Apresentação da Linguagem Python com foco em Orientação a Objetos

O QUE É PYLADIES?

PyLadies é um grupo internacional de mentoria com foco em ajudar mais mulheres a tornarem-se participantes ativas e líderes da comunidade Python.

pyladies
São Paulo

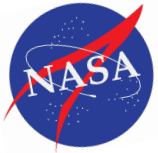
#souPyLadiesSP

O QUE É E ONDE SE USA PYTHON

pyladies São Paulo

python

Python é uma linguagem de programação com código aberto, de alto nível, tipicamente usada para aplicações web ou linguagens de scripts para administração de sistemas.




O QUE É E ONDE SE USA PYTHON

- Aplicações web, desktop e mobile
- Cálculos científicos
- Computação gráfica
- Automação de sistema
- Mineração de dados
- Big Data
- Machine learning
- Processamento de textos
- Tratamento e reconhecimento de imagens
- Animações 3D

O QUE É E ONDE SE USA PYTHON

Foi criada em 1989
por Guido Van Rossum

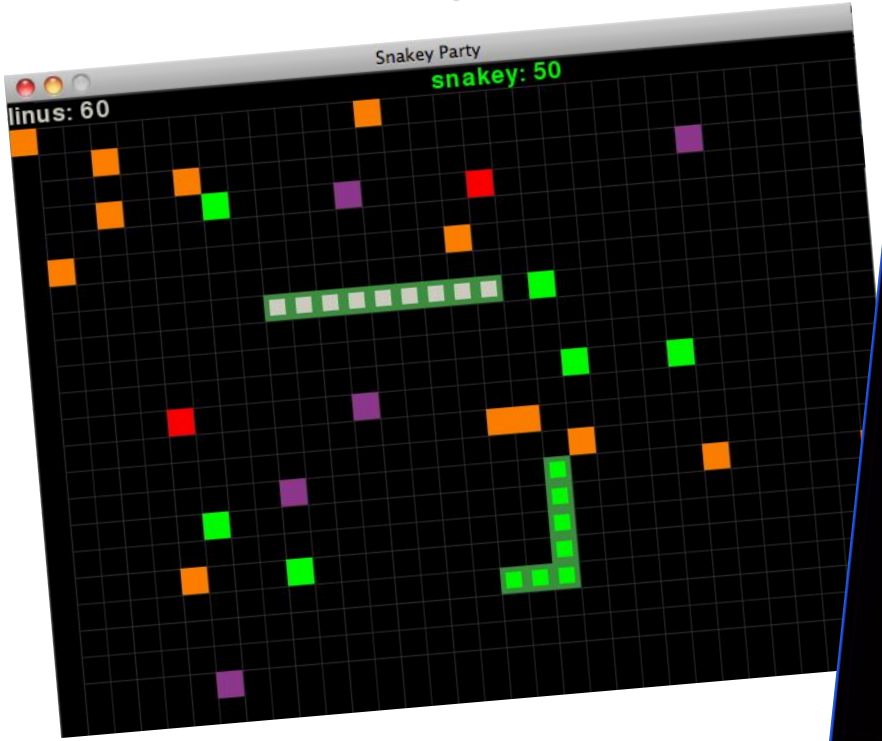
O nome Python
foi inspirado no
seriado britânico
Monty Python

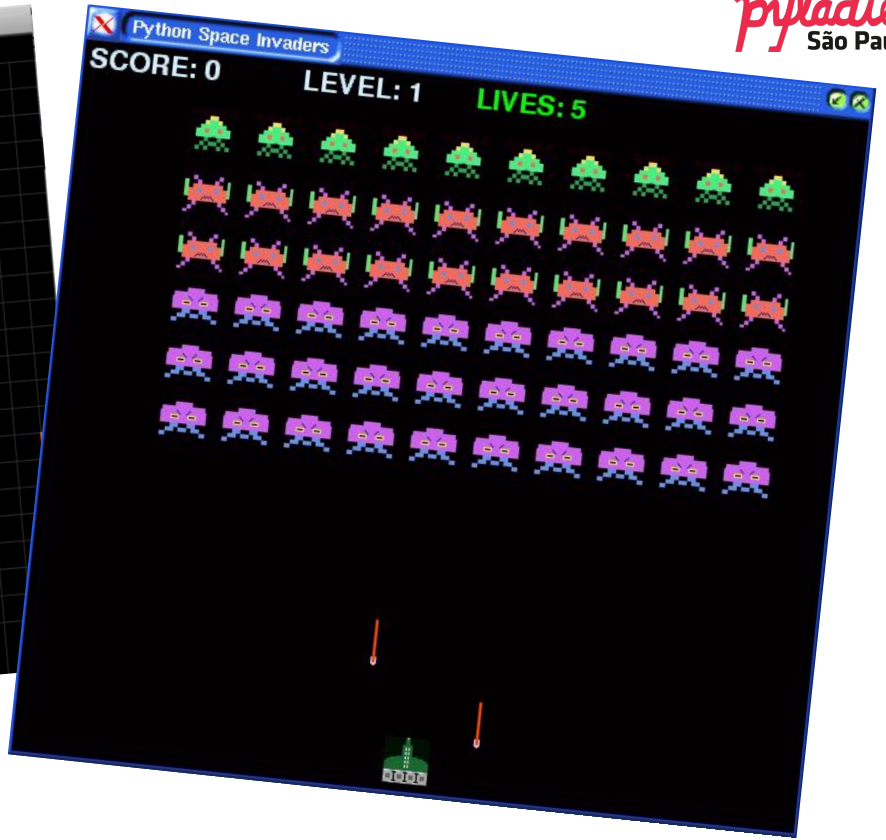
O QUE É E ONDE SE USA PYTHON

EXEMPLOS DE UTILIZAÇÃO

O QUE É E ONDE SE USA PYTHON

EXEMPLOS DE UTILIZAÇÃO





pyladies São Paulo

POR QUE PYTHON?

pyladies São Paulo

Exemplo do mesmo programa em várias linguagens:

Java

```java
public class Hello
{
    public static void main(String args[]) {
        java.util.Scanner s = new java.util.Scanner(System.in);
        System.out.print("Digite seu nome:");
        String nome = s.nextLine();
        System.out.println("Olá, " + nome);
    }
}
```

Pascal

```pascal
program HelloWorld(output);
var
    nome: string;
begin
    writeln('Digite seu nome:');
    read(nome);
    writeln('Olá, ', nome);
end.
```

C

```c
#include <stdio.h>
int main()
{
    char nome[200];
    printf("Digite seu nome:");
    scanf("%s", nome);
    printf("Olá, %s\n", nome);
    return 0;
}
```

Python

```python
nome = input('Digite seu nome:')
print ('Olá,', nome)
```

Fonte: https://pythonhelp.wordpress.com/por-que-python/

POR QUE PYTHON?

pyladies São Paulo

"Simples é melhor que complexo"
(Tim Peters)

ESCREVER NO TERMINAL

A linha de comando é indicada pelas setas **>>>**
É nela que você deve dar os comandos.

Sua primeira tarefa é executar um `print`, o comando para imprimir uma mensagem na tela.

O resultado da execução, deve sair logo abaixo, dessa forma:

```
>>> print('Hello!')
Hello!
```

PROGRAMANDO EM PYTHON

VARIÁVEIS

- **Sintaxe:**

```
<nome da variável> = <valor que quero armazenar>
```

- **Exemplo:**

```
comida = 'hamburguer'
preço = 8.7
vegetariano = True
ingredientes = ['alface', 'tomate', 'lentilha', 'soja',
'cebola', 'queijo', 'pão']
```

PROGRAMANDO EM PYTHON

VARIÁVEIS

```
>>> comida = 'hamburguer'
>>> type(comida)
<class 'str'>


>>> preço = 8.7
>>> type(nota)
<class 'float'>


vegetariano = True
>>> type(vegetariano)
<class 'bool'>
```

```
ingredientes = ['alface', 'tomate',
'lentilha', 'soja', 'cebola', 'queijo',
'pão']
>>> type(ingredientes)
<class 'list'>


dados = {
  'produto': comida,
  'preço': preço,
  'composição': ingredientes,
}
>>> type(dados)
<class 'dict'>
```

VARIÁVEIS


```
ingredientes = ['alface', 'tomate', 'lentilha', 'soja', 'cebola',
'queijo', 'pão']
>>> type(ingredientes)
<class 'list'>
```

mutável

```
ingredientes = ('alface', 'tomate', 'lentilha', 'soja', 'cebola',
'queijo', 'pão')
>>> type(ingredientes)
<class 'tuple'>
```

Imutável (constante)

VARIÁVEIS

Em Python, podemos modificar os tipos de uma variável ao longo do programa, por isso chamamos Python de uma linguagem dinamicamente tipada.

```
a = 'banana'
a = 10
```

Podemos modificar o tipo de variável ou criar uma variável de outro tipo, usando seus "tipos" como comandos, desde que compatíveis:

```
int( )
float( )
str( )
tuple( )
dict( )
list( )
```

PROGRAMANDO EM PYTHON

VARIÁVEIS

- **Exemplos:**

```
>>> a = 2
>>> type(a)
<class 'int'>
>>> b = float(a)
>>> type(b)
<class 'float'>
>>> b
2.0
```

```
>>> c = str(a)
>>> c
'2'
>>> d = int(c)
>>> d
2
>>> type(d)
<class 'int'>
```

```
>>> letras = list('hamburguer')
>>> letras
['h', 'a', 'm', 'b', 'u', 'r', 'g', 'u', 'e', 'r']
>>> type(letras)
<class 'list'>
```

OPERADORES

Podemos fazer operações em Python.
Tanto numéricas, quanto lógicas.

- Operadores numéricos básicos:

Adição: +

Subtração: -

Divisão: /

Multiplicação: *

Potenciação: **

Resto de uma divisão: %

Lembre-se do

PEMDAS

Parênteses

Expoentes

Multiplicação

Divisão

Adição

Subtração

OPERADORES

- **Operadores lógicos ou relacionais:** servem para fazer perguntas que possam ser respondidas com True ou False (verdadeiro/falso)

  Maior que: >
  Menor que: <
  Maior ou igual a: >=
  Menor ou igual a: <=
  Idêntico: ==
  Diferente de: !=
  Não: not
  E: and (todas as condições precisam ser satisfeitas)
  Ou: or (apenas uma das condições precisa ser satisfeita)

PROGRAMANDO EM PYTHON

OPERADORES

- **Exemplos:**

```
>>> a = 2
>>> b = 3
>>> c = 3 / 2
>>> c
1.5
>>> c = 3 // 2
>>> c
1
>>> a > b
False
>>> b > a
True
>>> a == b
False
```

```
>>> 'a' + 'b'
'ab'
>>> 'PyLadies ' + 'São Paulo'
'PyLadies São Paulo'
```

PROGRAMANDO EM PYTHON

OPERAÇÕES COM STRINGS

```
>>> nome = 'Penny...'
>>> print(nome * 5)
>>> 'Penny...Penny...Penny...Penny...Penny...'
```

EM PYTHON, TUDO É OBJETO

- **Comando dir:** lista os métodos de um objeto

- **Sintaxe:**

```
dir(<objeto>)
```

- **Exemplo:**

```
>>> dir(5)
```


```
>>> dir(5)
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__',
'__delattr__', '__dir__', '__divmod__', '__doc__', '__eq__', '__float__',
'__floor__', '__floordiv__', '__format__', '__ge__', '__getattribute__',
'__getnewargs__', '__gt__', '__hash__', '__index__', '__init__',
'__int__', '__invert__', '__le__', '__lshift__', '__lt__', '__mod__',
'__mul__', '__ne__', '__neg__', '__new__', '__or__', '__pos__', '__pow__',
'__radd__', '__rand__', '__rdivmod__', '__reduce__', '__reduce_ex__',
'__repr__', '__rfloordiv__', '__rlshift__', '__rmod__', '__rmul__',
'__ror__', '__round__', '__rpow__', '__rrshift__', '__rshift__',
'__rsub__', '__rtruediv__', '__rxor__', '__setattr__', '__sizeof__',
'__str__', '__sub__', '__subclasshook__', '__truediv__', '__trunc__',
'__xor__', 'bit_length', 'conjugate', 'denominator', 'from_bytes', 'imag',
'numerator', 'real', 'to_bytes']
```

PROGRAMANDO EM PYTHON

EM PYTHON, TUDO É OBJETO


```
>>> 5 + 3
8
```

 # Métodos com __ (dunder, ou seja, duplo under) é para uso interno do python

```
>>> 5 .__add__(3)
8
```

EM PYTHON, TUDO É OBJETO

- **Comando dir:** lista os métodos de um objeto

- **Sintaxe:**

```
dir(<objeto>)
```

- **Exemplo:**

```
>>> dir('Felino')
```


```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__',
'__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
'__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__',
'__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__',
'__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode',
'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index',
'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'isidentifier', 'islower',
'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join',
'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind',
'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate',
'upper', 'zfill']
```

COMO CONSIGO SABER MAIS SOBRE UM OBJETO?

- **Comando help:** mostra a documentação de um "objeto" (mesmo que não seja nativo do Python)

- **Sintaxe:**

```
help(<objeto>)
```

- **Exemplo:**

```
>>> help('PyLadies'.upper)
Help on built-in function upper:

upper(...) method of builtins.str instance
    S.upper() -> str

    Return a copy of S converted to uppercase.
```

```
>>> def batatinha():
...     """
...     Funcao criada por mim para ilustrar o comando help()
...     """
...     print('batata' * 3)
...
>>> help(batatinha)

Help on function batatinha in module __main__:

batatinha()
    Funcao criada por mim para ilustrar o comando help()
```

MAIS SOBRE STRINGS

- **Uppercase:** toda a string em maiúscula

- **Sintaxe:**

```
'<texto>'.upper()
```

- **Exemplo:**

```
>>> 'Batata frita'.upper()
'BATATA FRITA'
```

MAIS SOBRE STRINGS

- **Lowercase:** toda a string em letra minúscula

- **Sintaxe:**

```
'<texto>'.lower()
```

- **Exemplo:**

```
>>> 'Coxinha'.lower()
'coxinha'
```

MAIS SOBRE STRINGS

- **Começa com (startswith)**

  ou

  Termina com (endswith): este comando testa se um texto começa/termina com um elemento (é um teste lógico)

- **Sintaxe:**

```
<variável>.startswith('elemento que procuro')
```

ou

```
<variável>.startswith('elemento que procuro')
```

MAIS SOBRE STRINGS

- **Exemplo:**

```
>>> saudacao = 'Vida longa e próspera!'
>>> saudacao.startswith('V')
True
>>> saudacao.startswith('v')
False
>>> saudacao.endswith('!')
True
```

PROGRAMANDO EM PYTHON

MAIS SOBRE STRINGS

- **Índice em uma string:** número que indica a posição de cada caractere na string

- **Sintaxe:**

```
<variável tipo string>[número]
```

- **Exemplo:**

```
>>> serie = 'DOCTOR WHO'
>>> serie[0]
'D'
>>> serie[2]
'C'
>>> serie[6]
' '
>>> serie[-1]
'O'
```

O número será 0 (zero) para o primeiro caractere na string, 1 para o segundo, 2 para o terceiro, etc.

MAIS SOBRE STRINGS

- **Fatias de uma string:** retorna parte da string, começando pelo primeiro índice e terminando no anterior ao segundo.

- **Sintaxe:**

```
<variável>[índice1:índice2]
```

- **Exemplo:**

```
>>> serie[6:9]
' WH'
>>> serie[4:10]
'OR WHO'
```

```
>>> serie[7:]
'WHO'
>>> serie[-1:]
'O'
```

Quando omitimos um índice, é mostrado o caractere do extremo correspondente

MAIS SOBRE STRINGS

- **Comando Replace:** troca uma string por outra dentro de um texto.

- **Sintaxe:**

```
<variável>.replace('string que quero mudar', 'nova string')
```

- **Exemplo:**

```
>>> spock = 'Fascinante, capitão Kirk'
>>> spock.replace('Fascinante' , 'Incrível')
'Incrível, capitão Kirk'
```

LISTAS

- **Listas:** permitem armazenar várias informações diferentes (número, string, lógico) em uma mesma variável.

- **Sintaxe:**

```
<variável> = [info1, info2, info3]
```

- **Exemplo:**

```
>>> meubicho = ['Gato', 9, True]
```

- **Fatiando listas -** para obter apenas um trecho da lista:

- **Exemplo 1:**

```
>>> meubicho = ['Gato', 9, True]
>>> meubicho[0]
'Gato'
```

Para chamar um dos elementos, uso o índice entre colchetes como faço com strings

- **Exemplo 2:**

```
>>> meubicho = ['Gato', 9, True, ['preto', 'branco']]
>>> meubicho[3]
['preto', 'branco']
```

O elemento 3 da lista meubicho é uma outra lista

Exemplo de lista com um string, um inteiro, um booleano e uma lista

- **Comando Append:** acrescenta dados ao final de uma lista.

- **Sintaxe:**

```
<variável1>.append(<variável2>)
```

- **Exemplo:**

```
>>> nomedaserie = ['Gotham', 'A', 'Dark']
>>> nomedaserie.append('Knight')
>>> print(nomedaserie)
['Gotham', 'A', 'Dark', 'Knight']
```

- **Comando Join:** gruda os elementos de uma sequência de strings, usando um parâmetro fornecido

Funciona apenas com strings

- **Sintaxe:**

```
'<parâmetro que quero usar>'.join(<nome da sequência>)
```

- **Exemplo:**

```
>>> herois = ['Flash', 'Arrow', 'Supergirl']
>>> ' e '.join(herois)
'Flash e Arrow e Supergirl'
```

- **Comando Split:** separa uma string em pontos onde existam separadores de texto (espaço, tab, enter, '/', +, etc.), criando uma lista de strings.

- **Sintaxe:**

```
'<string>'.split('<separador>')
```

- **Exemplo:**

```
>>> '1,2,3,4'.split(',')
['1', '2', '3', '4']
```

- é um modelo de programação baseado no conceito de objetos que contém dados (atributos) e funcionalidades (métodos)

Animal

nome
nivel de alimentacao
nivel de saude
nivel de humor

} atributos (substantivos)

brincar
alimentar
informar_humor

} métodos (verbos, ações sobre o objeto, ou seja, sobre seus atributos)

CLASSES

(object) é opcional, é para manter compatibilidade com Python 2.7

```
class BichoVirtual(object):

    def __init__(self, nome='sem nome'):
        self.nome = nome
        self.nivel_saude = 50
        self.nivel_alimentacao = 50

    @property
    def nivel_humor(self):
        return (self.nivel_saude + self.nivel_alimentacao) / 2

    # continua
```

nome é um parâmetro opcional

@property "transforma" um método em um atributo

```
# continuacao
def informar_humor(self):
    if self.nivel_humor < 30:
        print('Estou mal!')
    elif self.nivel_humor < 50:
        print('Não estou bem...')
    elif self.nivel_humor < 70:
        print('Estou bem!')
    else:
        print('Estou ótimo!!!')
```

self.nivel_humor

é atributo, por isso não usa ()


```
# continuacao
def brincar(self):
    if self.nivel_alimentacao > 10:
        self.nivel_alimentacao -= 10
        self.nivel_saude += 10
        if self.nivel_saude > 100:
            self.nivel_saude = 100
    else:
        print('Está sem energia!!!')
```

CLASSES


```
# continuacao
def alimentar(self):
    if self.nivel_alimentacao < 90:
        self.nivel_alimentacao += 10
    else:
        print('Estou cheio!!!')

def mostrar_estatisticas(self):
    print('Nome: {}'.format(self.nome))
    print('Nível de saude: {}'.format(self.nivel_saude))
    print('Nível de alimentação:
{}'.format(self.nivel_alimentacao))
```

PROGRAMANDO EM PYTHON

INSTÂNCIAS DE UMA CLASSE

```
import random
```

modulo para lidar com aleatório

```
def criar_bichinhos(nomes):
    bichinhos = []
    for nome in nomes:
        bichinhos.append(BichoVirtual(nome))
    return bichinhos
```

```
# opcao pythonica
(4 linhas em 1):
return [BichoVirtual(nome) for nome
in nomes]
```

```
def escolher_bicho_aleatorio(bichos):
    return random.choice(bichos)
```

bichos é uma lista de BichoVirtual

INSTÂNCIAS DE UMA CLASSE


```
# código fora da classe

def mostrar_menu():
    print('Selecione uma opção:')
    print(' A para alimentar')
    print(' B para brincar')
    print(' C para criar bichos')
    print(' H para ver humor')
    print(' P para parar')
```

Classes

```
# código fora da classe
op = None
bichos = None
while op != 'P':
    mostrar_menu()
    if bichos is None or len(bichos) == 0:
    op = 'C'
    else:
        op = input('Escolha opção: ')
    if op == 'C':
        nomes = input('Escreva pelo menos 1 nome, use vírgula para
separar os nomes: ')
        bichos = criar_bichinhos(nomes.split(','))
# ... continua
```

nomes é str, então precisa usar o método split() com vírgula para obter lista de nomes para os bichinhos

Classes

```
# continuação
    elif op != 'P':
        bicho = escolher_bicho_aleatorio(bichos)
        print(bicho.nome)
        if op == 'A':
            bicho.alimentar()
        elif op == 'B':
            bicho.brincar()
        elif op == 'H':
            bicho.informar_humor()
```

Classes


```
# código fora da classe

for bicho in bichos:
    print('-'*10)
    bicho.mostrar_estatisticas()
```

ONDE ESTUDAR ONLINE

- www.codecademy.com/pt
- www.sololearn.com/Course/Python
- pythontutor.com
- www.pycursos.com/python-para-zumbis
- coursera.org
- www.urionlinejudge.com.br/judge/pt/login

REFERÊNCIAS

- http://wiki.python.org.br/PrincipiosFuncionais
- Curso Python para Zumbis
- Curso “An Introduction to Interactive Programming in Python” - Coursera
- http://www.peachpit.com/articles/article.aspx?p=1312792&seqNum=6
- https://docs.python.org/release/2.3.5/whatsnew/section-slices.html
- http://www.bbc.co.uk/education/guides/zqh49j6/revision/3
- https://www.youtube.com/watch?v=SYioCdLPmfw
- https://pt.wikibooks.org/wiki/Python/Conceitos_b%C3%A1sicos/Tipos_e_operadores
- http://www.dcc.ufrj.br/~fabiom/mab225/02tipos.pdf
- http://pt.stackoverflow.com/questions/62844/como-se-insere-n%C3%BAmeros-complexosem-python
- www.arquivodecodigos.net/principal/dicas_truques_categoria2.php?linguagem=12&categoria1=1&categoria2=59

REFERÊNCIAS

- www.arquivodecodigos.net/principal/dicas_truques_categoria2.php?linguagem=12&categoria1=1&categoria2=51
- www.dotnetperls.com/lower-python
- https://pt.wikipedia.org/wiki/Algoritmo
- http://wiki.python.org.br/SoftwarePython
- http://wiki.python.org.br/EmpresasPython
- https://powerpython.wordpress.com/2012/03/16/programas-e-jogos-feitos-em-python/
- http://tutorial.djangogirls.org/pt/python_installation/index.html
- https://powerpython.wordpress.com/2012/03/19/aula-python-17-estrutura-de-decisao/
- https://under-linux.org/entry.php?b=1371
- http://aprenda-python.blogspot.com.br/2009/10/nova-formatacao-de-strings.html
- https://docs.python.org/3/library/string.html#formatspec
- https://www.python.org/dev/peps/pep-3101/

PyLadiesSP

PyLadiesSãoPaulo

PyLadiesSP

@PyLadiesSP

Meetup PyLadiesSP

saopaulo@pyladies.com

Mulheres que amam programar e ensinar Python