Analises de Sinais
Introdução ao Python

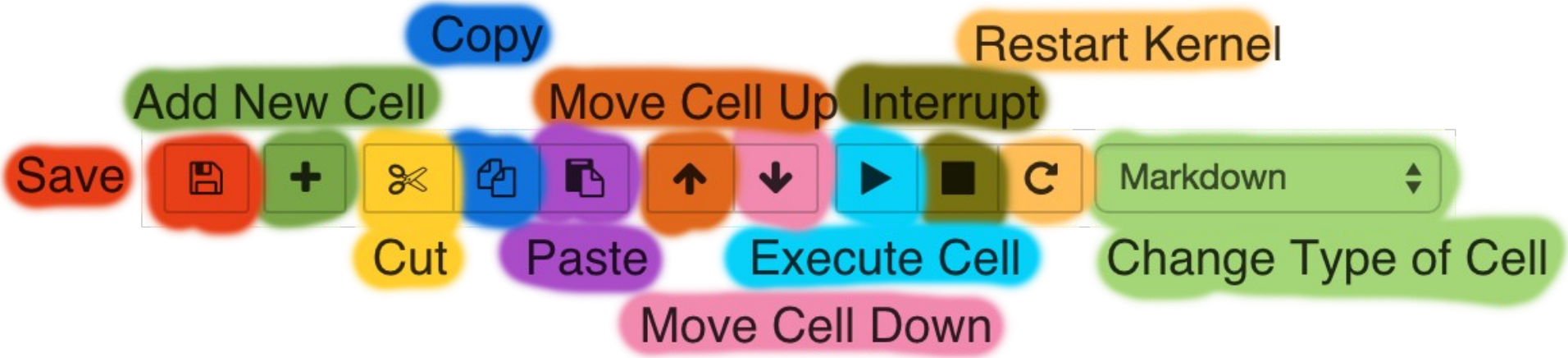
Prof. George Sand Franca
Profa. Susanne Maciel

Introdução ao Python

Introdução muito rápido para Python e em particular o seu sistema científico para o caso de você nunca ter visto isso antes. É, além disso, concede a possibilidade de conhecer o **IPython**.

Existe um monte de escrita motivacional sobre Porque Python? Python é uma “bom” e fácil de aprender, de código aberto, a linguagem programação de propósito geral que passa a ser muito bom para muitas módulos científicas (devido à sua vasto sistema científico).

introdução ao Python para windows



Referência rápida sobre Como usar este o teclado:

Shift + Enter: Executar celular e saltar para a próxima célula

Ctrl / Cmd + Enter: Executar celular e não saltar para a próxima célula

introdução ao Python

Dica

Escrever *scripts* para arquivos de texto e executá-las com o interpretador Python, claro, também funciona:

$python comandos.py

Outra alternativa é o uso interativo na linha de comando:

$ipython

introdução ao Python

Teste 1

Imprimindo alguma coisa no terminal:

print("Ola")

no Python 2 e 3.

introdução ao Python

```
$ipython

in []: from __future__ import print_function, division
       import matplotlib.pylab as plt
```

O primeiro comando para não ter problema com versões do Python.

introdução ao Python

Este parte do curso é relativamente não-interativo e serve para levá-lo até a entendimento com Python supondo que você tenha experiência com a prática de programação com pelo menos em outra linguagem. No entanto, vamos executar em um terminal e é a única maneira de realmente aprender!

A primeira parte irá apresentá-lo ao núcleo da linguagem Python. Este tutorial usa Python 3, mas quase todas as coisas podem ser transferidos para Python 2. Se possível escolher Python 3 para o seu próprio trabalho!

introdução ao Python - 1. Numbers

Python é digitado dinamicamente e atribuir algo a uma variável irá dar-lhe esse tipo.

in[]:

```
#Três tipos básicos
a = 1              # Inteiros
b = 2.0            # Pontos flutuantes
c = 3.0 + 4j       # Número complexo, observe o uso do j para parte complexa
# Ondem é int -> float -> complex
d = a + b          # (int + float = float)
print(d)

e = c ** 2         # c para potencia de 2, executa uma multiplicação complexa.
print(e)
```

Introdução ao Python - 2. Strings

Basta colocar algo em aspas simples ou duplas e ela se tornará uma *string*. Em Python 3 o padrão do strings é unicode, por exemplo, não aceita alfabetos latinos e outros símbolos.

in[]:

```
# Pode usar aspas simples ou duplas para criar strings.
local = "Natal RN"
# Concatenando strings com +.
Onde_eu_sou = 'Eu sou de' + local
# função print().
print(local, 1, 2)
print(Onde_eu_sou)
# Strings tem muito métodos adicionados para manipulações.
print(local.lower())
# Acessar itens com um duplo parênteses. Índice negativo é de trás pra frente.
print(local[0], local[-1])
# Strings pode ser cortado ou selecionado.
print(local[4:])
```

introdução ao Python 3. Lists

Python tem dois principais tipos de coleção: Lista e dicionários. O primeiro é apenas uma coleção ordenada de objetos e é introduzido aqui.

```
#Lista são usadas com duplo colchetes e são ordenados como uma coleção de coisas
tudo = [a, b, c, 1, 2, 3, "ola"]
# Acesa os elementos com a mesma índice de notação dos strings.
# Observe o índice zero!
print(tudo[0])
print(tudo[:3])
print(tudo[2:-2])
# Índices negativos são contados de trás para frente.
print(tudo[-3:])
# Pode adicionar mais elementos com método append.
tudo.append("Voce")
print(tudo)
```

introdução ao Python 4. Dictionaries

Eles são semelhantes aos associativa matrizes ou mapas em outras linguagens. Cada entrada é um par: chave-valor.

```
# Dicionários tem nomeado campos e não há ordem.

# Mesmo caso de listas, pode ter qualquer coisa.
informe = {
      "nome": "George",
      "Nome da Familia": "Franca",
      "idade": ??,
      "Filhos": [1, 2]
}
```

```
# Acesso aos itens usando a chave em parenteses.
print(informe["filhos"])

# Adicionar novo elemento.
print(informe)
information["musica"] = "forro"
print(informe)

# Deleta elementos usando o operador del
del informe["idade"]
print(informe)
```

Introdução ao Python 5. Functions

A chave para conquistar um grande problema é dividi-lo em muitos menores e enfrentá-los um por um. Isso geralmente é conseguido através de funções.

```
# Funções são definidas usando a palavra chave def.
def multi(a, b):
    return a * b

# e chamamos com os argumentos entre parênteses.
print(multi(2, 3))
```

```
# Função pode também ter argumentos opcionais.
def pote_multi(a, b, potencia=1):
    return (a * b) ** potencia

print(pote_multi(2, 3))
print(pote_multi(2, 3, potencia=3))

# Para a função mais complexa é muitas vezes uma boa idéia para nomear explicitamente os argumentos. Isto é mais fácil de ler e menos propenso a erros.
print(pote_multi(a=2, b=3, potencia=3))
```

introdução ao Python 6. Imports

Para utilizar as funções e objetos que não fazem parte do padrão, você tem que importar. Você vai ter que fazer isso muito.

```
# Import qualquer coisa
import math
a = math.cos(4 * math.pi)
# Voce pode selecionar somente o que vc deseja importar
from math import pi
b = 3 * pi
# E até renomear
from math import cos as cosseno
c = cosseno(b)
```

introdução ao Python 7. Controle de Fluxo (Control Flow)

Loops e condicionais são necessários para qualquer tarefa não-trivial. Por favor, observe atentamente que os espaços em branco ** em Python **. Tudo o que é recuado no mesmo nível é parte do mesmo bloco. De longe, os laços mais comuns em Python são **for**-each loops. Loops **While** também existem, mas são raramente utilizados.

introdução ao Python 7. Controle de Fluxo (Control Flow)

```
temp = ["a", "b", "c"]

# Loop típico do Python é For-each, e.g.
for item in temp:
        # tudo com a mesma indexacao faz parte do loop.
        novo_item = item + " " + item
        print(novo_item)

print("Nao faz parte do loop")
----------------------------------------
# Funcao range().
for i in range(5):
        print(i)
```

introdução ao Python 7. Controle de Fluxo (Control Flow)

A segunda estrutura de controle de fluxo fundamental são os condicionais **if/else** e eles funcionam da mesma forma em qualquer outra linguagem.

```
# If/else.
idade = 77

if idade >= 0 and idade < 10:
        print("menor de 10 anos.")
elif age >= 10:
        print("maior que 10 anos.")
else:
        print("Alguma coisa errada!")
```

```
# lista são uma boa maneira de escrever loops de compactos. Entender isso é necessário, pois é muito comum em Python.

a = list(range(10))
print(a)
b = [i for i in a if not i % 2]
print(b)

# Equivalente loop para b.
b = []
for i in a:
        if not i % 2:
           b.append(i)
print(b)
```

Os sistemas científicos do Python

O [SciPy stack] (https://www.scipy.org/stackspec.html) forma a base essencialmente para todas as aplicações científica do Python. Vamos mostrar rapidamente as três bibliotecas centrais:

NumPy

SciPy

Matplotlib

O SciPy stack ainda contém `pandas` (biblioteca para análise de dados em dados tabulares e de séries temporais) e` sympy` (pacote para a matemática simbólica), ambos os pacotes muito poderosos, mas vamos omitir-los neste tutorial.

8. NumPy

Grandes partes do uso NumPy, é para um pacote de matriz oferecendo N-dimensional, digitado matrizes e funções úteis para a álgebra linear, transformadas de Fourier, números aleatórios, e outras tarefas científicas básicas.

```
import numpy as np

#Criar um grande vetor(array) com com 1 milhão de amostras
x = np.linspace(start=0, stop=100, num=1E6, dtype=np.float64)

# A maioria das operações é por elemento.
y = x ** 2

# Usuariso de C e Fortran
print(y.sum())

# FFT e Inversa
x = np.random.random(100)
large_X = np.fft.fft(x)
x = np.fft.ifft(large_X)
```

9.SciPy

SciPy, em contraste com NumPy que só oferece rotinas numéricas básicos, contém uma grande quantidade de funcionalidades adicionais necessários para o trabalho científico. Exemplos são soluções para equações diferenciais básicas, integração numérica e de otimização, matrizes de reposição, rotinas de interpolação, métodos de processamento de sinal, e um monte de outras coisas.

```
from scipy.interpolate import interp1d

x = np.linspace(0, 10, num=11, endpoint=True)
y = np.cos(-x ** 2 / 9.0)
# Interpolação cubica spline para novos pontos.
f2 = interp1d(x, y, kind='cubic')(np.linspace(0, 10, num=101, endpoint=True))
```

10.Matplotlib

Gráficos são feitos usando Matplotlib, um pacote para a criação de lotes estáticas de alta qualidade. Ele tem uma interface que imita Matlab que muitas pessoas estão familiarizados.

```
import matplotlib.pylab as plt

plt.plot(np.sin(np.linspace(0, 2 * np.pi, 2000)), color="green",
      label="Some Curve")
plt.legend()
plt.ylim(-1.1, 1.1)
plt.show()
```