Notas de aula

Introdução à Física Computacional

Prof. Gerson – UFU – 2019

Atendimento:

- Sala 1A225
- Email: gersonjferreira@ufu.br
- Webpage: http://gjferreira.wordpress.com
- Horário: sextas-feiras 16:00 – 16:50

Basic operations in python

1) Assignment

```
a = 3
b = 4
c = a**2 + b**2
print('c = ', c)
```

2) if statement

```
if a > b:
    print('a is larger than b')
elif a < b:
    print('b is larger than a')
else:
    print('a is equal to b')
```

3) operators

Arithmetic: + , - , * , /
Exponentiation: ** ,
ex: 1024 == 2**10
Assignment: = , += , -= , *= , /=
ex: x *= 3 → x = x*3
Comparison: ==, !=, >, >=, <, <=
Logical: and , or , not
Membership: in , not in
ex: 5 in [3, 2, 9] → False

4) Lists (for vectors → see numpy)

```
fruits = ['banana', 'orange']
fruits.append('apple')
fruits.sort()
print('Length:', len(fruits))
print('first:', fruits[0])
print('last:', fruits[-1])
```

5) For loop over lists

```
for fruit in fruits:
    print('we have ', fruit)
```

6) Membership

```
if 'grape' in fruits:
    print('yes, we have grapes')
else:
    print('no, we don’t have grapes')
```

7) Lists over range of integers

syntax:

range(n) → 0,1,..n-1

range(i, f, s)

- i → initial
- f → final, not included
- s → step

```
for i in range(3, 15, 2):
    print(i)
```

```
for i in range(len(fruits)):
    print(i, fruits[i])
```

Compreehensions

short form for loops to create lists and matrices (suggestion, use with numpy)

[g(i) for i in some_list]

Lists

General notation → mylist = np.array([f(x) for x in xlist])

Example:

```
x2 = np.array([ x**2 for x in range(0, 20, 2)])
```

Matrices

columns lines

General notation → mymatrix = np.array([[g(x,y) for y in ylist] for x in xlist])

Example:

```
xy = np.array([ [ x*y for y in range(0, 10, 1)] for x in range(0, 10, 1)])
```

Exercise

1) Define a function as: $f(i,j) = \begin{cases} +2, & \text{if } i = j \\ -1, & \text{if } |i-j| = 1 \\ 0, & \text{otherwise} \end{cases}$

2) Use the compreehensions and the function f(i,j) to create the matrix:

$$M = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$

We'll see later on:

This matrix is the finite differences representation of the second derivative of a function

Dictionaries

very useful to store/organize input parameters

We won't use it yet… but soon we'll be able to use it to read/store input parameters

The general structure is the following:

```
params = {} # defines an empty dictionary
params['temperature'] = 300 # creates an entry 'temperature' and stores the value 300
params['mag. field'] = 10
params['method'] = 'RK4'

print(params)
print('T = ', params['temperature'])
```

import numpy as np

the most essential library for scientific computing in python

→ N-dimensional arrays (lists / matrices / tensors)
→ Easy broadcasting of functions over lists
→ Linear Algebra
→ Fourier transforms
→ Random numbers and matrices

Broadcasting:

```
x = np.linspace(0, 2*np.pi, 10)
y = np.sin(x)
```

- x is a list of numbers
- no need for loops
- np broadcast the function and calculates the sine on every element of x

Vector and matrix elements

Vectors

```
v1 = np.array([6, 3, 4, 2, 9, 8])

print(v1[0]) # to access an specific element, 0=first
print(v1[-3]) # can count backwards, -1=end, and so on

v1[-3] = 5 # or change its value
print(v1)

print(v1[1:3]) # open interval (i:f(
print(v1[:3]) # i=0 if omitted
print(v1[1:]) # f=-1 if omitted
print(v1[0:-2]) # counting backwards
```

Matrices

```
m1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

print(m1)
print(m1[1,2])
# line 1, column 2 : recalling that starts from 0

print(m1[:,2])
# [:,2] = [0:-1,2] = all lines, column 2

print(m1[1,:]) # line 1, all columns

print(m1[1:, 1:]) # extracting a submatrix
```

Linear Algebra

from numpy import linalg as LA

[link for the full documentation]

scalar, vector products

….

To calculate the eigenvalues and eigenvectors:

evals, evecs = LA.eig(M)

list of eigenvalues

matrix of eigenvectors stored in columns
→ evecs[:, i] is the eigenvector corresponding to the eigenvalues evals[i]

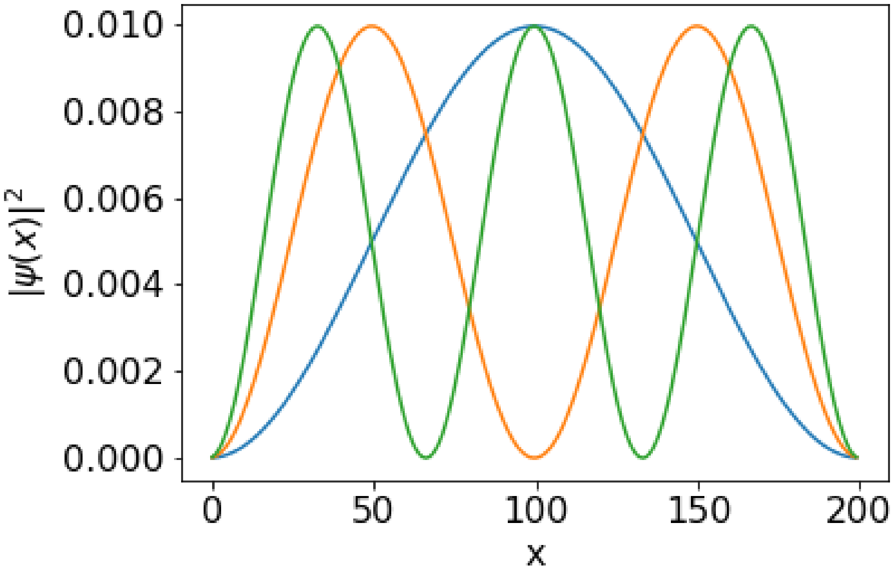
Exercise

Let's use the matrix M from before

1) Calculate the eigenvalues and eigenvectors of M

2) Plot the square of the first 3 eigenvectors

$$M = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$



← using a 200x200 M matrix

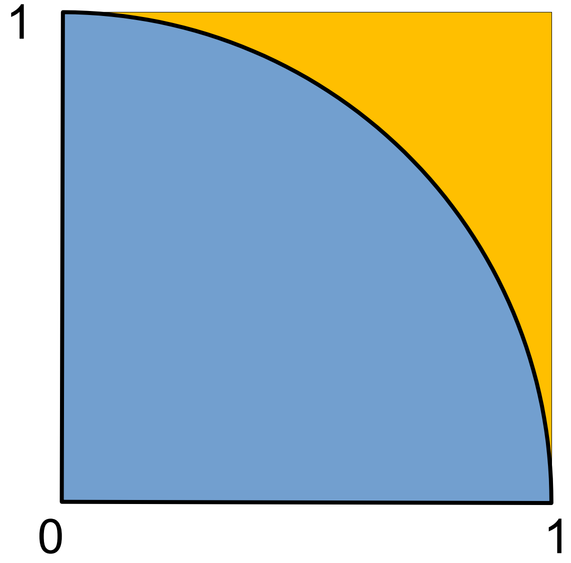
Random numbers

[link for documentation]

from numpy import random

To shuffle a random number between [0,1[→ print(random.rand())

Exercise: calculate pi = 3.141592...



→ How can we use random numbers to calculate pi?

→ What is the probability of a random point (x,y) being inside the blue region?

→ P = (blue area)/(total area) = pi/4

→ or… pi = 4.P

→ use random numbers to calculate P