12.010 Computational Methods of Scientific Programming 2021

Tom Herring, tah@mit.edu

Chris Hill, cnh@mit.edu

Lecture 4: Algorithms and common problems

Topics

- Numpy arrays
- Pandas Data frames: An advancement on simple arrays
- Functions

Python Data types

- Numpy arrays are the most useful of the basic data types for collecting things
  - Try Lec04_numpy.ipynb

Python Dataframes

- To do more beyond primitive types (float, array etc…), we need to introduce more concepts (functions, classes, objects)
- First, look at Dataframes from Pandas () module.
  - Pandas Dataframes use functions, classes, and objects to work and show some nice things that can be encapsulated in a library this way.

- Pandas Data frames – (https://pandas.pydata.org)
  - Provides an intuitive table-like representation of data with column headings and rows
  - Provides a Python-style interface to the table

Python Dataframes try

- Lec04_DataFrames.ipynb

Python Functions, Classes and Object

- With arrays and existing libraries like Dataframes (and also for example, Scipy) can do a lot.

- Can do even more by learning about other core programming concepts
  - Functions – a way to create generic "operations" that can be applied to different instances of similar things.
  - Classes – way to group variables with functions to create more advanced function behaviors
  - Objects – "instances" of classes that are variables with functions and internal variables all grouped together. A Pandas dataframe is an example of an object

- *Functions, classes, and objects, together with variables, data types, and control structures (if, for etc…) are the key elements of programming in Python.*

Python Functions*

- A basic function is a way to encapsulate some set of steps into a special collection of code
- Like in a recipe – a separate step might be “make a cheese sauce,” which might be the same for several recipes.
- Simplest form syntax

  def printhi():
  print(hi)

  def - says this is a function

  : - starts function body

  everything indented is part of the function e.g.

```
def printhi():
    print("hi")
```

```
printhi()
```

```
hi
```

Python Functions

- Most things about functions are best tried with examples
- Some key aspects are
- Functions can take arguments, e.g.

```
def printhiname(name):
    print("hi", name)
```

```
printhiname("Chris")
printhiname("Tom")
printhiname()
```

- the last example will give an error because the argument does not have a default setting.

Python Functions

- Adding a default for an argument is easy!

```
def printhiname(name="NO NAME"):
    print("hi", name)
```

```
printhiname()
printhiname("Chris")
printhiname(name="Chris")
```

Python Functions

- Functions can return a result

```
def calc_sum(arr):
    asum=0.
    for aval in arr:
        asum=asum+aval
    return asum
```

```
calc_sum([0,1,2])
```

Python Functions

- Special syntax allows for variable numbers of arguments

```
# One final piece of function syntax
def print_vars(*args):
    for a in args:
        print(a)
    return

print_vars(1)
print("")
print_vars(1,2)

print("")

def print_vars_with_kw(*args,**kwargs):
    for a in args:
        print(a)
    print(kwargs.keys())
    print(kwargs["x"])

    return

print_vars_with_kw(1,x=7)
```

Python Functions

- Functions can “call” themselves (recursion)

```
# This function calls itself recursvely with n reduced by 1 each time until n is 1.
def factorial(n):
    if n==1 or n==0:
        return 1
    else:
        return n * factorial(n-1)
```

Python Functions

- **Lab exercises**

Lec04_functions.ipynb

Summary

- Input/Output: Simply first, more later
- Programming approach: Algorithm development
  - Example of how to think about the process: Speed of coding versus speed of execution
- Example with polyarea.ipynb
- Verification of code. Expecting the unexpected.
- Common problems that can occur.

MIT OpenCourseWare

https://ocw.mit.edu

12.010 Computational Methods of Scientific Programming, Fall 2024