INSTITUTO DE INFORMÁTICA
Universidade Federal de Goiás

Introdução a Python

Altino Dantas

PARTE I

INSTITUTO DE INFORMÁTICA
UFG

UFG
UNIVERSIDADE FEDERAL DE GOIÁS

Objetivos da aula

- Conhecer os tipos de operadores em Python;
- Aprender as estruturas condicionais e de repetição em Python.

Tipos de operadores

No Python há alguns tipos de operadores:

- Os **operadores matemáticos** que são utilizados quando vamos fazer algum tipo de cálculo.
- Os **operadores relacionais** que são utilizados para comparações, retornando **true** ou **false** em seus resultados.
- Os **operadores lógicos** que são utilizados para testes condicionais.

Tem muito + https://docs.python.org/3/library/operator.html#mapping-operators-to-functions

Operadores matemáticos

Simples

Operação	Operador
adição	+
subtração	-
multiplicação	*
divisão	/

Compostos

Operação	Operador
exponenciação	**
parte inteira	//
módulo	%

```
print(10+10)
print(10+(50+50))
print(10-10*5)
print(1000%80)
print(10**5)
print(10/6)
print(10//6)
```

Operadores relacionais

Descrição	Operador
Maior que	>
Menor que	<
Igual a	==
Maior ou igual a	>=
Menor ou igual a	<=
Diferente	!=

```
x = ""
if(x):
    print("O valor de x é igual a vazio")
else:
    print("A variável 'x' contém caracteres “)
```

Operadores lógicos

```
var1 and var2: # retorna True se var1 e var2 forem TRUE;
var1 or var2:  # retorna True se num1 ou num2 forem
TRUE;
not var:       # retorna TRUE se var for False;
```

Atribuições

Descrição	Exemplo
Simples	a = 2
	a = b = 5
Múltipla	a, b, c = 2, 5, 3
Composta	a += 1
	a -= 3
	a *= 4
	a /= 3
	a %= 10
	a //= 60

Atualização de variáveis

- Python checa da direita para a esquerda.

```
>>> x = x + 1
```

```
>>> x = x + 1
NameError: name 'x' is not defined
```

```
>>> x = 0
>>> x = x + 1
```

Ordem de operações

operação	Exemplo	Resultado
Parênteses	2 * (3-1)	4
Exponenciação	2 * 3 ** 2	18
Multiplicação e Divisão	6 + 4 / 2	8
Mesma prioridade	Processa-se da esquerda para a direita	

Estruturas de decisão

- Estruturas de decisão servem para manipular o fluxo de execução de código em uma aplicação, tendo como base um teste lógico, que espera um valor verdadeiro ou falso.

```
if condição:
    print(“verdadeiro”)
else:
    print(“falso”)
```

```
Verdadeiro if condição else Falso
```

https://docs.python.org/3/tutorial/controlflow.html

Estruturas de decisão

A estrutura de decisão **if** irá testar uma condição e caso ela seja verdadeira, o que estiver dentro do seu bloco de código será executado, como por exemplo:

```
idade = 18
if idade == 18:
    print('Você é maior de idade')
else:
    print('Você é menor de idade')
```

Estruturas de decisão

combinando operadores

- Podemos atender mais de um requisito em uma estrutura de decisão utilizando operadores **and** e **or**.

```
idade = 18
sobrio = True
if idade >= 18 and sobrio == True:
    print ('Você pode dirigir')
else:
    print ('Você não pode dirigir')
```

Estruturas de decisão

Encadeamento de condição (*elif*)

- Em programação existe ainda o que nós chamamos de encadeamento de condições, ou seja, podemos ter diversos **if**s dentro de uma mesma estrutura.

```
if condição:
    print('verdadeiro')
elif condicao:
    print('falso')
else:
    print('Faça isso')
```

Mesmo se mais de uma condição for verdade, só o primeiro ramo verdadeiro é executado.

Estruturas de decisão

Aninhado	`if 0 < x:`
	`  if x < 10:`
	`  print('x um dígito menor que 10')`

Operado lógico	`if 0 < x and x < 10:` `  print('x um dígito menor que 10')`
Paytonicamente	`if 0 < x < 10:` `  print('x um dígito menor que 10')`

Estruturas de repetição

Laço de repetição *while*

- O **while** permite a execução de um bloco de código, desde que a expressão que foi passada como parâmetro seja verdadeira.

```
enquanto(condição) faça
    comandos;
fimenquanto
```

```
while condição:
    comandos
```

Estruturas de repetição

Utilizando o while

Condição de continuidade

```
x = 1
while x < 10:
    print("Número: %s" %x)
    x += 1
print(”O while acabou :)”)
```

Quando não se sabe o momento de parar

```
while True:
    line = input('>')
    if line == 'done':
        break
    print(line)
print('Done!')
```

Estruturas de repetição

Laço de repetição for

- A instrução **for** é perfeita quando trabalha-se com listas no *python*, mas ela também pode ser utilizada da maneira tradicional, indo de um valor inicial para um valor final.

```
para(valor/condicao)
    comandos;
fimpara
```

```
for item in lista:
    print(item)
```

- **Atenção:** O laço for é utilizado para executar um montante fixo. Como por exemplo percorrer uma lista ou executar até uma quantidade de vezes.

Estruturas de repetição

Utilizando o for

Podemos utilizar o for da seguinte forma:

```
fruta = ["Laranja", "Melancia", "Uva"]
for f in fruta:
    print(f)
```

Estrutura de repetição

No for podemos utilizar uma função chamada “range”, ela retornará uma sequência:

```
for i in range(10, 0, -1):
    print(i)
```

Sua sintaxe será a seguinte:

```
for i in range(inicio, fim, incremento)
```

A função range() é uma forma muito mais fácil de gerar uma sequência de números sem ter que utilizar o *while*.

Exercício

- Declare uma lista contendo as notas de 10 alunos;
- Declare uma lista contendo os nomes dos 10 alunos;
- Mostre a média das notas;
- Mostre a maior nota e o nome do respectivo aluno;
- Monte uma lista com alunos aprovados e outra com alunos reprovados (critério para aprovação é nota maior ou igual a 7).

Obrigado

altinobasilio@inf.ufg.br

Dúvidas ou sugestões?

FAPEG
Fundação de Amparo à Pesquisa do Estado de Goiás

INSTITUTO DE INFORMÁTICA
UFG

UFG
UNIVERSIDADE FEDERAL DE GOIÁS