Mini curso sobre python

Jesus O. Cunha

CMA/NOVA Math, Portugal

NOVAMATH
CENTER FOR MATHEMATICS
+ APPLICATIONS

Outline

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

O que é python?

- **Python** é uma linguagem de programação interpretada, orientada a objetos, de alto nível e com semântica dinâmica.
- **Python** suporta módulos e pacotes, que encoraja a programação modularizada e reuso de códigos.
- É considerada uma linguagem popular para análise de dados que conquistou a comunidade científica.
- Dropbox e Instagram são exemplos de programas escritos em **Python**.
- **Python** foi criada em 1990 por Guido Van Rossum(principal autor) no Centro de Matemática Stichting - CWI (http://www.cwi.nl) na Holanda como uma sucessora da linguagem ABC.
- Todos os lançamentos de **Python** são de código aberto (`http://opensource.org`).

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Instalação

- Dica: digitar no buscador "install python3 in windows".
- Guia para instalação do python: Install Python.
- Anaconda Plataform: "a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment" (Wikipedia).

Terminal

```
$ python3 --version
$ python3
>>>
>>> exit()
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Primeiros passos

- Modo iterativo(console): terminal (Linux e MacOS) ou o prompt de comando (Windows).

Terminal

```
$ python3
>>> print("programando em python")
>>> exit()
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Modo script

- Modo script: executa um conjunto de instruções contidas em um arquivo .py.

Script (script2_1.py)

```
print("programando em python")
```

Terminal

```
$ python3 script2_1.py
```

NOVA MATH
CENTER FOR MATHEMATICS + APPLICATIONS

Primeiros passos

- Notebook: ambiente computacional interativo no qual podemos executar um determinado trecho de código.

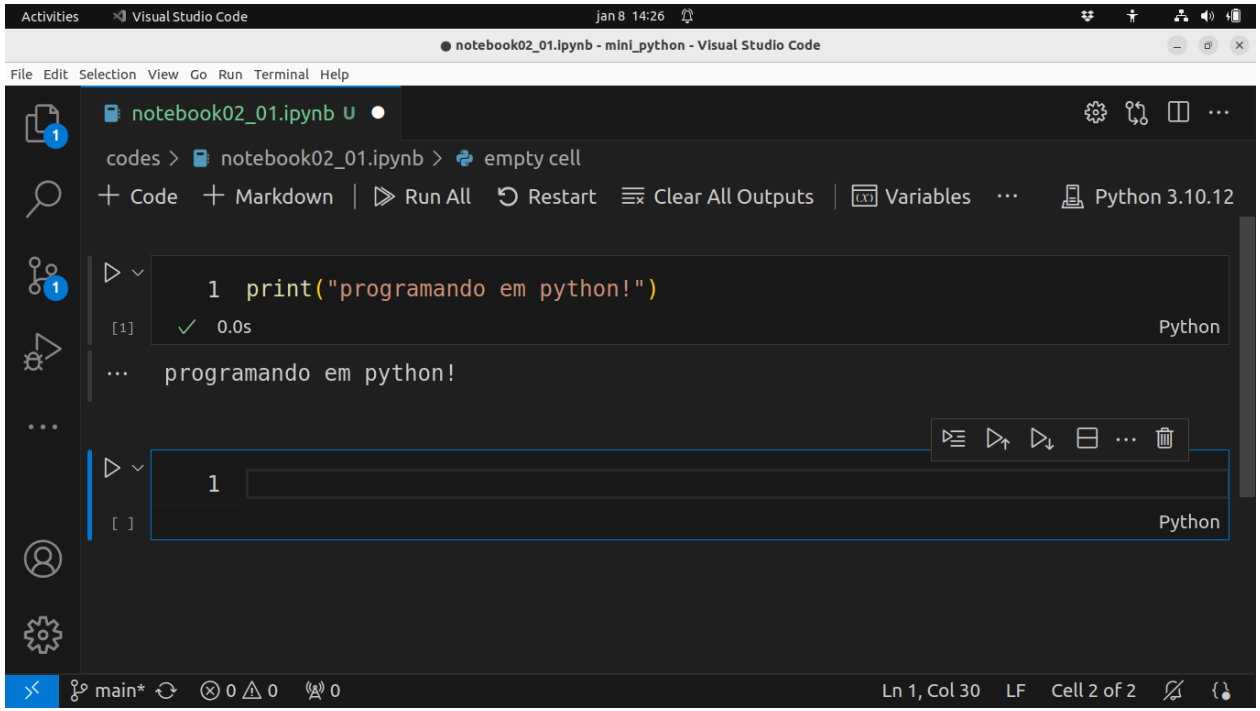

Figure: Notebooks

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Variáveis

- **tipos built-ins:** objetos nativos da linguagem. Recursos que já vêm prontos para uso.

Terminal

```
>>> type(2)
>>> type('python')
>>> type(3.14)
>>> type('3.14')
>>> type(2 + 3j)
```

NOVA MATH
CENTER FOR MATHEMATICS + APPLICATIONS

Variáveis

- **variáveis:** um nome que faz referência a um valor.

Terminal

```
>>> var0 = 2
>>> var1 = "python"
>>> var2 = 3.14
>>> var3 = "3.14"
>>> var4 = 2 + 3j
>>> type(var0)
>>> type(var1)
>>> type(var2)
>>> type(var3)
>>> type(var4)
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Variáveis

- Relação das 33 palavras reservadas em **Python**.

and	as	assert	break	class	continue
for	del	from	None	True	elif
global	nonlocal	try	else	if	not
while	except	import	or	with	False
in	pass	yield	finaly	is	raise
def	lambda	return			

Table: Palavras reservadas

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Instruções

- **instrução:** unidade de código que o Python pode executar.

Terminal

```
>>> x = 6
>>> y = 2
>>> soma = x + y
>>> sub = x - y
>>> print("soma = ", soma)
>>> print("sub = ", sub)
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Instruções

- **script:** contém uma sequência de instruções

Script (script3_1.py)

```
x = 6
y = 2
soma = x + y
sub = x - y
print("soma = ", soma)
print("subtração = ", sub)
```

Execute

Terminal

```
$ python3 script3_1.py
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Operadores

- **operadores aritméticos:** símbolos que representam cálculos.

Terminal

```
>>> 1 + 1
>>> 2 * 3
>>> x = 1
>>> y = 3
>>> x + y
>>> x - y
>>> x * y
>>> x / y
>>> x ** y
>>> 7 // 2
>>> 7 % 2
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Operadores

Operação	Nome	Descrição
$x + y$	adição	soma entre x e y
x - y	subtração	diferença entre x e y
x * y	multiplicação	produto entre x e y
x / y	divisão	divisão entre x e y
x // y	divisão inteira	divisão inteira entre x e y
x % y	módulo	resto da divisão entre x e y
x ** y	exponenciação	x elevado a potência de y

Table: Principais operadores

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Input

- **input()**: função que captura a entrada de valores.

Terminal

```
>>> entrada = input("digite algo: \n")
>>> print(entrada)
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Input

- Crie o script a seguir

Script (script3_2.py)

```
numero = input('Digite um número: \n')
print('Número digitado: ', numero)
```

- Execute o script script3_2.py

Terminal

```
$ python3 script3_2.py
```

NOVA MATH
CENTER FOR MATHEMATICS + APPLICATIONS

Input

- Crie o script a seguir

Script (script3_3.py)

```
nome = input('Digite seu nome: \n')
apelido = input('Digite seu apelido: \n')
print('Nome: ' + nome + '. Apelido: ' + apelido)
```

- execute script3_3.py

Terminal

```
$ python3 script3_3.py
```

NOVA MATH
CENTER FOR MATHEMATICS + APPLICATIONS

Constantes

- **True, False:** valores booleanos(verdadeiro e falso, respectivamente).
- **bool():** retorna **True** quando o argumento passado é verdadeiro e **False**, caso contrário.
- **None:** valor do tipo **NoneType**, representa a abstenção de um valor

Terminal

```
>>> 1 == '1'
>>> 2 > 1
>>> 2 < 1
>>> bool(3 > 5)
>>> bool(1 == 1)
>>> bool(0)
>>> bool(1)
>>> bool(None)
```

NOVA MATH
CENTER FOR MATHEMATICS + APPLICATIONS

Constantes

Operação	Descrição
$x == y$	x igual a y
$x != y$	x diferente y
$x < y$	x menor do que y
$x > y$	x maior do que y
$x <= y$	x menor ou igual a y
$x >= y$	x maior ou igual a y
x is y	x True se x e y são idênticos
x is not y	x True se x e y não são idênticos
x in y	x True se x é membro de y
x not in y	x True se x não é membro de y

Table: Operadores de comparação

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Comandos

- **if:** operador condicional que representa a palavra **se**

Script (script4_1.py)

```
numero = 22
palpite = input("digite um numero: ")
if palpite == numero:
    print("palpite correto")
```

- **else:** operador condicional que representa a palavra **senão**

Script (script4_2.py)

```
numero = 22
palpite = input("digite um numero: ")
if palpite == numero:
    print("palpite correto!")
else:
    print("palpite errado!")
```

Comandos

- Convertendo uma **string** em **inteiro**.

Script (script4_3.py)

```
numero = 22
palpite = int(input("digite um numero: "))
if palpite == numero:
    print("palpite correto")
```

Script (script4_4.py)

```
numero = 22
palpite = int(input("digite um numero: "))
if palpite == numero:
    print("palpite correto")
else:
    print("palpite errado")
```

CENTER FOR MATHEMATICS + APPLICATIONS

Comandos

- Comando **elif**.

Script (script4_5.py)

```
numero = 22
palpite = int(input("digite um palpite: "))
if (palpite == numero):
    print("palpite correto")
elif (palpite > numero):
    print("palpite maior que o número")
else:
    print("palpite menor que o número")
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Comandos

- O comando **while**: loop que faz repetir um conjunto de instruções dentro de um bloco **enquanto** uma condição for verdadeira.

Script (script4_6.py)

```
x = 7
while(x > 1):
    print("x = ", x)
    x = x - 1
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Comandos

Script (script4_7.py)

```
numero = 22
qtd_tentativas = 5
tmp = 1
while (tmp <= qtd_tentativas):
    palpite = int(input("digite um palpite: "))
    if (palpite == numero):
        print("palpite correto")
    elif (palpite > numero):
        print("palpite maior que o número")
    elif (palpite < numero):
        print("palpite menor que o número")
    tmp += 1
```

NOVA MATH
CENTER FOR MATHEMATICS + APPLICATIONS

Comandos

- Comando **break**: para a execução de um conjunto de instruções.

Script (script4_8.py)

```
numero = 22
qtd_tentativas = 5
tmp = 1
while (tmp < qtd_tentativas+1):
    palpite = int(input("digite um palpite: "))
    if (palpite == numero):
        print("palpite correto")
        break
    elif (palpite > numero):
        print("palpite maior que o número")
    elif (palpite < numero):
        print("palpite menor que o número")
    tmp += 1
```

Comandos

- loop **for**: inicia com um valor e incremente esse valor até chegar a um valor final.
- função **range()**: gera um intervalo de valores inteiros.

Script (script4_9.py)

```
print("intervalo [1,10), de 1 em 1 ", end = ": ")
for t in range(1,10):
    print(t, end = ", ")
print("\n")

print("intervalo [1,10), de 2 em 2:", end = ": ")
for t in range(1,10,2):
    print(t, end = ", ")
print("\n")
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Comandos

Script (script4_10.py)

```
numero = 22
qtd_tentativas = 5
for t in range(1,qtd_tentativas+1):
    palpite = int(input("digite um palpite: "))
    if (palpite == numero):
        print("palpite correto")
        break
    elif (palpite > numero):
        print("palpite maior que o número")
    elif (palpite < numero):
        print("palpite menor que o número")
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Estrutura de dados: **list**

- Tipos básicos de sequência(container):
  - **list** (lista)
  - **tuple** (tupla)
  - **range** (objeto de intervalo)
  - **strings** (sequência de texto)
- **list:** sequência mutável de elementos.

Notebook

```
lista0 = []
type(lista0)
```

Notebook

```
lista1 = [1,2,3,4,5]
print(lista1)
```

NOVA MATH
CENTER FOR MATHEMATICS + APPLICATIONS

Estrutura de dados: **list**

Notebook

```
lista3 = ["julia","python",3,5.1]
print(lista3)
```

Notebook

```
lista4 = list("python")
print(lista4)
```

Notebook

```
print(lista4[0])
print(lista4[4])
```

Notebook

```
for i int lista4:
    print(i)
```

Estrutura de dados: **list**

Notebook

```
lista5 = [2,3,5,7,9,11,13]
lista5[0]
```

Notebook

```
lista5[-1]
```

Notebook

```
lista5[0:2]
lista5[:2]
lista5[2:]
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Estrutura de dados: **list**

- Operações sobre **list** usando funções embutidas.

Notebook

```
lista6 = []
lista6.append("um")
lista6.append("dois")
print(lista6)
```

Notebook

```
lista6.extend(["tres","quatro"])
for item in lista6:
    print(item)
```

Notebook

```
lista6.remove("tres")
print("lista6 = ", lista6)
```

Estrutura de dados: **list**

Notebook

```
meses = ['Jan','Fev','Mar','Abr','Jun','Jul','Ago','Set','Out'
print("Escolha três meses do anos!")
for t in range(0,3):
    mes = int(input("Escolha um mês (1-12): "))
    if (1 <= mes <= 12):
        print('O mês escolhido foi {}'.format(meses[mes-1]))
    else:
        print("escolha errada")
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Estrutura de dados: **tuple**

- **tuple:** sequência de elementos que é imutável.

Notebook

```
tupla0 = ()
type(tupla0)
```

Notebook

```
tupla1 = tuple("python")
print(tupla1)
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Estrutura de dados: **range**

Notebook

```
lista0 = [1,2,3]
tupla2 = tuple(lista0)
print(tupla2)
```

Notebook

```
tupla3 = (1,2,lista0)
print(tupla3)
```

Notebook

```
lista0[0] = 4
print(tupla3)
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Estrutura de dados: **range**

- **range:** sequência imutável de números.(Já vista junto com o comando **for**)

Notebook

```
intervalo0 = range(1,5)
for item in intervalo0:
    print(item)
```

Notebook

```
intervalo1 = range(1,10,2)
for item in intervalo1:
    print(item)
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Estrutura de dados: **set**

- **set:** coleção não ordenada que não admite elementos repetidos.

Notebook

```
C = {'a','b','c', 'd','e','f','g'}
print("C =", C )
```

Notebook

```
type(C)
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Estrutura de dados: **set**

- **set:** coleção não ordenada que não admite elementos repetidos.

Notebook

```
A = set('abacate')
B = set('abacaxi')
print("A = ", A)
print("B = ", B)
```

Notebook

```
print("A - B = ", A - B)
print("A | B = ", A | B)
print("A & B = ", A & B)
print("A ^ B = ", A ^ B)
```

NOVA MATH
CENTER FOR MATHEMATICS + APPLICATIONS

Estrutura de dados: **dict**

- **dict:** pares de valor-chave não ordenados.

Notebook

```
dic0 = {'nome':'Joao', 'rua':'São João', 'numero':'33'}
print(dic0)
```

Notebook

```
dic0['nome']
dic0['rua']
```

Notebook

```
dic0['cidade'] = 'Almada'
print(dic0)
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Estrutura de dados: **dict**

Notebook

```
print("keys = ", dic0.keys())
print("values = ", dic0.values())
```

Notebook

```
dic1 = dict(um=1, dois=2, tres=3, quatro=4)
print(dic1)
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Funções

- **Função:** uma sequência de instruções que computa um ou mais resultados

Notebook

```
def velocidade(espaco, tempo):
    v = espaco/tempo
    print('velocidade: {} m/s'.format(v))

velocidade(100, 20)
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Funções: parâmetros

- Parâmetros: lista com nenhum ou mais elementos que podem ser obrigatórios ou opcionais.
- Um parâmetro pode ser opcional com um valor padrão.

Notebook

```
def numeros(x, y=None):
    print('x = {}'.format(x))
    if(y is not None):
        print('y = {}'.format(y))
    else:
        print('y = vazio')
```

Notebook

```
numeros(4,5)
numeros(4)
```

CENTER FOR MATHEMATICS + APPLICATIONS

Funções: **return**

- Uma função pode retorna um valor através do comando **return**.

Notebook

```
def velocidade(espaco, tempo):
    v = espaco/tempo
    return v

resultado = velocidade(100, 20)
print(resultado)
```

NOVA MATH
CENTER FOR MATHEMATICS + APPLICATIONS

Funções: **return**

- Uma função pode conter mais de um **return**.

Notebook

```
def calculadora0(x, y=None):
    if(y is None):
        return x
    else:
        return x**y
```

Notebook

```
calculadora0(2)
calculadora1(2,4)
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Funções: múltiplos **return**

- Uma função pode retornar múltiplos valores.

Notebook

```
def calculadora(x, y):
    return x+y, x-y
```

Notebook

```
resultado = calculadora(1, 2)
```

Notebook

```
print(resultado)
```

Notebook

```
type(resultado)
```

CENTER FOR MATHEMATICS + APPLICATIONS

Funções: return

- Uma função pode retornar um dicionário

Notebook

```
def calculadora(x, y):
    return {'soma':x+y, 'subtração':x-y}
```

Notebook

```
resultados = calculadora(1, 2)
```

Notebook

```
for key in resultados:
    print('{}: {}'.format(key, resultados[key]))
```

NOVA MATH
CENTER FOR MATHEMATICS + APPLICATIONS

Arquivos: escrita de um arquivo

- Função **open()**: abre um arquivo.
- A função **open()** recebe dois parâmetros: o nome do arquivo, e modo(escrita ou leitura) que queremos trabalhar com esse arquivo.
- O modo "w" é usado para escrita no arquivo.
- Função **close()**: fechar um arquivo aberto.

Notebook

```
arquivo = open('grupos.txt', 'w')
arquivo.write("algebra e logic")
arquivo.write('analysis')
arquivo.write("operations research")
arquivo.write('statistics and risk management')
arquivo.close()
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Arquivos: escrita de um arquivo

- O modo "a"(append) é usado para escrita em um arquivo inserindo uma nova linha.
- Os caracteres "\n" insere uma quebra de linha no arquivo.

Notebook

```
arquivo = open('grupos.txt', 'w')
arquivo.write("algebra e logic \n")
arquivo.write('analysis \n')
arquivo.close()
```

Notebook

```
arquivo = open('grupos.txt', 'a')
arquivo.write("operations research\n")
arquivo.write('statistics and risk management\n')
arquivo.close()
```

Arquivos: leitura de um arquivo

- O modo "r" é usado para leitura do arquivo.

Notebook

```
arquivo = open('grupos.txt', 'r')
arquivo.read()
arquivo.close()
```

- Um arquivo é como um fluxo de linhas:
  - Começa no início do arquivo como se fosse um cursor.
  - Ele vai descendo e lendo o arquivo.
  - Após ler tudo, ele fica posicionado no final do arquivo.

Notebook

```
arquivo = open('grupos.txt', 'r')
arquivo.read()
arquivo.read()
arquivo.close()
```

Arquivos: leitura de um arquivo

- O loop **for** pode ser usado para ler cada linha do arquivo.
- Obs: um arquivo é uma sequência de linhas.

Notebook

```
arquivo = open('grupos.txt', 'r')
for linha in arquivo:
    print(linha)
```

NOVA MATH
CENTER FOR MATHEMATICS + APPLICATIONS

Arquivos: leitura de um arquivo

- Função **readline()**: lê apenas uma linha do arquivo.

Notebook

```
arquivo = open('grupos.txt', 'r')
linha = arquivo.readline()
linha
```

- Função **readlines()**: lê as linhas de um arquivo.

Notebook

```
arquivo = open('grupos.txt', 'r')
linhas = arquivo.readlines()
linhas
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Arquivos: leitura de um arquivo

- Função **strip()**: elimina espaços em branco e caracteres especiais, como o \n, no início e no fim da string.

Notebook

```
arquivo = open('grupos.txt', 'r')
nova_math = []
for linha in arquivo:
    linha = linha.strip()
    nova_math.append(linha)

arquivo.close()
nova_math
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

Arquivos: comando **with**

- Comando **with**: fecha um arquivo mesmo que aconteça algum erro no código dentro de seu escopo.

Notebook

```
with open('grupos.txt') as arquivo:
    for linha in arquivo:
        print(linha)
```

NOVAMATH
CENTER FOR MATHEMATICS + APPLICATIONS

joc.silva@campus.fct.unl.pt

NOVAMATH
CENTER FOR MATHEMATICS
+ APPLICATIONS