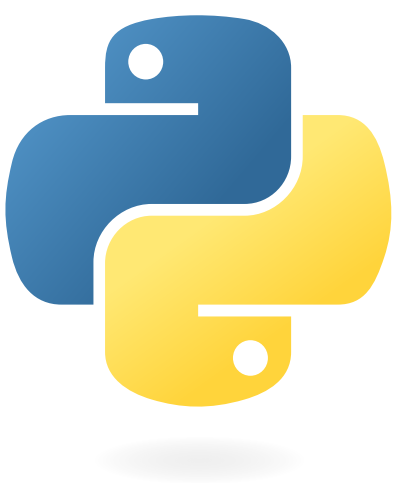
Introdução à Programação em



python™

https://evandro-crr.github.io/intro-python

Evandro Chagas Ribeiro da Rosa

Toda Variável Possui um Tipo

Verificar o Tipo

```
>>> mensagem = "Olá"
>>> print(type(mensagem))
<class 'str'>
>>> inteiro = 10
>>> print(type(inteiro))
<class 'int'>
>>> real = 2.50
>>> print(type(real))
<class 'float'>
```

Conversão de Tipos

```
>>> numero = "13.50"
>>> print(type(numero))
<class 'str'>
>>> numero = float(numero)
>>> print(type(numero))
<class 'float'>
>>> numero = int(numero)
>>> print(type(numero))
<class 'int'>
```

Recebendo Valores do Terminal

```
>>> entrada = input("Por favor, digite um número: ")
Por favor, digite um número: 50
>>> print(entrada)
50
>>> print(type(entrada))
<class 'str'>
```

- A função `input()` sempre retorna uma string.
- É necessário converter o valor para o tipo adequado.
- Cuidado com erros de conversão ⚠️

```
>>> numero = int("trinta")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: 'trinta'
```

Lista em Python

Coleção de valores em uma ordem específica

```
>>> animais = ["gato", "cavalo", "lagarto"]
>>> print(animais)
['gato', 'cavalo', 'lagarto']
```

Acessando Valores

```
>>> print(animais)
['gato', 'cavalo', 'lagarto']
>>> print(animais[0])
gato
>>> print(animais[1])
cavalo
>>> print(animais[2])
lagarto
>>> print(animais[-1])
lagarto
```

```
>>> print(type(animais))
<class 'list'>
>>> print(type(animais[0]))
<class 'str'>
```

```
>>> ns = [40, 25, 12]
>>> print("Média", (ns[0] + ns[1] + ns[2]) / 3)
Média 25.666666666666668
```

Modificando Elementos da Lista

```
>>> lista = [1, 2, 3, 4, 5]
>>> lista[2] = 33
>>> print(lista)
[1, 2, 33, 4, 5]
>>> print(lista[2])
33
>>> lista[-1] = 55
>>> print(lista)
[1, 2, 33, 4, 55]
```

Adicionando Elementos à Lista

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

Adicionando ao Final

`.append(valor)`

```
>>> lista.append(6)
>>> print(lista)
[1, 2, 3, 4, 5, 6]
```

Inserindo na Posição

`.insert(idx, valor)`

```
>>> print(lista[3])
4
>>> lista.insert(3, 10)
>>> print(lista[3])
10
>>> print(lista)
[1, 2, 3, 10, 4, 5]
```

- O tamanho da lista é dinâmico e cresce conforme o necessário.
- Inserir elementos na lista atualiza os índices dos valores.

Removendo Elementos da Lista

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

Removendo do Final

`.pop()`

```
>>> valor = lista.pop()
>>> print(lista)
[1, 2, 3, 4]
>>> print(valor)
5
```

Removendo na Posição

`del`

```
>>> lista = [1, 2, 3, 4, 5]
>>> del lista[2]
>>> print(lista)
[1, 2, 4, 5]
```

Removendo por Valor

```
>>> lista = [1, 2, 3, 4, 5]
>>> lista.remove(4)
>>> print(lista)
[1, 2, 3, 5]
```

Funções Úteis para Lista

```
lista = [20, 16, 26, 98, 45, 6, 17, 27, 98, 87]
```

- `len(lista)` - Retorna o tamanho da lista
- `.sort()` - Ordena os valores da lista
- `.reverse()` - Inverte a ordem dos valores da lista
- `sum(lista)` - Soma todos os valores da lista
- `min(lista)` - Retorna o menor valor da lista
- `max(lista)` - Retorna o maior valor da lista

⚙ **Exercício**

Em um arquivo `.py`, teste cada um desses comandos.

O Tipo `range`

Sequência imutável de números

- O range se comporta como um objeto do tipo `list`.
- É possível construir uma lista de números com `range`.

```
>>> de_0_a_9 = list(range(10))
>>> print(de_0_a_9)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> de_5_a_15 = list(range(5, 16))
>>> print(de_5_a_15)
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
>>> pares_de_4_a_24 = list(range(4, 25, 2))
>>> print(pares_de_4_a_24)
[4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24]
```

O Laço de Repetição `for`

Iterar sobre elementos de um conjunto

```
>>> for i in range(10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
```

```
for variável in lista:
    <corpo>
```

- O corpo do `for` é repetido várias vezes.
- A cada iteração, a `variável` definida no `for` assume um valor da `lista`.
- A indentação é necessária ⚠️

Trabalhando com Listas

```
>>> quadrados = []
>>> for numero in range(1, 11):
...     quadrados.append(numero**2)
...
>>> print(quadrados)
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
>>> quadrados = list(range(1, 11))
>>> print(quadrados)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> for i in range(len(quadrados)):
...     quadrados[i] *= quadrados[i]
...
>>> print(quadrados)
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Compreensão de Lista

```
>>> quadrados = [valor**2 for valor in range(1, 11)]
>>> print(quadrados)
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

⚙ Exercícios (1/2)

1. Escreva um laço `for` que imprima os números de 1 a 20.
2. Crie uma lista contendo os números de 1 até 1.000.000. Use um loop `for` para imprimir cada número.[1]
3. Verifique se a lista começa em 1 e termina em 1.000.000 usando as funções `min()` e `max()`. Em seguida, use a função `sum()` para calcular a soma de todos os números da lista, depois, calcule a media dos valores da lista.

Se a saída demorar muito, você pode interromper o processo pressionando `CTRL-C` ou fechando o terminal.

⚙ Exercícios (2/2)

1. Gere uma lista com todos os números ímpares entre 1 e 20 usando `range()` com um valor de passo apropriado. Use um loop `for` para imprimir cada número.
2. Crie uma lista com os múltiplos de 3 entre 3 e 30. Use um loop `for` para imprimir cada múltiplo.
3. Um número elevado ao cubo é chamado de cubo. Por exemplo, `2**3` representa o cubo de 2. Crie uma lista com os cubos dos 10 primeiros números inteiros (de 1 a 10) e use um loop `for` para imprimir cada cubo.
4. Use uma compreensão de lista para gerar uma lista com os cubos dos 10 primeiros números inteiros.

Fatiando Listas

Trabalhar com parte de uma lista

```
>>> numeros = list(range(100))
>>> inicio = 30
>>> fim = 40
>>> sublista = numeros[inicio:fim]
>>> print(sublista)
[30, 31, 32, 33, 34, 35, 36, 37, 38, 39]
>>> print(sublista[:4])
[30, 31, 32, 33]
>>> print(sublista[5:])
[35, 36, 37, 38, 39]
```

- Podemos usar `[inicio:fim]` para pegar um pedaço da lista.
- Se o início for omitido, por padrão é zero.
- Se o fim for omitido, por padrão é o final da lista.

Copiando Listas

```
>>> a = list(range(10))
>>> print(a)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> b = a
>>> print(b)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> b.append(10)
>>> print(b)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> print(a)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> b = a[:]
>>> print(b)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> b.append(11)
>>> print(b)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
>>> print(a)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

- Quando uma lista é atribuída a uma variável, é criada uma referência.
- Para copiar os valores de uma lista, podemos usar `[:]`.

Tuplas

Sequência imutável de elementos

```
>>> tupla = (1, 2, 3, 4, 5)
>>> print(tupla)
(1, 2, 3, 4, 5)
>>> print(type(tupla))
<class 'tuple'>
>>> print(tupla[1])
2
>>> tupla[1] = 10
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Operador `in`

Verifica a presença de um elemento em uma sequência (listas, tuplas, strings).

```
>>> lista = [1, 2, 3, 4, 5]
>>> print(3 in lista)
True
>>> print(10 in lista)
False
>>> mensagem = "Olá, mundo!"
>>> print("Olá" in mensagem)
True
```

Função `enumerate()`

Retorna um iterador de pares de índice e valor de uma sequência.

```
>>> lista = ["a", "b", "c"]
>>> for index, valor in enumerate(lista):
...     print(f"O índice de {valor} é {index}")
...
O índice de a é 0
O índice de b é 1
O índice de c é 2
```

```
>>> quadrados = list(range(1, 11))
>>> print(quadrados)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> for i, valor in enumerate(quadrados):
...     quadrados[i] = valor**2
...
>>> print(quadrados)
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Função `zip()`

Junta elementos de duas ou mais sequências.

```
>>> nomes = ["Ana", "João", "Maria"]
>>> idades = [28, 34, 23]
>>> for nome, idade in zip(nomes, idades):
...     print(f"{nome} tem {idade} anos")
...
Ana tem 28 anos
João tem 34 anos
Maria tem 23 anos
```

Listas Heterogêneas

Listas que podem conter elementos de diferentes tipos.

```
>>> lista = [1, "texto", 3.14, True]
>>> print(lista)
[1, 'texto', 3.14, True]
>>> tipos = [type(item) for item in lista]
>>> print(tipos)
[<class 'int'>, <class 'str'>, <class 'float'>, <class 'bool'>]
```

Listas Compostas

Listas que contêm outras listas

```
>>> matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> print(matriz)
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> print(matriz[0][1])
2
```

Separando Strings

Juntando Strings

```
nome_completo = "João Carlos da Costa Silva"
```

.split()

```
>>> nomes = nome_completo.split()
>>> print(nomes)
['João', 'Carlos', 'da', 'Costa', 'Silva']
>>> print("O sobrenome é", nomes[-1])
O sobrenome é Silva
```

" ".join(lista)

```
>>> nome = " ".join(nomes[:2])
>>> print(nome)
João Carlos
```

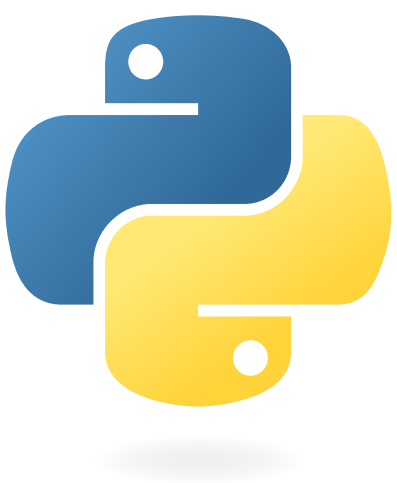
⚙ Exercícios

Construa um programa que leia uma lista de números usando `input()`. Por exemplo:

```
>>> input("Forneça diversos números separados por espaço: ")
Forneça diversos números separados por espaço: 10 4 44 12 75 96
```

- Some todos os números.
- Encontre o maior e o menor valor.
- Calcule a média desconsiderando o maior e o menor valor.
- Imprima todos os resultados na tela.

Introdução à Programação em



python™

https://evandro-crr.github.io/intro-python

Evandro Chagas Ribeiro da Rosa