

Python Cheat Sheet

Para iniciantes

by Marco Antonio Sanches - 2023

Geral

- Python é Case Sensitive, ou seja, a variável **nome** é diferente de **Nome**.
- Os índices sempre começam por 0.
- O Python utiliza-se de espaços em branco (tabulação ou espaços) para indentar o código, em substituição ao uso de chaves {}.

Operadores matemáticos

+	Soma	3 + 3 = 6
-	Subtração	4 - 2 = 2
*	Multiplicação	4 * 2 = 8
/	Divisão	22 / 8 = 2.75
**	Exponenciação	2 ** 3 = 8
%	Resto da divisão	10 % 3 = 1
//	Divisão de inteiro	10 // 3 = 3

Saída de dados: função print()

```
>>> print('Olá mundo')
Olá mundo
```

```
>>> idade = 10
>>> print('Minha idade é', idade, 'anos')
Minha idade é 10 anos
```

```
>>> media = 8.75
>>> print(f'Minha média final foi {media}')
Minha média final foi 8.75
```

```
>>> peso = 70
>>> print('Meu peso é %f kg' % peso)
Meu peso é 70 kg
```

Entrada de dados: função input()

```
>>> print('Qual o seu nome?')
>>> nome = input()
>>> print(f'Olá {nome}, como você está?')
Qual o seu nome: Arthur
Olá Arthur, como você está?
```

Operadores relacionais

x == y	igualdade	x igual a y
x > y	maior que	x maior que y
x >= y	maior ou igual a	x maior ou igual a y
x < y	menor que	x menor que y
x <= y	menor ou igual a	x menor ou igual a y
x != y	diferente	x diferente de y

Tipos de dados

int	-2, -1, 0, 1, 2
float	-2.1, -1.5, 0, 1.3, 1.8, 2.8
String	"Hello", "Marco"

- Nas Strings podemos utilizar aspas simples, duplas ou triplas.
- Strings são uma sequência de caracteres e, portanto, devem ser tratadas como qualquer outra sequência.
- Podemos fazer coerção de dados utilizando-se int(), float() ou str().
- A entrada de dados por meio da função input() será sempre uma String. Caso o tipo desejado seja número, você deverá fazer a coerção.

```
>>> num = int(input('Digite um n° inteiro:'))
>>> dobro = num * 2
>>> print(f'O dobro de {num} é {dobro}.')
Digite um n° inteiro: 10
O dobro de 10 é 20.
```

Estrutura de decisão

Sintaxe:

```
>>> if condição: #expressão lógica
#executa o bloco de instruções V
>>> else:
#executa o bloco de instruções F
```

Exemplo:

```
>>> idade = int(input('Digite a idade: '))
>>> if idade >= 18:
print('Você pode ter CNH')
>>> else:
print('Você não pode ter CNH')
```

Digite a idade: 14

Você não pode ter CNH

Operador ternário:

```
>>> n = int('Digite um número:')
>>> resp = 'par' if n%2==0 else 'ímpar'
>>> print(f'O número {n} é {resp}!')
```

Digite um número: 3

O número 3 é ímpar!

Decisão aninhada:

```
>>> if condição1: #expressão lógica
#executa o bloco de instruções V
>>> elif condição2: #expressão lógica
#executa o bloco de instruções V
>>> else:
#executa o bloco de instruções F
```

Laços contados (for)

- Pode iterar sobre os itens de uma sequência (lista ou string):

```
>>> meses = ['jan', 'fev', 'mar', 'abr']
>>> for mes in meses:
>>>     print(mes, end='')
```

jan fev mar abr

- Assim como outras linguagens, também pode iterar sobre sequências numéricas com uso da função **range**:

```
>>> for i in range(10):
>>>     print(i, end=' ')
```

0 1 2 3 4 5 6 7 8 9

Laços condicionais (while)

- Utilizado quando não sabemos exatamente a quantidade de iterações. Neste caso, uma expressão booleana é utilizada para controlar o laço.

```
>> i=0
>> while i < 10:
>>     print(i, end= ' ')
>>     i+=1
0 1 2 3 4 5 6 7 8 9
```

Break x continue

- A instrução **break** oferece a possibilidade de sair do laço mais interno a qualquer momento.

```
>> num=0
>> for num in range(5):
>>     if num == 3:
>>         break #encerra o laço
>>     print(f'Número: {num}')
>> print('Laço encerrado!')
Número: 0
Número: 1
Número: 2
Laço encerrado!
```

- A instrução **continue** pode ser usada para ignorar os comandos e executar a próxima iteração ou passo do laço mais interno.

```
>> num=0
>> for num in range(5):
>>     if num == 3:
>>         continue #pula o laço
>>     print(f'Número: {num}')
>> print('Laço encerrado!')
Número: 0
Número: 1
Número: 2
Número: 4
Laço encerrado!
```

Métodos

- O conceito de método (ou função, ou procedimento) está relacionado à divisão de um problema em diversos subproblemas.
- Um método em Python é definido pela instrução **def**, seguida pelo nome e parêntesis, que pode (ou não) conter a lista de parâmetros (**opcional**).

```
>> def soma(a, b):
>>     return a + b
>> print(soma(4, 5))
9
```

- Uma expressão **lambda** permite escrever métodos anônimos usando apenas uma linha de código.

```
>> soma = lambda a, b: a + b
>> print(soma(4, 5))
9
```

Listas

- Uma lista em Python é uma estrutura que armazena vários dados, que **podem ser de um mesmo tipo ou não**.
- Listas são construções de linguagens de programação que servem para armazenar vários dados de forma simplificada.

```
>> lista1 = [10, 20, 30]
>> lista2 = ['programação', 'olá', 'mundo']
>> lista3 = ['olá', 'mundo', 2023]
```

- A utilização de uma lista está associada a uma estrutura de repetição.

```
>> livros = ['Java', 'Python', 'C++']
>> for livro in livros:
>>     print(livro)
Java
Python
C++
```

Principais métodos usados com Listas

- **append(item)**: adiciona um item ao final da lista.

```
>> livros.append('Sql')
>> livros
['Java', 'Python', 'C++', 'Sql']
```

- **insert(pos, item)**: insere um novo item na posição desejada.

```
>> livros.insert(0, 'Adroid')
>> livros
['Adroid', 'Java', 'Python', 'C++', 'Sql']
```

- **pop()**: remove o último item de uma lista.

```
>> livros.pop()
>> livros
['Adroid', 'Java', 'Python', 'C++']
```

- **pop(pos)**: remove o item na posição desejada.

```
>> livros.pop(0)
>> livros
['Java', 'Python', 'C++']
```

- **remove(item)**: remove a primeira ocorrência de um item.

```
>> livros.remove(0, 'C++')
>> livros
['Java', 'Python']
```

- **count(item)**: retorna o número de ocorrências de um item.

```
>> livros.count('Java')
1
```

- **reverse()**: inverte a posição dos itens da lista.

```
>> livros.reverse()
>> livros
['Python', 'Java']
```

- **sort()**: ordena a lista.

```
>> livros.sort()
>> livros
['Java', 'Python']
```

- **index(item)**: retorna a posição da primeira ocorrência de um item.

```
>> livros.index('Java')
0
```

Funções Matemáticas

- math** é o módulo do Python que reúne as funções matemáticas.
- É utilizado somente para números não complexos.
 - Para utilizá-lo, devemos fazer a importação da biblioteca **math**: `import math`
 - Algumas das principais funções são:

Método	Descrição
<code>sqrt(x)</code>	Retorna a raiz quadrada de x.
<code>pow(x, y)</code>	Retorna x elevado a y (<code>x**y</code>)
<code>sin(x)</code>	Retorna o seno de x.
<code>cos(x)</code>	Retorna o cosseno de x.
<code>tan(x)</code>	Retorna a tangente de x.
<code>asin(x)</code>	Retorna o arco seno de x.
<code>acos(x)</code>	Retorna o arco cosseno de x.
<code>atan(x)</code>	Retorna o arco tangente de x.
<code>radians(x)</code>	Converte o ângulo x de graus para radianos.
<code>floor(x)</code>	Retorna o maior número inteiro menor ou igual a x.
<code>fabs(x)</code>	Retorna o valor absoluto de x.
<code>ceil(x)</code>	Retorna o menor número inteiro maior ou igual a x.
<code>factorial(x)</code>	Retorna o fatorial de x.

- Lembre-se que as funções trigonométricas trabalham com ângulos em radianos.
- Caso você precisar trabalhar com ângulos em graus, use as funções **degrees()** e **radians()** para converter entre as unidades de medida.

Veja alguns exemplos:

```
>>> math.sqrt(25)
5.0
>>> math.pow(2,3)
8.0
>>> math.sin(math.radians(60))
0.5
>>> math.cos(math.pi)
-1.0
```

```
>>> math.tan(math.pi/4)
0.9999999999999999
>>> math.floor(2.8)
2
>>> math.ceil(2.1)
3
>>> math.fabs(-4)
4.0
```