



Universidade Federal de Viçosa
Campus Rio Paranaíba
Instituto de Ciências Exatas e Tecnológicas

SIN 141

Computação

Orientada a Objetos

Sistemas de Informação
Alan Diêgo Aurélio Carneiro
alan.carneiro@ufv.br



Universidade Federal de Viçosa
Campus Rio Paranaíba
Instituto de Ciências Exatas e Tecnológicas

Aula de Hoje

Introdução à linguagem Python

Strings

Todo caractere de uma string é indexado, começando do primeiro caractere (índice 0) à esquerda.

Notação: string[indice]

Exemplo: var = "Pedro dos Santos"

	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
<u>Índice</u>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Var	P	e	d	r	o		d	o	s		S	a	n	t	o	s



Strings

Representação: `s = "12346"` ou `s = '123456'`

len(s): retorna o tamanho de uma string.

Operador +: concatena strings. Ex: `'ab' + 'cd' = 'abcd'`

Operador *: repete strings. Ex: `'a'*5 = 'aaaaa'`

Fatias (Slices): `[start:end:step]` # o que `[::-1]` faz?

lower(): retorna a string com todos os caracteres convertidos para minúsculos.

upper(): retorna a string com todos os caracteres convertidos para maiúsculos.

Listas

Tipo de dados mais versátil do Python.

Uma lista é representada como uma sequência de valores entre colchetes e separados por vírgula.

Os elementos de uma lista podem ser de tipos de dados diferentes.

Listas são **mutáveis** !!!

Exemplo

```
1 >>> lista1 = [ 'calculo ', 'fisica ', 'computacao ' ]
2 >>> lista2 = [ 'notas ', 5.4, 'aprovado ' ]
3 >>> lista2[1] = 6
4 >>> lista2
5 [ 'notas ', 6, 'aprovado ' ]
```

Listas

Tipo de dados mais versátil do Python.

Uma lista é representada como uma sequência de valores entre colchetes e separados por vírgula.

Os elementos de uma lista podem ser de tipos de dados diferentes.

Listas são **mutáveis** !!!

Exemplo

```
1 >>> lista1 = [ 'calculo ', 'fisica ', 'computacao ' ]
2 >>> lista2 = [ 'notas ', 5.4, 'aprovado ' ]
3 >>> lista2[1] = 6
4 >>> lista2
5 [ 'notas ', 6, 'aprovado ' ]
```

Listas

```
>>> c=[-45, 6, 3, 0, 1, 19, 32, -23, 12, 5, -3, 8, 2]
>>> c[3]
0
>>> c[9]==c[-4]
True
>>> len(c)
13
```

```
>>> [1,2] + [3]
[1, 2, 3]
>>> [1,2] + [[3]]
[1, 2, [3]]
>>> [[1,2]] + [[3]]
[[1, 2], [3]]
>>> [1,2] * 3
[1, 2, 1, 2, 1, 2]
```

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Listas

A função `range(...)` pode ter 1, 2 ou 3 argumentos:

- `range(numero)`: retorna uma lista contendo uma sequência de valores de `0` a `numero-1`

```
1 >>> list(range(10))
2      [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- `range(inf,sup)`: retorna uma lista contendo uma sequência de valores de `inf` a `sup-1`

```
1 >>> list(range(3, 8))
2      [3, 4, 5, 6, 7]
```

- `range(inf, sup, inc)`: retorna uma lista contendo uma sequência de valores de `inf` a `sup-1` com incremento de `inc`

```
1 >>> list(range(3, 8, 2))
2      [3, 5, 7]
```


Listas

Podemos usar a notação de fatias (slices) em listas:

- `[start : end]` : vai do índice start até o índice end
- `[start :]` : vai de start até o final da lista
- `[: end]` : vai do início da lista até end
- `[:]` : copia a lista toda

	C[0]	→	-45
C[-12]	C[1]	→	6
C[-11]	C[2]	→	3
C[-10]	C[3]	→	0
C[-9]	C[4]	→	1
C[-8]	C[5]	→	19
C[-7]	C[6]	→	32
C[-6]	C[7]	→	-23
C[-5]	C[8]	→	12
C[-4]	C[9]	→	5
C[-3]	C[10]	→	-3
C[-2]	C[11]	→	8
C[-1]	C[12]	→	2

```
1 >>> c = [-45,6,3,0,1,19,32,-23,12,5,-3,8,2]
2
3 >>> c[-1:-5:1]
4
5
6 >>> c[-5:-1:1]
```

Listas

Além dos operadores + (concatenação) e * (usado para múltiplas concatenações) podemos manipular listas usando:

append: outra forma de concatenação. Neste caso, a lista é tratada como uma fila.

extend: permite adicionar os elementos de uma lista a outra.

del : remover elemento de uma lista.

Listas

```
>>> lista = []
>>> list.append(lista, 'a')
>>> lista
['a']

>>> list.append(lista, 2)
>>> lista
['a', 2]

>>> list.append(lista, [3, 'f'])
>>> lista
['a', 2, [3, 'f']]
```

Listas

- `list.insert(lista, índice, elemento)`: insere *elemento* na lista na posição indicada por *índice*.

```
1 >>> lista = [0,1,2,3]
2
3 >>> list.insert(lista,1,'dois')
4
5 >>> lista
6 [0,'dois', 1, 2, 3]
```

- Como o *extend*, altera a lista ao invés de retornar a lista. O valor retornado é **None**!
- Atribuições a fatias servem para a mesma finalidade mas são menos legíveis.

```
1 >>> lista = [0,1,2,3]
2
3 >>> lista [1:1] = ['dois']
4
5 >>> lista
6 [0,'dois', 1, 2, 3]
```

Listas

- **list.remove(lista, elemento):** Remove da lista o primeiro elemento igual a **elemento**. Se não existe tal elemento, um erro é gerado.

```
>>> lista = ['oi', 'alo', 'ola']
```

```
>>> list.remove(lista, 'alo')
```

```
>>> lista  
['oi', 'ola']
```

```
>>> list.remove(lista, 'oba')
```

```
Traceback (most recent call last):
```

```
File "<pyshell#116>", line 1, in <module>
```

```
list.remove(lista, "oba")
```

```
ValueError: list.remove(x): x not in list
```

Listas

- **list.pop(lista, índice)**: Remove da lista o elemento na posição **índice** e o retorna. Se **índice** não for mencionado, é assumido o último.

```
>>> lista = [1,2,3,4]
>>> list.pop(lista)
4
>>> lista
[1,2,3]
>>> deletado = list.pop(lista,1)
>>> deletado
2
>>> lista
[1,3]
```

A diferença entre **del** e **pop** é que este retorna o elemento deletado, enquanto o **del** não.

Listas

- **list.count(lista, elemento)**: Retorna quantas vezes o elemento aparece na lista.

```
1 >>> lista = [9,8,33,12,33]
2 >>> list.count(lista,33)
3 2
```

- **list.index(elemento)**: Retorna o índice da **primeira** ocorrência de elemento na lista. Um erro ocorre se elemento não consta da lista.

```
1 >>> list.index(lista, 33)
2 2
3 >>> list.index(lista, 7)
4 Traceback (most recent call last):
5   File "<pyshell#110>", line 1, in <module>
6     lista.index(7)
7   ValueError: 7 is not in list
```

Listas

- **OBSERVAÇÃO:** Usar o index para saber se o elemento está numa lista não é uma boa idéia, porque se não estiver, dará erro.
- Uma forma de saber se um elemento está numa lista é usar o **in**, conforme exemplificado abaixo:

```
1 >>> lista = [1,4,8,3,2]
2 >>> 2 in lista
3 True
4
5 >>> 10 in lista
6 False
```


Listas

ATENÇÃO

Algumas funções que manipulam listas não possuem valor de retorno:

- `list.append`
- `list.extend`
- `list.insert`
- `list.remove`
- `list.reverse`
- `list.sort`

Enquanto outras possuem:

- `list.pop`
- `list.count`
- `list.index`

Estrutura de Repetição *while*

Permite que o programador especifique que a função deve repetir um conjunto de comandos **enquanto** uma dada condição for verdadeira.

```
while <condição>:  
    <sequência de comandos>
```

Exemplo

```
1 def exemplo1(numero):  
2     """ Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e  
3         igual a numero.  
4         Parametro de entrada: int  
5         Valor de Retorno: list """  
6     listax = []  
7     while numero > 0:  
8         list.append(listax, 'x')  
9         numero = numero - 1  
10    return listax
```

Estrutura de Repetição *while*

```
1 def exemplo1(numero):
2     """Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e
   igual a numero.
3     Parametro de entrada: int
4     Valor de Retorno: list"""
5
6     listax = []
7     while numero > 0:
8         list.append(listax, 'x')
9         numero = numero - 1
10    return listax
```

- exemplo1(2)

Estrutura de Repetição *while*

```
1 def exemplo1(numero):
2     """Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e
3         igual a numero.
4     Parametro de entrada: int
5     Valor de Retorno: list"""
6
7     listax = []
8     while numero > 0:
9         list.append(listax, 'x')
10        numero = numero - 1
11    return listax
```

- exemplo1(2)
- listax=[]

Estrutura de Repetição *while*

```
1 def exemplo1(numero):
2     """Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e
        igual a numero.
3     Parametro de entrada: int
4     Valor de Retorno: list"""
5
6     listax = []
7     while numero > 0:
8         list.append(listax, 'x')
9         numero = numero - 1
10    return listax
```

- exemplo1(2)
- listax=[]
- 2 > 0

Estrutura de Repetição *while*

```
1 def exemplo1(numero):
2     """Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e
   igual a numero.
3     Parametro de entrada: int
4     Valor de Retorno: list"""
5
6     listax = []
7     while numero > 0:
8         list.append(listax, 'x')
9         numero = numero - 1
10    return listax
```

- exemplo1(2)
- listax=[]
- $2 > 0 \Rightarrow \text{True}$

Estrutura de Repetição *while*

```
1 def exemplo1(numero):
2     """Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e
   igual a numero.
3     Parametro de entrada: int
4     Valor de Retorno: list"""
5
6     listax = []
7     while numero > 0:
8         list.append(listax, 'x')
9         numero = numero - 1
10    return listax
```

- exemplo1(2)
- listax=[]
- $2 > 0 \Rightarrow \text{True}$
 - listax=['x']

Estrutura de Repetição *while*

```
1 def exemplo1(numero):
2     """Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e
3         igual a numero.
4     Parametro de entrada: int
5     Valor de Retorno: list"""
6
7     listax = []
8     while numero > 0:
9         listax.append('x')
10        numero = numero - 1
11    return listax
```

- exemplo1(2)
- listax=[]
- $2 > 0 \Rightarrow \text{True}$
 - listax=['x']
 - numero = 1

Estrutura de Repetição *while*

```
1 def exemplo1(numero):
2     """Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e
3         igual a numero.
4     Parametro de entrada: int
5     Valor de Retorno: list"""
6
7     listax = []
8     while numero > 0:
9         list.append(listax, 'x')
10        numero = numero - 1
11    return listax
```

- exemplo1(2)
- listax=[]
- $2 > 0 \Rightarrow \text{True}$
 - listax=['x']
 - numero = 1
- $1 > 0$

Estrutura de Repetição *while*

```
1 def exemplo1(numero):
2     """Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e
3         igual a numero.
4     Parametro de entrada: int
5     Valor de Retorno: list"""
6
7     listax = []
8     while numero > 0:
9         list.append(listax, 'x')
10        numero = numero - 1
11    return listax
```

- `exemplo1(2)`
- `listax=[]`
- `2 > 0 ⇒ True`
 - `listax=['x']`
 - `numero = 1`
- `1 > 0 ⇒ True`

Estrutura de Repetição *while*

```
1 def exemplo1(numero):
2     """Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e
3         igual a numero.
4     Parametro de entrada: int
5     Valor de Retorno: list"""
6
7     listax = []
8     while numero > 0:
9         list.append(listax, 'x')
10        numero = numero - 1
11    return listax
```

- exemplo1(2)
- listax=[]
- 2 > 0 ⇒ True
 - listax=['x']
 - numero = 1
- 1 > 0 ⇒ True
 - listax=['x','x']

Estrutura de Repetição *while*

```
1 def exemplo1(numero):
2     """Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e
3         igual a numero.
4     Parametro de entrada: int
5     Valor de Retorno: list"""
6
7     listax = []
8     while numero > 0:
9         list.append(listax, 'x')
10        numero = numero - 1
11    return listax
```

- `exemplo1(2)`
- `listax=[]`
- `2 > 0 ⇒ True`
 - `listax=['x']`
 - `numero = 1`
- `1 > 0 ⇒ True`
 - `listax=['x','x']`
 - `numero = 0`

Estrutura de Repetição *while*

```
1 def exemplo1(numero):
2     """Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e
3         igual a numero.
4     Parametro de entrada: int
5     Valor de Retorno: list"""
6
7     listax = []
8     while numero > 0:
9         list.append(listax, 'x')
10        numero = numero - 1
11    return listax
```

- `exemplo1(2)`
- `listax=[]`
- `2 > 0 ⇒ True`
 - `listax=['x']`
 - `numero = 1`
- `1 > 0 ⇒ True`
 - `listax=['x','x']`
 - `numero = 0`
- `0 > 0`

Estrutura de Repetição *while*

```
1 def exemplo1(numero):
2     """Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e
   igual a numero.
3     Parametro de entrada: int
4     Valor de Retorno: list"""
5
6     listax = []
7     while numero > 0:
8         list.append(listax, 'x')
9         numero = numero - 1
10    return listax
```

- `exemplo1(2)`
- `listax=[]`
- `2 > 0 ⇒ True`
 - `listax=['x']`
 - `numero = 1`
- `1 > 0 ⇒ True`
 - `listax=['x','x']`
 - `numero = 0`
- `0 > 0 ⇒ False`

Estrutura de Repetição *while*

```
1 def exemplo1(numero):
2     """Funcao que cria uma lista formada por strings 'x'. A quantidade de strings 'x' e
3         igual a numero.
4     Parametro de entrada: int
5     Valor de Retorno: list"""
6
7     listax = []
8     while numero > 0:
9         list.append(listax, 'x')
10        numero = numero - 1
11    return listax
```

- `exemplo1(2)`
- `listax=[]`
- `2 > 0 ⇒ True`
 - `listax=['x']`
 - `numero = 1`
- `1 > 0 ⇒ True`
 - `listax=['x','x']`
 - `numero = 0`
- `0 > 0 ⇒ False`

`return ['x','x']`

Estrutura de Repetição *while*

```
while <condição>:  
    <sequência de comandos>
```

- A <**condição**> é uma expressão ou dado do tipo booleano (**True** ou **False**), tal como os testes usados com o comando **IF**.
- Estrutura também conhecida como **laço de repetição** ou “**loop**”: o bloco de comandos é sequencialmente repetido tantas vezes quanto o teste da condição for verdadeiro.
- Somente quando a condição se torna falsa a próxima instrução após o bloco de comandos associado ao **while** é executada (fim do laço).

Estrutura de Repetição *while*

```
while <condição>:  
    <sequência de comandos>
```

- Se a <condição> da estrutura while já for falsa desde o início, o bloco de <sequência de comandos> associado a ela nunca é executado.
- Deve haver algum processo dentro do bloco de <sequência de comandos> que torne a **condição** falsa e a repetição seja encerrada, ou um erro GRAVE ocorrerá: sua função ficará rodando para sempre!!

Estrutura de Repetição *while*

```
1 def exemplowhile0(numero):
2
3     """ Parametro de entrada: int
4     Valor de Retorno: list """
5
6     listanum = [ ]
7     while numero > 0:
8         listanum[numero-1] = numero
9         numero = numero -1
10    return listanum
```

Qual o problema desta função?

Estrutura de Repetição *while*

```
1 def exemplowhile0(numero):
2
3     """ Parametro de entrada: int
4     Valor de Retorno: list """
5
6     listanum = [ ]
7     while numero > 0:
8         listanum[numero-1] = numero
9         numero = numero -1
10    return listanum
```

Qual o problema desta função?

```
1 def exemplowhile1(numero):
2
3     """ Parametro de entrada: int
4     Valor de Retorno: list """
5
6     listanum = numero * [0]
7     while numero > 0:
8         listanum[numero-1] = numero
9         numero = numero -1
10    return listanum
```

Estrutura de Repetição *while*

```
1 def exemplowhile0(numero):
2
3     """ Parametro de entrada: int
4     Valor de Retorno: list """
5
6     listanum = [ ]
7     while numero > 0:
8         listanum[numero-1] = numero
9         numero = numero -1
10    return listanum
```

Qual o problema desta função?

```
1 def exemplowhile2(numero):
2
3     """ Parametro de entrada: int
4     Valor de Retorno: list """
5
6     listanum = [ ]
7     while numero > 0:
8         list.insert(listanum,0,numero)
9         numero = numero -1
10    return listanum
```

Estrutura de Repetição *while*

Exemplo

```
1 def exemplo2(numero):
2
3     """ Funcao que conta quantas vezes se pode reduzir em 1 o valor do numero passado como
4         parametro ate chegar a zero.
5     Parametro de entrada: int
6     Valor de retorno: str """
7
8     contador = 0 # variavel contadora
9     while numero > 0:
10         numero = numero - 1
11         contador = contador + 1
12     return "A funcao rodou " + str(contador) + " vezes."
```

Estrutura de Repetição *while*

Faça uma função que determina a soma de todos os números pares desde 100 até 200.

Estrutura de Repetição *while*

Faça uma função que determina a soma de todos os números pares desde 100 até 200.

```
1 def somaPares():
2
3     """Funcao que calcula a soma dos numeros pares de 100 a 200
4     Parametro de entrada: nao tem
5     Valor de retorno: int"""
6
7     soma = 0 # variavel acumuladora
8     contador = 100 # o contador nao precisa começar de zero
9     while contador <= 200:
10         soma = soma + contador
11         contador = contador + 2 # o contador nao precisa ir de 1 em 1
12     return soma
```

Estrutura de Repetição *while*

A função abaixo apresenta algum problema?

```
1 def exemplo3():
2
3     """ Parametro de entrada: nao tem
4     Valor de retorno: int """
5
6     x = 10
7     while x > 8:
8         x = x+ 2
9     return x
```


Estrutura de Repetição *while*

A função abaixo apresenta algum problema?

```
1 def exemplo3():
2
3     """ Parametro de entrada: nao tem
4     Valor de retorno: int """
5
6     x = 10
7     while x > 8:
8         x = x + 2
9     return x
```

- Sendo X igual a 10, o teste $X > 8$ é inicialmente verdadeiro.
- Enquanto a condição for verdadeira, apenas o comando $X = X + 2$ será executado. Porém incrementar a variável X não altera a validade da condição $X > 8$.
- Logo, a repetição segue **indefinidamente! (Loop infinito)**

Estrutura de Repetição *while*

O que faz a seguinte função ?

```
1 def soma(numero):  
2  
3     """ Parametro de entrada: int  
4     Valor de retorno: int """  
5  
6     soma = 0  
7     contador = 0  
8     while contador < numero:  
9         soma = soma + contador  
10        contador = contador + 1  
11    return soma
```

Estrutura de Repetição *while*

Faça uma função que gere números aleatórios entre 1 e 10 e calcule a soma destes números até que seja gerado o número 5.

Use a função **randint(início,fim)** do módulo **random** para gerar um número aleatório, onde os valores de (início,fim) representam o intervalo desejado para os números a serem gerados.

Exemplo: randint(1,10) → gera um número aleatório entre 1 e 10, inclusive.

Estrutura de Repetição *while*

Faça uma função que gere números aleatórios entre 1 e 10 e calcule a soma destes números até que seja gerado o número 5.

Use a função **randint(início,fim)** do módulo **random** para gerar um número aleatório, onde os valores de (início,fim) representam o intervalo desejado para os números a serem gerados.

Exemplo: randint(1,10) → gera um número aleatório entre 1 e 10, inclusive.

```
1 from random import randint
2
3 def somaAleatoria():
4
5     """ Parametro de entrada: nao tem
6     Valor de retorno: int """
7
8     soma = 0
9     numero = randint(1,10)
10    while numero != 5:
11        soma = soma + numero
12        numero = randint(1,10)
13    return soma
```

Estrutura de Repetição *while*

Faça uma função que gere números aleatórios entre 1 e 10 e calcule a soma destes números até que seja gerado o número 5.

Use a função **randint(início,fim)** do módulo **random** para gerar um número aleatório, onde os valores de (início,fim) representam o intervalo desejado para os números a serem gerados.

Exemplo: randint(1,10) → gera um número aleatório entre 1 e 10, inclusive.

Solução usando listas.

```
1 from random import randint
2
3 def somaAleatoria():
4
5     """ Parametro de entrada: nao tem
6     Valor de retorno: int """
7
8     lista = [ ]
9     numero = randint(1,10)
10    while numero != 5:
11        lista.append(lista , numero)
12        numero = randint(1,10)
13    return sum(lista)
```

Estrutura de Repetição *while*

Faça uma função que some 10 números gerados aleatoriamente no intervalo de 1 a 5.

Estrutura de Repetição *while*

Faça uma função que some 10 números gerados aleatoriamente no intervalo de 1 a 5.

```
1 from random import randint
2
3 def soma10():
4
5     """ Parametro de entrada: nao tem
6     Valor de retorno: int """
7
8     soma = 0
9     contador = 0
10    while contador < 10:
11        numero = randint(1,5)
12        soma = soma + numero
13        contador = contador + 1
14    return soma
```

Estrutura de Repetição *while*

Para cada um dos itens abaixo, faça uma tabela mostrando os valores que i , j e n assumem depois de cada execução do laço *while*.

```
def ...  
    i = 0  
    j = 10  
    n = 0  
    while i < j:  
        i = i + 1  
        j = j - 1  
        n = n + 1
```

```
def ...  
    i = 0  
    j = 0  
    n = 0  
    while i < 10:  
        i = i + 1  
        n = n + i + j  
        j = j + 1
```


Estrutura de Repetição *while*

Faça uma função que dada uma lista de tamanho desconhecido contendo as notas de uma turma de alunos, retorne a média dessas notas.

Autores

- **João C. P. da Silva** ▶ Lattes
- **Carla Delgado** ▶ Lattes
- **Ana Luisa Duboc** ▶ Lattes

Colaboradores

- **Anamaria Martins Moreira** ▶ Lattes
- **Fabio Mascarenhas** ▶ Lattes
- **Leonardo de Oliveira Carvalho** ▶ Lattes
- **Charles Figueiredo de Barros** ▶ Lattes
- **Fabrício Firmino de Faria** ▶ Lattes

Computação I - Python

Aula 8 - Teórica: Estrutura de Repetição : for

João C. P. da Silva

Carla A. D. M. Delgado

Ana Luisa Duboc

Dept. Ciência da Computação - UFRJ

Estrutura de Repetição *while*

Estrutura que permite a repetição de um conjunto de comandos. Até o momento vimos o *while*:

```
while <condição>:  
    <sequência de comandos>
```

- Com *while* podemos implementar qualquer algoritmo que envolva repetição.
- **DICA:** o *while* é mais recomendado quando não se sabe ao certo quantas vezes a repetição será feita, pois a **condição** é um teste booleano qualquer e não necessariamente uma contagem.

Estrutura de Repetição *while*

Lembre: Faça uma função que gere números aleatórios entre 1 e 10 e calcule a soma destes números até que seja gerado o número 5.

```
1 from random import randint
2
3 def somaAleatoria():
4
5     """ Parametro de entrada: nao tem
6     Valor de retorno: int """
7
8     soma = 0
9     numero = randint(1,10)
10    while numero != 5:
11        soma = soma + numero
12        numero = randint(1,10)
13    return soma
```

O número de repetições dos comandos associados ao laço *while* depende de quando sair o número 5. Podem ser 2 vezes ou 1000 vezes!

Estrutura de Repetição *while*

Faça uma função *somaPares* que recebe uma tupla de números inteiros e calcula a soma de todos os números pares que ocorrem nesta tupla.

Por exemplo, a chamada *somaPares*((3, 1, 2, 4, 6, 7, 2)) deve retornar 14 e *somaPares*((1, 3, 5, 7)) deve retornar 0.

Estrutura de Repetição *while*

Faça uma função *somaPares* que recebe uma tupla de números inteiros e calcula a soma de todos os números pares que ocorrem nesta tupla.

Por exemplo, a chamada *somaPares*((3, 1, 2, 4, 6, 7, 2)) deve retornar 14 e *somaPares*((1, 3, 5, 7)) deve retornar 0.

```
1 def somaPares(tupla):
2
3     """ Parametro de entrada: tupla
4     Valor de retorno: int """
5
6     soma = 0
7     indice = 0
8     while indice < len(tupla):
9         if tupla[indice] % 2 == 0:
10             soma = soma + tupla[indice]
11             indice = indice + 1
12     return soma
```

Estrutura de Repetição *while*

Faça uma função *somaPares* que recebe uma tupla de números inteiros e calcula a soma de todos os números pares que ocorrem nesta tupla.

Por exemplo, a chamada *somaPares((3, 1, 2, 4, 6, 7, 2))* deve retornar 14 e *somaPares((1, 3, 5, 7))* deve retornar 0.

```
1 def somaPares(tupla):
2
3     """ Parametro de entrada: tupla
4     Valor de retorno: int """
5
6     soma = 0
7     indice = 0
8     while indice < len(tupla):
9         if tupla[indice] % 2 == 0:
10             soma = soma + tupla[indice]
11             indice = indice + 1
12     return soma
```

Estamos usando a variável *indice* para percorrer a *tupla*.

É possível acessar os elementos na tupla sem precisarmos da variável *indice*!

Estrutura de Repetição *for*

Faça uma função *somaPares* que recebe uma tupla de números inteiros e calcula a soma de todos os números pares que ocorrem nesta tupla.

Por exemplo, a chamada *somaPares*((3, 1, 2, 4, 6, 7, 2)) deve retornar 14 e *somaPares*((1, 3, 5, 7)) deve retornar 0.

Com o comando *for*, podemos pegar um a um os elementos que formam a tupla dada como entrada:

```
1 def somaPares(tupla):
2
3     """ Parametro de entrada: tupla
4     Valor de retorno: int """
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """ Parametro de entrada: tupla
4     Valor de retorno: int """
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

● somaPares((10,21,32,43))

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """ Parametro de entrada: tupla
4     Valor de retorno: int """
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

- somaPares((10,21,32,43))
- soma = 0

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """ Parametro de entrada: tupla
4     Valor de retorno: int """
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

- somaPares((10,21,32,43))
- soma = 0
- for elemento in (10,21,32,43):

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """ Parametro de entrada: tupla
4     Valor de retorno: int """
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

- `somaPares((10,21,32,43))`
- `soma = 0`
- `for elemento in (10,21,32,43):`
 - `if 10 % 2 == 0:`

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """ Parametro de entrada: tupla
4     Valor de retorno: int """
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

- `somaPares((10,21,32,43))`
- `soma = 0`
- `for elemento in (10,21,32,43):`
 - `if 10 % 2 == 0: (True)`

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """ Parametro de entrada: tupla
4     Valor de retorno: int """
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

- `somaPares((10,21,32,43))`
- `soma = 0`
- `for elemento in (10,21,32,43):`
 - `if 10 % 2 == 0: (True)`
 - `soma = 0 + 10 = 10`

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """Parametro de entrada: tupla
4     Valor de retorno: int"""
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

- `somaPares((10,21,32,43))`
- `soma = 0`
- `for elemento in (10,21,32,43):`
 - `if 10 % 2 == 0: (True)`
 - `soma = 0 + 10 = 10`
 - `if 21 % 2 == 0:`

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """Parametro de entrada: tupla
4     Valor de retorno: int"""
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

- `somaPares((10,21,32,43))`
- `soma = 0`
- `for elemento in (10,21,32,43):`
 - `if 10 % 2 == 0: (True)`
 - `soma = 0 + 10 = 10`
 - `if 21 % 2 == 0: (False)`

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """Parametro de entrada: tupla
4     Valor de retorno: int"""
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

- `somaPares((10,21,32,43))`
- `soma = 0`
- `for elemento in (10,21,32,43):`
 - `if 10 % 2 == 0: (True)`
 - `soma = 0 + 10 = 10`
 - `if 21 % 2 == 0: (False)`
 - `if 32 % 2 == 0:`

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """Parametro de entrada: tupla
4     Valor de retorno: int"""
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

- `somaPares((10,21,32,43))`
- `soma = 0`
- `for elemento in (10,21,32,43):`
 - `if 10 % 2 == 0: (True)`
 - `soma = 0 + 10 = 10`
 - `if 21 % 2 == 0: (False)`
 - `if 32 % 2 == 0: (True)`

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """Parametro de entrada: tupla
4     Valor de retorno: int"""
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

- `somaPares((10,21,32,43))`
- `soma = 0`
- `for elemento in (10,21,32,43):`
 - `if 10 % 2 == 0: (True)`
 - `soma = 0 + 10 = 10`
 - `if 21 % 2 == 0: (False)`
 - `if 32 % 2 == 0: (True)`
 - `soma = 10 + 32 = 42`

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """Parametro de entrada: tupla
4     Valor de retorno: int"""
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

- `somaPares((10,21,32,43))`
- `soma = 0`
- `for elemento in (10,21,32,43):`
 - `if 10 % 2 == 0: (True)`
 - `soma = 0 + 10 = 10`
 - `if 21 % 2 == 0: (False)`
 - `if 32 % 2 == 0: (True)`
 - `soma = 10 + 32 = 42`
 - `if 43 % 2 == 0:`

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """Parametro de entrada: tupla
4     Valor de retorno: int"""
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

- `somaPares((10,21,32,43))`
- `soma = 0`
- `for elemento in (10,21,32,43):`
 - `if 10 % 2 == 0: (True)`
 - `soma = 0 + 10 = 10`
 - `if 21 % 2 == 0: (False)`
 - `if 32 % 2 == 0: (True)`
 - `soma = 10 + 32 = 42`
 - `if 43 % 2 == 0: (False)`

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """Parametro de entrada: tupla
4     Valor de retorno: int"""
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

- `somaPares((10,21,32,43))`
- `soma = 0`
- `for elemento in (10,21,32,43):`
 - `if 10 % 2 == 0: (True)`
 - `soma = 0 + 10 = 10`
 - `if 21 % 2 == 0: (False)`
 - `if 32 % 2 == 0: (True)`
 - `soma = 10 + 32 = 42`
 - `if 43 % 2 == 0: (False)`
- `return 42`

Estrutura de Repetição *for*

```
1 def somaPares(tupla):  
2  
3     """Parametro de entrada: tupla  
4     Valor de retorno: int"""  
5  
6     soma = 0  
7     for elemento in tupla:  
8         if elemento % 2 == 0:  
9             soma = soma + elemento  
10    return soma
```

Poderíamos ter usado uma lista ao invés de uma tupla ?

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """Parametro de entrada: tupla
4     Valor de retorno: int"""
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

Poderíamos ter usado uma lista ao invés de uma tupla ? SIM !

Estrutura de Repetição *for*

```
1 def somaPares(tupla):
2
3     """Parametro de entrada: tupla
4     Valor de retorno: int"""
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

Poderíamos ter usado uma lista ao invés de uma tupla ? SIM !

```
1 def somaParesLista(lista):
2
3     """Parametro de entrada: list
4     Valor de retorno: int"""
5
6     soma = 0
7     for elemento in tupla:
8         if elemento % 2 == 0:
9             soma = soma + elemento
10    return soma
```

`somaParesLista([10,21,32,43])` retorna 42 também !

Estrutura de Repetição *while*

Faça uma função que some 10 números gerados aleatoriamente no intervalo de 1 a 5.
Como seria essa função com **while**?

Estrutura de Repetição *while*

Faça uma função que some 10 números gerados aleatoriamente no intervalo de 1 a 5. Como seria essa função com **while**?

```
1 from random import randint
2
3 def soma10():
4
5     """ Funcao que soma 10 numeros gerados aleatoriamente no intervalo de 1 a 5
6     Parametro de entrada: nao tem
7     Valor de retorno: int"""
8
9     contador = 0
10    soma = 0
11    while contador < 10:
12        numero = randint(1,5)
13        soma = soma + numero
14        contador = contador + 1
15    return soma
```

O número de repetições será 10 em qualquer execução da função, independente dos números aleatórios gerados.

Estrutura de Repetição *for*

Faça uma função que some 10 números gerados aleatoriamente no intervalo de 1 a 5.
Como seria essa função com **for** ?

Estrutura de Repetição *for*

Faça uma função que some 10 números gerados aleatoriamente no intervalo de 1 a 5. Como seria essa função com **for** ? Na função, a variável *contador* vai assumir os valores 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9.

```
1 from random import randint
2
3 def soma10usandofor():
4
5     """ Funcao que soma 10 numeros gerados aleatoriamente no intervalo de 1 a 5 usando o
6         comando for
7     Parametro de entrada: nao tem
8     Valor de retorno: int"""
9
10    soma = 0
11    for contador in range(10):
12        numero = randint(1,5)
13        soma = soma + numero
14    return soma
```

<pre>for var in range(n): <comandos></pre>	OU	<pre>for var in [0,..., n-1]: <comandos></pre>
--	----	--

Estrutura de Repetição *for*

- A função `range(...)` pode ter 1, 2 ou 3 argumentos:
 - `range(numero)`: faz com que a variável do `for` assuma valores de 0 a `numero-1`

`for x in range(10):` → x recebe 0,1,2,...,9

- `range(inf,sup)`: faz com que a variável do `for` assuma valores de `inf` a `sup-1`

`for x in range(3,8):` → x recebe 3,4,5,6,7

- `range(inf, sup, inc)`: faz com que a variável do `for` assuma valores de `inf` a `sup-1` com incremento de `inc`

`for x in range(3,8,2):` → x recebe 3,5,7

Estrutura de Repetição *for*

Faça uma função que determina a soma de todos os números pares desde 100 até 200. (Usando **for** ao invés de **while**)

Estrutura de Repetição *for*

Faça uma função que determina a soma de todos os números pares desde 100 até 200. (Usando **for** ao invés de **while**)

```
1 def somaPares():
2
3     """Funcao que soma todos os numeros pares de 100 ate 200
4     Parametro de entrada: nao tem
5     Valor de retorno: int"""
6
7     soma = 0
8     for par in range(100,202,2):
9         soma = soma + par
10    return soma
```

OU

```
1 def somaPares():
2
3     """Funcao que soma todos os numeros pares de 100 ate 200
4     Parametro de entrada: nao tem
5     Valor de retorno: int"""
6
7     soma = 0
8     lista = range(100,202,2)
9     for par in lista:
10        soma = soma + par
11    return soma
```

Estrutura de Repetição

IMPORTANTE: diferença de uso entre *while* e *for*:

- **while**: decisão sobre repetir ou não baseia-se em teste booleano.
Risco de loop infinito. :-)
- **for**: Contagem automática do número de repetições.

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

● `vogaisPalavra('testando')`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `vogaisPalavra('testando')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `vogaisPalavra('testando')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'], resposta = []`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `vogaisPalavra('testando')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'], resposta = []`
- `for letra in 'testando':`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a','e','i','o','u','A','E','I','O','U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta , letra)
12     return resposta
```

- `vogaisPalavra('testando')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'], resposta = []`
- `for letra in 'testando':`
 - `if 't' in ['a','e','i','o','u','A','E','I','O','U']:`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """ Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list """
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `vogaisPalavra('testando')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'], resposta = []`
- `for letra in 'testando':`
 - `if 't' in ['a','e','i','o','u','A','E','I','O','U']:` (False)

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `vogaisPalavra('testando')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'], resposta = []`
- `for letra in 'testando':`
 - `if 't' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 'e' in ['a','e','i','o','u','A','E','I','O','U']:`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `vogaisPalavra('testando')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'], resposta = []`
- `for letra in 'testando':`
 - `if 't' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 'e' in ['a','e','i','o','u','A','E','I','O','U']:` (True)

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `vogaisPalavra('testando')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'], resposta = []`
- `for letra in 'testando':`
 - `if 't' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 'e' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'e')`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `vogaisPalavra('testando')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'], resposta = []`
- `for letra in 'testando':`
 - `if 't' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 'e' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'e')`
 - `if 's' in ['a','e','i','o','u','A','E','I','O','U']:`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `vogaisPalavra('testando')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'], resposta = []`
- `for letra in 'testando':`
 - `if 't' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 'e' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'e')`
 - `if 's' in ['a','e','i','o','u','A','E','I','O','U']:` (False)

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `vogaisPalavra('testando')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'], resposta = []`
- `for letra in 'testando':`
 - `if 't' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 'e' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'e')`
 - `if 's' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 't' in ['a','e','i','o','u','A','E','I','O','U']:`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `vogaisPalavra('testando')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'], resposta = []`
- `for letra in 'testando':`
 - `if 't' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 'e' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'e')`
 - `if 's' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 't' in ['a','e','i','o','u','A','E','I','O','U']:` (False)

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `for letra in 'testando':`
 - `if 'a' in ['a','e','i','o','u','A','E','I','O','U']:`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `for letra in 'testando':`
 - `if 'a' in ['a','e','i','o','u','A','E','I','O','U']:` (True)

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

- `for letra in 'testando':`
 - `if 'a' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append(['e'],'a')`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a','e','i','o','u','A','E','I','O','U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta , letra)
12     return resposta
```

- `for letra in 'testando':`
 - `if 'a' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append(['e'],'a')`
 - `if 'n' in ['a','e','i','o','u','A','E','I','O','U']:`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

• `for letra in 'testando':`

- `if 'a' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (True)
 - `list.append(['e'], 'a')`
- `if 'n' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (False)

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

• `for letra in 'testando':`

- `if 'a' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (True)
 - `list.append(['e'], 'a')`
- `if 'n' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (False)
- `if 'd' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

• `for letra in 'testando':`

- `if 'a' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (True)
 - `list.append(['e'], 'a')`
- `if 'n' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (False)
- `if 'd' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (False)

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

• `for letra in 'testando':`

- `if 'a' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (True)
 - `list.append(['e'], 'a')`
- `if 'n' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (False)
- `if 'd' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (False)
- `if 'o' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

• `for letra in 'testando':`

- `if 'a' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (True)
 - `list.append(['e'], 'a')`
- `if 'n' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (False)
- `if 'd' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (False)
- `if 'o' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (True)

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

• `for letra in 'testando':`

- `if 'a' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (True)
 - `list.append(['e'], 'a')`
- `if 'n' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (False)
- `if 'd' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (False)
- `if 'o' in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:` (True)
 - `list.append(['e', 'a'], 'o')`

Estrutura de Repetição *for*

Faça uma função que dada uma palavra retorna uma lista formada por todas as vogais que aparecem na palavra.

```
1 def vogaisPalavra(palavra):
2
3     """Funcao que retorna todas as vogais que aparecem em uma palavra
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
8     resposta = [ ]
9     for letra in palavra:
10         if letra in vogais:
11             list.append(resposta, letra)
12     return resposta
```

• **for** letra **in** 'testando':

• **if** 'a' **in** ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']: (True)

• **list.append**(['e'], 'a')

• **if** 'n' **in** ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']: (False)

• **if** 'd' **in** ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']: (False)

• **if** 'o' **in** ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']: (True)

• **list.append**(['e', 'a'], 'o')

• **return** ['e', 'a', 'o']

Estrutura de Repetição *for*

Faça uma função que dada uma string retorna uma lista formada por todas as palavras que começam com vogais.

Dica: use a função `str.split` para separar as palavras na frase.

Estrutura de Repetição *for*

Faça uma função que dada uma string retorna uma lista formada por todas as palavras que começam com vogais.

Dica: use a função `str.split` para separar as palavras na frase.

```
1 def palavrasComecandoComVogal(frase):
2
3     """Funcao que retorna todas as palavras que come am com vogais
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a','e','i','o','u','A','E','I','O','U']
8     lista = str.split(frase)
9     resposta = [ ]
10    for palavra in lista:
11        if palavra[0] in vogais:
12            list.append(resposta , palavra)
13    return resposta
```

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ) :
2
3 """ Funcao que retorna todas as palavras que come am com vogais
4 Parametro de entrada: str
5 Valor de retorno: list """
6
7 vogais = [ 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' ]
8 lista = str.split( frase )
9 resposta = [ ]
10 for palavra in lista :
11     if palavra[0] in vogais :
12         list.append( resposta , palavra )
13 return resposta
```

● palavrasComecandoComVogal('Amanhã será outro dia.')

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal(frase):
2
3     """Funcao que retorna todas as palavras que come am com vogais
4     Parametro de entrada: str
5     Valor de retorno: list"""
6
7     vogais = ['a','e','i','o','u','A','E','I','O','U']
8     lista = str.split(frase)
9     resposta = [ ]
10    for palavra in lista:
11        if palavra[0] in vogais:
12            list.append(resposta , palavra)
13    return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ) :
2
3 """ Funcao que retorna todas as palavras que come am com vogais
4 Parametro de entrada: str
5 Valor de retorno: list """
6
7 vogais = [ 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' ]
8 lista = str.split( frase )
9 resposta = [ ]
10 for palavra in lista :
11     if palavra[0] in vogais :
12         list.append( resposta , palavra )
13 return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'],`

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ) :
2
3 """ Funcao que retorna todas as palavras que come am com vogais
4 Parametro de entrada: str
5 Valor de retorno: list """
6
7 vogais = [ 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' ]
8 lista = str.split( frase )
9 resposta = [ ]
10 for palavra in lista :
11     if palavra[0] in vogais :
12         list.append( resposta , palavra )
13 return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'], resposta = []`

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ) :
2
3 """ Funcao que retorna todas as palavras que come am com vogais
4 Parametro de entrada: str
5 Valor de retorno: list """
6
7 vogais = [ 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' ]
8 lista = str.split( frase )
9 resposta = [ ]
10 for palavra in lista :
11     if palavra[0] in vogais :
12         list.append( resposta , palavra )
13 return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'], resposta = []`
- `for palavra in ['Amanhã','será','outro','dia.']:`

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ) :
2
3 """ Funcao que retorna todas as palavras que come am com vogais
4 Parametro de entrada: str
5 Valor de retorno: list """
6
7 vogais = [ 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' ]
8 lista = str.split( frase )
9 resposta = [ ]
10 for palavra in lista :
11     if palavra[0] in vogais :
12         list.append( resposta , palavra )
13 return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'], resposta = []`
- `for palavra in ['Amanhã','será','outro','dia.']:`
 - `if 'A' in ['a','e','i','o','u','A','E','I','O','U']:`

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ):  
2  
3 """Funcao que retorna todas as palavras que come am com vogais  
4 Parametro de entrada: str  
5 Valor de retorno: list"""  
6  
7 vogais = [ 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' ]  
8 lista = str.split( frase )  
9 resposta = [ ]  
10 for palavra in lista:  
11     if palavra[0] in vogais:  
12         list.append( resposta , palavra )  
13 return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'], resposta = []`
- `for palavra in ['Amanhã','será','outro','dia.']:`
 - `if 'A' in ['a','e','i','o','u','A','E','I','O','U']:` (True)

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ):  
2  
3 """Funcao que retorna todas as palavras que come am com vogais  
4 Parametro de entrada: str  
5 Valor de retorno: list"""  
6  
7 vogais = [ 'a','e','i','o','u','A','E','I','O','U']  
8 lista = str.split(frase)  
9 resposta = [ ]  
10 for palavra in lista:  
11     if palavra[0] in vogais:  
12         list.append(resposta , palavra)  
13 return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'], resposta = []`
- `for palavra in ['Amanhã','será','outro','dia.']:`
 - `if 'A' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'Amanhã')`

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ):  
2  
3 """ Funcao que retorna todas as palavras que come am com vogais  
4 Parametro de entrada: str  
5 Valor de retorno: list """  
6  
7 vogais = [ 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' ]  
8 lista = str.split( frase )  
9 resposta = [ ]  
10 for palavra in lista:  
11     if palavra[0] in vogais:  
12         list.append( resposta , palavra )  
13     return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'], resposta = []`
- `for palavra in ['Amanhã','será','outro','dia.']:`
 - `if 'A' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'Amanhã')`
 - `if 's' in ['a','e','i','o','u','A','E','I','O','U']:`

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ) :
2
3 """ Funcao que retorna todas as palavras que come am com vogais
4 Parametro de entrada: str
5 Valor de retorno: list """
6
7 vogais = [ 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' ]
8 lista = str.split( frase )
9 resposta = [ ]
10 for palavra in lista :
11     if palavra[0] in vogais :
12         list.append( resposta , palavra )
13 return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'], resposta = []`
- `for palavra in ['Amanhã','será','outro','dia.']:`
 - `if 'A' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'Amanhã')`
 - `if 's' in ['a','e','i','o','u','A','E','I','O','U']:` (False)

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ) :
2
3 """ Funcao que retorna todas as palavras que come am com vogais
4 Parametro de entrada: str
5 Valor de retorno: list """
6
7 vogais = [ 'a','e','i','o','u','A','E','I','O','U' ]
8 lista = str.split( frase )
9 resposta = [ ]
10 for palavra in lista :
11     if palavra[0] in vogais :
12         list.append( resposta , palavra )
13 return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'], resposta = []`
- `for palavra in ['Amanhã','será','outro','dia.']:`
 - `if 'A' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'Amanhã')`
 - `if 's' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 'o' in ['a','e','i','o','u','A','E','I','O','U']:`

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ):  
2  
3 """ Funcao que retorna todas as palavras que come am com vogais  
4 Parametro de entrada: str  
5 Valor de retorno: list """  
6  
7 vogais = [ 'a','e','i','o','u','A','E','I','O','U']  
8 lista = str.split(frase)  
9 resposta = [ ]  
10 for palavra in lista:  
11     if palavra[0] in vogais:  
12         list.append( resposta , palavra )  
13     return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'], resposta = []`
- `for palavra in ['Amanhã','será','outro','dia.']:`
 - `if 'A' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'Amanhã')`
 - `if 's' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 'o' in ['a','e','i','o','u','A','E','I','O','U']:` (True)

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ) :
2
3 """ Funcao que retorna todas as palavras que come am com vogais
4 Parametro de entrada: str
5 Valor de retorno: list """
6
7 vogais = [ 'a','e','i','o','u','A','E','I','O','U' ]
8 lista = str.split(frase)
9 resposta = [ ]
10 for palavra in lista :
11     if palavra[0] in vogais :
12         list.append( resposta , palavra )
13 return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'], resposta = []`
- `for palavra in ['Amanhã','será','outro','dia.']:`
 - `if 'A' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'Amanhã')`
 - `if 's' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 'o' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append(['Amanhã'],'outro')`

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ) :
2
3 """ Funcao que retorna todas as palavras que come am com vogais
4 Parametro de entrada: str
5 Valor de retorno: list """
6
7 vogais = [ 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' ]
8 lista = str.split( frase )
9 resposta = [ ]
10 for palavra in lista :
11     if palavra[0] in vogais :
12         list.append( resposta , palavra )
13 return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'], resposta = []`
- `for palavra in ['Amanhã','será','outro','dia.']:`
 - `if 'A' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'Amanhã')`
 - `if 's' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 'o' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append(['Amanhã'],'outro')`
 - `if 'd' in ['a','e','i','o','u','A','E','I','O','U']:`

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ) :
2
3 """ Funcao que retorna todas as palavras que come am com vogais
4 Parametro de entrada: str
5 Valor de retorno: list """
6
7 vogais = [ 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' ]
8 lista = str.split( frase )
9 resposta = [ ]
10 for palavra in lista :
11     if palavra[0] in vogais :
12         list.append( resposta , palavra )
13 return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'], resposta = []`
- `for palavra in ['Amanhã','será','outro','dia.']:`
 - `if 'A' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'Amanhã')`
 - `if 's' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 'o' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append(['Amanhã'],'outro')`
 - `if 'd' in ['a','e','i','o','u','A','E','I','O','U']:` (False)

Estrutura de Repetição *for*

```
1 def palavrasComecandoComVogal ( frase ):
2
3     """ Funcao que retorna todas as palavras que come am com vogais
4     Parametro de entrada: str
5     Valor de retorno: list """
6
7     vogais = [ 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' ]
8     lista = str.split( frase )
9     resposta = [ ]
10    for palavra in lista:
11        if palavra[0] in vogais:
12            list.append( resposta , palavra )
13    return resposta
```

- `palavrasComecandoComVogal('Amanhã será outro dia.')`
- `vogais = ['a','e','i','o','u','A','E','I','O','U'],`
`lista = ['Amanhã','será','outro','dia.'], resposta = []`
- `for palavra in ['Amanhã','será','outro','dia.']:`
 - `if 'A' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append([], 'Amanhã')`
 - `if 's' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
 - `if 'o' in ['a','e','i','o','u','A','E','I','O','U']:` (True)
 - `list.append(['Amanhã'],'outro')`
 - `if 'd' in ['a','e','i','o','u','A','E','I','O','U']:` (False)
- `return ['Amanhã','outro']`

Estrutura de Repetição *for*

Resolva usando *for*:

1. Dada uma lista de nomes e um número inteiro n , faça uma função que conte quantos nomes de tamanho maior que n aparecem nesta lista.
2. Faça uma função que calcule o valor de $N!$, onde N é passado como parâmetro. (Sem usar o `factorial` do módulo *math*).
3. Faça uma função que calcule e retorne o valor de

$$S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + \cdots + \frac{99}{50}$$

4. Faça uma função que calcule e retorne o valor de

$$S = \frac{1}{1} - \frac{2}{4} + \frac{3}{9} - \frac{4}{16} + \cdots - \frac{10}{100}$$

Atenção ao tipo de dado !!!

Autores

- **João C. P. da Silva** ▶ Lattes
- **Carla Delgado** ▶ Lattes
- **Ana Luisa Duboc** ▶ Lattes

Colaboradores

- **Anamaria Martins Moreira** ▶ Lattes
- **Fabio Mascarenhas** ▶ Lattes
- **Leonardo de Oliveira Carvalho** ▶ Lattes
- **Charles Figueiredo de Barros** ▶ Lattes
- **Fabício Firmino de Faria** ▶ Lattes

Computação I - Python

Aula 8 - Teórica: Estrutura de Repetição : for

João C. P. da Silva

Carla A. D. M. Delgado

Ana Luisa Duboc

Dept. Ciência da Computação - UFRJ