A Description of Python Programming Language Essentials

Python is a general-purpose, high-level programming language, used in everything from web programming to scientific research labs. This introduction assumes that you have some experience in another high-level programming language, such as C++ or Java. We will examine essential Python syntax and explore some of the built-in functions in the Python standard library.

Comments

Single line comments in Python begin with the `#` symbol. Mutliline comments use three double quotes as delimiters.

Variables

Variables allow you to store and reuse values. String literals are represented with either single quotes or double quotes.

```
#  A single line comment

""" A multiline comment can be
    written by using three quotes
    in sequence, and then by ending
    with the same.
"""
```

```
'This is a single quoted string'
"This is a doubly quoted string"


1987654  # Represents the number 1,987,654
3.14159  # Represents the number 3.14159
```

Output

Python uses the `print` statement in order to display to standard output. You can combine strings and literals with the `.format()` function. Notice the curly braces inside the string literal.

```
x = 10
y = 36
z = "Python is cool!"

x = 42
print x   # Outputs `42`

print "Hello World"

print "The answer is {}".format(42)
```

Conditionals

The `if` statement allows you to execute only portions of the code.

```
x = 42
if x <= 42:
    #  Indentation matters!
    print "x is less than or equal to 42."
```

You can also test for multiple conditions using `or` and `and` operators.

```
x = 10
y = 20

if x < y and y == 20:
    print "x is less than y and y is equal to 20."
```

For greater granularity of control, you can also use `elif` and `else`.

```
x = 10

if x < 10:
    print "x is less than 10."
elif x < 20:
    print "x is less than 20."
else:
    print "x is greater than or equal to 20."
```

You can nest multiple `if` statements with increased indentation.

```
x = 10

if x < 15:
    if x < 10:
        print "x is less than 10."
    else:
        print "x is less than 15, but not less than 10."
```

Lists

Python allows you to store multiple values in a list.

```
my_list = [3, 1, 4, 5]
```

To access elements in the list, use the indexing `[]` operator.

```
print my_list[0]  # Outputs `3`
print my_list[1]  # Outputs `1`
print my_list[2]  # Outputs `4`
print my_list[3]  # Outputs `5`
```

Indexing the `-1` elements yields the last element of the list.

```
print my_list[-1]  # Outputs `5`
```

Loops

The `while` and `for` keywords allow you to execute a block of code multiple times. The `while` keyword executes a block of code until a condition evaluates to `False`.

```
x = 0
while x < 5:
    print "Happy day!"
```

A `for` loop allows you to loop through a list.

```
my_list = [1, 2, 3, 4, 5]
for num in my_list:
    print num
```

Use the `xrange()` function to loop a specified number of times

```
# Outputs the numbers 10 through 14
for i in xrange(10, 15):
    print i
```

Sets

Sets allow fast access to elements, but only retain unique elements. A set makes no guarantee of ordering.

```
S = set()
S.add(5)
S.add(3)
S.add(1)

# Prints out the numbers 5, 3, 1 in no particular order
for element in S:
    print "{} is in the set".format(element)

S.remove(3)
S.remove(5)
S.remove(1)
```

Dictionaries

Dictionaries offer fast key-value lookup. To declare a dictionary, use curly `{}` braces.

```
my_dict = {}
my_dict['A'] = 1
my_dict['B'] = 2
my_dict['C'] = 3

# Or alternatively:

my_dict = {
    'A': 1, 'B': 2, 'C': 3
}
```

Iterate through a dictionary with the `.iteritems()` method.

```
for key, value in my_dict.iteritems():
    print "{} maps to {}".format(key, value)
```

Conclusion

We've only covered the basics of Python in this tutorial. To learn more about the Python programming language, visit www.python.org.

View this tutorial online at
https://github.com/krrg/python-tutorial
The Python Logo is a trademark of the Python Software Foundation and is used within their guidelines.