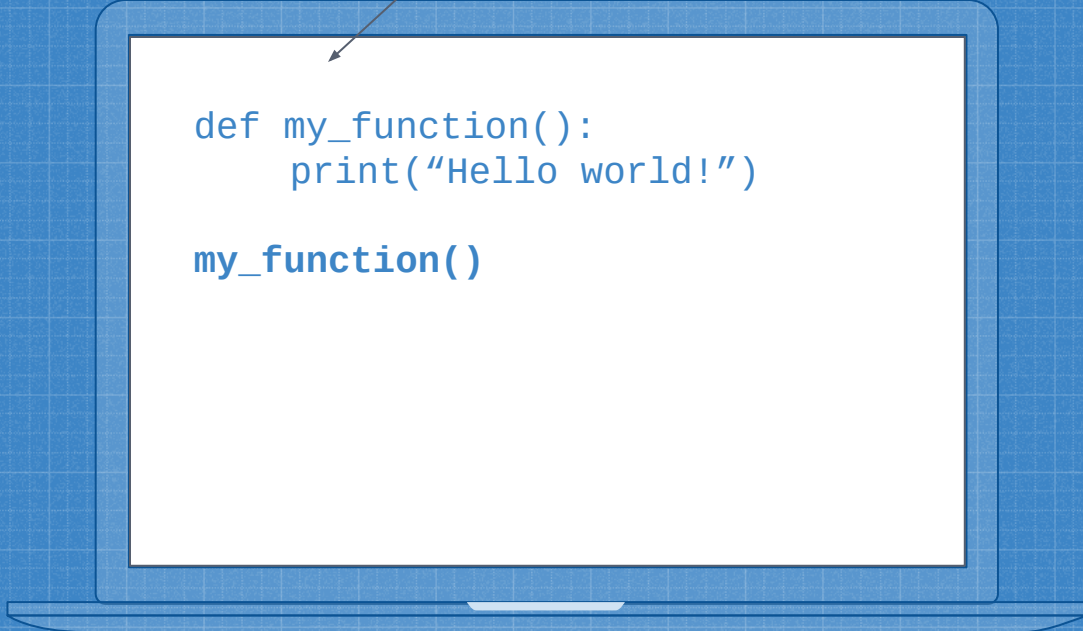


FUNCTIONS

Creating a function is easy! Don't forget to indent everything you want to include.

To use a function, all you have to do is write its name followed by parentheses: ()

How to define a simple function



```
def my_function():  
    print("Hello world!")  
  
my_function()
```


FUNCTIONS

Your functions can also take arguments, or placeholders for data values. This is really useful for when you have to do the same operation multiple times on different data!

```
def my_function(aname):  
    print("Hello " + aname)  
  
my_function("Amelia")  
my_function("friend")  
my_function("world!")
```


FUNCTIONS

Here's an example of a slightly more complicated function which can be used to find the error propagation for an area calculation.

```
import math

def u_area(x,dx,y,dy):
    area = x * y
    x_comp = (dx / abs(x)) ** 2
    y_comp = (dy / abs(y)) ** 2
    ans = area * math.sqrt(x_comp +
                           y_comp)
    return ans

print(u_area(5.3, 0.1, 8.5, 0.1))
```


MODULES

Modules are collections of useful methods and variables that must be imported into Python to be used.

Put import statements at the beginning

```
import math
import numpy
import matplotlib.pyplot as plt
import scipy.stats as stat
```

You can nickname modules with long names for convenience

MODULES

Most of the time,
you will be
importing modules
for their functions.

Numpy has useful
functions, but we
also use it for
constants, arrays,
and much more!

```
import numpy

x = numpy.sin(numpy.pi/2.0)
y = numpy.hypot(3.0, 4.0)

print("sin(pi/2) = " + x)
print("the hypotenuse is " + y)
```

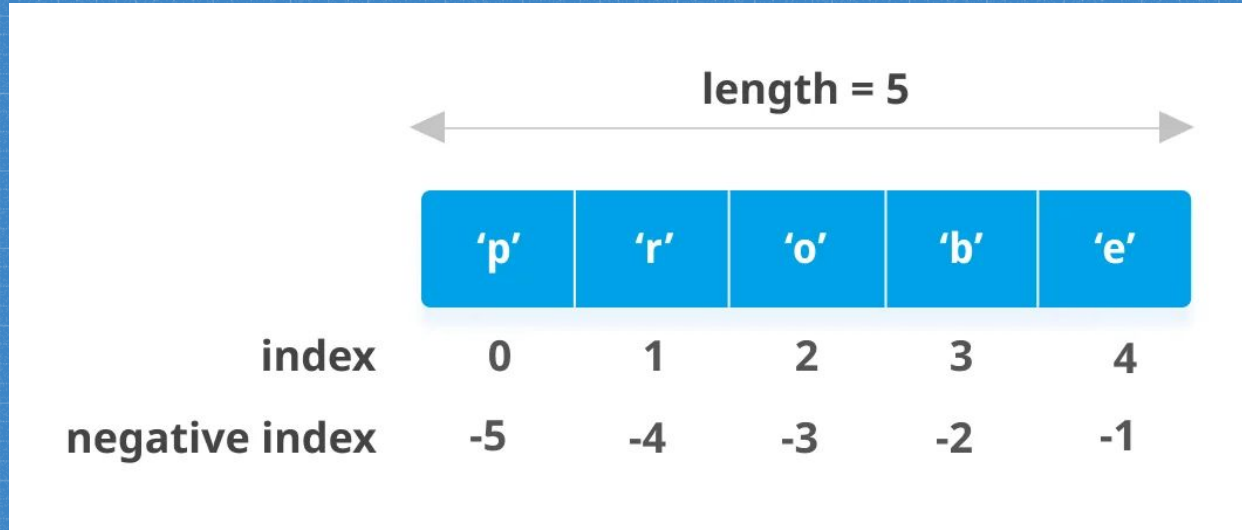



3

ARRAYS + LOOPS



Featuring numpy!



INDEXING

In most programming languages, including Python, the first item in a list is actually at the 0th index!

ARRAYS

Arrays are simply ordered lists of items - integers, floats, strings, etc.

Use the name of the array followed by square brackets to reference individual items in the array.

```
import numpy as np

x = np.array([1,2,3])
colors = np.array(["red", "blue"])

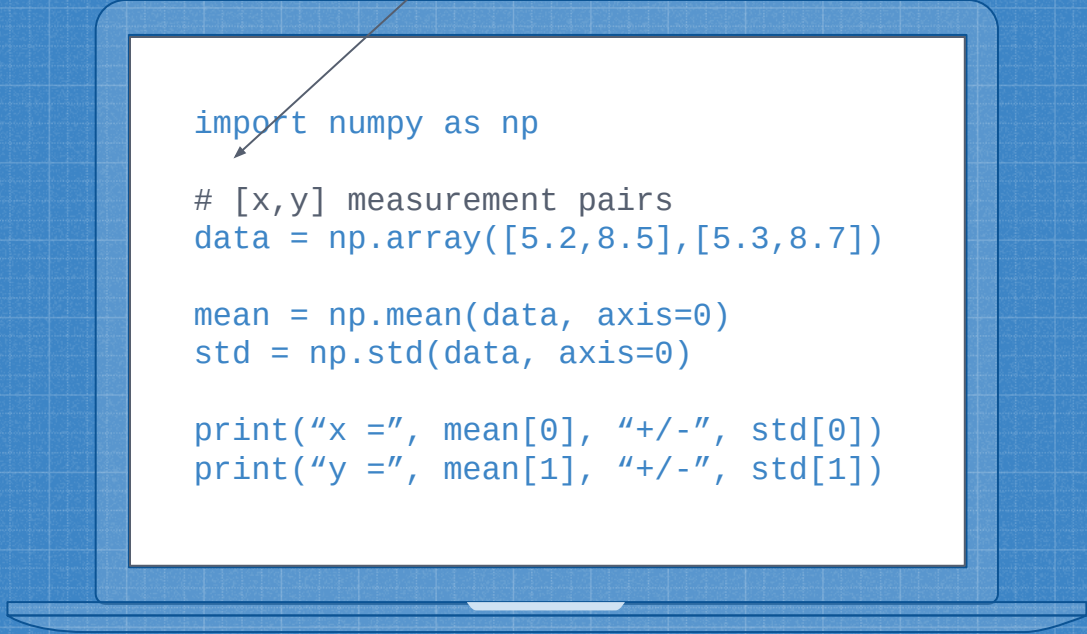
# print the first item in colors
print(colors[0])

# add the second + third items of x
print(x[1] + x[2])
```


ARRAYS

Numpy gives us an array structure with a lot of data analysis functions included. In this example, we find the mean and standard deviation of each column, and store these values in new arrays.

You can store data tables as multidimensional arrays



```
import numpy as np

# [x,y] measurement pairs
data = np.array([5.2,8.5],[5.3,8.7])

mean = np.mean(data, axis=0)
std = np.std(data, axis=0)

print("x =", mean[0], "+/-", std[0])
print("y =", mean[1], "+/-", std[1])
```

Don't worry if this doesn't make sense just yet!

LOOPS

A loop is used when you need to iterate over some list or array, or when you need to perform the same operation a set number of times.

A string in Python is a list of letters

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

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

What do you expect the output of this loop to be?

LOOPS

Loops are most useful when used with arrays.

```
import numpy as np

sum = 0
nums = np.array([4,3,2,1])

for x in nums:
    sum = sum + x

print(sum)
```

What do you expect this output to be?

MORE RESOURCES

- Codecademy - interactive beginner lessons
- Datacamp - same deal, data science focus
- W3schools - great quick reference

And your classmates and instructors!