



Linguagem de Programação Python

SITEC 2019



Um pouco da história...



Criada em 1991 por Guido van Rossum

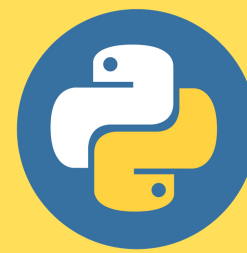


Origem do nome veio em homenagem ao Monty Python



Desenvolvido para leigos

"O difícil é criar uma linguagem que faça tanto sentido para outro ser humano quanto faz para uma máquina ler" — Guido van Rossum



Uma pequena comparação



Java

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```



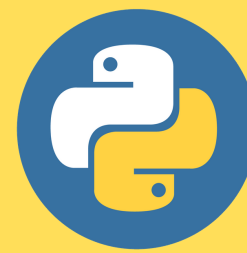
C

```
#include <stdio.h>  
  
int main() {  
    printf("Hello, World!");  
    return 0;  
}
```



Python

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



Ambiente de Desenvolvimento



Python versão ^3.5



Visual Studio Code



VS Code Plugin Python



Características e Especificidades



Script

Multiparadigma

Alto Nível

Interpretada

**Tipagem Dinâmica
e Forte**



Comentários

Comentários

Comentários de uma linha: #

Comentários Multilinha:

""" """ ou ''' '''



Tipos e Variáveis

Tipos

- Definido **dinamicamente** na atribuição
- Tipos comuns:
 - `int`: { -1, 0, 1, ... }
 - `float`: { -1.5, 0.0, 2.32, ... }
 - `str` (string): { "Python", "SITEC", ... }
 - `bool` (boolean): { True, False }
 - `None` (null) : { None }
- Função **type()**

Variáveis

- Case Sensitive: "Var" e "var" são **diferentes**
- Criadas no momento da atribuição
- Podem ser convertidas (casting)
- Operações com variáveis de mesmo tipo (Tipagem forte)
- Atribuição individual ou múltipla
- Tudo em Python é objeto



Operadores Aritméticos

Operadores Aritméticos

- **+** : Adição # $1 + 2 = 3$
- **-** : Subtração # $3 - 1 = 2$
- ***** : Multiplicação # $3 * 2 = 6$
- **/** : Divisão # $4 / 2 = 2$
- **%** : Módulo # $3 \% 2 = 1$
- **//** : Divisão floor # $5 / 2 = 2$
- ****** : Potenciação # $5 ** 2 = 25$



Entrada e Saída

Entrada

- `input('Entrada: ')`
- `raw_input('Entrada: ')` # Indicada na versão 2.x

Saída

- `print('Hello, World!')`
- Formatação:
 - `print(f'Valor: {variavel}')` # 3.6+
 - `printf('Valor: {}'.format(variavel))` # 3.5-



Strings

0	1	2	3	4	5	6
P	Y	T	H	O	N	3

Manipulação e Funções de String:

- `str[0:3]` : Substring ([0:6] = "Python")
- `len(str)` : Retorna tamanho da String (Python3 = 7)
- `str.split("y")` : Divide a String por "y" (Python3 = ["P", "thon3"])
- `str.replace('3', '4')` : Substitue '3' por '4' (Python3 = Python4)
- `str.lower()` : Letras Minusculas (Python3 = 'python3')
- `str.upper()` : Letras Maiusculas (Python3 = 'PYTHON3')



Conditionals

Condicionais

Definição

As estruturas de condição permitem a execução de diferentes linhas de código (diferentes fluxos de execução) a partir da avaliação de uma condição

Blocos `if/elif/else`

```
if <conditional>:  
    pass  
elif <conditional>:  
    pass  
else:  
    pass
```



Expressões e Operadores Lógicos

Expressões Lógicas

- `==` : Igualdade # `1 == 2` : False
- `!=` : Diferença # `1 != 2` : True
- `>=` : Maior ou igual # `3 >= 2` : True
- `<=` : Menor ou igual # `3 * 2 = 6`
- `>` : Maior # `4 / 2 = 2`
- `<` : Menor # `3 % 2 = 1`

Operadores Lógicos

- `and` : Ambas Condições devem ser atendidas
- `not` : Nega a condição
- `or` : Uma das condições devem ser atendidas
- `is` : Avalia a referência entre objetos



Operadores Lógicos **and** & **or**

and

and	a = True	a = False
b = True	True	False
b = False	False	False

or

or	a = True	a = False
b = True	True	True
b = False	True	False



Loops

Loops

Definição

As estruturas de repetição permite executar o mesmo bloco de código enquanto uma condição é atendida (verdadeira).

Blocos `for/while`

```
for number in range(<start>, <stop>):  
    pass
```

```
while <conditional>:  
    pass
```



Lists / Arrays

Index	0	1	2	3	4	5	6
Items	P	Y	T	H	O	N	3

Listas

Definição

As listas ou arrays são uma estrutura de dados que armazena uma coleção de elementos de tal forma que cada um dos elementos possa ser identificado por, pelo menos, um índice ou uma chave.

Declaração e Manipulação `[]`

```
list = ["Oficina", "Python"]  
list[0] # Oficina
```

Observações

- Lista é ordenada e mutável
- Permite duplicação de items



Dictionaries

Dicionários

Definição

É uma estrutura de dados composta de chave e valor comparável a lista (separados por ':')

Declaração e Manipulação { : }

```
dict = { "name": "John", "age": 18 }  
list["name"] # John
```

Observações

Outras Python Collections:

Tuple e ***Set***

- É não-ordenado e mutável
- Não permite duplicação de items
- Similar ao JSON



Functions

Funções

Definição

Funções são blocos de códigos encapsulados que ao serem chamados executam as instruções contidas no mesmo.

Exemplos de Funções:

- `print()`
- `input()`

Bloco `def`

```
def sayHello(name):  
    print(f'Hello {name}!')
```

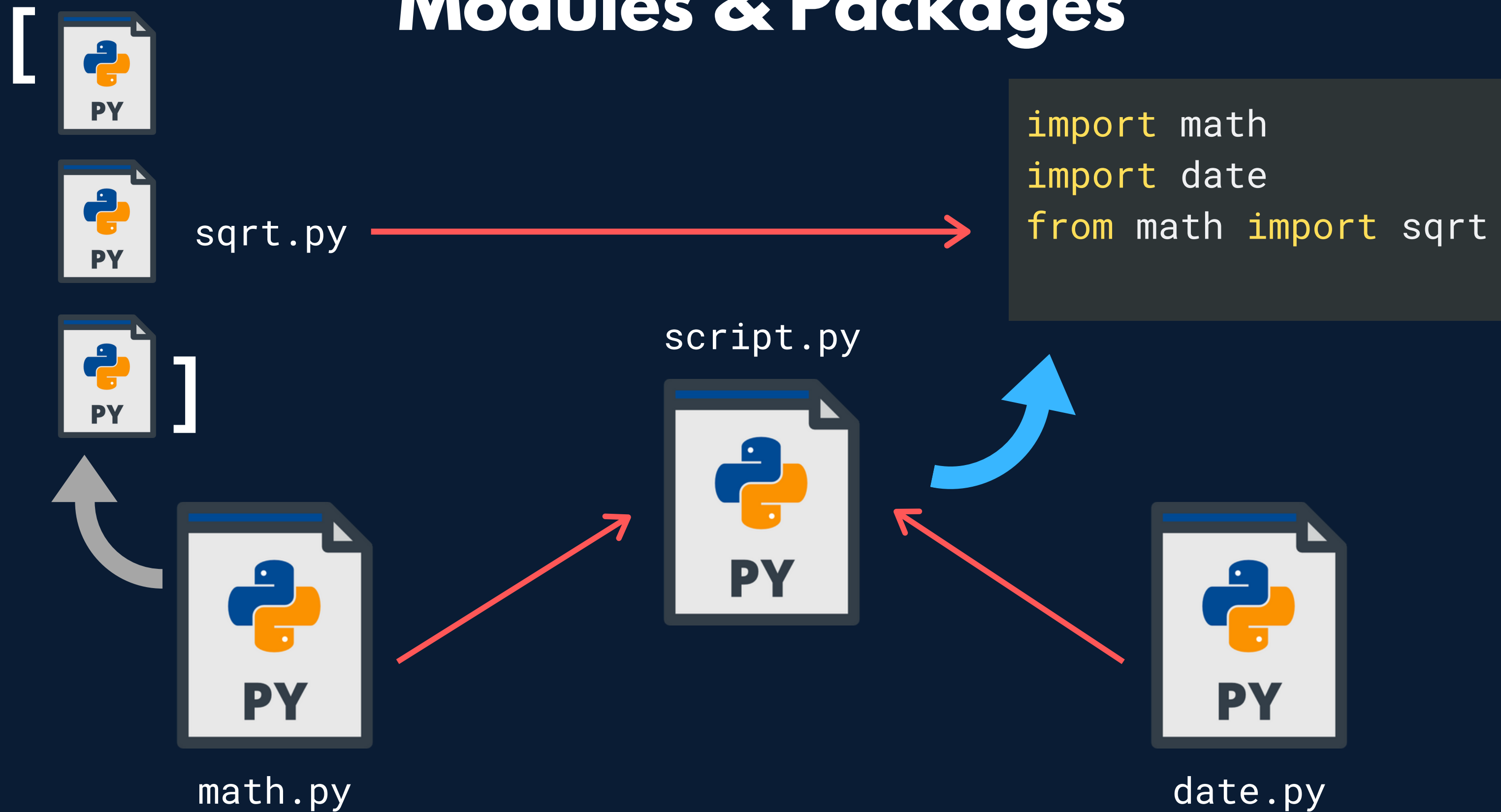
Chamando a Função: `sayHello("John")`

Observações

- Uma função pode conter ou não parametros ou retorno



Modules & Packages





Modules & Packages

Módulos & Pacotes

Definição

Em Python um módulo é um arquivo Python contendo definições e instruções. Um pacote é um conjunto de módulos.

Exemplos de Módulos e Pacotes:

- `import math`
- `from math import sqrt`

Declarações `import/from`

Permitem importar para um arquivo .py um módulo ou pacote.

Declarar no início:

- `import <module || package>`
- `from <package> import <module || *>`



Classes & Objects

Classes

Definição

Classes definem um novo tipo de dado que contém estado (atributos) e comportamento (métodos).

Objetos são variáveis o qual o tipo é uma classe, logo um objeto é uma instância de uma classe.

Bloco `class`

Definem uma classe:

```
class Person:  
    pass
```



Classes & Objects

Atributos e Métodos Especiais

Métodos Especiais

`__init__()`, `__str__()`, `self`

- `__init__(self, <atributes>)`: Método construtor, permite ao instanciar um novo objeto definir seu estado.
- `__str__(self)`: Representação do objeto em string.

Definem uma classe:

```
class Person:  
    __init__(self, name):  
        self.name = name
```

Atributos Especiais

- `self`: Referencia a própria classe (semelhante ao `this` de outras linguagens)

```
__str__(self):  
    return self.name
```



Classes & Objects

Herança

Definição

A herança permite que uma classe filha herde todos os atributos e métodos da sua classe pai.

- `super()`: Invoca a classe pai herdada.

Extends

```
class <Class>(<Parent>):  
    __init__(self, <attributes>, attribute):  
        super().__init__(<attributes>)  
        self.attribute = attribute  
    ...
```



Prática

Atividade Prática: Calculadora

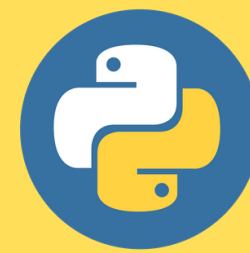


Lista de Exercícios

Lista de Exercícios

A Python Brasil disponibiliza uma lista completa para exercitar os diversos conceitos citados nessa apresentação.

Lista completa: <https://wiki.python.org.br/ListaDeExercicios>



Obrigado!

Dúvidas?

by Yuri Ziemba