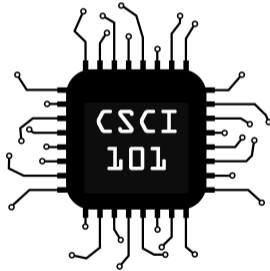


Intro to Python & Programming



Before We Start: A Note on Copy-Paste

When small code examples are shown on the slides, it's probably better for you to type in the code examples yourself than try to copy-paste them. You'll learn more by doing so!

Larger code examples will be provided on the course website rather than in the slides.

Not to mention, some PDF viewers throw crazy characters at you when you try and copy-paste; Python won't like this!

Your First Python Program

Create a *new* file (File menu) and type the following contents:

```
print("Hello, World!")
```

Save this file as `helloworld.py` and run in shell:

```
$ python3 helloworld.py
```

If you don't have a shell open, click the '+' sign and 'New Terminal'

Your First Python Program

Create a *new* file (File menu) and type the following contents:

```
print("Hello, World!")
```

Save this file as `helloworld.py` and run in shell:

```
$ python3 helloworld.py
```

If you don't have a shell open, click the '+' sign and 'New Terminal'

- `print` is a **function**, it takes arguments, does things (in this case, print to our console), and returns things

Your First Python Program

Create a *new* file (File menu) and type the following contents:

```
print("Hello, World!")
```

Save this file as `helloworld.py` and run in shell:

```
$ python3 helloworld.py
```

If you don't have a shell open, click the '+' sign and 'New Terminal'

- `print` is a **function**, it takes arguments, does things (in this case, print to our console), and returns things
- The **arguments** to our `print` call is just the **string** `"Hello, World"`.

Variables

In math, when we write =, we mean **relation**:

$$x^3 - x + 1 = 0$$

Variables

In math, when we write =, we mean **relation**:

$$x^3 - x + 1 = 0$$

This is **not** the same in Python! In Python, = means **assignment**. We let the *variable on the left* take the *value on the right*.

```
x = 10
y = x + 2
x = 11
print(x, y)
```

Variables

In math, when we write =, we mean **relation**:

$$x^3 - x + 1 = 0$$

This is **not** the same in Python! In Python, = means **assignment**. We let the *variable on the left* take the *value on the right*.

```
x = 10
y = x + 2
x = 11
print(x, y)
```

```
11 12
```

Type into `helloworld.py` and run to make sure you get the same answer.

Variables

```
x = 10  
y = x + 2  
x = 11  
print(x, y)
```

```
11 12
```

Type into `helloworld.py` and run to make sure you get the same answer.

Multiple Arguments to print

Notice when we called `print` with multiple arguments, Python printed **all of the arguments on the same line**, separated by spaces.

Naming Variables

Python lets you name your variables more than one letter. In fact, there's a very specific set of rules for valid variable names:

- 1 You may use any amount of **letters, digits, and underscores** (`_`)
- 2 Your variable names **must not start with a digit**
- 3 You may not use spaces in your variable names
- 4 Python has some **reserved words** you cannot name your variables (`in`, `if`, `for`, `class`, to name a few)
- 5 Variable names are **case-sensitive**, so `Jack` is a separate variable from `jack`

Practice: Naming Variables

Which of the following are valid variable names?

- 1 dogcow
- 2 clarus_the_dogcow
- 3 Clarus the Dogcow
- 4 8ball
- 5 _8ball
- 6 if

Practice: Naming Variables

Which of the following are valid variable names?

- 1 dogcow
- 2 clarus_the_dogcow
- 3 Clarus the Dogcow
- 4 8ball
- 5 _8ball
- 6 if

Answer: 1, 2, and 5 are valid. 3 is not valid since it contains spaces, 4 is not valid since it starts with a digit, and 6 is not valid since it is a reserved word for Python.

Accepting User Input

The `input` function takes a prompt string, and returns the string the user types.

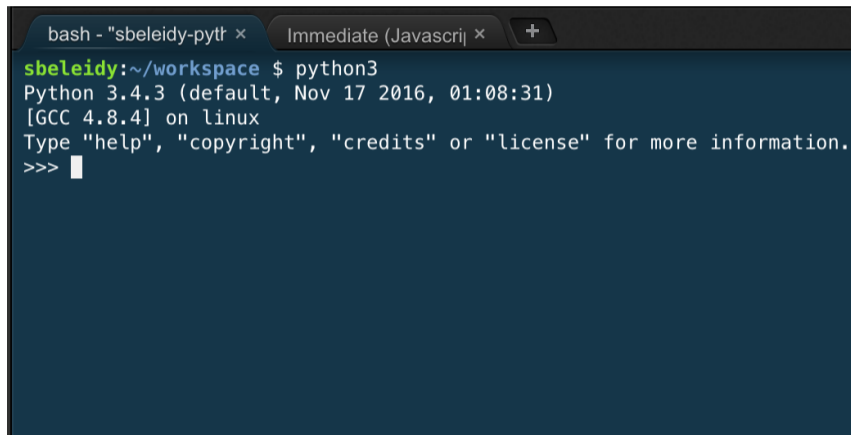
```
name = input("What is your name? ")  
print("Nice to meet you", name)
```

Python provides a simple notation to write mathematical statements.

+	Addition
-	Subtraction
*	Multiplicaton
**	Exponentation
/	Division
//	Integer Division
%	Modulus (division remainder)

Cloud 9 REPL

Open the **interactive interpreter** in Cloud9 IDE by typing `python3` as shown below:



```
bash - "sbeleidy-pytr" × Immediate (Javascrj) × +
sbeleidy:~/workspace $ python3
Python 3.4.3 (default, Nov 17 2016, 01:08:31)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Operators Example in Python REPL

Try these examples and play with a few of your own.

```
>>> (4 + 3) * 6
```

```
42
```

```
>>> 4 ** 3
```

```
64
```

```
>>> 33 / 2
```

```
16.5
```

```
>>> 33 // 2
```

```
16
```

```
>>> 74 % 8
```

```
2
```

Notice the Notation

The `>>>` symbol is used to indicate lines typed at REPL. No need to type `>>>` yourself.

Comments

Comments are a way for programmers to leave notes for others (and sometimes even themselves) in their code. In Python, you can write comments using the # symbol. Anything from # to the end of line will be ignored in Python.

```
# This defines x to the value 10  
x = 10  
x = x + 1   # add one to x
```

Comments

Comments are a way for programmers to leave notes for others (and sometimes even themselves) in their code. In Python, you can write comments using the # symbol. Anything from # to the end of line will be ignored in Python.

```
# This defines x to the value 10  
x = 10  
x = x + 1    # add one to x
```

When should I leave comments in my code?

Python ignores comments, so you never *have* to leave comments in your source. However, if you feel that a piece of code needs explanation for other Python programmers to understand it, usually it's a good idea to leave a comment.

Data Types

Notice we've been using two data types: **integers** and **strings**. Python provides a way to change between the two types:

- 1 Call `int` on a string to convert to an integer

```
an_integer = int("42")
```

Data Types

Notice we've been using two data types: **integers** and **strings**. Python provides a way to change between the two types:

- 1 Call `int` on a string to convert to an integer

```
an_integer = int("42")
```

- 2 Call `str` on an integer to convert to a string

```
a_string = str(42)
```

Data Types

Notice we've been using two data types: **integers** and **strings**. Python provides a way to change between the two types:

- 1 Call `int` on a string to convert to an integer

```
an_integer = int("42")
```

- 2 Call `str` on an integer to convert to a string

```
a_string = str(42)
```

Can you add an integer to a string? Try in the Python REPL and see if you get an error. For example:

```
"26" + 26
```

The len Function

The `len` function takes a sequence (such as a string) and returns its length. For example:

```
>>> len("Jack")
4
>>> len("Hello World!")
12
```