Python

Jarek Szlichta

http://data.science.uoit.ca/

Python Language

- **Python is a straightforward language**
  - very simple syntax
- **It encourages programmers to program without boilerplate (prepared) code**

Python 2 and 3

- **There are two major Python versions, Python 2 and Python 3.**
  - **For example, one difference between Python 2 and 3 is the print statement.**
    - In Python 2, the "print" statement is not a function, it is invoked without parentheses.
    - However, in Python 3, it is a function, and must be invoked with parentheses.

Print

- **To print a string, just write:**

```
print "This line will be printed."
```

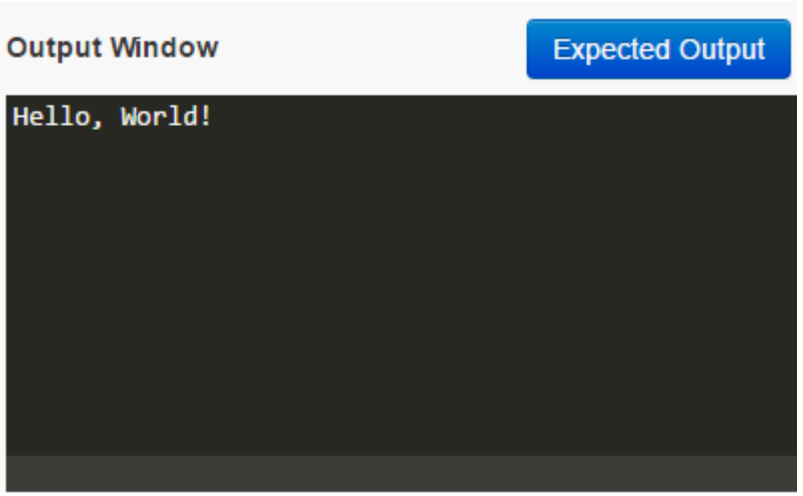
Indentation

- **Python uses indentation for blocks, instead of curly braces.**
  - Both tabs and spaces are supported, but the standard indentation requires standard Python code to use four spaces.

```
x = 1
if x == 1:
    # indented four spaces
    print "x is 1."
```

Exercise – Hello World!

- Use the "print" command to print the line "Hello, World!“
  - http://www.learnpython.org/
    - http://www.learnpython.org/en/Hello%2C_World%21



Solution

Code Window

Solution

```
1 print "Hello, World!"
```

Variables and Types

- **Python is completely object oriented, and not "statically typed“**
  - **For Static typed languages type checking is done at compile-time, whereas for dynamic typed languages type checking is done at run-time**
  - You do not need to declare variables before using them, or declare their type
  - Every variable in Python is an object

Numbers

- Python supports two types of numbers - integers and floating point numbers
  - To define an integer, use the following syntax:

```
myint = 7
```

  - To define a floating point number, you may use one of the following notations:

```
myfloat = 7.0
myfloat = float(7)
```

Strings

- **Strings are defined either with a single quote or a double quote.**

```
mystring = 'hello'
mystring = "hello"
```

- **Using double quotes makes it easy to include apostrophes**
  - whereas these would terminate the string if using single quotes

```
mystring = "Don't worry about apostrophes"
```

Operators

- **Standard operators can be executed on numbers and strings:**

```
one = 1
two = 2
three = one + two

hello = "hello"
world = "world"
helloworld = hello + " " + world
```

Assignments

- **Assignments can be done on more than one variable "simultaneously" on the same line**

```
a, b = 3, 4
```

- **Mixing operators between numbers and strings is not supported:**

```
# This will not work!
print one + two + hello
```

Exercise

- Create a string, an integer, and a floating point number.
  - The string should be named **mystring** and should contain the word ***hello***.
  - The floating point number should be named **myfloat** and should contain the number **10**,
  - And the integer should be named **myint** and should contain the number **20**.

Exercise

- http://www.learnpython.org/en/Variables_and_Types

Code Window

```
# change this code
mystring = None
myfloat = None
myint = None

# testing code
if mystring == "hello":
    print "String: %s" % mystring
if isinstance(myfloat, float) and myfloat == 10.0:
    print "Float: %d" % myfloat
if isinstance(myint, int) and myint == 20:
    print "Integer: %d" % myint
```

Output Window Expected Output

```
String: hello
Float: 10
Integer: 20
```

Powered by Sphere Engine ™

Solution

Code Window

Solution

```
# change this code
mystring = "hello"
myfloat = 10.0
myint = 20

# testing code
if mystring == "hello":
    print "String: %s" % mystring
if isinstance(myfloat, float) and myfloat == 10.0:
    print "Float: %d" % myfloat
if isinstance(myint, int) and myint == 20:
    print "Integer: %d" % myint
```

Lists

- Lists are very similar to arrays
  - They can contain any type of variable,
  - and they can contain as many variables as you wish
- Lists in Python can also be iterated over in a simple manner

Lists

```
# empty list
mylist = []
mylist.append(1)
mylist.append(2)
mylist.append(3)
print(mylist[0]) # prints 1
print(mylist[1]) # prints 2
print(mylist[2]) # prints 3

# prints out 1,2,3
for x in mylist:
    print x
```

Exceptions

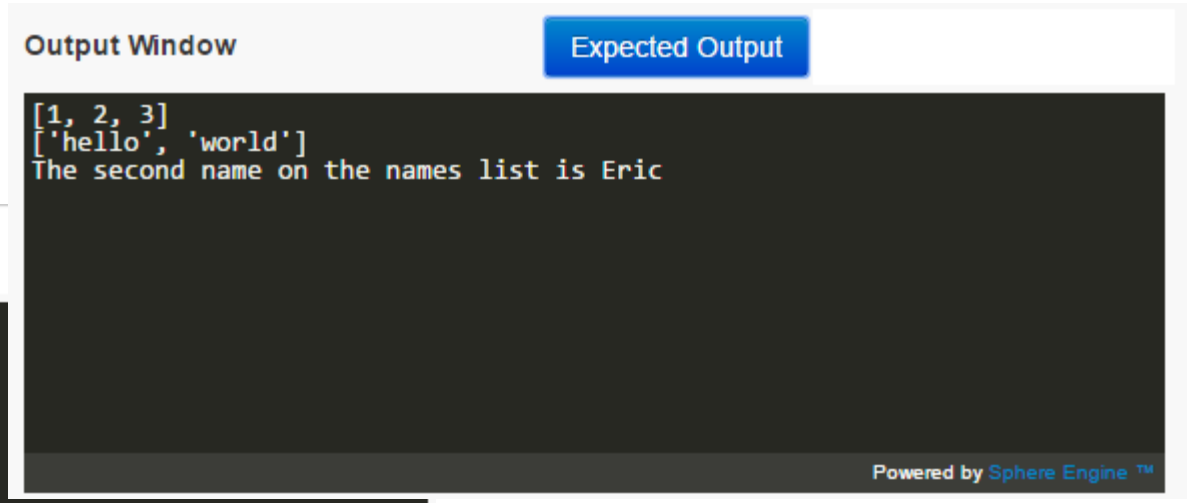
- **Accessing an index which does not exist generates an exception (an error)**

```
mylist = [1,2,3]
print(mylist[10])
```

You will also have to fill in the variable second_name with the second name in the names list, using the brackets operator [].

Exercise

- http://www.learnpython.org/en/Lists
  - Add numbers and strings to the correct lists using the "append" list method. Add the numbers 1,2, and 3 to the **numbers list**, and the words 'hello' and 'world' to the **strings** variable
  - Fill in the variable **second_name** with the second name in the **names** list, using the brackets operator [ ]



Code Window

```
numbers = []
strings = []
names = ["John", "Eric", "Jessica"]

# write your code here
second_name = None


# this code should write out the filled arrays and the second name in
print(numbers)
print(strings)
print("The second name on the names list is %s" % second_name)
```

Solution

Code Window

Solution

```
numbers = []
strings = []
names = ["John", "Eric", "Jessica"]

# write your code here
numbers.append(1)
numbers.append(2)
numbers.append(3)

strings.append("hello")
strings.append("world")

second_name = names[1]

# this code should write out the filled arrays and the second name i
```

Arithmetic Operators

- **The addition, subtraction, multiplication, and division operators can be used with numbers**

```
number = 1 + 2 * 3 / 4.0
```

  - Try to predict what the answer will be?
- **Modulo (%) operator returns the integer remainder of the division**

```
remainder = 11 % 3
```

- **Using two multiplication symbols makes a power relationship**

```
squared = 7 ** 2
cubed = 2 ** 3
```

Using Operators With Strings

- **Python supports concatenating strings using the addition operator**

```
helloworld = "hello" + " " + "world"
```

- **Python also supports multiplying strings to form a string with a repeating sequence:**
  - “hellohellohellohellohellohellohellohellohellohello”:

```
lotsofhellos = "hello" * 10
```

Using Operators with Lists

- **Lists can be joined with the addition operators:**

```
even_numbers = [2,4,6,8]
odd_numbers = [1,3,5,7]
all_numbers = odd_numbers + even_numbers
```

- **Just as in strings, Python supports forming new lists with a repeating sequence using the multiplication operator:**

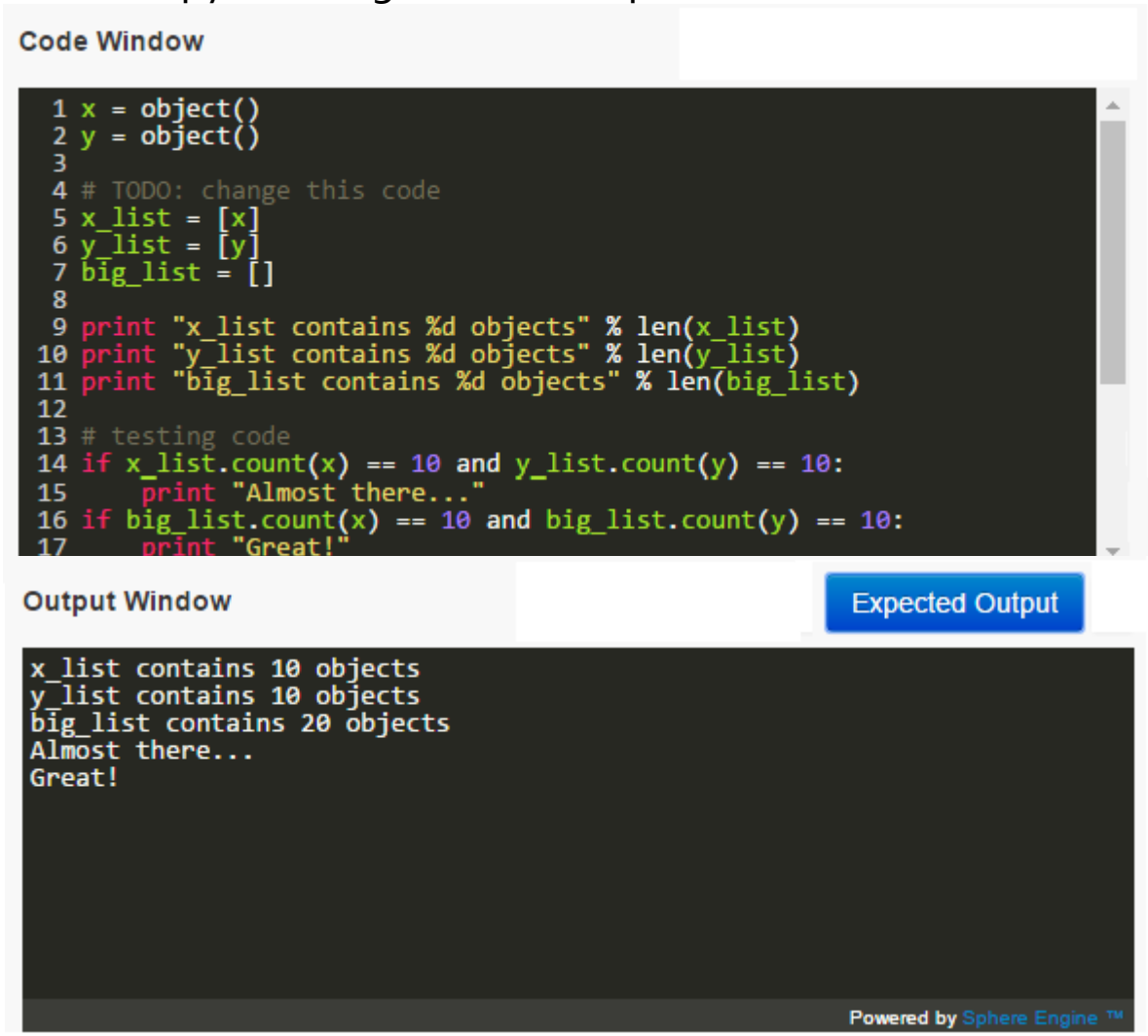
```
print [1,2,3] * 3
```

Exercise

- Create two lists called `x_list` and `y_list`, which contain 10 instances of the variables `x` and `y`, respectively.
  - Use multiplication operator.
- You are also required to create a list called `big_list`, which contains the variables `x` and `y`, 10 times each, by concatenating the two lists you have created.
  - Use addition operator

Exercise

- http://www.learnpython.org/en/Basic_Operators



Solution

Code Window

Solution

```
1 x = object()
2 y = object()
3
4 # TODO: change this code
5 x_list = [x] * 10
6 y_list = [y] * 10
7 big_list = x_list + y_list
8
9 print "x_list contains %d objects" % len(x_list)
10 print "y_list contains %d objects" % len(y_list)
11 print "big_list contains %d objects" % len(big_list)
12
13 # testing code
14 if x_list.count(x) == 10 and y_list.count(y) == 10:
15     print "Almost there..."
16 if big_list.count(x) == 10 and big_list.count(y) == 10:
17     print "Great!"
```

String Formatting

- **Python uses C-style string formatting to create new, formatted strings.**
  - special symbols like "%s" and "%d“ (argument specifiers)
  - Let's say you have a variable called "name" with your user name in it:

```
# This prints out "Hello, John!"
name = "John"
print "Hello, %s!" % name
```

Two or More Argument Specifiers

- **To use two or more argument specifiers, use a tuple (parentheses):**

```
# This prints out "John is 23 years old."
name = "John"
age = 23
print "%s is %d years old." % (name, age)
```

- **Any object which is not a string can be formatted using the %s operator as well.**

```
# This prints out: A list: [1, 2, 3]
mylist = [1,2,3]
print "A list: %s" % mylist
```

Argument Specifiers

- Here are some basic argument specifiers you should know:
  - %s - String (or any object with a string representation, like numbers)
  - %d - Integers
  - %f - Floating point numbers
  - %.<number of digits>f - Floating point numbers with a fixed amount of digits to the right of the dot.
  - %x / %X - Integers in hex representation (lowercase/uppercase)

Exercise

- Format string which prints out the data using the following syntax:
  - Hello John Doe. Your current balance is 53.44$.

Exercise

- http://www.learnpython.org/en/String_Formatting
  - Change 2nd line of the code

Code Window

```
1 data = ("John", "Doe", 53.44)
2 format_string = "Hello"
3
4 print format_string % data
```

Output Window Expected Output

```
Hello John Doe. Your current balance is 53.44$.
```

Powered by Sphere Engine ™

Solution

Code Window

Solution

```
1 data = ("John", "Doe", 53.44)
2 format_string = "Hello %s %s. Your current balance is %s$."
3
4 print format_string % data
```

Basic String Operations

- **Strings are bits of text. They can be defined as anything between quotes:**

```
astring = "Hello world!"
astring2 = 'Hello world!'
```

- You can also use single quotes to assign a string.
  - However, you will face problems if the value to be assigned itself contains single quotes
  - you need to use double quotes only like this

```
print "single quotes are ' '"
```

Len, Index and Count

- That prints out 12, because "Hello world!" is 12 characters long, including punctuation and spaces

```
print len(astring)
```

- That prints out 4, because the location of the first occurrence of the letter "o" is 4 characters away from the first character.

```
print astring.index("o")
```

- This counts the number of l's in the string. Therefore, it should print 3.

```
print astring.count("l")
```

Slicing

- **This prints a slice of the string, starting at index 3, and ending at index 6**

```
print astring[3:7]
```

- **Reverse a string**

```
print astring[::-1]
```

This is used to determine whether the string starts with something or ends with something, respectively.

```
print astring.startswith("Hello")
print astring.endswith("rld")
```

Exercise

- Try to fix the code to print out the correct information by changing the string.
  - Note: These make a new string with all letters converted to uppercase and lowercase:
    - Print astring.upper()
    - Print astring.lower()

Exercise

- Try to fix the code to print out the correct information by changing the string.
  - http://www.learnpython.org/en/Basic_String_Operations

Code Window

```
s = "Hey there! what should this string be?"
# Length should be 20
print "Length of s = %d" % len(s)

# First occurrence of "a" should be at index 8
print "The first occurrence of the letter a = %d" % s.index("a")

# Number of a's should be 2
print "a occurs %d times" % s.count("a")

# Slicing the string into bits
print "The first five characters are '%s'" % s[:5] # Start to 5
print "The next five characters are '%s'" % s[5:10] # 5 to 10
print "The twelfth character is '%s'" % s[12] # Just number 12
print "The characters with odd index are '%s' " %s[1::2] #(0-based i
print "The last five characters are '%s'" % s[-5:] # 5th-from-last t

# Convert everything to uppercase
print "String in uppercase: %s" % s.upper()

# Convert everything to lowercase
print "String in lowercase: %s" % s.lower()

# Check how a string starts
if s.startswith("Str"):
    print "String starts with 'Str'. Good!"

# Check how a string ends
if s.endswith("ome!"):
    print "String ends with 'ome!'. Good!"

# Split the string into three separate strings,
# each containing only a word
print "Split the words of the string: %s" % s.split(" ")
```

Solution

Code Window

Solution

```
1 s = "Strings are awesome!"
2 # Length should be 20
3 print "Length of s = %d" % len(s)
4
5 # First occurrence of "a" should be at index 8
6 print "The first occurrence of the letter a = %d" % s.index("a")
7
8 # Number of a's should be 2
9 print "a occurs %d times" % s.count("a")
10
11 # Slicing the string into bits
12 print "The first five characters are '%s'" % s[:5] # Start to 5
13 print "The next five characters are '%s'" % s[5:10] # 5 to 10
14
```

Output Window

Expected Output

```
Length of s = 20
The first occurrence of the letter a = 8
a occurs 2 times
The first five characters are 'Strin'
The next five characters are 'gs ar'
The thirteenth character is 'a'
The characters with odd index are
The characters with odd index are tig r wsm!
The last five characters are 'some!'
String in uppercase: STRINGS ARE AWESOME!
String in lowercase: strings are awesome!
String starts with 'Str'. Good!
String ends with 'ome!'. Good!
Split the words of the string: ['Strings', 'are', 'awesome!']
```

Powered by Sphere Engine

Conditions

- **Python uses boolean variables to evaluate conditions**
- The boolean values True and False are returned when an expression is evaluated
  - variable assignment is done using a single equals operator "="
  - comparison between two variables is done using the double equals operator "=="
  - The "not equals" operator is marked as "!="

```
x = 2
print x == 2 # prints out True
print x == 3 # prints out False
print x < 3  # prints out True
```

Reading List

- **Review Slides!**
- **Recommended**
  - Python Tutorial
    - http://www.learnpython.org/
- **Optional**
  - History and General Information
    - https://en.wikipedia.org/wiki/History_of_Python
    - https://www.python.org/doc/essays/comparisons
    - http://www.programmerinterview.com/index.php/general-miscellaneous/whats-the-difference-between-a-compiled-and-an-interpreted-language/
  - Documentation
    - https://docs.python.org/2/tutorial/