Estruturas de Controle de Fluxo

Introdução a Programação

Listas

Python possui alguns tipos de dados **compostos**, ou seja que podem agrupar outros valores. O mais versátil é o tipo lista (*list*)

Listas

- Além dos tipos básicos (`int` , `float` , `str` , etc) existem os tipos de dados compostos
  - `dictionary`
  - `sets`
  - `lists`
- Para definir uma lista basta utilizar `[` e `]` como delimitares e separar os valores com `,`
- Exemplo:

```
1  squares = [1, 4, 9, 16, 25]
```

Listas

- Uma única lista pode conter ítens de tipos distintos
- Exemplos

```
1  spam = ['bacon', 'eggs', 42]
2  l = [1, 3.14, ['a', 'b', 'c']]
```

Indexação

- As listas permitem o acesso a um elemento individual através da **indexação**
- Exemplo:

```
1  squares = [1, 4, 9, 16, 25]
2  squares[0] # retorna o primeiro elemento
3  squares[1] # retorna o segundo elemento
4  squares[-1] # retorna o ultimo elemento
```

`Strings` e Listas

- As `strings` são um tipo especial de lista
- Também permitem a indexação
- Exemplos:

```
1  my_text = 'hello'
2  my_text[0] # retorna h
3  my_text[1] # retorna e
4  my_text[-1] # retorna o
```

- As `strings` são imutáveis (*immutable*), as listas são mutáveis (*mutable*)

Tipos Mutáveis x Imutáveis

- Exemplo de tipo de mutável

```
1  cubes = [1, 8, 27, 65, 125] # como corrigir o quarto elem
2  cubes[3] = 64
```

- Exemplo de tipo imutável

```
1  my_str = 'hellu' # como corrigir o ultimo elemento?
2  my_str[-1] = 'o' # qual o erro apresentado ?
```

Concatenação de Listas

```
1  squares = [1, 4, 9, 16, 25]
2  squares + [36, 49, 64, 81, 100]
```

Métodos para Listas

```
1  squares = [1, 4, 9, 16, 25]
2  squares.append(36) # adiciona novo item ao final da lista
3  squares.pop() # remove o ultimo elemento da lista
4  squares.clear() # remove todos os itens da lista
5  squares.count() # retorna a quantidade de itens da lista
```

Controle de Fluxo

Estruturas de controle de fluxo permitem que partes do código sejam executadas repetidas vezes ou não, de acordo com condições especificadas pelo programador.

`for`

- A estrutura `for` **itera** sobre uma lista ou *string* realizando operações na mesma sequência em que os itens estão ordenados
- Exemplo:

```
1  words = ['gato', 'cachorro', 'janela']
2  for w in words:
3      print(w, len(w))
```

`range()`

- Caso não tenha sido definido uma lista, pode-se utilizar uma sequência de números simples com `range()`
- Exemplos:

```
for i in range(5):
    print(i)
```

```
squares = []
for i in range(5):
    squares.append(i**2)
```

```
list(range(5, 10))
list(range(0, 10, 3))
list(range(-10, -100, -30))
```

`while`

- A estrutura `while` executa comandos em ciclicamente (em *loop*)enquanto a condição estabelecida for válida (`true`)
- Exemplo:

```
1  i = 1
2  while i < 6:
3    print(i)
4    i += 1
```

`while` com `break`

- A declaração `break` finaliza a execução laço (*loop*)
- Exemplo:

```
1  i = 1
2  while i < 6:
3    print(i)
4    if i == 3:
5      break
6    i += 1
```

`while` com `continue`

- A declaração `continue` finaliza a iteração atual e dá continuidade a execução laço na próxima iteração
- Exemplo:

```
1  i = 0
2  while i < 6:
3    i += 1
4    if i == 3:
5      continue
6    print(i)
```

`while` com `else`

- A declaração `else` permite que um bloco de código seja executado quando a condição for falsa (`False`)
- Exemplo:

```
1  i = 1
2  while i < 6:
3    print(i)
4    i += 1
5  else:
6    print("i não vale mais 6")
```

Exemplo 1

Escrever um *script* em Python que recebe um número inteiro positivo (`n`) do usuário e exibe o quadrado dos números de 1 até `n`.

Solução

```
1  n = input('Digite um número inteiro positivo:')
2  if n.isdigit():
3      n = int(n)
4      for i in range(n+1):
5          print(f'O quadrado de {i} vale {i**2}.')
6  else:
7      print('Número digitado não é inteiro positivo')
```

Solução

```
1  n = input('Digite um número inteiro positivo:')
2  if n.isdigit():
3      n = int(n)
4      for i in range(n+1):
5          print(f'O quadrado de {i} vale {i**2}.')
6  else:
7      print('Número digitado não é inteiro positivo')
```

Solução

```
1  n = input('Digite um número inteiro positivo:')
2  if n.isdigit():
3      n = int(n)
4      for i in range(n+1):
5          print(f'O quadrado de {i} vale {i**2}.')
6  else:
7      print('Número digitado não é inteiro positivo')
```

Solução

```
1  n = input('Digite um número inteiro positivo:')
2  if n.isdigit():
3      n = int(n)
4      for i in range(n+1):
5          print(f'O quadrado de {i} vale {i**2}.')
6  else:
7      print('Número digitado não é inteiro positivo')
```

Solução

```
1  n = input('Digite um número inteiro positivo:')
2  if n.isdigit():
3      n = int(n)
4      for i in range(n+1):
5          print(f'O quadrado de {i} vale {i**2}.')
6  else:
7      print('Número digitado não é inteiro positivo')
```

Solução

```
1  n = input('Digite um número inteiro positivo:')
2  if n.isdigit():
3      n = int(n)
4      for i in range(n+1):
5          print(f'O quadrado de {i} vale {i**2}.')
6  else:
7      print('Número digitado não é inteiro positivo')
```

Exemplo 2

Refazer o exercício anterior com as seguintes modificações:

- Utilizar `while` ao invés de `for`
- Ao invés de exibir os valores na saída, guardar numa lista (`l`)
- Exibir a lista ao final do *script*

Solução

```
1  n = input('Digite um número inteiro positivo:')
2  if n.isdigit():
3      n = int(n)
4      i = 1
5      l = []
6      while i ⩽ n:
7          l.append(i**2)
8          i += 1
9      print(l)
10 else:
11     print('Número digitado não é inteiro positivo')
```

Solução

```
1  n = input('Digite um número inteiro positivo:')
2  if n.isdigit():
3      n = int(n)
4      i = 1
5      l = []
6      while i ⩽ n:
7          l.append(i**2)
8          i += 1
9      print(l)
10 else:
11     print('Número digitado não é inteiro positivo')
```

Solução

```
1  n = input('Digite um número inteiro positivo:')
2  if n.isdigit():
3      n = int(n)
4      i = 1
5      l = []
6      while i ⩽ n:
7          l.append(i**2)
8          i += 1
9      print(l)
10 else:
11     print('Número digitado não é inteiro positivo')
```

Solução

```
1  n = input('Digite um número inteiro positivo:')
2  if n.isdigit():
3      n = int(n)
4      i = 1
5      l = []
6      while i ⩽ n:
7          l.append(i**2)
8          i += 1
9      print(l)
10 else:
11     print('Número digitado não é inteiro positivo')
```

Solução

```
1  n = input('Digite um número inteiro positivo:')
2  if n.isdigit():
3      n = int(n)
4      i = 1
5      l = []
6      while i ⩽ n:
7          l.append(i**2)
8          i += 1
9      print(l)
10 else:
11     print('Número digitado não é inteiro positivo')
```

Solução

```
n = input('Digite um número inteiro positivo:')
if n.isdigit():
    n = int(n)
    i = 1
    l = []
    while i ⩽ n:
        l.append(i**2)
        i += 1
    print(l)
else:
    print('Número digitado não é inteiro positivo')
```

Solução

```
1  n = input('Digite um número inteiro positivo:')
2  if n.isdigit():
3      n = int(n)
4      i = 1
5      l = []
6      while i ⩽ n:
7          l.append(i**2)
8          i += 1
9      print(l)
10 else:
11     print('Número digitado não é inteiro positivo')
```

Exercício 1

Escrever um *script* em Python que exiba a tabuada de Soma ou Multiplicação, conforme opção do usuário. Além da operação, o usuário também deverá escolher qual o número da tabuada. A saída deve similar a mostrada a seguir:

- 1 + 1 = 2
- 1 + 2 = 3 ...
- 1 + 10 = 11

Incluir título e mensagem de encerramento. Verificar se números inteiros positivos são digitados pelo usuário. Caso contrário, mensagem de erro ou opção de digitar novamente.

Referências

- Python.ORG
- W3 Schools