

Computação 1 - Python

Aula 3 - Teórica: Tipos de dados, Strings, Estrutura Condicional

Tipos de Dados

Dados Numéricos

- **Números Inteiros:** Int/Long
- **Ponto Flutuante:** Float
- **Números Complexos:** Complex

Operações com dados de um mesmo tipo tendem a gerar resultados do mesmo tipo dos operandos;

Operações com dados de diferentes tipos geram resultados do tipo mais complexo;

Tipos de Dados

Sequência de caracteres: Str

- Constantes string são escritas usando aspas simples ou duplas

Exemplo:

"a" ou 'a'

- O operador + pode ser usado para concatenar strings

Exemplo

"a"+"b" é o mesmo que "ab"

- O operador * pode ser usado para repetir strings

Exemplo

"a"*10 é o mesmo que "aaaaaaaaaa"

Tipos de Dados

Conversão entre tipos de dados

- Dados numéricos não são convertidos automaticamente para o tipo string

Exemplo:

```
>>> "Minha idade é " + 15 + " anos"
TypeError: Can't convert 'int' object to string implicitly
>>> "Minha idade é " + str(15) + " anos"
"Minha idade é 15 anos"
```

- Para converter uma string em inteiro ou float podemos usar:

Exemplo

```
>>> int("15")
15
>>> float("3.14")
3.14
```

Exercício

- Escreva uma função que receba como parâmetro o nome e a idade de uma pessoa, e que retorne a frase:

“Olá fulano, meu nome é Python e eu tenho x anos. ”

onde fulano e x são, respectivamente, o nome e o dobro da idade do usuário.

String

Exercício

- Escreva uma função que receba como parâmetro o nome e a idade de uma pessoa, e que retorne a frase:

“Olá fulano, meu nome é Python e eu tenho x anos. ”

onde fulano e x são, respectivamente, o nome e o dobro da idade do usuário.

```
# Função que recebe nome e idade e  
# escreve uma frase  
# str,int → str  
def olafulano(nome,idade):  
    return "Olá " + nome + ", meu nome é Python, e tenho " +  
    str(2*int(idade)) + " anos."
```

Tipos de Dados

Booleano: Bool

- Assume apenas dois valores: verdadeiro (True) ou falso (False)
- É o tipo de dado resultante das operações de comparação.

Exemplo:

```
>>> 3>2
```

```
True
```

```
>>> 10 <= 5
```

```
False
```

Relações e Expressões Booleanas

Relações

- Operadores: $>$, $<$, $==$ (igual), $!=$ (diferente), $>=$, $<=$

ATENÇÃO

- $X == Y$: operador relacional \Rightarrow X É IGUAL A Y
- $X = Y$: operador de atribuição \Rightarrow ATRIBUIR A X O VALOR DE Y

Relações e Expressões Booleanas

Relações

- Operadores: $>$, $<$, $==$ (igual), $!=$ (diferente), $>=$, $<=$

Expressões Booleanas

Retornam como resultado de sua avaliação os valores **verdadeiro** (**True**) ou **falso** (**False**)

Operadores

- 1 **not** (negação)
- 2 **and** (e)
- 3 **or** (ou) (nesta ordem de precedência)

Operadores Lógicos

Operadores: **not** (negação), **and** (e), **or** (ou)

- **x and y**: verdadeiro se, e somente se x e y forem ambos verdadeiros.
- **x or y**: falso se, e somente se x e y forem ambos falsos.
- **not x**: falso se x for verdadeiro, e verdadeiro se x for falso.

Observe que x e y podem ser variáveis booleanas ou podem ser expressões booleanas compostas de operadores relacionais e operadores lógicos.

Expressões Booleanas

Tabela Verdade

Exp 1	Exp 2	Exp 1 and Exp 2	Exp 1 or Exp 2	not Exp 1
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

Estrutura Condicional Simples

Faça uma função que, dado um número inteiro X passado como parâmetro, retorna a string “*positivo*” caso X seja um número positivo, e “*não positivo*” caso contrário.

Estrutura Condicional Simples

Faça uma função que, dado um número inteiro X passado como parâmetro, retorna a string “*positivo*” caso X seja um número positivo, e “*não positivo*” caso contrário.

```
# Função que recebe um número inteiro e  
# determina se ele é positivo  
# int → str  
def positivo(X):  
    if X > 0 :  
        return 'positivo'  
    return 'não positivo'
```

Estrutura Condicional Simples

Faça uma função que, dado um número inteiro X passado como parâmetro, retorna a string “*positivo*” caso X seja um número positivo, e “*não positivo*” caso contrário.

```
# Função que recebe um número inteiro e  
# determina se ele é positivo  
# int → str  
def positivo(X):  
    if X > 0 :  
        return 'positivo'  
    return 'não positivo'
```

Estrutura Condicional Simples

```
if expressão :  
    comandos
```

Estrutura Condicional Simples

Faça uma função que, dado um número inteiro X passado como parâmetro, retorna a string “positivo” caso X seja um número positivo, e “não positivo” caso contrário.

```
# Função que recebe um número inteiro e  
# determina se ele é positivo  
# int → str  
def positivo(X):  
    if X > 0 :  
        return 'positivo'  
    return 'não positivo'
```

Estrutura Condicional Simples

```
if expressão :  
    comandos
```

expressão na estrutura condicional é um tipo especial de expressão chamado **expressão booleana**, que pode ser verdadeira ou falsa.

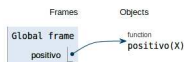
Python Tutor

```
1 # Exemplo - Estrutura Condicional Simples
2
3 # Faça uma função que,
4 # dado um número inteiro X
5 # passado como parâmetro,
6 # retorna a string 'positivo'
7 # caso X seja um número positivo,
8 # e 'não positivo' caso.
9
10 def positivo(X):
11     if X > 0 :
12         return 'positivo'
13     return 'não positivo'
14
15 # Caso de Teste 1
16 positivo(3)
17
18 # Caso de Teste 2
19 positivo(-5)
```

[Edit code](#)



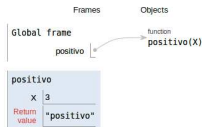
→ line that has just executed
→ next line to execute



Python Tutor

```
1 # Exemplo - Estrutura Condicional Simples
2
3 # Faça uma função que,
4 # dado um número inteiro X
5 # passado como parâmetro,
6 # retorna a string 'positivo'
7 # caso X seja um número positivo,
8 # e 'não positivo' caso.
9
10 def positivo(X):
11     if X > 0 :
12         return 'positivo'
13     return 'não positivo'
14
15 # Caso de Teste 1
16 positivo(3)
17
18 # Caso de Teste 2
19 positivo(-5)
```

→ line that has just executed
→ next line to execute



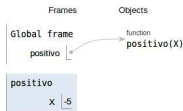
Python Tutor

```
1 # Exemplo - Estrutura Condicional Simples
2
3 # Faça uma função que,
4 # dado um número inteiro X
5 # passado como parâmetro,
6 # retorna a string 'positivo'
7 # caso X seja um número positivo,
8 # e 'não positivo' caso.
9
10 def positivo(X):
11     if X > 0 :
12         return 'positivo'
13     return 'não positivo'
14
15 # Caso de Teste 1
16 positivo(3)
17
18 # Caso de Teste 2
19 positivo(-5)
```

[Edit code](#)

<< First < Back Step 9 of 11 Forward > Last >>

→ line that has just executed
→ next line to execute



Python Tutor

```
1 # Exemplo - Estrutura Condicional Simples
2
3 # Faça uma função que,
4 # dado um número inteiro X
5 # passado como parâmetro,
6 # retorna a string 'positivo'
7 # caso X seja um número positivo,
8 # e 'não positivo' caso.
9
10 def positivo(X):
11     if X > 0 :
12         return 'positivo'
13     return 'não positivo'
14
15 # Caso de Teste 1
16 positivo(3)
17
18 # Caso de Teste 2
19 positivo(-5)
```

[Edit code](#)

<< First

< Back

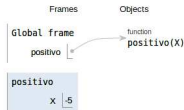
Step 10 of 11

Forward >

Last >>

→ line that has just executed

→ next line to execute



Python Tutor

The image displays the Python Tutor web application. On the left, a code editor shows a Python function `positivo(X)` with comments in Portuguese. The function returns 'positivo' if `X > 0` and 'não positivo' otherwise. The code is being executed step-by-step, with line 13 highlighted. Below the code editor is a progress bar and navigation buttons: '<< First', '< Back', 'Step 11 of 11', 'Forward >', and 'Last >>'. A legend indicates that a green arrow points to the line just executed and a red arrow points to the next line to execute.

On the right, the 'Frames' and 'Objects' panels are visible. The 'Frames' panel shows the 'Global frame' and a local frame for the function `positivo`. The 'Objects' panel shows the function object `positivo(X)`. The local frame for `positivo` contains a variable `x` with the value `-5` and a 'Return value' of `'não positivo'`.

```
1 # Exemplo - Estrutura condicional simples
2
3 # Faça uma função que,
4 # dado um número inteiro X
5 # passado como parâmetro,
6 # retorna a string 'positivo'
7 # caso X seja um número positivo,
8 # e 'não positivo' caso.
9
10 def positivo(X):
11     if X > 0 :
12         return 'positivo'
13     return 'não positivo'
14
15 # Caso de Teste 1
16 positivo(3)
17
18 # Caso de Teste 2
19 positivo(-5)
```

Global frame

positivo

function
positivo(X)

positivo

x	-5
Return value	"não positivo"

<< First < Back Step 11 of 11 Forward > Last >>

→ line that has just executed
→ next line to execute

Estrutura Condicional

Faça uma função que determina se um número inteiro X passado como parâmetro é positivo, negativo ou zero. O valor de retorno da função deve ser uma dentre as strings *“positivo”*, *“negativo”* ou *“zero”*.

Estrutura Condicional

Faça uma função que determina se um número inteiro X passado como parâmetro é positivo, negativo ou zero. O valor de retorno da função deve ser uma dentre as strings “positivo”, “negativo” ou “zero”.

```
# Função que recebe um número inteiro e  
# determina se ele é positivo, negativo ou zero  
# int → str  
def PosNegZero(X):  
    if X > 0 :  
        return 'positivo'  
    if X < 0 :  
        return 'negativo'  
    if X == 0 :  
        return 'zero'
```

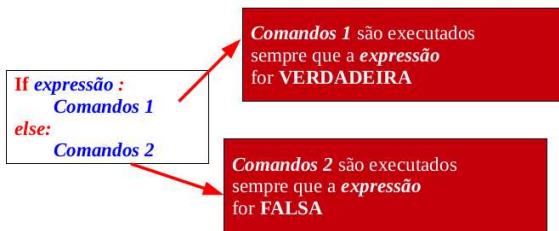
Estrutura Condicional

Faça uma função que determina se um número inteiro X passado como parâmetro é positivo, negativo ou zero. O valor de retorno da função deve ser uma dentre as strings “positivo”, “negativo” ou “zero”.

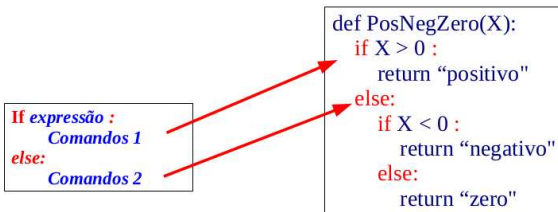
```
# Função que recebe um número inteiro e  
# determina se ele é positivo, negativo ou zero  
# int → str  
def PosNegZero(X):  
    if X > 0 :  
        return 'positivo'  
    if X < 0 :  
        return 'negativo'  
    if X == 0 :  
        return 'zero'
```

Podemos simplificar o código ? Como ?

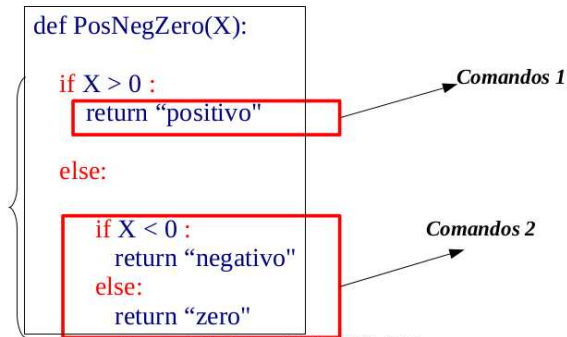
Estrutura Condicional Composta



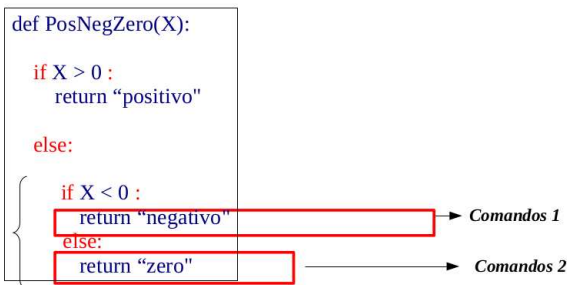
Estrutura Condicional Composta



Estrutura Condicional Composta



Estrutura Condicional Composta



Estrutura Condicional Composta

Faça uma função que determina se um número inteiro X passado como parâmetro é positivo, negativo ou zero. O valor de retorno da função deve ser uma dentre as strings “positivo”, “negativo” ou “zero”.

```
# Função que recebe um número inteiro e  
# determina se ele é positivo, negativo ou zero  
# int → str  
def PosNegZero(X):  
    if X > 0 :  
        return 'positivo'  
    else:  
        if X < 0 :  
            return 'negativo'  
        else:  
            return 'zero'
```

Estrutura Condicional Composta

Faça uma função que determina se um número inteiro X passado como parâmetro é positivo, negativo ou zero. O valor de retorno da função deve ser uma dentre as strings “positivo”, “negativo” ou “zero”.

```
# Função que recebe um número inteiro e  
# determina se ele é positivo, negativo ou zero  
# int → str  
def PosNegZero(X):  
    if X > 0 :  
        return 'positivo'  
    elif X < 0 :  
        return 'negativo'  
    else:  
        return 'zero'
```

Teste no Python Tutor para os seguintes casos: *PosNegZero(0)*,
PosNegZero(2) e *PosNegZero(-12)*

Estrutura Condicional

1. Faça uma função que receba como entrada o código de uma mercadoria e o preço e retorne como saída o preço da mercadoria, sendo que se o código for '00' um desconto de 10% no preço deve ser aplicado.
2. Faça uma função que receba como entrada dois números e retorne o maior deles. Os valores são, por definição, diferentes entre si.
3. Faça uma função que receba como entrada dois números e retorne o maior deles. Caso os números sejam iguais, retorne *“Os números são iguais”*.

Computação 1 - Python

Aula 3 - Teórica: Tipos de dados, Strings, Estrutura Condicional