

Boolean Expressions

Boolean expressions are expressions that evaluate to one of two Boolean values: True or False

```
<class 'bool'>
```

True

False

Some languages use 0 or 1

Others have a data type

Python has literals True and False

Boolean Expressions

```
>>> 2 < 3
```

True

```
>>> 3 < 2
```

False

```
>>> 5 - 1 > 2 + 1
```

True

```
>>> 3 == 3
```

True

```
>>> 3 + 5 == 4 + 4
```

True

```
>>> 3 == 5 - 3
```

False

>>> 3 <= 4

True

>>> 3 >= 4

False

>>> 3! = 4

True

<	<	less than
<=	≤	less than or equal to
>	>	greater than
>=	≥	greater than or equal to
==	=	equal to
!=	≠	not equal to

Boolean expressions can be combined together using Boolean operators and, or, and not to form larger Boolean expressions.

>>> 2 < 3 and 4 > 5

False

>>> 2 < 3 and True

True

a and b or c

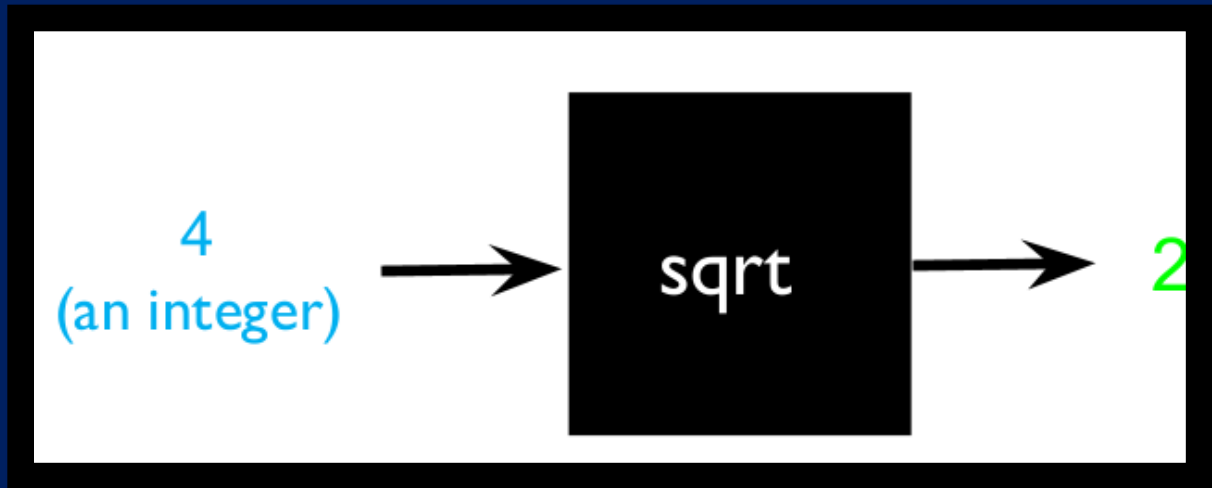
(a and b) or c not a and (b or c)

Functions

Function is a named sequence of statements that performs a computation.

Functions are “self contained” modules of code that accomplish specific task.

“take in” data ☐ process it ☐ return result



Function components

Name of the function

The sequence of statements that

Perform a computation

Abstraction

Function types

There are two kinds of functions in Python.

Built-in functions that are provided as part of

Python `type()`, `float()`, `int()` ...

User-defined functions that we define ourselves

and then use

Built-in Functions

```
abs(-2)      2
```

```
max(7,8,12)  12
```

“Calling” functions

```
>>> type(32)
```

Argument

Arguments

An argument is a value we pass into the function as its input when we call the function

We use arguments so we can direct the function to do

Different kinds of work when we call it at different times

We put the arguments in parenthesis after the name of the function

Modules Import

Includes predefined functions in workspace

```
>>> import math
```

```
>>> math.sqrt(20)
```

Math module

```
>>> math.exp(math.log(10))
```

```
>>> math.cos(60)
```

Definitions and Uses

Once we have defined a function, we can call (or invoke) it as many times as we like

This is the store and reuse pattern

Function Definition (Python)

def function name (parameters):

Statements

Say hello twice

```
def hello_twice():
```

```
    print ('hello')
```

```
    print ('hello')
```

```
>>> hello_twice()
```

```
def print_chorus():
```

```
    print("Girls hit your hallelujah ...Woo!.")
```

```
    print("Girls hit your hallelujah ...Woo!.")
```

```
print("Cause uptown funk gon' give it to you")
```

```
>>> print('Break it down')
```

```
>>> print_chorus()
```

```
>>> print_chorus()
```

Square function

```
def square(num):
```

```
    print(num*num)
```

```
>>> square()
```

```
>>> square(3)
```