

Python no terminal

# Importante

- ▶ Scripts Python são arquivos de texto puro.
- ▶ Não use editores de documentos como Word ou LibreOffice Writer para editá-los
- ▶ Use editores como gedit, gvim, kate(não tá instalado) ou o bloco de notas

# Dicas pra quem quiser usar o editor vim ou gvim para editar códigos Python

- ▶ Pacote **vim-X11** (use o instalador de pacotes da sua distribuição)
- ▶ Edite o arquivo chamado **.vimrc** do seu homedir e adicione as seguintes linhas no final
- ▶ Se você for o administrador (root), pode fazer a configuração para todos os usuários editando o arquivo **/etc/vimrc**

```
syntax on  
filetype indent plugin on  
set modeline  
:set tabstop=8 expandtab shiftwidth=4 softtabstop=4
```

# Outros editores

- ▶ Kate (leve e multiplataforma)
- ▶ PyCharm (Community edition é free e cheio de recursos)
- ▶ Gedit (só faz colorir a sintaxe)
- ▶ Bloco de notas no Windows (quebra um galho)
- ▶ Ou ambientes mais sofisticados e pesados se estiver trabalhando num projeto com múltiplos scripts e pacotes (Ex: PyDev for Eclipse)

Aqui iremos usar vim, gvim ou gedit (ou qualquer outro que encontrarem aí instalado)

# Que versão do python?

- ▶ `python --version`
- ▶ `python3 --version`
- ▶ Às vezes o sistema tem python2 e python3 simultaneamente
- ▶ Comando: `which python` (retorna o caminho do interpretador python que está sendo usado por default quando vc digita python no terminal)
- ▶ `which python3`
- ▶ Podemos mudar a variável de ambiente `PATH` para encontrar primeiro um ou outro interpretador python. Deixe isso pra lá se não estiver acostumado com variáveis de ambiente no LINUX.

# Como rodar um script python no terminal

- ▶ crie um diretório para colocar seu(s) script(s)
  - ▶ `mkdir aula5`
- ▶ `cd aula5`
- ▶ Usando o editor de texto de sua preferência crie um script chamado `programa1.py`
- ▶ A primeira linha do script deve ter o “shebang” com o caminho do interpretador python.
- ▶ O programa `env` que vem junto com sistemas Linux pode ser usado para encontrar o caminho do interpretador python do ambiente
  - ▶ `#!/usr/bin/env python3`

# programa1.py

```
#!/usr/bin/env python3  
print ("Olá. Eu sou um programa em python")  
a = 3  
b = 5  
c=a+b  
print(f"A soma de {a} e {b} é {c}")
```

# Dê permissão de execução e rode seu programa

- ▶ No diretório do programa1.py digite:
  - ▶ `chmod +x programa1.py`
- ▶ Agora rode com: `./programa1.py`
- ▶ Alternativamente: `python3 programa1.py`
- ▶ Ou se seu interpretador python padrão já for da versão 3, simplesmente: `python programa1.py`

# Recebendo parâmetros da linha de comando programa2.py

```
#!/usr/bin/env python3
import sys
print ("Olá. Eu sou um programa em python")
#sys.argv contém uma lista [nome_do_programa, arg1, arg2 .. argN]
a = int(sys.argv[1])
b = int(sys.argv[2])
c = a+b
print(f"A soma de {a} e {b} é {c}")
```

# Como saber quantos parâmetros foram passados?

- ▶ `len(sys.argv)`
- ▶ Boa prática: testar se seu programa recebeu o número de argumentos correto

# programa3.py

```
#!/usr/bin/env python3
import sys
#sys.argv contém uma lista dos argumentos passados
numargs = len(sys.argv)
if numargs < 3:
    print("Usage: ", sys.argv[0] , "<num1> <num2>") #ou sys.stderr.write("message")
    exit(1) #Qualquer código de saída diferente de zero significa que o programa deu erro
else:
    a = int(sys.argv[1])
    b = int(sys.argv[2])
    c = a+b
    print(f"A soma de {a} e {b} é {c}")
```

# imc.py

- ▶ Faça um `imc.py` que receba dois argumentos da linha de comando (**altura** em m e **peso** em Kg) , calcule o IMC e imprima o valor na tela.
- ▶ O programa deve ter função chamada `calc_imc` que retorna o `imc` como um float. No programa principal, chame a função `calc_imc`, pegue o resultado e imprima na tela.

```
def calc_imc(altura,peso):  
    imc = ALGUM CALCULO  
    return imc
```

```
./imc.py altura peso
```

# programa4.py (usando import)

- ▶ Programa imc.py tem uma função chamada `calc_imc`
- ▶ Faça um programa4.py que use a função `calc_imc` do `imc.py`
- ▶ Importante: leia o slide seguinte para estruturar adequadamente seu programa

Três alternativas que irão mudar a forma como a função `calc_imc` será chamada:

- ▶ `import imc`

Chamada: `imc.calc_imc(altura,peso)`

- ▶ `import imc as i`

Chamada: `i.calc_imc(altura,peso)`

- ▶ `from imc import calc_imc`

Chamada: `calc_imc(altura,peso)`

# Como estrutura um programa para poder ser importado ou rodado diretamente

```
#!/usr/bin/env python3
```

```
import sys
```

```
def calc_imc(peso, altura):  
    return peso/altura**2
```

```
def main():  
    código principal do programa
```

```
#Se o programa for rodado diretamente, a variável implícita __name__ == '__main__'
```

```
#Se for importado, __name__ será diferente de __main__ e o método main() não será executado
```

```
if __name__ == '__main__':  
    main()
```

# Virtual environment

- ▶ É útil colocar seu programa junto com todos os módulos que ele usa num ambiente virtual
- ▶ No ambiente virtual é possível instalar pacotes que só serão visíveis dentro do ambiente virtual, sem influenciar o resto do sistema
- ▶ Você pode ter quantos ambientes virtuais quiser com cada um instalado em um diretório diferente.

# Como criar um ambiente virtual?

- ▶ Entre no diretório onde você quer criar o ambiente virtual
- ▶ `virtualenv -p python3 NOME_DO_AMBIENTE_VIRTUAL`
  
- ▶ Ex: `virtualenv -p python3 venv`
- ▶ A linha acima irá criar um subdiretório `venv` contendo o interpretador python, a biblioteca padrão e tudo mais pro python funcionar

# Ativando o ambiente virtual

```
source venv/bin/activate
```

```
pip install numpy
```

```
pip install matplotlib
```

```
...
```

```
python meu_programa.py (ou ./programa.py)
```

# Teste do numpy

- ▶ `#!/usr/bin/env python3`
- ▶ `import numpy as np`
- ▶ `X = np.array([[1,2],[3,4]])`
- ▶ `Y = np.array([[2,3],[5,8]])`
- ▶ `e = np.dot(X,Y)`
- ▶ `print(e)`

# Desativando o ambiente virtual

- ▶ Comando: `deactivate`
- ▶ Você pode criar vários ambientes virtuais, um para cada programa, e instalar os módulos que cada um usa com o pip (instalador de pacotes python)
- ▶ Python Package Index <https://pypi.org/>
- ▶ Trabalhando desta forma, você poderá usar o python numa máquina compartilhada sem mexer nas bibliotecas do sistema, evitar conflitos de versões de pacotes etc.