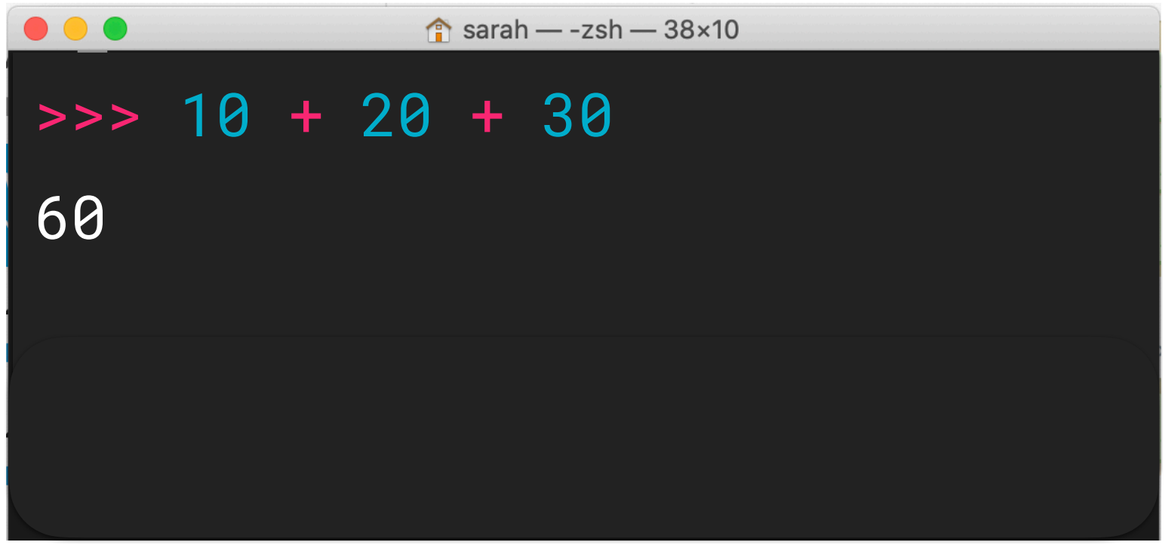
Practical Python for Beginners

DATA TYPES, INPUT, AND OUTPUT

Sarah Holderness
PLURALSIGHT AUTHOR
@dr_holderness

Where Do We Write Python Code?

The Python Interpreter



The Python interpreter let's you run Python lines of code one at time.

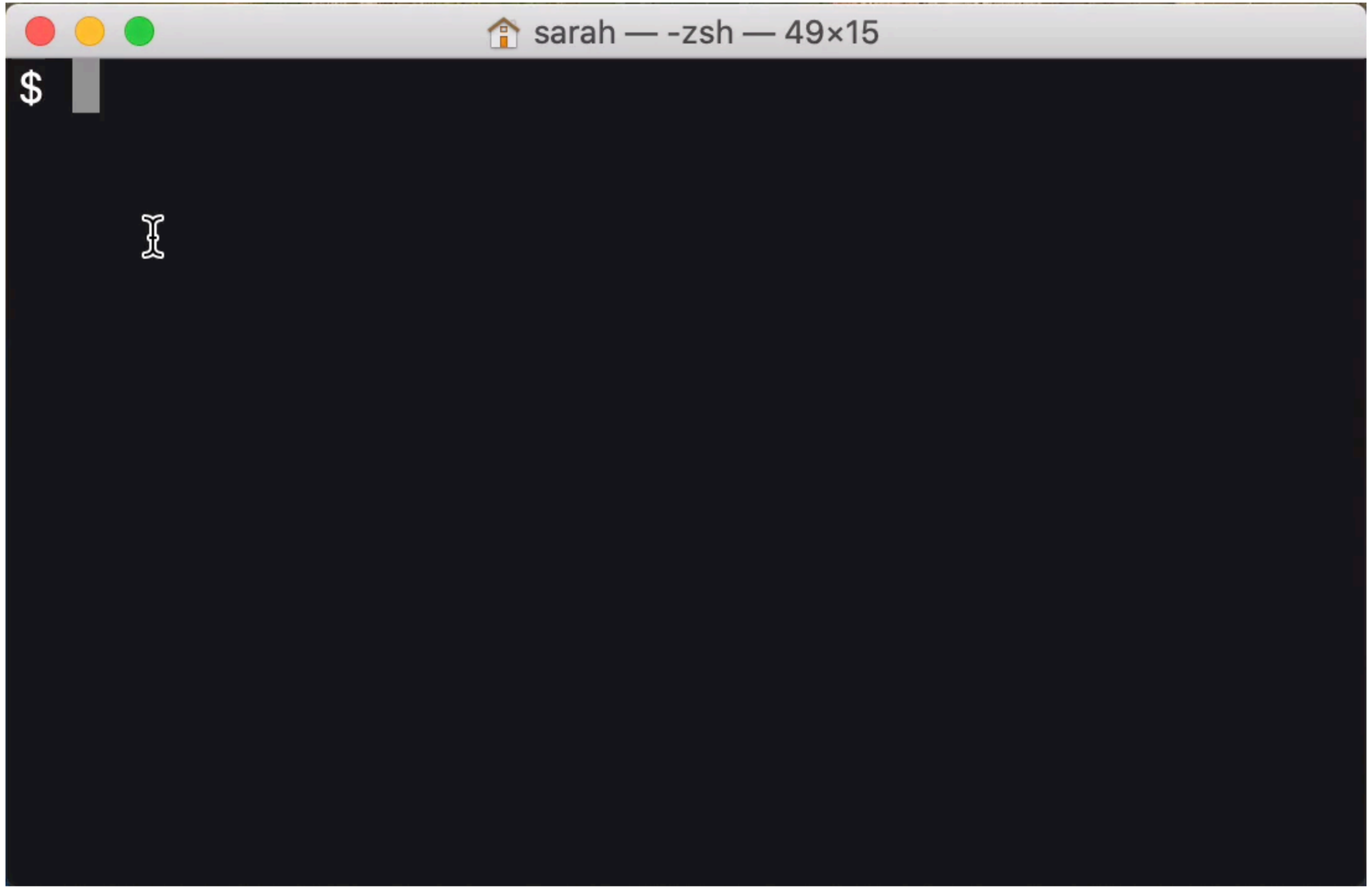
A Python Script

script.py

```
amount = 10 + 20 + 30
print(total)
```

A Python script is where you will create longer Python programs.

The Python Interpreter



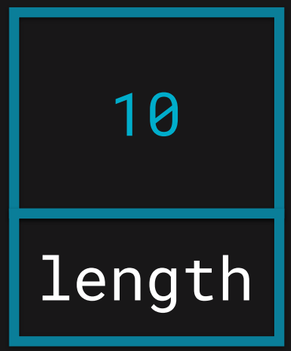
To start the Python interpreter on a Mac enter `python3` from the Terminal.

To start the Python interpreter in Windows enter `py` from the Command Prompt.

Saving Numbers to Variables

Assigning the value `10` *to the variable* `length`.

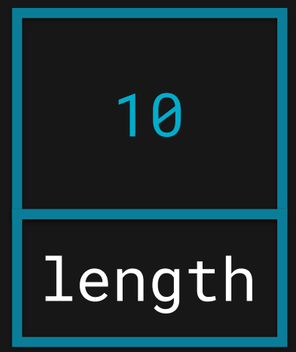
```
>>> length = 10
```



Now on your computer there is a piece of memory labeled `length` *that stores the value* `10`.

Saving Numbers to Variables

```
>>> length = 10
>>> length
10
```



From the interpreter we can enter the name of the variable `length` *to see its value and see that it's actually* `10`.

Saving Numbers to Variables

```
>>> length = 10
>>> width = 20
```

Let's also add the width of the rectangle. Now we have another variable stored in memory.

Saving Numbers to Variables

```
>>> length = 10
>>> width = 20
>>> area = length*width
```

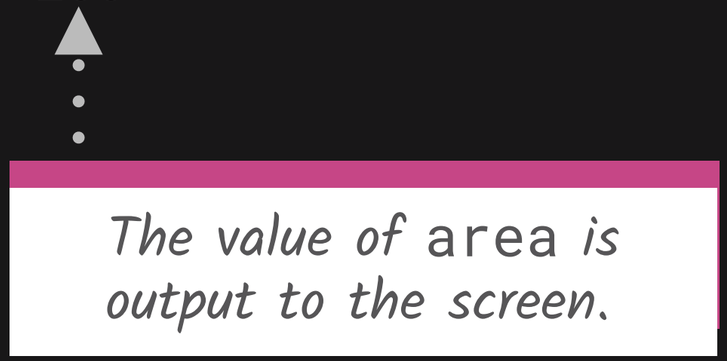
*Now we can calculate the area with the multiplication, * , operator.*

And now we have another variable stored in memory.

*The arithmetic operators in Python are mostly the same ones you know already from a calculator: +, -, *, /*

Saving Numbers to Variables

```
>>> length = 10
>>> width = 20
>>> area = length*width
>>> area
200
```



The value of `area` *is output to the screen.*

Primitive Data Types

Python assumes the type of a variable based on the assigned value

int

```
>>> amount = 10
```

Python infers that `amount` *is an* `int` *since it is a whole number*

float

```
>>> amount = 10.50
```

Python infers that `amount` *is a* `float` *since it is decimal*

Saving an `int` and a `float` to Variables

```
>>> amount = 10
>>> tax = .06
```

Calculating the Total with Sales Tax

```
>>> amount = 10
>>> tax = .06
>>> total = amount + amount*tax
>>> total
10.6
```

A Python Script

A file containing code written in Python

sales_tax.py

```
amount = 10
tax = .06
total = amount + amount*tax
```

```
> python3 sales_tax.py
```

How to run a Python script - `python` followed by the filename.

(The command can also be `python3` on Mac or `py` on Windows.)

A Python Script

sales_tax.py

```
amount = 10
tax = .06
total = amount + amount*tax
total
```

How do we output a value from a Python script?

Just typing the value we want doesn't work like in the shell.

```
> python3 sales_tax.py
```

Notice there's no output.

A Python Script

sales_tax.py

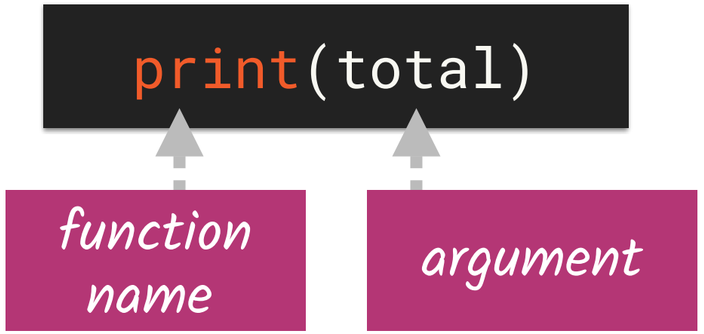
```
amount = 10
tax = .06
total = amount + amount*tax
print(total)
```

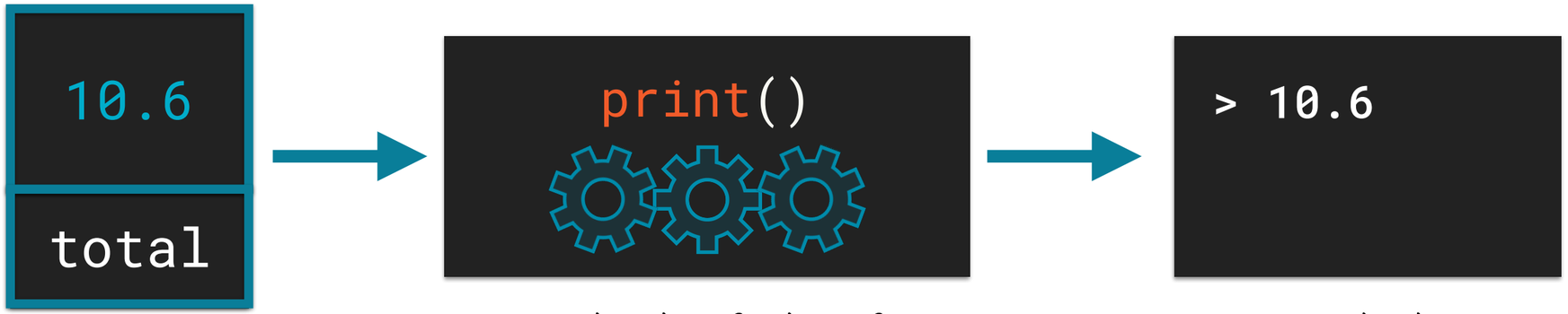
We can call the `print()` function

```
> python3 sales_tax.py
10.6
```

Now the value of `total` is printed to the screen

Python `print()` Function





You can think of this function as a black box machine. We don't know how it works...

But we do know it will output the given value to the screen.

Data Type Conversion Functions

What if we want to convert a float to an int? Or vice versa?

int()

```
>>> amount = int(10.6)
>>> amount
10
```

Use the `int()` *conversion function*

float()

```
>>> amount = float(10)
>>> amount
10.0
```

Use the `float()` *conversion function*

A `String` Stores Text

greeting.py

```
name = 'Sarah'
print(name)
```

Creating a `String` *with single quotes*

The string `'Sarah'` *is saved to the variable* `name`

```
> python3 greeting.py
Sarah
```

The value of `name` *prints without quotes.*

The quotes are only used tell Python that anything inside them is a `String`.

Create Strings with Single or Double Quotes

greeting.py

```
store_name = "Sarah's Store"
print(store_name)

store_name = 'Sarah's Store'
print(store_name)
```

Double quotes can be useful if a single quote is literally part of the `String`

This would cause an error because the second single quote would end the `String` *and Python wouldn't know what to do with the rest.*

String Concatenation

greeting.py

```
hello = "Hello"
name = "Sarah"
greeting = hello + name
print(greeting)
```

Concatenate two Strings with a +

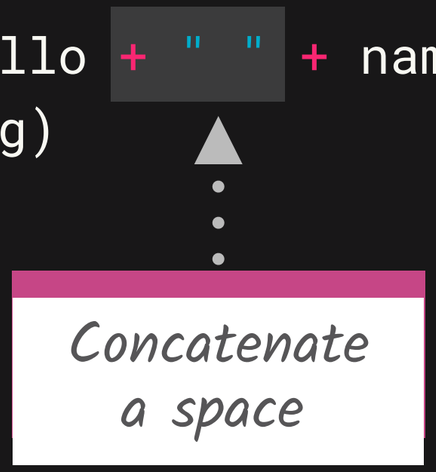
```
> python3 greeting.py
HelloSarah
```

Notice how the Strings are smushed together? We need a space between them.

Fixing Our Program

greeting.py

```
hello = "Hello"
name = "Sarah"
greeting = hello + " " + name
print(greeting)
```




```
> python3 greeting.py
Hello Sarah
```

Fixing Our Program

greeting.py

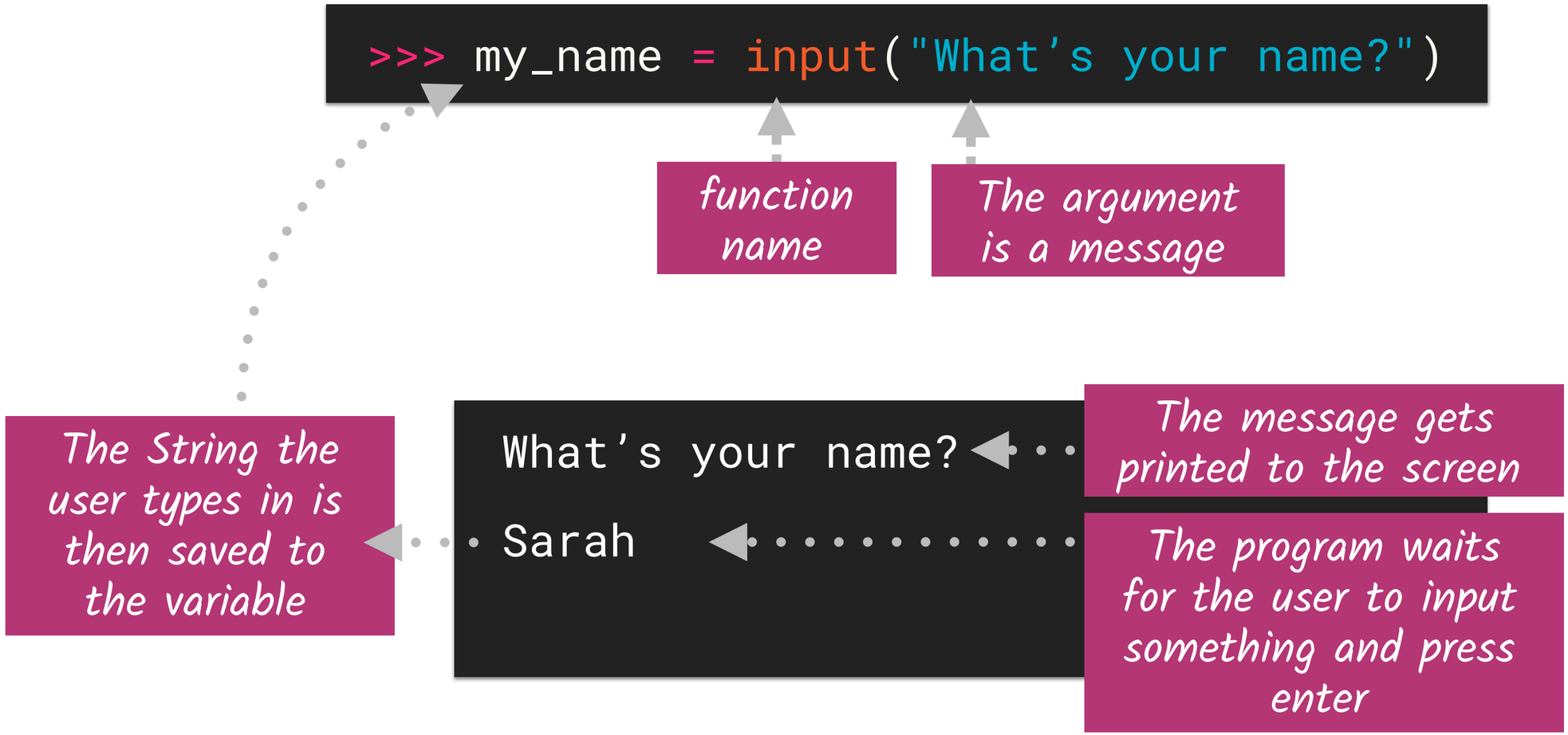
```
hello = "Hello"
name = "Sarah"
greeting = hello + " " + name
print(greeting)
```

Let's ask the user for their name.

```
> python3 greeting.py
Hello Sarah
```

How can we customize this program for other names?

Python **input()** Function



Console Input

greeting.py

```
hello = "Hello"
name = input("What's your name?")
greeting = hello + " " + name
print(greeting)
```

`input()` *prints the statement, then waits for a value from the console*

```
> python3 greeting.py
  What's your name?Bob
  Hello Bob
```

Notice how the name **Bob** *is now printed inside of the greeting.*

Console Input

greeting.py

```
hello = "Hello"
name = input("What's your name?")
greeting = hello + " " + name
print(greeting)
```

```
> python3 greeting.py
  What's your name?Bob
  Hello Bob
```

This looks bad. Can we enter the name on the next line?

Improving Our Program

greeting.py

```
hello = "Hello"
name = input("What's your name?\n")
greeting = hello + " " + name
print(greeting)
```

/n is a special character for a

```
> python3 greeting.py
  What's your name?
  Bob
  Hello Bob
```

Now input is entered on the next line.

Summary of Primitive Data Types

int

```
>>> amount = 10
```

float

```
>>> tax = .06
```

string

```
>>> name = "Sarah"
```

Summary of Input and Output

Input

```
>>> name = input("What's your name?\n")
What's your name?
Sarah
```

Output

```
>>> print("Hello " + name + "!!")
Hello Sarah!!
```