Hello, World!

An Easy Intro to Python & Programming

Jack Rosenthal (jrosenth@mines.edu)

Don't just buy a new video game, make one.
Don't just download the latest app, help design it.
Don't just play on your phone, program it.
No one is born a computer scientist, but with a little hard work and some math and science, just about anyone can become one.

— Barack Obama

Your First Python Program

Create a file `hello.py` and type the following contents:

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

Your First Python Program

Create a file `hello.py` and type the following contents:

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

- `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 file `hello.py` and type the following contents:

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

- `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"`.

Your First Python Program

Create a file `hello.py` and type the following contents:

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

- `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"`.
- A string can be written with either single or double quotes, so this program does the same:

```
print('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 programming! In most programming languages, including 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 programming! In most programming languages, including 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
```

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)
```

Operators

Python provides a simple notation to write mathematical statements.

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

Operators Example in Python REPL

```
>>> (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 the interactive interpreter (right side of repl.it). 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.

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
```

Another Kind of Equals

`==` is for **equality testing**. An expression `a == b` will be `True` iff `a` has the same value as `b`, `False` otherwise.

Another Kind of Equals

`==` is for **equality testing**. An expression `a == b` will be `True` iff `a` has the same value as `b`, `False` otherwise.

```
name = input('What is your name? ')
print('It is', name == 'Jack',
      'that your name is the best.')
```

Branching

```
if condition:
    # do something
elif other_condition:
    # do something else
else:
    # do something else
```

Branching

```
if condition:
    # do something
elif other_condition:
    # do something else
else:
    # do something else
```

```
name = input('What is your name? ')
if name == 'Jack':
    print('Your name is the best!')
elif len(name) == 4:
    print('Your name is 4 letters!')
else:
    print('Pleased to meet you', name)
```

More Comparison Operators

`<`	Less than	`<=`	Less or equal	`!=`	Not equal
`>`	Greater than	`>=`	Greater or equal	`is`	Same instance

More Comparison Operators

`<`	Less than	`<=`	Less or equal	`!=`	Not equal
`>`	Greater than	`>=`	Greater or equal	`is`	Same instance

```
age = int(input('What is your age? '))
if age < 0:
    print('I don\'t think so')
elif age <= 10:
    print('Wow! You\'re young!')
elif age != 16:
    print('Cool cool.')
else:
    print('Sweet sixteen.')
```

Indentation Denotes Scope

In Python, indentation not only provides style to help yourself and others read your code, but also provides functionality by **denoting the scope of the operation**. Consider the following example:

```
# i was defined previously in this program
if i > 0:
    print("i is positive")
    if i % 2 == 0:
        print("i is even")
    print("hello")
print("goodbye")
```

Indentation Denotes Scope

In Python, indentation not only provides style to help yourself and others read your code, but also provides functionality by **denoting the scope of the operation**. Consider the following example:

```
# i was defined previously in this program
if i > 0:
    print("i is positive")
    if i % 2 == 0:
        print("i is even")
    print("hello")
print("goodbye")
```

1. What will be printed if `i` is `3`
2. What will be printed if `i` is `-2`
3. What will be printed if `i` is `4`