Workshop básico em Python

PyLadies Campinas

Agenda

- Sobre programação
  - O que é?
  - Para que serve?
  - Linguagens de programação
- Python
- Conceitos iniciais
  - Variáveis
  - Operações com variáveis
  - Listas e Dicionários
- Estruturando seu código
  - If, else
  - Loops
  - Funções
- Pacotes
  - Por que usar?
  - De onde vem?
  - Imports
- Mão no código!
  - Atividade com variáveis e listas
  - Atividade com listas e loops
  - Atividade com funções
  - Atividade legal e emocionante

Sobre programação

O que é?
Pra que serve?
Linguagens de programação

python™

Conceitos iniciais

Conceitos Iniciais

- Nós utilizaremos Python para mandar o computador executar instruções
- Essas instruções são executadas de forma sequencial, uma após a outra
- A linguagem Python oferece uma série de expressões que podemos usar
- Nós combinamos essas expressões para montar qualquer tipo de programa que quisermos

Conceitos Iniciais

```
print('Hello, world!')
```

```
>>> Hello, world!
```

- Neste código, utilizamos da expressão `print()` para imprimir um texto na tela

Usando o terminal do Python

-

Variáveis - Tipos Básicos

- Inteiro
- Float
- Booleano
- String

Operações com variáveis

+ - * / % ** [ ]

Mais variáveis

- Listas
- Dicionários

Estruturando seu código

Estruturando Seu Código - if/else

- As vezes precisamos que pedaços do nosso código sejam executados apenas em algumas condições
- Para isso, nós podemos utilizar estruturas de controle
- No Python fazemos isso com `if` e `else`

Estruturando Seu Código - if/else

```
if <condição>:
    código executado caso ‘condição’ seja verdadeira
else:
    código executado caso ‘condição’ seja falsa
```

Estruturando Seu Código - if/else

- A condição pode ser qualquer expressão booleana, ou seja, qualquer coisa que retorne um valor ‘Verdadeiro’ ou ‘Falso

Estruturando Seu Código - if/else

- Nós podemos também fazer expressões compostas
- Essas expressões vão ter o seu valor dependente de outras

Estruturando Seu Código - if/else

- Existem dois conectores fundamentais, **AND** e **OR**
- Suponha a seguinte expressão
- `dia == 'domingo' and hora == 12`
- Ela só será 'verdadeira' caso o dia seja domingo e a hora seja 12
- Caso qualquer um dos lados seja falso, ela será falsa também

Estruturando Seu Código - if/else

- Já o conectivo **or** vai ter um retorno verdadeiro caso qualquer uma das duas expressões seja verdadeira
- `dia == 'domingo' or hora > 18`
- A expressão será verdadeira caso o dia seja domingo ou seja mais tarde que 18h
- Importante: na computação, o **or** em geral é um **ou inclusivo**, ou seja, se as duas partes forem verdadeiras simultaneamente, a expressão ainda é verdadeira
- No exemplo, caso seja domingo e seja depois das 18h, o valor será verdadeiro

Estruturando Seu Código - if/else

- Outro operador é o **not**
- Ele apenas inverte o valor de verdade de uma expressão

Estruturando Seu Código - if/else

- Você pode utilizar os três operadores para construir expressões complexas nos códigos

```
if dia == ‘domingo’ or dia == ‘sábado’ or hora > 18:
    print(‘Tô de folga!’)
else:
    print(‘#partiu #trabalhar’)
```

Repetindo seu código!

- Usando Python nós também podemos *repetir* um certo pedaço de código enquanto uma condição for verdadeira
- Para isso podemos utilizar a instrução **`while`**

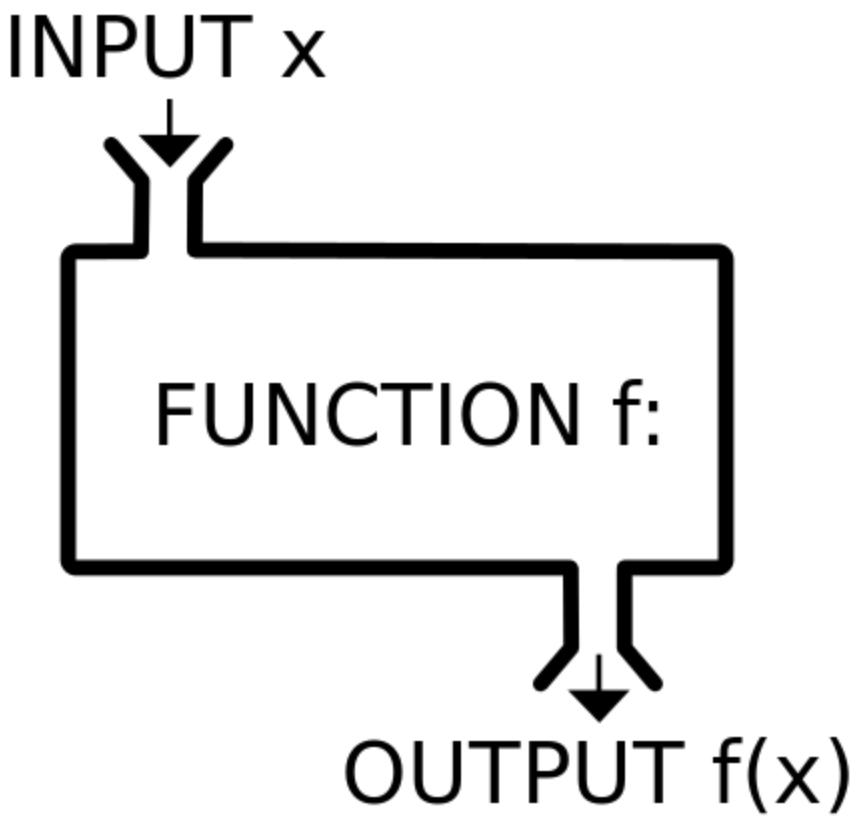
```
while <condição>:

    <repetir esse bloco de código>
```

Repetindo seu código!

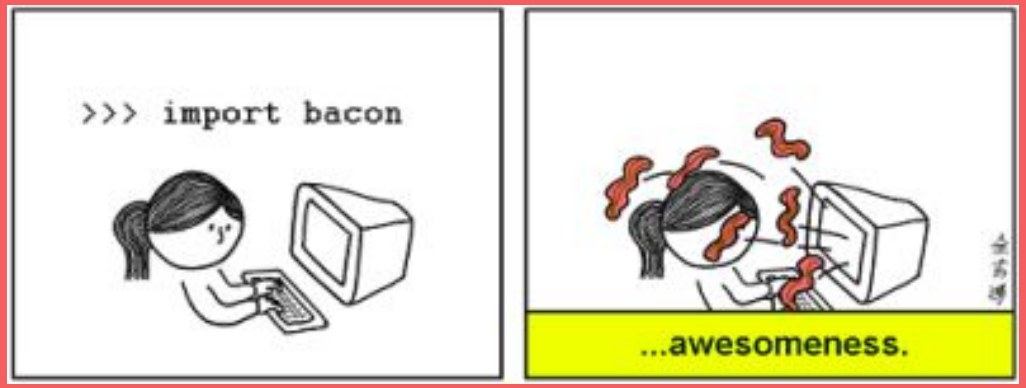
- Outra forma de repetir o código é com `for`
- A principal diferença é que o for aceita uma condição iterável

Dando apelidos pro seu código

- Caso existam pedaços do seu código que você queira reutilizar, você pode criar **funções**
- Uma função permite que você dê um “apelido” para um pedaço de código e invoque-o quando quiser
- Você pode usar a chamada `return` para definir a saída da função


INPUT x
FUNCTION f:
OUTPUT f(x)

Pacotes



Por que pacotes?

- Agilizar MUITO desenvolver coisas úteis
  - Evita reinventar a roda todo o dia
  - Permite focar no que é importante

De onde vem?

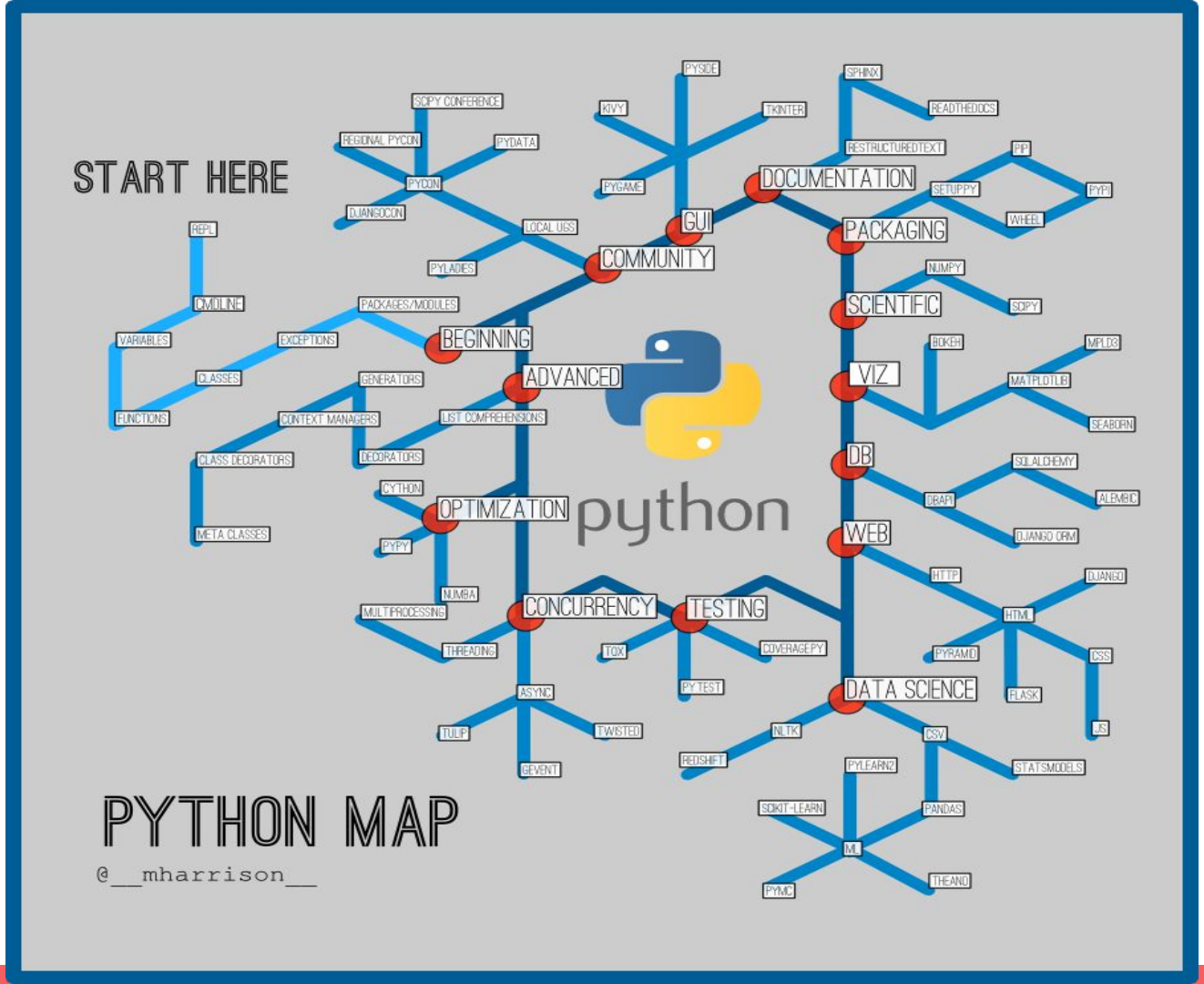
- A comunidade do Python é muito ativa
- Existem pacotes para (quase) tudo

Instalando pacotes (na linha de comando)

```
$ pip --user install meu_pacote
```

O que tem dentro?

- Variáveis
- Funções
- Outros Pacotes
- ...


START HERE
PYTHON MAP
@__mharrison__
python
PYSIDE
SCIPY CONFERENCE
KIVY
TKINTER
SPHINX
READTHEDOCS
REGIONAL PYCON
PYDATA
PYGAME
RESTRUCTUREDTEXT
PIP
PYCON
DOCUMENTATION
SETUPPY
PYPI
DJANGOCON
GUI
WHEEL
REPL
LOCAL US
PACKAGING
PYLADIES
COMMUNITY
NUMPY
IPYTHON
PACKAGES/MODULES
SCIENTIFIC
SCIPY
VARIABLES
EXCEPTIONS
BEGINNING
BOKEH
MPLD3
CLASSES
GENERATORS
ADVANCED
VIZ
MATPLOTLIB
FUNCTIONS
CONTEXT MANAGERS
LIST COMPREHENSIONS
SEABORN
CLASS DECORATORS
DECORATORS
DB
SQLALCHEMY
CYTHON
DBAPI
ALEMBIC
OPTIMIZATION
META CLASSES
DJANGO ORM
PYPY
WEB
HTTP
DJANGO
NUMBA
CONCURRENCY
TESTING
HTML
MULTIPROCESSING
PYRAMID
CSS
THREADING
TOX
COVERAGEPY
FLASK
PYTEST
DATA SCIENCE
ASYNC
JS
NLTK
CSV
TULIP
TWISTED
REDSHIFT
STATSMODELS
GEVENT
PYLEARN2
SCIKIT-LEARN
PANDAS
ML
PYMC
THEANO

Imports

Jeito 1:

```
>>> import numpy

>>> numpy.pi

3.141592653589793

>>> numpy.cos(numpy.pi)

-1.0
```

Jeito 2:

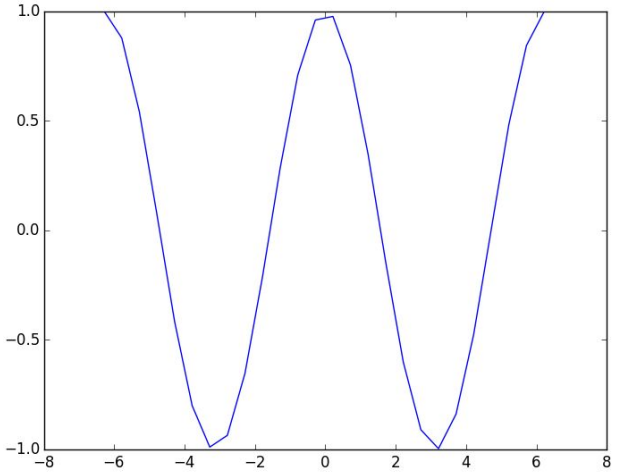
```
>>> from numpy import pi, cos

>>> cos(pi)

-1.0
```

Jeito 3

```
>>> import numpy as np
```

Alguns exemplos: gráficos

```
>>> import numpy as np

>>> x = np.arange(-2*pi, 2*pi, 0.5) # igual ao range p/ floats

>>> y = cos(x)

>>> import matplotlib.pyplot as plt

>>> plt.plot(x,y)

>>> plt.show()
```



Alguns exemplos: lendo tabelas

```
>>> import csv

>>> f = open("dados.csv")

>>> dados = list(csv.reader(f))
```

E agora?

- Python para Zumbis
- Automate the Boring Stuff
- Curso no Codecademy