Introduction to Scientific Computing

with Python

Utku Kaya

Department of Mathematics
University of Kiel, Germany

25 02 2020

Today’s Overview

- functions
- conditional statements
- loops
- classes/objects
- reading & writing files

Functions

The keyword def is used to define a **function**.

```
>>> def my_func():
...     print("This is a function")
>>> my_func()
```

Functions

```
>>> def multiply(x, y):
...     return x * y
```

The function `multiply` expects two arguments: `x` and `y`.

It returns `x * y`.

```
>>> multiply(3, 4)
```

If statements

```
>>> x = 20
>>> y = 19
>>> if x > y:
...     print(x, 'is greater than', y)
```

If statements

```
>>> def compare(x, y):
...     if x > y:
...         print(x, 'is_greater_than', y)
...     elif x < y:
...         print(x, 'is_less_than', y)
...     else:
...         print(x, 'is_equal_to', y)
```

If statements

```
>>> if (x % 2 == 0) and (x % 3 == 0):
...   print(x,'can be divided by 6')
... else:
...   print(x,'cannot be divided by 6')
```

Strings

```
>>> animals=['dog', 'cat', 'sheep', 'alpaca']
>>> sorted(animals)
>>> len(animals)
>>> len(animals[1])
>>> animals[1]
```

While loops

```
>>> i = 0
>>> while (i < 10):
...     print(i)
...     i += 1
```

While loops

```
>>> i = 0
>>> while True:
...     if i > 20:
...         break
...     else:
...         i *= 1
...         print(i)
```

For loops

```
>>> for count in [1, 2, 3]:
...   print(count)
```

Note that [*expression*1, *expression*2, . . . , *expressionN*] is a **`list`** in Python.

```
>>> arr = [1, 2, 3]
>>> for count in arr:
...   print(count)
>>> for i in range(10):
...   print(i)
>>> for color in ['red', 'blue', 'green']:
...   print(color)
```

For loops

```
>>> num = 1
>>> for color in ['red', 'blue', 'green']:
...     print(color, num)
...     num += 1
```

Booleans

Boolean in Python can have two values: `True` or `False`.

```
>>> condition = False
>>> if condition == True:
...     print("The condition is True")
... else:
...     print("The condition is False")
```

Note following equivalence:

```
>>> if condition == True:
>>> if condition:
```

Lists

```
>>> assorted_list = [True, False, 1, 1.1, 1+2j, \
... 'Learn', b'Python']
>>> first_element = assorted_list[0]
>>> print(first_element)
>>> print(assorted_list)
>>> for item in assorted_list:
...   print(type(item))
>>> nested = [[1,1,1], [2,2,2], [3,3,3]]
>>> for items in nested:
...   for item in items:
...     print(item, end='␣')
```

Fibonacci sequence

```
>>> a, b = 0, 1
>>> while a < 1000:
...     print(a, end = '␣')
...     a, b = b, a + b
```

Guess which number!

Save following code in game.py and run

```
>>> import random
>>> number = random.randint(1, 10)
>>> niter = 0
>>> print('Guess␣which␣number␣I␣chose␣between␣1␣to␣10!')
>>> while True:
...   guess = int(input('Your␣guess:'))
...   niter += 1
...   if guess > number:
...     print('Too␣high')
...   elif guess < number:
...     print('Too␣low')
...   elif guess == number:
...     print('Needed␣{}␣iterations'.format(niter))
...     break
```

Python Classes

Python is an object oriented programming language.
Almost everything in Python is an object, with its properties and methods.

```
>>> s = 'Hello!'
>>> s.upper()
>>> s
>>> s2 = s.upper()
>>> s2
>>> s
>>> text = 'AAaaAA'
>>> text.count('AA')
>>> text.lower.count('a')
>>> text.count('aA')
```

Classes/objects

Each object in Python is created from a class. Classes can be seen as object constructors, or "blueprints".

Create a class named `MyClass`, with a property named `n`:

```
>>> class MyClass:
...   n = 5
```

Create an object named `LaMolina`, and print the value of `n`:

```
>>> LaMolina = MyClass()
>>> print(LaMolina.n)
```

Classes/objects

```
>>> LaMolina.learningPython=True
>>> type(LaMolina.learningPython)
>>> LaMolina.learningPython
```

Classes/objects

When the **class** Person is being initiated, firstly the `__init__()` function is executed.

```
>>> class Person:
...   def __init__(self, n, a):
...     self.name = n
...     self.age = a
>>> p1 = Person("Dandy", "Very young")
>>> print(p1.name)
>>> print(p1.age)
```

`p1` is an **object**. The `__init__()` function is used to assign values to object properties.

Classes/objects

```
>>> class Person:
...   def __init__(self, name, age):
...     self.name = name
...     self.age = int(age)
>>> p1 = Person("Dandy", "Very young")  #Returns error!
>>> p1 = Person("Dandy", "35")
>>> print(p1.name, p1.age)
```

The first parameter of `__init__()` function is a reference to the current instance of the class.

It does not have to be named `self`, but has to be the first parameter of any function in the class.

Classes/objects

In a class, one can declare functions.

```
>>> class Person:
...   def __init__(self, name, age,field ):
...     self.name = name
...     self.age = int(age)
...     self.field = field
...   def introduce(self):
...     print("Hello␣my␣name␣is␣" + self.name+ \
...     "._␣I␣am␣" , self.age ,\
...     "years␣old._␣My␣field␣is␣" + self.field )
>>> p1 = Person("Dandy", "35")
>>> p1 = Person("Dandy", "35", "Math.")
>>> p1.introduce()
```

The function `introduce` is a **method** of the objects created from the class `Person`.

Python Classes

```
>>> class Car:
...   def __init__(self, b, c, i):
...     self.brand = b
...     self.color = c
...     self.iselectric = i
...   def makeElectric(self):
...     self.iselectric = True
...   def makeDiesel(self):
...     self.iselectric = False
```

Classes/objects

Wit the del command you can delete properties and also objects.

```
>>> p1.age(40)
>>> del p1.age
>>> p1.age(40)  #returns error!
>>> del p1
```

Reading files

```
>>> file = open('file_example.txt', 'r')
>>> contents = file.read()
>>> print(contents)
>>> file.close()
```

open -> opens a file and returns an object that knows how to get information from the file

r -> read

w -> write

a -> append

Reading files

```
>>> import os
>>> os.getcwd()
>>> os.chdir('...')
>>> os.getcwd()
```

`...` is the directory where the `file_example.txt` is saved.

```
>>> file = open('file_example.txt', 'r')
>>> contents1 = file.read(10)
>>> contents2 = file.read()
>>> print(contents1)
>>> print(contents2)
>>> file.close()
```

Method call `file.read(10)` moves the file cursor, so the next call, `file.read()`, reads everything from character 11 to the end of the file.

Thanks for your attention!