



Python:
Variables and Operators

Today's agenda

- Variables & Operators
- Naming Variables
- Displaying Output
- Lab

Variables

- Strings

```
>>> x = 'Hello World'
```

- Numerics

```
>>> x = 3.1415
```

- Booleans

```
>>> x = True
```

- Lists

```
>>> x = ['Hello', True, 3.1415]
```

- And many more...

Variables

- Python is a “dynamically typed” language
 - A variable’s data type is not declared.
 - “Statically typed” languages like Java must declare a variable’s data type

```
String x = “Hello World”;
```

- Get a variable’s data type with the `type` function

```
>>> x = ‘Hello World’
```

```
>>> type(x)
```

```
<type 'str'>
```

Strings

- A string is a piece of text.
- Encase with quotes
 - Single-quotes
 - >>> x = 'abc'
 - Double-quotes
 - >>> x = "abc"
 - Triple single-quotes or triple double-quotes
 - >>> x = '''abc'''
 - >>> x = """abc"""

Strings

- Use double-quotes to encase text containing single-quotes

```
>>> "It's a string with a single-  
quote!"
```

- What is wrong with this statement?

```
>>> x = abc
```

Common String operations

```
>>> x = 'Hello'  
>>> y = 'My name is Mike'
```

```
# Concatenate two strings
```

```
>>> x + '.'  
'Hello.'  
>>> x + ' ' + y  
'Hello. My name is Mike'
```

```
# Equality
```

```
>>> x == 'Hello'  
True  
>>> x == y  
False
```

Common String operations

```
>>> x = 'Hello'  
>>> y = 'My name is Mike'
```

```
# length of a string
```

```
>>> len(x)
```

```
5
```

```
# Convert to lowercase
```

```
>>> x.lower()
```

```
'hello world'
```

```
# Convert to uppercase
```

```
>>> x.upper()
```

```
'HELLO WORLD'
```


Numerics

- Integers

```
>>> x = 10
```

```
>>> type(x)
```

```
<type 'int'>
```

```
>>> y = 10000000000
```

```
>>> type(y)
```

```
<type 'long'>
```

- Decimals

```
>>> x = 3.1415
```

```
>>> type(x)
```

```
<type 'float'>
```

Numerics

- Complex numbers

– $1j$ represents $\sqrt{-1}$

```
>>> x = 5 + 1j          # 5 +  $\sqrt{-1}$ 
```

```
>>> type(x)
```

```
<type 'complex'>
```

Basic Arithmetic Operations

```
>>> x = 5
```

```
>>> y = 8
```

- Addition

```
>>> x + y
```

```
13
```

- Subtraction

```
>>> x - y
```

```
-3
```

- Multiplication

```
>>> x * y
```

```
40
```

Basic Arithmetic Operations

```
>>> x = 5
```

```
>>> y = 8
```

- Modulo division

```
>>> y % x
```

```
3
```

```
>>> -8 % 5
```

```
2
```

Division

- Float division

```
>>> x = 10.0
```

```
>>> y = 8.0
```

```
>>> x / y
```

```
1.25
```

- Integer division. The result is rounded down to the nearest integer.

```
>>> x = 10
```

```
>>> y = 8
```

```
>>> x / y
```

```
1 # 1.25 rounded down
```

```
>>> x = -10
```

```
>>> x / y
```

```
-2 # -1.25 rounded down
```

Division

- If one variable is a float, then do float division.
- This is known as “type coercion”, i.e. coercion of integers to float.

```
>>> x = 10
```

```
>>> y = 8.0
```

```
>>> x / y
```

```
1.25
```

Order of numeric operations

- Same as standard arithmetic writing
 1. Parenthesis
 2. ** (Exponent)
 3. *, / (Multiplication, division)
 4. +, - (Addition, subtraction)
 5. - (Negative)
- If operations have equal precedence, then evaluate from left to right.
- Evaluate
 - >>> $3 + 6 / 3 * (1 + 1)$
7

Booleans

- Variables with two values
 - True
 - False

```
# It's a sunny day!
```

```
>>> is_sunny = True
```

```
>>> type(is_sunny)
```

```
<type 'bool'>
```

```
# It's not raining!
```

```
>>> is_raining = False
```

```
>>> type(is_raining)
```

```
<type 'bool'>
```


Boolean logic

the not statement

```
>>> a = True
```

```
>>> b = True
```

```
>>> c = False
```

```
>>> d = False
```

```
# not x := the opposite of x
```

```
>>> not a
```

```
False
```

```
>>> not c
```

```
True
```

Boolean logic

the and statement

```
>>> a = True
>>> b = True
>>> c = False
>>> d = False
```

```
# x and y := Evaluate x. If x is False, return x. If not, return y
```

```
#           := True only when both x and y are True
```

```
>>> a and b
```

```
True
```

```
>>> a and c
```

```
False
```

```
>>> c and d
```

```
False
```

Boolean logic

the or statement

```
>>> a = True
>>> b = True
>>> c = False
>>> d = False
```

```
# x or y := Evaluate x. If x is True, return x. If not, return y
```

```
# := False only when both x and y are False.
```

```
>>> a or b
```

```
True
```

```
>>> a or c
```

```
True
```

```
>>> c or d
```

```
False
```

Boolean logic practice

```
>>> ((a or d) and c)
```

```
False
```

```
>>> (b and c or d) and a
```

```
False
```

Boolean Coercion

- `0` and `''` are considered False in a Boolean context.
- All other numbers and Strings are considered True.

```
# x and y := Evaluate x.  If x is False, return x.  
                If not, return y.
```

```
>>> '' and 2  
''
```

```
>>> 2 and 0  
0
```

```
>>> True and 4  
4
```

Boolean Coercion

```
# not x := the opposite of x
```

```
>>> not 2
```

```
False
```

```
>>> not ''
```

```
True
```

```
# x or y := Evaluate x. If x is True, return x. If not,  
return y
```

```
>>> '' or 2
```

```
2
```

```
>>> 3 or 0
```

```
3
```

```
>>> False or 0
```

```
0
```

Today's agenda

- Variables & Operators
- Naming Variables
- Displaying Output
- Lab

Naming your variables

- Name your variables to indicate what they're storing
 - Not helpful

```
>>> x = 'Ghana'
```
 - Informative

```
>>> country = 'Ghana'
```
- Use lowercase_with_underscores for multi-word functions and variable names
 - Encouraged

```
>>> soccer_team = 'Black Stars'
```


Naming your variables

- First character must be a letter
 - Invalid
 - >>> 1country = 'Ghana'
 - >>> €five = 5
 - Valid
 - >>> one_country = 'Ghana'
- Keep the name short for readability
 - Too long:
 - >>> the_capital_city_of_ghana = 'Accra'
 - Shorter
 - >>> capital_ghana = 'Accra'

Today's agenda

- Variables & Operators
- Naming Variables
- Displaying Output
- Lab

Output

- Just print it out!

```
# print a string
```

```
>>> print 'Goooooal!'
```

```
Goooooal!
```

```
# without a print, the quotes remain
```

```
>>> 'Goooooal!'
```

```
'Goooooal!'
```

```
# print other data types
```

```
>>> print 3.1415
```

```
3.1415
```

Output

- Print newlines with the `\n` character

```
>>> print 'First line\nSecond line'
First line
Second line
```
- Separate multiple phrases with commas

```
>>> players = 11
>>> print 'There are', players, 'players'
There are 11 players on each team
```

Today's agenda

- Variables & Operators
- Naming Variables
- Displaying Output
- Lab