

INTRODUÇÃO A PYTHON

PYTHON

- Criado em 1990 por Guido Van Rossum

PYTHON

- Criado em 1990 por Guido Van Rossum
- Código aberto

PYTHON

- Criado em 1990 por Guido Van Rossum
- Código aberto
- Linguagem de programação de alto nível

PYTHON

- Criado em 1990 por Guido Van Rossum
- Código aberto
- Linguagem de programação de alto nível
- Linguagem interpretada

FILOSOFIA

The zen of python, Tim
Peters

```
import this
```

VARIÁVEIS

Na programação variável é um objeto capaz de representar um valor ou expressão.

```
x = 14
```

```
y = 12.4
```

```
z = "LCCV"
```

VARIÁVEIS (NOMENCLATURAS)

VARIÁVEIS (NOMENCLATURAS)

123x = "NO" 

pu^{dim}* = 13 

import = 21.33333333 

VARIÁVEIS (NOMENCLATURAS)

123x = "NO" 

pudim* = 13 

import = 21.33333333 

import keyword

print(keyword.kwlist)

VALORES E TIPOS

Tipo	Descrição
int	Números Inteiro
float	Ponto Flutuante(Números reais)
bool	Valores Booleanos
complex	Números complexos
dict	conjunto associativo
list	Sequência de objetos, que podem ser identificados por índice, que podem ser alterados
tuple	Sequência de objetos imutáveis
str	Cadeia de caracteres imutáveis

VALORES E TIPOS

Tipo	Exemplo
int	x = 1
float	x = 1.111111
bool	x = True, x = False
complex	x=3+2j
dict	x = {'key1': 1, 'key2': 2}
list	x = [1, 2, 3, 4]
tuple	x = (1, 2, 3, 4)
str	x = "LCCV"


OPERADORES ARITMÉTICOS

Operação	Operador
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
exponenciação	**
Parte Inteira	//
Módulo	%
=	Atribuição


AVALIAÇÃO DE EXPRESSÕES

`3**(2+1) == 3**2+1?`

AVALIAÇÃO DE EXPRESSÕES

$3 ** (2 + 1) == 3 ** 2 + 1?$ 


AVALIAÇÃO DE EXPRESSÕES

$3 ** (2 + 1) == 3 ** 2 + 1?$ 

Ordem de avaliação:

- P (Parênteses)

AVALIAÇÃO DE EXPRESSÕES

$3 ** (2 + 1) == 3 ** 2 + 1?$ 

Ordem de avaliação:

- P (Parênteses)
- E (Exponenciação)

AVALIAÇÃO DE EXPRESSÕES

$$3**(2+1) == 3**2+1 \quad \text{X}$$

Ordem de avaliação:

- P (Parênteses)
- E (Exponenciação)
- MDAS (multiplicação, divisão, adição, subtração)

AVALIAÇÃO DE EXPRESSÕES

$3 \times (2+1) == 3 \times 2 + 1?$ 

Ordem de avaliação:

- P (Parênteses)
- E (Exponenciação)
- MDAS (multiplicação, divisão, adição, subtração)
- PEMDAS

OPERAÇÕES STRINGS

x = "linux"

y = " < windows"

OPERAÇÕES STRINGS

```
x = "linux"
```

```
y = " < windows"
```

```
print(x + y)
```

OPERAÇÕES STRINGS

```
x = "linux"
```

```
y = " < windows"
```

```
print(x + y)
```

```
print("penguin"*3)
```

LISTAS

[1, 2, 3, 4]

['python', 2, 3, 4]

['python', 2.2, 4, 5]

LISTAS - ACESSO

```
l = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9,]
```

```
l[0]
```

```
l[-1]
```

```
l[1:10]
```


LISTAS - SLICES

`l[1:3]`

`l[:3]`

`l[1:]`

`l[:]`

LISTAS - OPERAÇÕES

`a = [1, 2, 3]`

`b = [3, 4, 5]`

`a+b`

`a*4`

REMOÇÕES NUMA LISTA

```
a = [1, 2, 3, 4, 5, 6, 7]
```

```
del a[0]
```

```
del a[0:2]
```

LISTAS ANINHADAS - MATRIZES

```
a = [1, 2, 3, 4]
```

```
a = [1, 2, [3, 4]]
```

```
a = ['python', 2, [3, 4]]
```

```
a = [[1, 2], [3, 4]]
```

DICIONÁRIOS

```
d = {'linguagem': 'python', 'versão': '3.7.2'}
```

DICIONÁRIOS

```
d = {'linguagem': 'python', 'versão': '3.7.2'}
```

```
d['linguagem']
```

DICIONÁRIOS

```
d = {'linguagem': 'python', 'versão': '3.7.2'}
```

```
d['linguagem']
```

```
del d['linguagem']
```

DICIONÁRIOS

```
d = {'linguagem': 'python', 'versão': '3.7.2'}
```

```
d['linguagem']
```

```
del d['linguagem']
```

```
d.keys()
```

```
d.values()
```


TUPLAS

Se comportam de forma semelhante as listas, mas são imutáveis

APELIDOS

a = 2

b = a

b += 1

a = [1, 2, 3]

b = a

b[2] = 5

CASTING DE VARIÁVEIS

CASTING DE VARIÁVEIS

`1 == 1.00?`

CASTING DE VARIÁVEIS

`1 == 1.00`  Não Exatamente

`1 == int(1.00)`

`1.00 == "1.00"` 

`1.00 == float(1.00)`

EXPRESSÕES BOLEANAS

EXPRESSÕES BOLEANAS

5 == 1

2 > 3

2 < 6

4 != 3

OPERADORES RELACIONAIS

Descrição	Operador
Maior que	>
menor que	<
igual a	==
diferente de	!=
maior ou igual a	>=
menos igual a	<=

OPERADORES LÓGICOS

OPERADORES LÓGICOS

x = True

y = False

x and y

OPERADORES LÓGICOS

x = True

y = False

x and y

x or y

OPERADORES LÓGICOS

`x = True`

`y = False`

`x and y`

`x or y`

`x = [1, 2, 3, 4, 5]`

`1 in x`

BOOLEANS

True, False

BOOLEANS

True, False

‘ ’?

‘python’?

5 - 2?

[]?

CONDICIONAIS

CONDICIONAIS

```
if expressão_logica:
```

```
    Comandos
```


CONDICIONAIS

```
if expressão_logica:
```

```
    Comandos
```

```
else:
```

```
    Comandos
```

CONDICIONAIS

```
if expressão_logica:
```

```
    Comandos
```

```
elif expressão_logica:
```

```
    Comandos
```

```
else:
```

```
    Comandos
```

LOOPS

LOOPS

```
for i in iterator:
```

Comandos

LOOPS

```
for i in iterator:
```

Comandos

```
while expressão_logica:
```

Comandos

COMENTÁRIOS

Comentário de Linha

‘ ’ ’

comentário de bloco

‘ ’ ’