Visualize ComplexCities

Introduction to Python

Chair of Information Architecture
ETH Zürich

October 18, 2013

First Steps

Python Basics

Conditionals Statements

Loops

User Input

Functions

Programming?

Programming is the interaction between the programmer and the computer. The Computer evaluates the written code and executes the instructions.

Python

- Interactive programming language.
- Script based programming possible.
- Object-oriented programming.
- Can be used to control Blender.

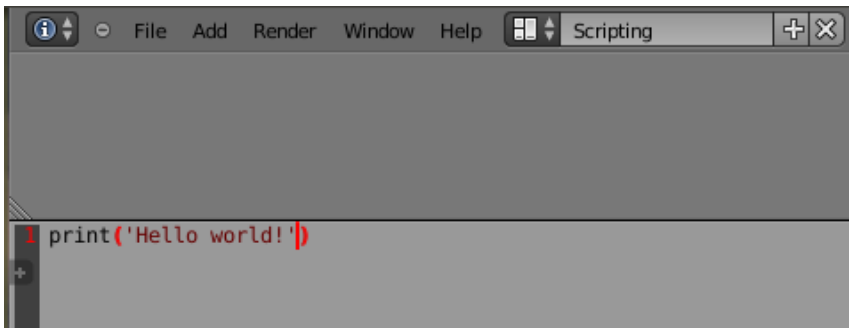
Interactive Mode Programming: Hello world!

1. Go to the Scripting mode.

2. In the Console type: print('Hello world!')
3. Hit Enter.
4. The Console should print: *Hello world!*

Script Based Programming: Hello world!

1. First go to Window and click *Toggle System Console*.
2. In the editor type: print('Hello world!')



3. Press the "Run Script"-Button.
4. The output should be *Hello world!* in the windows command line.

Variables

Variables are locations to store values. The thing on the left of ”=” is the variable and the thing on the right side of ”=” the value stored in the variable. For example:

```
bar = 23
foo = "Hello world!"
ali = True
abcd = [1,2,3,4,5]
```

Data Types

We saw three data types on the previous slide: an integer, a string and a list. There exist many different data types. The standard data types are:

- Boolean
- Number
- String
- List
- Tuple
- Dictionary

Boolean

- A boolean can only have one of two values, it is either true or false.

```
iAmTrue = True
iAmFalse = False
```

- It is mostly used to store the results of comparisons.

```
a = 1
b = 12
foo = a == b
bar = a < b
ali = a > b
```

Number

There are two main ways of storing numbers.

- Integer number: Whole number.

```
foo = 42
```

- Floating point number: Real number.

```
bar = 2.7182818
```

String

- A string is a sequence of characters.

```
foo = "Hello world!"
```

- The character sequence has to be between quotation marks, so that python knows what the string is.

List

- A list is a sequence of values of any data type.

```
foo = [42, [1,2,3], "Hello world!", 23, 73]
```

- The elements in a list can be accessed with an index.

```
# Store the value of the second element to bar.
bar = foo[1]
```

- **Attention:** The first element has index 0, the second 1, and so on!

List continued

- We can change the elements in a list.

```
# Set the third element to 42.
foo[2] = 42
```

- We can also add new elements to the list.

```
# Append a string.
foo.append("The quick brown fox")
```

- To remove an element we remove the value of it.

```
# Remove the first appearance of 23
foo.remove(23)
```

Tuple

Similar to list but of fixed length. A tuple can not be changed after initialization.

```
foo = (42, [1,2,3], "Hello world!", 23, 73)
```

The only difference in initialization to a list is: tuples are enclosed within parentheses.

Dictionary

- A dictionary is a so called hash table. This means, that every element has a key, i.e. all elements are key-value pairs.

```
foo = {"answer":42, "who": "brown fox"}
```

- We can access the element by using their key.

```
# Set bar to the "answer"
bar = foo["answer"]
```

Conditional Statements

A very important construct in programming are the *if* statements.

```
if condition:
    statement1
    statement2
    ...
```

Python executes only the parts within an if statement, if it is valid. If the condition is not met, it proceeds with the part after the *if* block.

Interlude

The elements of a block have to be intended. In the *if* clause below, the statements one and two belong to the block, N does not. What does this mean?

```
if condition:
    statement1
    statement2

statementN
```

Conditional Statements ctd.

It is possible, to have alternative statements when the conditions do not hold.

```
if condition:
    statement1
else
    statement2
```

```
if condition1:
    statement1
elif condition2:
    statement2
else
    statement3
```

Loops

There are two major kinds of loops. The *while* loop and the *for* loop.

```
while condition:
    statement1
    statement2
    ...
```

```
for iterator in sequence:
    statement1
    statement2
    ...
```

What is the difference between them?

while Loop

This loops as long as the condition holds. What does the following do?

```
a = 1
while a < 128:
    a = a + a
    print(a)
```

while Loop

This loops as long as the condition holds. What does the following do?

```
a = 1
while a < 128:
    a = a + a
    print(a)
```

```
2
4
8
16
32
64
128
```

for Loop

This loops through all elements in the sequence. In every loop iteration it sets the iterator to one of the values in the sequence, beginning with the first element. What does the following do?

```
for a in [1,2,3,4]:
    print(a*a)
```

for Loop

This loops through all elements in the sequence. In every loop iteration it sets the iterator to one of the values in the sequence, beginning with the first element. What does the following do?

```
for a in [1,2,3,4]:
    print(a*a)
```

```
1
4
9
16
```

Helpful Keywords for Loops

- *break*: Terminates the current loop and resumes after the loop.
- *continue*: Terminates the current iteration and returns to the beginning of the loop.

Helpful Keywords for Loops

What does the following do?

```
for a in range(0, 10):
    if a < 3:
        print (a * 100)
    elif a == 5:
        continue
    elif a > 7:
        break
    else:
        print(a)
```

Helpful Keywords for Loops

What does the following do?

```
for a in range(0, 10):
    if a < 3:
        print (a * 100)
    elif a == 5:
        continue
    elif a > 7:
        break
    else:
        print(a)
```

```
0
100
200
3
4
6
7
```

User Input

The easiest way to have an interactive program is to use command line input. The program asks you to type something into the console. This can be done the following way.

```
# Ask the user to enter his age.
age = raw_input("Enter your age: ")
```

User Age

A small program which tells you if you are old or not.

```
age = int(raw_input("Enter your age: "))
if age < 20:
   print("You are young.")
elif age < 50:
   print("You are in a fine age.")
else:
   print("You are old.")
```

Functions

When you have a code, which has to be executed in different spots of your program, it is more applicable to create a function than rewrite the code all the time. Earlier we called the ”range()” function. You can define such functions also yourselves.

Functions

Function definitions start with the keyword *def*. This is followed by the function name and parentheses "()". When parameters are needed for the function, they are within the parentheses.

```
def functionName( parameters ):
    statements
    return variable
```

Example

Following a function, which squares the input.

```
def square(a):
    ret = a * a
    return ret
```

You can call the function from within the script by invoking the name of the function along with the right number and order of parameters. The call must happen after the function definition.

```
# Call the function square with the parameter 3.
square(3)
```

Helpful Links

- `http://www.tutorialspoint.com/python`
- `http://docs.python.org/3/`