

Victor Mesquita Lima de Santana

**Mundo 3 de Python:**  
**Curso Em Vídeo**



São Paulo

2021

## Variáveis Compostas

É o modo de atribuir diversos valores à mesma variável ao mesmo tempo.

Exemplo:



Nesse caso, a variável lanche recebe 4 informações distintas (hamburguer, suco, pizza e pudim). Outra coisa a se observar é que dá para identificar cada item da variável composta. Cada parte dela é numerada em índices, como representados na imagem.

Para representar somente algumas partes da variável composta, utiliza o nome da variável seguida do índice do item requisitado entre colchetes. Dessa forma, o código ficaria assim:



Também pode ser combado com a estrutura de repetição for:



As três formas de elaborar variáveis compostas são:

- Tuplas;
- Listas;
- Dicionários.

A representação de cada um deles é assim:



## Tuplas

Essa forma é caracterizada por ser imutável. Uma vez tendo os seus itens estabelecidos, não poderá voltar a traz para substituí-los.

```
aula16.py × aula16b.py × aula16c.py ×
1 lanche = ('Hamburguer', 'Suco', 'Pizza', 'Pudim', 'Batata Frita')
2 # Tuplas são imutáveis
3
4 for comida in lanche:
5     print(f'Eu vou comer {comida} ')
6
7 for cont in range(0, len(lanche)):
8     print(f'Eu vou comer {lanche[cont]} na posição {cont} ')
9
10 for pos, comida in enumerate(lanche):
11     print(f'Eu vou comer {comida} na posição {pos}')
12
13 print('Comi pra caramba!')
```

```
aula16.py × aula16b.py × aula16c.py ×
1 lanche = ('Hamburguer', 'Suco', 'Pizza', 'Pudim', 'Batata Frita')
2
3 # Sorted significa ordenar, nesse caso em ordem alfabética
4 print(sorted(lanche))
5
```

```
aula16.py × aula16b.py × aula16c.py ×
1 a = (2, 5, 4)
2 b = (5, 8, 1, 2)
3 c = a + b
4 # count é para contar quantas unidades um item tem
5 print(c.count(3))
6 # index é para saber a posição de um determinado item
7 print(c.index(8))
8
```

## Exercício 072 - Número por Extenso

**Crie um programa que tenha uma dupla totalmente preenchida com uma contagem por extenso, de zero até vinte. Seu programa deverá ler um número pelo teclado (entre 0 e 20) e mostrá-lo por extenso.**

**Desafio extra: Fazer ele rodar até o usuário dizer que não utilizará mais.**

Código:

```
1 numextenso = ('Zero', 'Um', 'Dois', 'Três', 'Quatro', 'Cinco', 'Seis', 'Sete',
2                 'Oito', 'Nove', 'Dez', 'Onze', 'Doze', 'Treze', 'Catorze', 'Quinze',
3                 'Desesseis', 'Desessete', 'Dezoito', 'Dezenove', 'Vinte')
4 while True:
5     escolha = int(input('Digite um número entre 0 e 20: '))
6     while escolha < 0 or escolha > 20:
7         escolha = int(input('Tente Novamente. Digite um número entre 0 e 20: '))
8     print(f'Você digitou o número {numextenso[escolha]}')
9     continuar = str(input('Quer continuar? [S/N] ')).strip().upper()[0]
10    while continuar not in 'SN':
11        continuar = str(input('Quer continuar? [S/N] ')).strip().upper()[0]
12    if continuar == 'N':
13        break
14
```

Run:

```
Run: ex072 ×
" C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\ex072.py "
Digite um número entre 0 e 20: 56
Tente Novamente. Digite um número entre 0 e 20: 23
Tente Novamente. Digite um número entre 0 e 20: 8
Você digitou o número Oito
Quer continuar? [S/N] s
Digite um número entre 0 e 20: 19
Você digitou o número Dezenove
Quer continuar? [S/N] j
Quer continuar? [S/N] l
Quer continuar? [S/N] n

Process finished with exit code 0
```

## Exercício 073 – Tuplas com Times de Futebol

Crie uma tupla preenchida com os 20 primeiros colocados da Tabela do Campeonato Brasileiro de Futebol, na ordem de colocação. Depois mostre:

- Os 5 primeiros times.
- Os últimos 4 colocados.
- Times em ordem alfabética.
- Em que posição está o time da Chapecoense.

Código:

```
ex073.py ×
1     times = ('Atlético-MG', 'Flamengo', 'Palmeiras', 'Fortaleza', 'Corinthians',
2             'Bragantino', 'Fluminense', 'América-MG', 'Athlético-GO', 'Santos',
3             'Ceará SC', 'Internacional', 'São Paulo', 'Athletico-PR', 'Cuiabá',
4             'Juventude', 'Grêmio', 'Bahia', 'Sport Recife', 'Chapecoense')
5     print('--'*15)
6     print('Lista de times do Brasileirão:', times)
7     print('--'*15)
8     print('Os 5 primeiros times são:', times[0:5])
9     print('--'*15)
10    print('Os 4 últimos são:', times[-4:])
11    print('--'*15)
12    print('Times em ordem alfabética:', sorted(times))
13    print('--'*15)
14    for pos, time in enumerate(times):
15        if time == 'Chapecoense':
16            print(f'O Chapecoense está na {pos + 1}º posição.')
17
```

Run:

```
Run: ex073 ×
" C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\python-3.10.1-6
-----+
Lista de times do Brasileirão: ('Atlético-MG', 'Flamengo', 'Palmeiras', 'Fort
-----+
Os 5 primeiros times são: ('Atlético-MG', 'Flamengo', 'Palmeiras', 'For
-----+
Os 4 últimos são: ('Grêmio', 'Bahia', 'Sport Recife', 'Chapecoense')
-----+
Times em ordem alfabética: ['América-MG', 'Athletico-PR', 'Athlético-GO
-----+
O Chapecoense está na 20º posição.

Process finished with exit code 0
```

## Exercício 074 – Maior e Menor Valores em Tupla

**Crie um programa que vai gerar cinco números aleatórios e colocar em uma tupla. Depois disso, mostre a listagem de números gerados e também indique o menor e o maior valor que estão na tupla.**

Código:

```
ex074.py ×
1  from random import randint
2  numeros = (randint(1, 10), randint(1, 10), randint(1, 10),
3              randint(1, 10), randint(1, 10))
4  print('Os valores sorteados foram: ', end='')
5  for n in numeros:
6      print(f'{n}', end=' ')
7  print(f'\nO maior valor sorteado foi {max(numeros)}')
8  print(f'O menor valor sorteado foi {min(numeros)}')
9
```

Run:

```
Run: ex074 ×
C:\Users\DELL\Desktop\Estudo de Python\Python\Exercícios\ex074.py
Os valores sorteados foram: 9 3 8 9 7
O maior valor sorteado foi 9
O menor valor sorteado foi 3
```

## Exercício 075 – Análise de Dados em uma Tupla

**Desenvolva um programa que leia quatro valores pelo teclado e guarde-os em uma tupla. No final, mostre:**

- A) Quantas vezes apareceu o valor 9;**
- B) Em que posição foi digitado o primeiro valor 3;**
- C) Quais foram os números pares.**

Código:

```
ex075.py ×
1  numeros = (int(input('Digite um número: ')), int(input('Digite outro número: ')),
2              int(input('Digite mais um número: ')), int(input('Digite o último número: ')))
3  print(f'O valor 9 apareceu {numeros.count(9)} vez(es)')
4  if 3 in numeros:
5      print(f'O valor 3 apareceu na {numeros.index(3)+1}ª posição')
6  else:
7      print('O valor 3 não foi digitado em nenhuma posição')
8  print(f'Os valores pares digitados foram ', end='')
9  for n in numeros:
10     if n % 2 == 0:
11         print(n, end=' ')
```

Run:

```
Run: ex075 ×
▶ ↑ "C:\Users\DELL\Desktop\Estudo de Pyt
Digitando um número: 9
Digitando outro número: 2
Digitando mais um número: 6
Digitando o último número: 3
O valor 9 apareceu 1 vez(es)
O valor 3 apareceu na 4ª posição
Os valores pares digitados foram
2 6
Process finished with exit code 0
```

## Exercício 76 – Lista de Preços com Tupla

**Crie um programa que tenha uma tupla única com nomes de produtos e seus respectivos preços, na sequência. No final, mostre uma listagem de preços, organizando os dados em forma tabular.**

Código:

```
1     materiais = ('Lápis', 1.75,
2                     'Borracha', 2.00,
3                     'Caderno', 15.90,
4                     'Estojo', 25.00,
5                     'Transferidor', 4.20,
6                     'Compasso', 9.99,
7                     'Mochila', 120.32,
8                     'Canetas', 22.30,
9                     'Livro', 34.90)
10    print('=' * 40)
11    print('LISTAGEM DE PREÇOS'.center(40))
12    print('=' * 40)
13    for pos in range(0, len(materiais)):
14        if pos % 2 == 0:
15            print(f'{materiais[pos]:<30}', end=' ')
16        else:
17            print(f'R${materiais[pos]:>0.2f}')
18    print('=' * 40)
19    |
```

Run:

```
Run: ex076 x
"C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\Exercicios/ex076.py"
=====
        LISTAGEM DE PREÇOS
=====
Lápis.....R$1.75
Borracha.....R$2.00
Caderno.....R$15.90
Estojo.....R$25.00
Transferidor.....R$4.20
Compasso.....R$9.99
Mochila.....R$120.32
Canetas.....R$22.30
Livro.....R$34.90
=====

Process finished with exit code 0
```

## Exercício 077 – Contando Vogais em Tupla

Crie um programa que tenha uma tupla com várias palavras (não usar acentos). Depois disso, você deve mostrar, para cada palavra, quais são as suas vogais.

Código:

```
Run: ex077.py x
1 palavras = ('APRENDER', 'PROGRAMAR', 'LINGUAGEM', 'PYTHON',
2                 'CURSO', 'GRATIS', 'ESTUDAR', 'PRATICAR',
3                 'TRABALHAR', 'MERCADO', 'PROGRAMADOR', 'FUTURO')
4 for p in palavras:
5     print(f'\nNa palavra {p} temos ', end=' ')
6     for letra in p:
7         if letra.upper() in 'AEIOU':
8             print(letra, end=' ')
```

Run:

```
Run: ex077 ×
C:\Users\DELL\Desktop\Estudo de Python\Python\PycharmProjects\Exercicios/ex077.py
Na palavra APRENDER temos A E E
Na palavra PROGRAMAR temos O A A
Na palavra LINGUAGEM temos I U A E
Na palavra PYTHON temos O
Na palavra CURSO temos U O
Na palavra GRATIS temos A I
Na palavra ESTUDAR temos E U A
Na palavra PRATICAR temos A I A
Na palavra TRABALHAR temos A A A
Na palavra MERCADO temos E A O
Na palavra PROGRAMADOR temos O A A O
Na palavra FUTURO temos U U O
```

## Listas

Diferente das tuplas, as listas se identificam substituindo os parênteses por colchetes, ainda com a presença de vírgulas e aspas separando as chaves:



Outra diferença, é a capacidade de alterar uma lista. As tuplas eram imutáveis, não podendo substituir os valores dela depois que formados. Já as listas não, se for atribuído novos valores, estes substituirão os atuais e os apagarão.

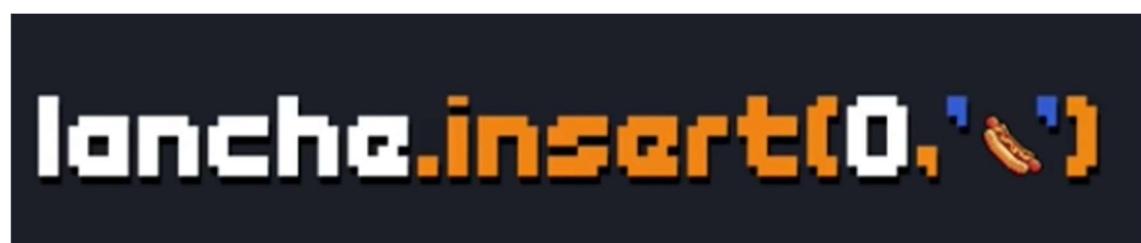
Para adicionar novos valores, utiliza-se um método especial, chamado de append:



Ele adicionará sempre um determinado valor ao final da sua lista



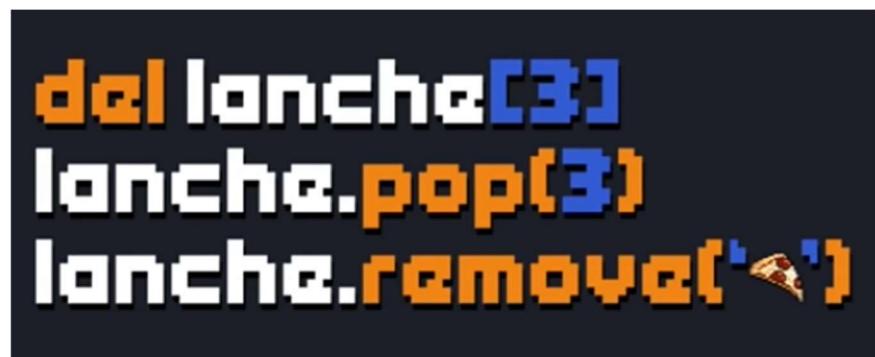
Agora quando se quer adicionar o valor em outra posição, basta utilizar o método `insert`, para inserir o item em qual quer local da lista.



Nesse caso, um cachorro-quente está sendo inserido na posição 0 e o restante está sendo empurrado para as próximas chaves.



Para apagar algum item de uma lista, temos várias possibilidades. Dentre elas, estão:



No método remove, você remove utilizando o nome do conteúdo, diferente dos outros dois que apagam utilizando a posição do item na lista.

Depois que os itens são apagados, a lista é reorganizada, apenas arrastando os itens para os espaços vazios que sobraram para ficar em ordem:



Caso o comando pop seja utilizado sem índice, a última posição é eliminada da lista:



Quando se quer criar uma lista em uma determinada progressão, começando em tal número e terminando em outro tal, utiliza-se o seguinte comando:



Ele vai ser ordenado automaticamente, sempre de forma crescente, como na imagem.

Agora, quando se quer algo mais em fora de ordem, o comando ficará mais ou menos desse jeito:

```
valores = [8, 2, 5, 4, 9, 3, 0]
```

valores							
8	2	5	4	9	3	0	
0	1	2	3	4	5	6	

Para reorganizar uma lista que está fora de ordem, o comando utilizado é o sort:

```
valores = [8, 2, 5, 4, 9, 3, 0]
valores.sort()
```

valores							
0	2	3	4	5	8	9	
0	1	2	3	4	5	6	

E se precisar ser na ordem inversa, tem solução também:

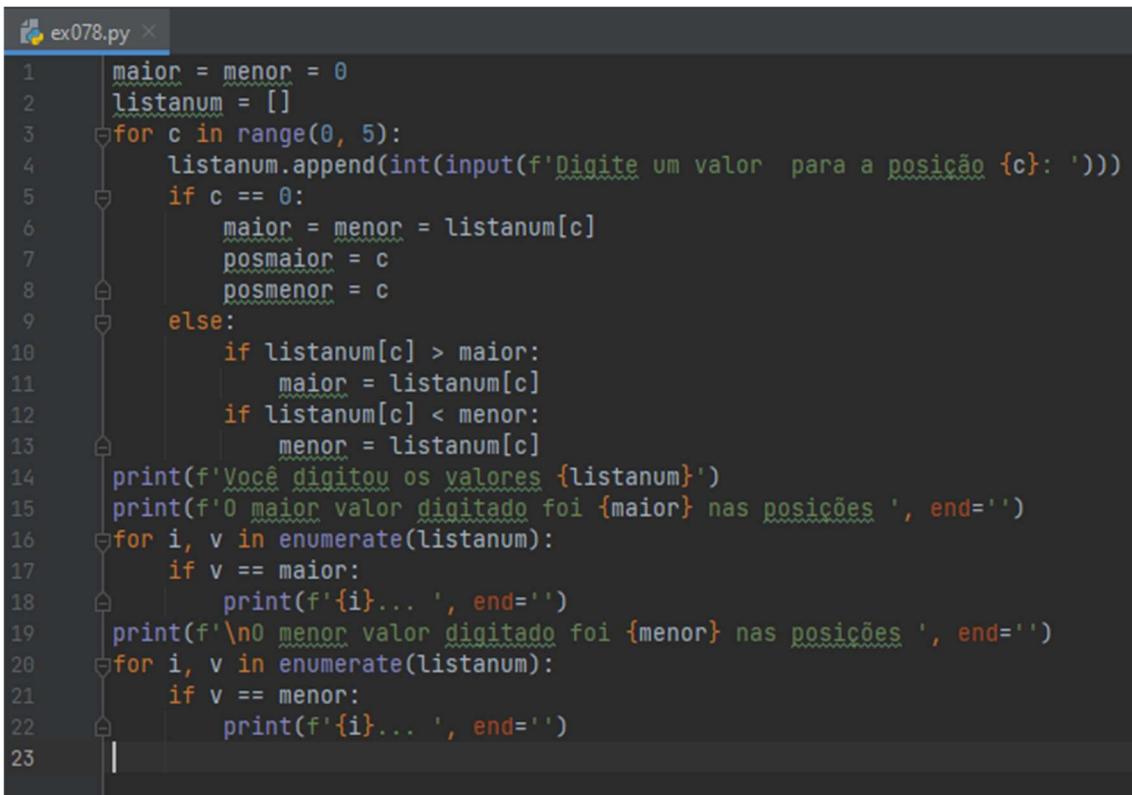
```
valores.sort(reverse=True)
```

valores							
9	8	5	4	3	2	0	
0	1	2	3	4	5	6	

## Exercício 078 – Maior e Menor Valores na Lista

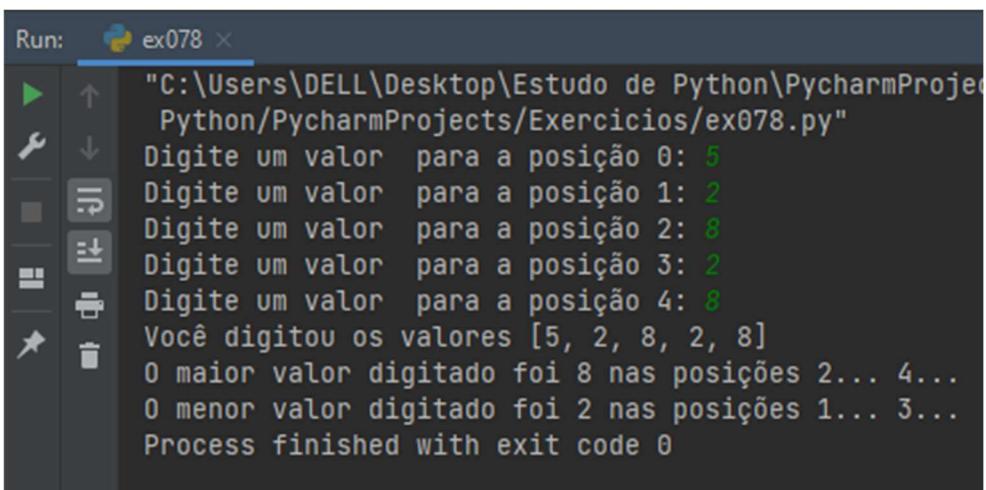
Faça um programa que leia 5 valores numéricos e guarde-os em uma lista. No final, mostre qual foi o maior e o menor valor digitado e as suas respectivas posições na lista.

Código:



```
ex078.py
1 maior = menor = 0
2 listanum = []
3 for c in range(0, 5):
4     listanum.append(int(input(f'Digite um valor para a posição {c}: ')))
5     if c == 0:
6         maior = menor = listanum[c]
7         posmaior = c
8         posmenor = c
9     else:
10        if listanum[c] > maior:
11            maior = listanum[c]
12        if listanum[c] < menor:
13            menor = listanum[c]
14    print(f'Você digitou os valores {listanum}')
15    print(f'O maior valor digitado foi {maior} nas posições ', end='')
16    for i, v in enumerate(listanum):
17        if v == maior:
18            print(f'{i}...', end=' ')
19    print(f'\nO menor valor digitado foi {menor} nas posições ', end='')
20    for i, v in enumerate(listanum):
21        if v == menor:
22            print(f'{i}...', end=' ')
23
```

Run:

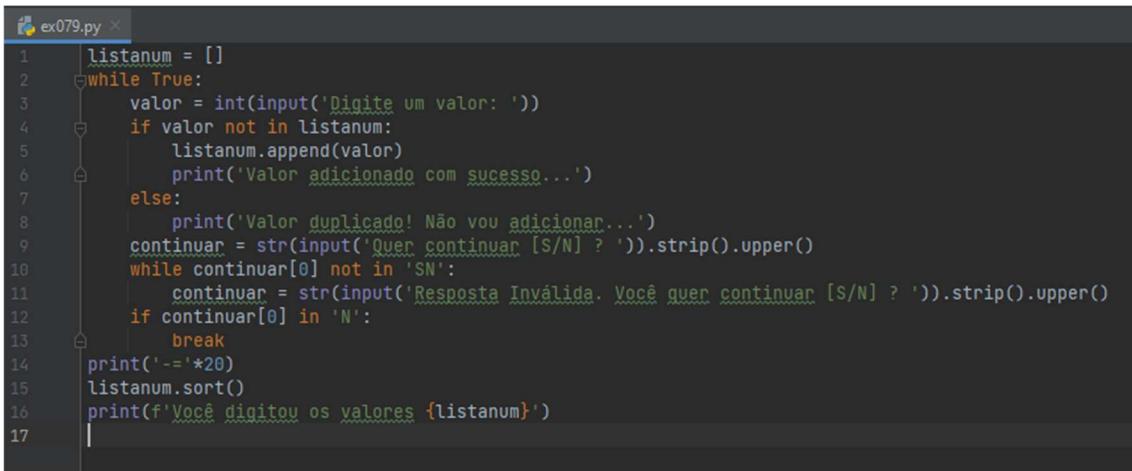


```
Run: ex078
C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\Python\PycharmProjects\Exercicios/ex078.py
Digite um valor para a posição 0: 5
Digite um valor para a posição 1: 2
Digite um valor para a posição 2: 8
Digite um valor para a posição 3: 2
Digite um valor para a posição 4: 8
Você digitou os valores [5, 2, 8, 2, 8]
O maior valor digitado foi 8 nas posições 2... 4...
O menor valor digitado foi 2 nas posições 1... 3...
Process finished with exit code 0
```

## Exercício 79 – Valores Únicos em uma Lista

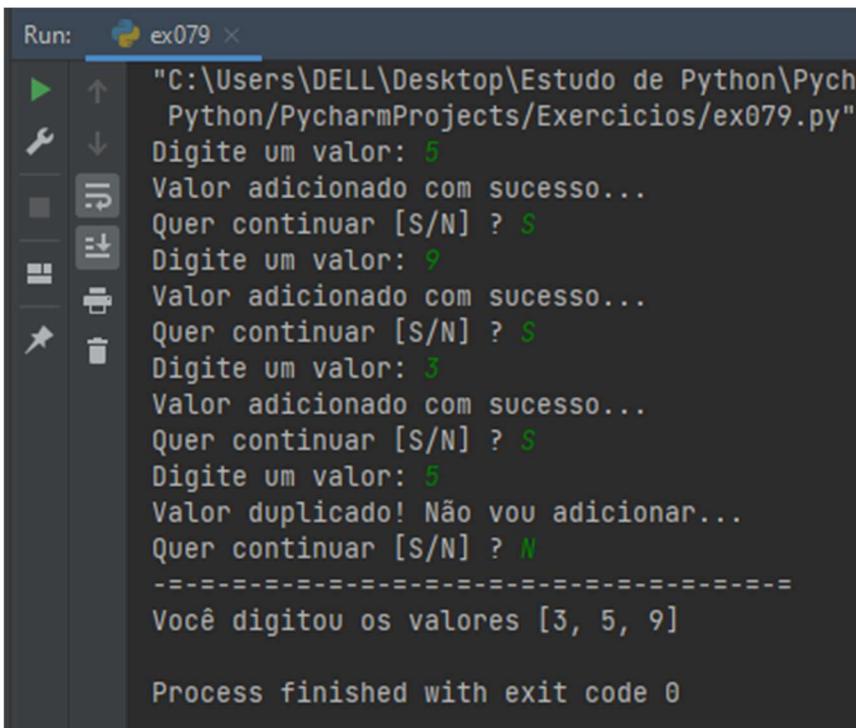
Crie um programa onde o usuário possa digitar vários valores numéricos e cadastre-os em uma lista. Caso o número já exista lá dentro, ele não será adicionado. No final, serão exibidos todos os valores únicos digitados, em ordem crescente.

Código:



```
1  listanum = []
2  while True:
3      valor = int(input('Digite um valor: '))
4      if valor not in listanum:
5          listanum.append(valor)
6          print('Valor adicionado com sucesso...')
7      else:
8          print('Valor duplicado! Não vou adicionar...')
9      continuar = str(input('Quer continuar [S/N] ? ')).strip().upper()
10     while continuar[0] not in 'SN':
11         continuar = str(input('Resposta Inválida. Você quer continuar [S/N] ? ')).strip().upper()
12     if continuar[0] in 'N':
13         break
14     print('-'*20)
15     listanum.sort()
16     print(f'Você digitou os valores {listanum}')
17 |
```

Run:



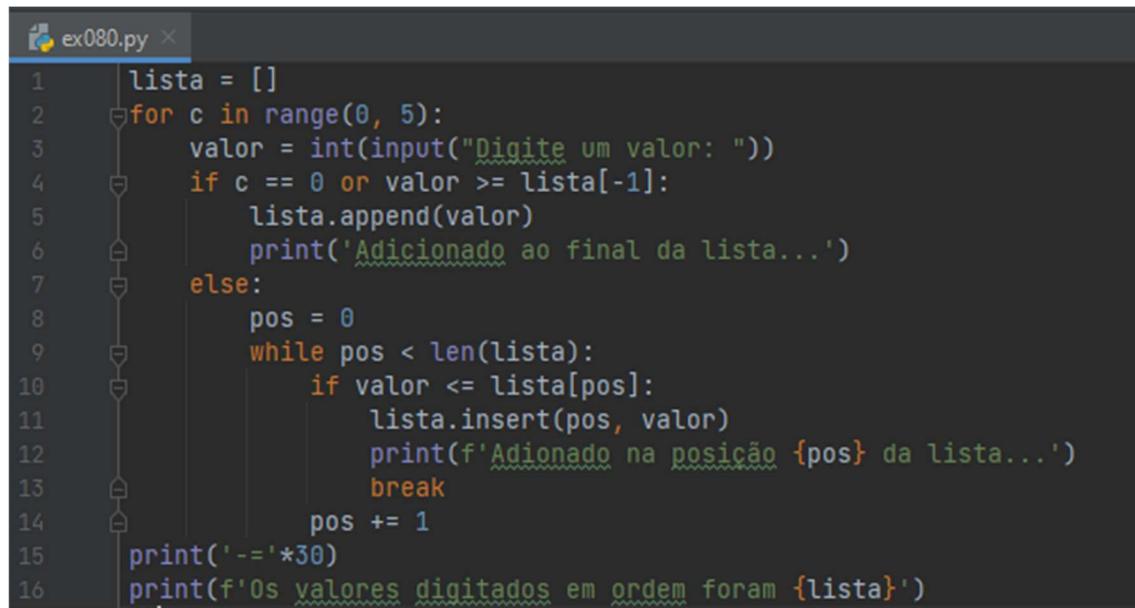
```
Run: ex079 ×
C:\Users\DELL\Desktop\Estudo de Python\Python\PycharmProjects\Exercicios\ex079.py"
Digite um valor: 5
Valor adicionado com sucesso...
Quer continuar [S/N] ? S
Digite um valor: 9
Valor adicionado com sucesso...
Quer continuar [S/N] ? S
Digite um valor: 3
Valor adicionado com sucesso...
Quer continuar [S/N] ? S
Digite um valor: 5
Valor duplicado! Não vou adicionar...
Quer continuar [S/N] ? N
=====
Você digitou os valores [3, 5, 9]

Process finished with exit code 0
```

## Exercício 80 – Lista Ordenada Sem Repetições

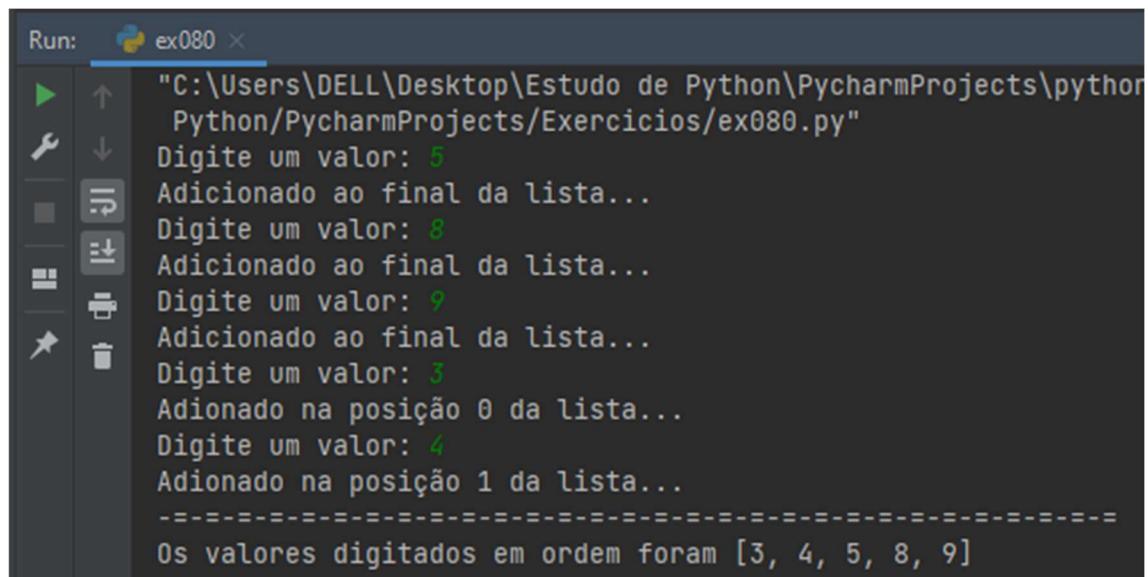
Crie um programa onde o usuário possa digitar cinco valores numéricos e cadastre-os em uma lista, já na posição correta de inserção (sem usar o `sort()`). No final, mostre a lista ordenada na tela.

Código:



```
ex080.py x
1 lista = []
2 for c in range(0, 5):
3     valor = int(input("Digite um valor: "))
4     if c == 0 or valor >= lista[-1]:
5         lista.append(valor)
6         print('Adicionado ao final da lista...')
7     else:
8         pos = 0
9         while pos < len(lista):
10            if valor <= lista[pos]:
11                lista.insert(pos, valor)
12                print(f'Adicionado na posição {pos} da lista...')
13                break
14            pos += 1
15
16 print('--'*30)
17 print(f'Os valores digitados em ordem foram {lista}')
```

Run:



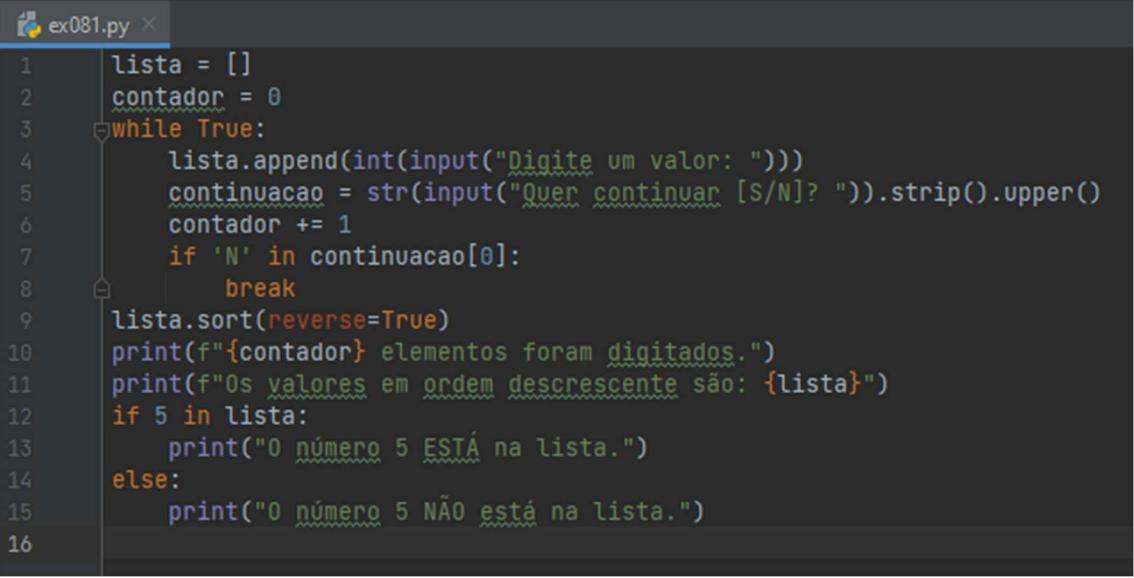
```
Run: ex080 x
C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\pythonfor
Python\PycharmProjects\Exercicios\ex080.py"
Digite um valor: 5
Adicionado ao final da lista...
Digite um valor: 8
Adicionado ao final da lista...
Digite um valor: 9
Adicionado ao final da lista...
Digite um valor: 3
Adicionado na posição 0 da lista...
Digite um valor: 4
Adicionado na posição 1 da lista...
=====
Os valores digitados em ordem foram [3, 4, 5, 8, 9]
```

## Exercício 081 – Extraíndo Dados de uma Lista

**Crie um programa que vai ler vários números e colocar em uma lista.  
Depois disso, mostre:**

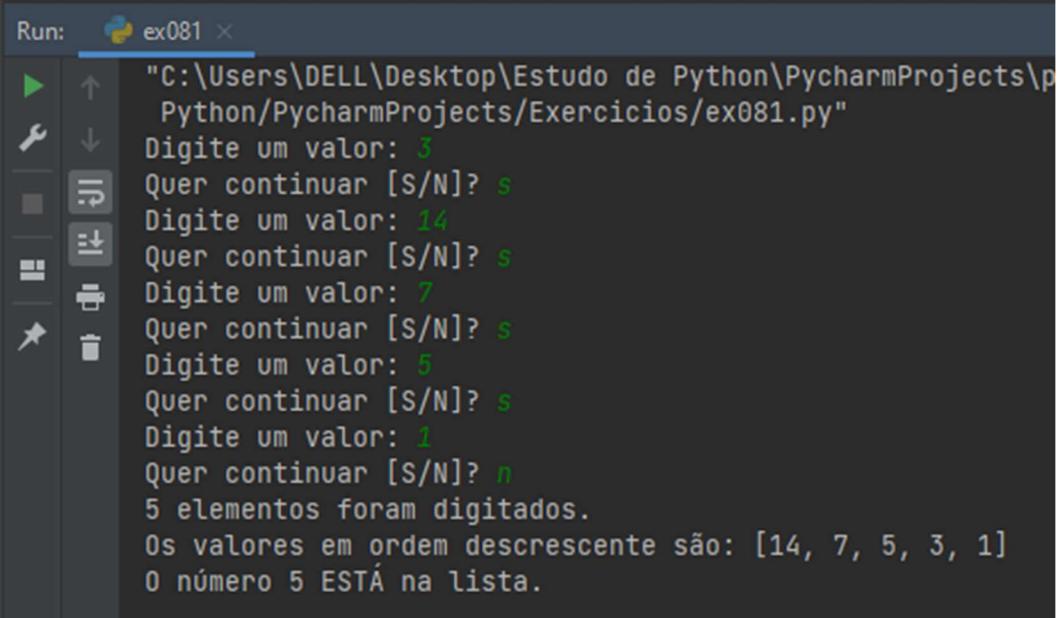
- **Quantos números foram digitados.**
- **A lista de valores, ordenada de forma decrescente.**
- **Se o valor 5 foi digitado e está ou não na lista.**

Código:



```
ex081.py ×
1  lista = []
2  contador = 0
3  while True:
4      lista.append(int(input("Digite um valor: ")))
5      continuaçao = str(input("Quer continuar [S/N]? ")).strip().upper()
6      contador += 1
7      if 'N' in continuaçao[0]:
8          break
9  lista.sort(reverse=True)
10 print(f"{contador} elementos foram digitados.")
11 print(f"Os valores em ordem descrescente são: {lista}")
12 if 5 in lista:
13     print("O número 5 ESTÁ na lista.")
14 else:
15     print("O número 5 NÃO está na lista.")
16
```

Run:



```
Run: ex081 ×
C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\Python\PycharmProjects\Exercicios/ex081.py
Digite um valor: 3
Quer continuar [S/N]? s
Digite um valor: 14
Quer continuar [S/N]? s
Digite um valor: 7
Quer continuar [S/N]? s
Digite um valor: 5
Quer continuar [S/N]? s
Digite um valor: 1
Quer continuar [S/N]? n
5 elementos foram digitados.
Os valores em ordem descrescente são: [14, 7, 5, 3, 1]
O número 5 ESTÁ na lista.
```

## Exercício 082 – Dividindo Valores em Várias Listas

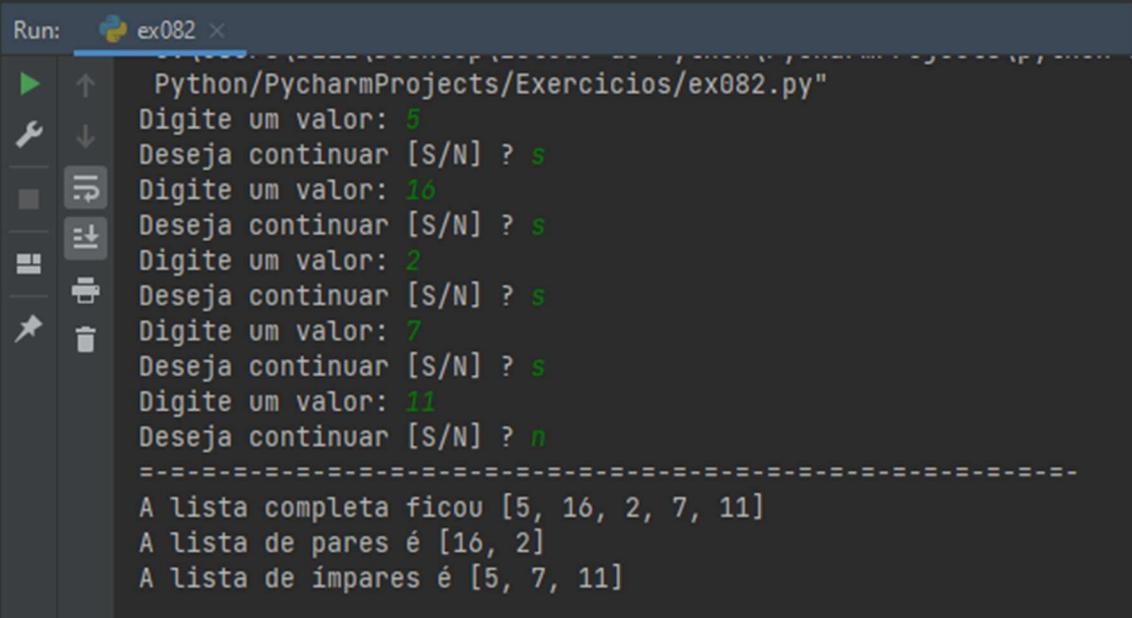
**Crie um programa que vai ler vários números e colocar em uma lista. Depois disso, crie duas listas extras que vão conter apenas os valores pares e os valores ímpares digitados, respectivamente. Ao final, mostre o conteúdo das três listas geradas.**

Código:



```
1 lista1 = []
2 listapar = []
3 listaimpar = []
4
5 while True:
6     valor = int(input("Digite um valor: "))
7     lista1.append(valor)
8     if valor % 2 == 0:
9         listapar.append(valor)
10    else:
11        listaimpar.append(valor)
12    continuaçao = str(input("Deseja continuar [S/N] ? ")).strip().upper()
13    if "N" in continuaçao:
14        break
15    print("-" * 30)
16    print(f"A lista completa ficou {lista1}")
17    print(f"A lista de pares é {listapar}")
18    print(f"A lista de ímpares é {listaimpar}")
19
```

Run:

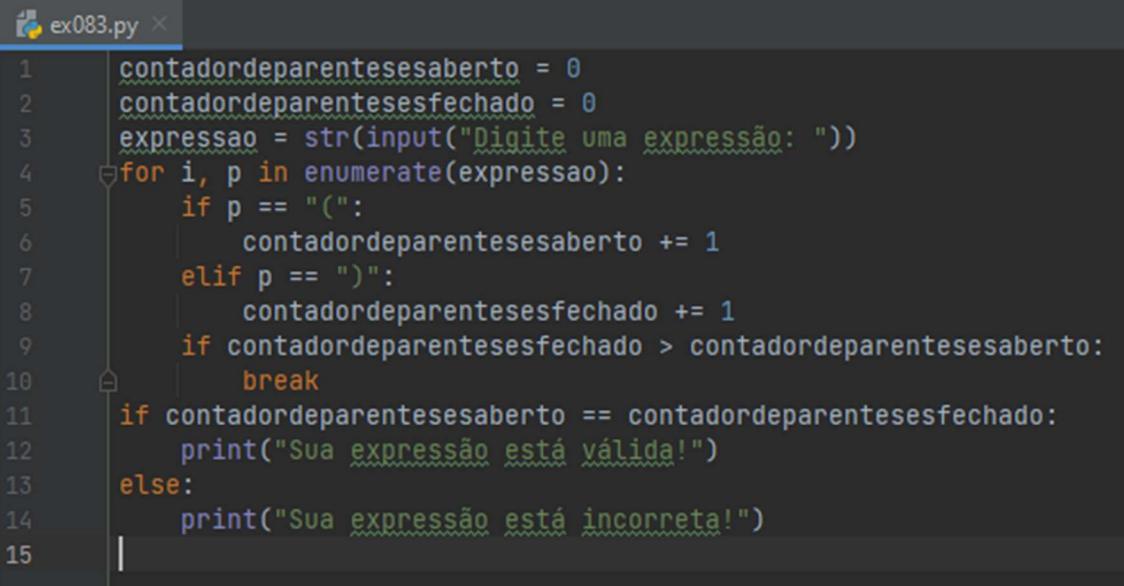


```
Run: ex082 ×
Python/PycharmProjects/Exercicios/ex082.py"
Digite um valor: 5
Deseja continuar [S/N] ? s
Digite um valor: 16
Deseja continuar [S/N] ? s
Digite um valor: 2
Deseja continuar [S/N] ? s
Digite um valor: 7
Deseja continuar [S/N] ? s
Digite um valor: 11
Deseja continuar [S/N] ? n
=====
A lista completa ficou [5, 16, 2, 7, 11]
A lista de pares é [16, 2]
A lista de ímpares é [5, 7, 11]
```

## Exercício 083 – Validando Expressões Matemáticas

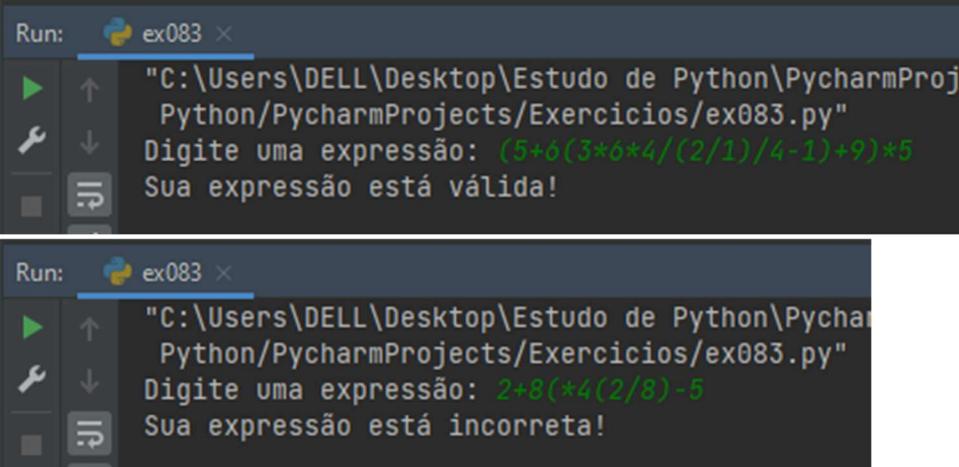
**Crie um programa onde o usuário digite uma expressão qualquer que use parênteses. Seu aplicativo deverá analisar se a expressão passada está com os parênteses abertos e fechados na ordem correta.**

Código:



```
1  contadordeparentesesaberto = 0
2  contadordeparentesesfechado = 0
3  expressao = str(input("Digite uma expressão: "))
4  for i, p in enumerate(expressao):
5      if p == "(":
6          contadordeparentesesaberto += 1
7      elif p == ")":
8          contadordeparentesesfechado += 1
9          if contadordeparentesesfechado > contadordeparentesesaberto:
10             break
11     if contadordeparentesesaberto == contadordeparentesesfechado:
12         print("Sua expressão está válida!")
13     else:
14         print("Sua expressão está incorreta!")
15 
```

Run:



Run:	Output
Run: ex083	"C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\Exercicios/ex083.py" Digite uma expressão: (5+6*(3*6*4/(2/1)/4-1)+9)*5 Sua expressão está válida!
Run: ex083	"C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\Exercicios/ex083.py" Digite uma expressão: 2+8(*4(2/8)-5 Sua expressão está incorreta!

## Listas Aninhadas

Dados é uma lista composta de duas chaves (0 e 1).

dados	
'Pedro'	25
0	1

Pessoas é uma outra lista, que nessa caso vai acoplar a lista dados dentro dela em uma única chave (dados[:]) significa que está havendo um fatiamento completo da estrutura de dados – vai copiar tudo que há nela).

pessoas	
'Pedro'	25
0	1
0	
<b>pessoas = list()</b>	
<b>pessoas.append(dados[:])</b>	

Assim, se pode repetir para as demais chaves com mais informações necessárias para o preenchimento completo de uma lista.

pessoas	
'Pedro'	25
0	1
0	
'Maria'	19
0	1
1	
'João'	32
0	1
2	

A lista, em código, fica representada dessa forma:

```
pessoas = [['Pedro', 25], ['Maria', 19], ['João', 32]]
```

Para printar na tela partes de uma lista ou chaves inteiras dela, utiliza-se o conceito de posição, como mostra a imagem abaixo:

**pessoas**

'Pedro' 25	'Maria' 19	'João' 32
0 1	0 1	0 1
0	1	2

```
print(pessoas[0][0])  
print(pessoas[1][1])  
print(pessoas[2][0])  
print(pessoas[1])
```

Pedro  
19  
João  
['Maria', 19]

O primeiro número faz referência à chave maior (em amarelo) e o segundo à chave menor (em branco). Quando se tem apenas um número entre colchetes, será apenas da chave maior, printando a inteira.

Nesse caso aqui, espera-se que o resultado desse uma lista assim:

[ [ "Gustavo", 40 ], [ "Maria", 22 ] ]

```
aula18.py  
1 teste = list()  
2 teste.append('Gustavo')  
3 teste.append(40)  
4 galera = list()  
5 galera.append(teste)  
6 teste[0] = 'Maria'  
7 teste[1] = 22  
8 galera.append(teste)  
9 print(galera)  
10  
Run: aula18  
/Library/Frameworks/Python.framework  
[['Maria', 22], ['Maria', 22]]
```

No entanto, ao fazer galera.append(teste) uma conexão é criada entre ambas as listas, e depois, ao tentar modificar as chaves 0 e 1 da lista teste, a modificação é feita na lista galera também. Para isso não ocorrer, é necessário utilizar a estrutura [:], que apenas faz uma cópia da lista e não gera uma conexão permanente entre elas.

The screenshot shows a PyCharm interface. In the top window, titled 'aula18.py', the following Python code is displayed:

```
1 teste = list()
2 teste.append('Gustavo')
3 teste.append(40)
4 galera = list()
5 galera.append(teste[:])
6 teste[0] = 'Maria'
7 teste[1] = 22
8 galera.append(teste[:])
9 print(galera)
10
```

In the bottom window, titled 'Run: aula18', the output of the code is shown:

```
/Library/Frameworks/Python.framework
[['Gustavo', 40], ['Maria', 22]]
```

Um exemplo prático utilizando listas aninhadas com a estrutura de cópia [:]:

The screenshot shows a PyCharm interface. In the top window, titled 'aula18.py', the following Python code is displayed:

```
1 galeria = list()
2 dado = list()
3 for c in range(0, 3):
4     dado.append(str(input('Nome: ')))
5     dado.append(int(input('Idade: ')))
6     galeria.append(dado[:])
7     dado.clear()
8
9 print(galeria)
```

In the bottom window, titled 'Run: aula18', the output of the code is shown:

```
Idade: 22
Nome: Maria
Idade: 33
Nome: Claudia
Idade: 55
[['Pedro', 22], ['Maria', 33], ['Claudia', 55]]
```

Outro exemplo:



```
aula18.py
1 galera = list()
2 dado = list()
3 totmai = totmen = 0
4 for c in range(0, 3):
5     dado.append(str(input('Nome: ')))
6     dado.append(int(input('Idade: ')))
7     galera.append(dado[:])
8     dado.clear()
9
10 for p in galera:
11     if p[1] >= 21:
12         print(f'{p[0]} é maior de idade.')
13         totmai += 1
14     else:
15         print(f'{p[0]} é menor de idade.')
16         totmen += 1
17
18 print(f'Temos {totmai} maiores e {totmen} menores de idade.')
19
```

## Exercício 084 – Lista Composta e Análise de Dados

**Faça um programa que leia nome e peso de várias pessoas, guardando tudo em uma lista. No final, mostre:**

- **Quantas pessoas foram cadastradas.**
- **Uma listagem com as pessoas mais pesadas.**
- **Uma listagem com as pessoas mais leves.**

Código:

```
 1  pessoaatual = []
 2  grupo = []
 3  pesadas = leves = 0
 4  while True:
 5      pessoaatual.append(str(input("Nome: ")))
 6      pessoaatual.append(float(input("Peso [em kg]: ")))
 7      if len(grupo) == 0:
 8          pesadas = leves = pessoaatual[1]
 9      else:
10          if pessoaatual[1] > pesadas:
11              pesadas = pessoaatual[1]
12          if pessoaatual[1] < leves:
13              leves = pessoaatual[1]
14      grupo.append(pessoaatual[:])
15      pessoaatual.clear()
16      continuacao = str(input("Quer continuar [S/N] ? ")).strip().upper()
17      if "N" in continuacao:
18          break
19
20      print("-" * 30)
21      print(f"Ao todo, você cadastrou {len(grupo)} pessoas.")
22      print(f"O maior peso foi de {pesadas}kg. Peso de ", end='')
23      for p in grupo:
24          if p[1] == pesadas:
25              print(f"[{p[0]}]", end=' ')
26      print()
27      print(f"O menor peso foi de {leves}kg. Peso de: ", end='')
28      for p in grupo:
29          if p[1] == leves:
30              print(f"[{p[0]}]", end="")
```

Run:

```
Run: ex084 ×
"C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\python
  Python\PycharmProjects\Exercicios/ex084.py"
Nome: Victor
Peso [em kg]: 70
Quer continuar [S/N] ? s
Nome: Fábio
Peso [em kg]: 52
Quer continuar [S/N] ? s
Nome: Ana
Peso [em kg]: 52
Quer continuar [S/N] ? s
Nome: Vladimir
Peso [em kg]: 92
Quer continuar [S/N] ? s
Nome: Juliana
Peso [em kg]: 92
Quer continuar [S/N] ? n
=====
Ao todo, você cadastrou 5 pessoas.
O maior peso foi de 92.0kg. Peso de [Vladimir][Juliana]
O menor peso foi de 52.0kg. Peso de: [Fábio][Ana]
Process finished with exit code 0
```

## Exercício 085 – Listas com Pares e Ímpares

Crie um programa onde o usuário possa digitar sete valores numéricos e cadastre-os em uma lista única que mantenha separados os valores pares e ímpares. No final, mostre os valores pares e ímpares em ordem crescente.

Código:

```
Run: ex085.py ×
1     listadenumeros = [[], []]
2     for c in range(1, 8):
3         n = int(input(f"Digite o {c}º valor: "))
4         if n % 2 == 0:
5             listadenumeros[0].append(n)
6         else:
7             listadenumeros[1].append(n)
8     listadenumeros[0].sort()
9     listadenumeros[1].sort()
10    print("-" * 30)
11    print(f"Os valores pares digitados foram: {listadenumeros[0]}")
12    print(f"Os valores ímpares digitados foram: {listadenumeros[1]}")
13
```

Run:

```
Run: ex085 x
"C:\Users\DELL\Desktop\Estudo de Python\PycharmProj
Python\PycharmProjects/Exercicios/ex085.py"
Digite o 1º valor: 5
Digite o 2º valor: 8
Digite o 3º valor: 3
Digite o 4º valor: 2
Digite o 5º valor: 9
Digite o 6º valor: 0
Digite o 7º valor: 7
Os valores pares digitados foram: [0, 2, 8]
Os valores ímpares digitados foram: [3, 5, 7, 9]
```

## Exercício 086 – Matriz em Python

**Crie um programa que declare uma matriz de dimensão 3×3 e preencha com valores lidos pelo teclado. No final, mostre a matriz na tela, com a formatação correta.**

Código:

```
ex086.py x
1 matriz = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
2 for linha in range(0, len(matriz)):
3     for coluna in range(0, len(matriz[0])):
4         matriz[linha][coluna] = int(input("Digite um número para [{0}, {1}]: ".format(linha, coluna)))
5 for linha in range(0, len(matriz)):
6     for coluna in range(0, len(matriz[0])):
7         print(f"[{matriz[linha][coluna]}]", end=' ')
8     print()
```

Run:

```
"C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects/Exercicios/ex086.py"
Digite um número para [0, 0]: 123
Digite um número para [0, 1]: 53
Digite um número para [0, 2]: 5432
Digite um número para [1, 0]: 448
Digite um número para [1, 1]: 3
Digite um número para [1, 2]: 8
Digite um número para [2, 0]: 549
Digite um número para [2, 1]: 54
Digite um número para [2, 2]: 7
[ 123 ][ 53 ][5432 ]
[ 448 ][ 3 ][ 8 ]
[ 549 ][ 54 ][ 7 ]
```

## Exercício 087 – Mais sobre Matrizes em Python

Aprimore o desafio anterior, mostrando no final:

A) A soma de todos os valores pares digitados.

B) A soma dos valores da terceira coluna.

C) O maior valor da segunda linha.

Código:

```
ex087.py x
1 matriz = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
2 somapar = somacoluna3 = maiorlinha2 = 0
3 for linha in range(0, len(matriz)):
4     for coluna in range(0, len(matriz[linha])):
5         matriz[linha][coluna] = int(input("Digite um número para [{linha}, {coluna}]: "))
6         if matriz[linha][coluna] % 2 == 0:
7             somapar += matriz[linha][coluna]
8             if coluna == 2:
9                 somacoluna3 += matriz[linha][coluna]
10            if linha == 1 and coluna == 0:
11                maiorlinha2 = matriz[linha][coluna]
12            elif linha == 1 and coluna > 0:
13                if matriz[linha][coluna] > maiorlinha2:
14                    maiorlinha2 = matriz[linha][coluna]
15    print("-" * 30)
16    for linha in range(0, len(matriz)):
17        for coluna in range(0, len(matriz[linha])):
18            print(f"[{matriz[linha][coluna]}]", end=' ')
19    print()
20    print("-" * 30)
21    print("A soma de todos os valores pares digitados foi ", somapar)
22    print("A soma dos valores da terceira coluna foi ", somacoluna3)
23    print("O maior valor da segunda linha foi ", maiorlinha2)
```

Run:

```

"C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\python-
Python\PycharmProjects\Exercicios/ex087.py"
Digite um número para [0, 0]: 1
Digite um número para [0, 1]: 2
Digite um número para [0, 2]: 3
Digite um número para [1, 0]: 4
Digite um número para [1, 1]: 5
Digite um número para [1, 2]: 6
Digite um número para [2, 0]: 7
Digite um número para [2, 1]: 8
Digite um número para [2, 2]: 9
=====
[ 1 ][ 2 ][ 3 ]
[ 4 ][ 5 ][ 6 ]
[ 7 ][ 8 ][ 9 ]
=====
A soma de todos os valores pares digitados foi 20
A soma dos valores da terceira coluna foi 18
O maior valor da segunda linha foi 6

```

## Exercício 088 – Palpites para a Mega Sena

**Faça um programa que ajude um jogador da MEGA SENA a criar palpites. O programa vai perguntar quantos jogos serão gerados e vai sortear 6 números entre 1 e 60 para cada jogo, cadastrando tudo em uma lista composta.**

Código:

```

ex088.py ×
1  from random import randint
2  numeros = []
3  listadejogos = []
4  print("=" * 30 + "\n" + "JOGA NA MEGA SENA".center(30, " ") + "\n" + "=" * 30 + "\n")
5  jogos = int(input("Quantos jogos você quer que eu sorteie? "))
6  print()
7  print("-" * 3 + f" SORTEANDO {jogos} JOGOS " + "-" * 3)
8  for sorteadordejogos in range(0, jogos):
9      for sorteadordenumeros in range(0, 6):
10         while True:
11             x = randint(1, 60)
12             if x not in numeros:
13                 numeros.append(x)
14                 break
15         numeros.sort()
16         listadejogos.append(numeros[:])
17         print(f"JOGO {sorteadordejogos + 1}: {listadejogos[sorteadordejogos]}")
18         numeros.clear()
19     print()
20     print("-" * 5 + " < BOA SORTE! > " + "-" * 5)
21

```

Run:

```
Run: ex088 x
"C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\Exercicios\ex088.py"
=====
    JOGA NA MEGA SENA
=====

Quantos jogos você quer que eu sorteie? 5
===== SORTEANDO 5 JOGOS =====
JOGO 1:[16, 27, 38, 40, 49, 53]
JOGO 2:[11, 14, 18, 32, 40, 54]
JOGO 3:[4, 17, 35, 38, 47, 53]
JOGO 4:[9, 22, 32, 33, 37, 53]
JOGO 5:[10, 15, 26, 36, 47, 52]

===== < BOA SORTE! > =====
```

## Exercício 89 – Boletim com Listas Compostas

**Crie um programa que leia nome e duas notas de vários alunos e guarde tudo em uma lista composta. No final, mostre um boletim contendo a média de cada um e permita que o usuário possa mostrar as notas de cada aluno individualmente.**

Código:

```
ex089.py x
1 print()
2 print("=" * 20 + " GERANDO BOLETIM DOS ALUNOS " + "=" * 20)
3 sala = []
4 aluno = []
5 contadordealunos = 0
6 while True:
7     aluno.append(str(input("Nome do aluno: ")).strip())
8     aluno.append(float(input("Nota 1: ")))
9     aluno.append(float(input("Nota 2: ")))
10    sala.append(aluno[:])
11    aluno.clear()
12    contadordealunos += 1
13    continuacao = str(input("Quer continuar [S/N]? ")).upper().strip()
14    if "N" in continuacao:
15        break
16    print("-" * 13)
17    print("No. NOME           MÉDIA   ")
18    print("-" * 26)
19    for aluno in range(0, contadordealunos):
20        media = (sala[aluno][1] + sala[aluno][2]) / 2
21        print(f"{aluno:<5}" + f"{sala[aluno][0]:<15}" + f"{media}")
22    while True:
23        print("-" * 40)
24        consultanota = int(input("Mostrar notas de qual aluno (999 interrompe) ? "))
25        if consultanota == 999:
26            break
27        print(f"As notas de {sala[consultanota][0]} foram {sala[consultanota][1]} e {sala[consultanota][2]}")
28    print("Finalizando...\n      <<= VOLTE SEMPRE =>>")
```

Run:

```
===== GERANDO BOLETIM DOS ALUNOS =====
Nome do aluno: Victor
Nota 1: 10
Nota 2: 9.5
Quer continuar [S/N]? s
Nome do aluno: Thomas
Nota 1: 8
Nota 2: 8.5
Quer continuar [S/N]? s
Nome do aluno: Marcos
Nota 1: 7
Nota 2: 9.5
Quer continuar [S/N]? n
-----
No. NOME MÉDIA
-----
0 Victor 9.75
1 Thomas 8.25
2 Marcos 8.25
-----
Mostrar notas de qual aluno (999 interrompe) ? 0
As notas de Victor foram 10.0 e 9.5.
-----
Mostrar notas de qual aluno (999 interrompe) ? 2
As notas de Marcos foram 7.0 e 9.5.
-----
Mostrar notas de qual aluno (999 interrompe) ? 999
Finalizando...
<<<= VOLTE SEMPRE =>>>
```

## Dicionários

Parecidos com as listas, os dicionários têm a função de armazenar valores em uma mesma estrutura, acessíveis por chaves literais, isto é, com uma personalização dos índices. É representado por chaves - {}.

```
dados
['Pedro' 25]
    nome  idade
```

```
dados = dict()
dados = { 'nome': 'Pedro', 'idade': 25 }
print(dados['nome'])  Pedro
print(dados['idade']) 25
```

Outra forma de adicionar novos elementos, tem-se também essa outra forma de declaração:

```
dados
['Pedro' 25 'M']
    nome  idade sexo
```

```
dados['sexo'] = 'M'
```

Para apagar espaços dentro dos dicionários, utiliza-se o comando del:

```
dados
['Pedro' 'M']
    nome  sexo
```

```
del dados['idade']
```

Outra coisa importante de se ter em mente ao estudar sobre dicionários é a diferença entre valores (values) , chaves (keys) e itens (items).

```
filme
['Star Wars' 1977 'George Lucas']
    titulo      ano      diretor
print(filme.values())
print(filme.keys())
print(filme.items())
```

O primeiro comando só retornaria: "Star Wars", 1977, "George Lucas".

O segundo: título, ano, diretor.

Já o terceiro, ambos os valores: título Star Wars, ano 1977, diretor "George Lucas".

Utilizando esses conceitos em uma estrutura de repetição, esses seriam os outputs:

```
filme
['Star Wars' 1977 'George Lucas']
    titulo      ano      diretor
for k, v in filme.items():
    print(f'O {k} é {v}')
```

```
O título é Star Wars
O ano é 1977
O diretor é George Lucas
```

Há também a possibilidade de combinar dicionários com tuplas e listas. Com listas, ficaria dessa forma, por exemplo:

The screenshot shows a Python code editor with the following content:

```
locadora = [
    {'titulo': 'Star Wars', 'ano': 1977, 'diretor': 'George Lucas'},
    {'titulo': 'Avengers', 'ano': 2012, 'diretor': 'Joss Whedon'},
    {'titulo': 'Matrix', 'ano': 1999, 'diretor': 'Wachowski'},

    print(locadora[0]['ano'])      1977
    print(locadora[2]['titulo'])   Matrix
```

## Exercício 090 – Dicionário em Python

Faça um programa que leia nome e média de um aluno, guardando também a situação em um dicionário. No final, mostre o conteúdo da estrutura na tela.

Código:

The screenshot shows a Python code editor with the file named `ex090.py`. The code defines a dictionary `aluno` and calculates the student's situation based on their average grade:

```
aluno = {}
aluno['nome'] = str(input("Nome: "))
aluno['média'] = float(input(f"Média de {aluno['nome']}: "))
if aluno['média'] >= 7:
    aluno['situação'] = 'APROVADO'
elif 5 <= aluno['média'] < 7:
    aluno['situação'] = 'RECUPERAÇÃO'
else:
    aluno['situação'] = 'NÃO APROVADO'
print("-=" * 20)
for k, v in aluno.items():
    print(f"- {k} é igual a {v}.")
```

Run:

```
Run: ex090 ×
C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\Exercicios\ex090.py
Nome: Victor
Média de Victor: 5
=====
- nome é igual a Victor.
- média é igual a 5.0.
- situação é igual a RECUPERAÇÃO.

Process finished with exit code 0
```

## Exercício 091 – Jogo de Dados em Python

Crie um programa onde 4 jogadores joguem um dado e tenham resultados aleatórios. Guarde esses resultados em um dicionário em Python. No final, coloque esse dicionário em ordem, sabendo que o vencedor tirou o maior número no dado.

Código:

```
1  from random import randint
2  from time import sleep
3  from operator import itemgetter
4  jogadas = {'jogador1': randint(1, 6), 'jogador2': randint(1, 6),
5             'jogador3': randint(1, 6), 'jogador4': randint(1, 6)}
6  print("Valores sorteados: ")
7  for k, v in jogadas.items():
8      print(f"O {k} tirou {v} no dado.")
9      sleep(1)
10 print("-" * 20)
11 print(" == RANKING DOS JOGADORES == ")
12 ranking = []
13 ranking = sorted(jogadas.items(), key=itemgetter(1), reverse=True)
14 for pos, v in enumerate(ranking):
15     print(f"    {pos+1}º lugar: {v[0]} com {v[1]}.")
16     sleep(1)
```

Run:

```
Run: ex091 ×
C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\Exercicios\ex091.py
Valores sorteados:
0 jogador1 tirou 5 no dado.
0 jogador2 tirou 5 no dado.
0 jogador3 tirou 2 no dado.
0 jogador4 tirou 1 no dado.
=====
== RANKING DOS JOGADORES ==
1º lugar: jogador1 com 5.
2º lugar: jogador2 com 5.
3º lugar: jogador3 com 2.
4º lugar: jogador4 com 1.

Process finished with exit code 0
```

## Exercício 92 – Cadastro de Trabalhador em Python

Crie um programa que leia nome, ano de nascimento e carteira de trabalho e cadastre-o (com idade) em um dicionário. Se por acaso a CTPS for diferente de ZERO, o dicionário receberá também o ano de contratação e o salário. Calcule e acrescente, além da idade, com quantos anos a pessoa vai se aposentar.

Código:

```
from datetime import datetime
ctps = {'nome': str(input('Nome: ')), 'idade': datetime.now().year - int(input('Ano de nascimento: ')),
        'carteira': int(input('Carteira de Trabalho (0 = não tem): '))}
if ctps['carteira'] != 0:
    ctps['contratação'] = int(input("Ano de contratação: "))
    ctps['salário'] = float(input("Salário: R$"))
    ctps['aposentadoria'] = ctps['contratação'] - (datetime.now().year - ctps['idade']) + 35
print("-" * 20)
for k, v in ctps.items():
    print(f"{k} tem o valor {v}.")
```

Run:

```
Run: ex092 ×
C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\Exercicios\ex092.py"
Nome: Victor
Ano de nascimento: 2002
Carteira de Trabalho (0 = não tem): 23154
Ano de contratação: 2021
Salário: R$885
=====
nome tem o valor Victor.
idade tem o valor 20.
carteira tem o valor 23154.
contratação tem o valor 2021.
salário tem o valor 885.0.
aposentadoria tem o valor 54.
```

## Exercício 093 – Cadastro de Jogador de Futebol

Crie um programa que gerencie o aproveitamento de um jogador de futebol. O programa vai ler o nome do jogador e quantas partidas ele jogou. Depois vai ler a quantidade de gols feitos em cada partida. No final, tudo isso será guardado em um dicionário, incluindo o total de gols feitos durante o campeonato.

Código:

```
1 contagemdegols = {'jogador': str(input("Nome do jogador: "))} # Nome do jogador
2 partidas = int(input(f"Quantas partidas {contagemdegols['jogador']} jogou? "))
3 golspartida = []
4 for c in range(0, partidas):
5     golspartida.append(int(input(f"    Quantos gols na partida {c}? ")))
6 contagemdegols['gols'] = golspartida[:]
7 contagemdegols['total'] = sum(golspartida) # sum: soma tudo que estiver dentro da lista
8 print("-" * 20)
9 print(contagemdegols)
10 print("-" * 20)
11 for k, v in contagemdegols.items():
12     print(f"O campo {k} tem valor {v}.")
13 print("-" * 20)
14 print(f"O jogador {contagemdegols['jogador']} jogou {len(contagemdegols['gols'])} partidas.")
15 for i, v in enumerate(contagemdegols['gols']):
16     print(f"    => Na partida {i}, fez {v} gols.")
17 print(f"Fez um total de {contagemdegols['total']} gols.")
```

Run:

The screenshot shows the PyCharm IDE interface with the 'Run' tab selected. The project path is 'C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\python-3\Python\PycharmProjects/Exercicios/ex093.py'. The output window displays the following text:

```
"C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\python-3\Python\PycharmProjects/Exercicios/ex093.py"
Nome do jogador: Mesquita
Quantas partidas Mesquita jogou? 5
    Quantos gols na partida 0? 3
    Quantos gols na partida 1? 4
    Quantos gols na partida 2? 1
    Quantos gols na partida 3? 2
    Quantos gols na partida 4? 0
=====
{'jogador': 'Mesquita', 'gols': [3, 4, 1, 2, 0], 'total': 10}
=====
O campo jogador tem valor Mesquita.
O campo gols tem valor [3, 4, 1, 2, 0].
O campo total tem valor 10.
=====
O jogador Mesquita jogou 5 partidas.
=> Na partida 0, fez 3 gols,
=> Na partida 1, fez 4 gols,
=> Na partida 2, fez 1 gols,
=> Na partida 3, fez 2 gols,
=> Na partida 4, fez 0 gols,
Fez um total de 10 gols.
```

## Exercício 094 – Unindo Dicionários e Listas

**Crie um programa que leia nome, sexo e idade de várias pessoas, guardando os dados de cada pessoa em um dicionário e todos os dicionários em uma lista. No final, mostre:**

- A) Quantas pessoas foram cadastradas;**
- B) A média de idade;**
- C) Uma lista com as mulheres**
- D) Uma lista de pessoas com idade acima da média.**

Código:

```
ex094.py x
1 pessoa = dict()
2 listadecadastro = list()
3 somaidade = 0
4 while True:
5     pessoa['nome'] = str(input("Nome: "))
6     while True:
7         pessoa['sexo'] = str(input("Sexo: [M/F] ")).strip().upper()[0]
8         if pessoa['sexo'] in "MF":
9             break
10        print("ERRO! Por favor, digite apenas M ou F.")
11    pessoa['idade'] = int(input("Idade: "))
12    somaidade += pessoa['idade']
13    while True:
14        continuar = str(input("Deseja continuar? [S/N] ")).strip().upper()
15        if continuar in "SN":
16            break
17        print("ERRO! Responda com S ou N.")
18    listadecadastro.append(pessoa.copy())
19    if continuar == "N":
20        break
21 mediaidade = somaidade / len(listadecadastro)
22 print("-" * 20)
23 print(f"A) Ao todo temos {len(listadecadastro)} pessoas cadastradas.")
24 print(f"B) A média de idade é de {mediaidade:.2f} anos.")
25 print("C) As mulheres cadastradas foram", end=' ')
26 for p in listadecadastro:
27     if p['sexo'] == 'F':
28         print(p['nome'], end=' ')
29 print("\nD) Lista das pessoas que estão acima da média:")
30 for p in listadecadastro:
31     if p['idade'] > mediaidade:
32         print(' ', end='')
33         for k, v in p.items():
34             print(f"{k} = {v};", end=' ')
35         print()
36 print("<< ENCERRADO >>")
37
```

Run:

```
Run: ex094 ×
C:\Users\DELL\Desktop\Estudo de Python\Pyc
Nome: Victor
Sexo: [M/F] m
Idade: 20
Deseja continuar? [S/N] s
Nome: Vanessa
Sexo: [M/F] f
Idade: 24
Deseja continuar? [S/N] s
Nome: Anderson
Sexo: [M/F] m
Idade: 12
Deseja continuar? [S/N] s
Nome: Laura
Sexo: [M/F] f
Idade: 32
Deseja continuar? [S/N] s
Nome: Mariana
Sexo: [M/F] f
Idade: 27
Deseja continuar? [S/N] n
=====
```

```
=====
A) Ao todo temos 5 pessoas cadastradas.
B) A média de idade é de 23.00 anos.
C) As mulheres cadastradas foram Vanessa Laura Mariana
D) Lista das pessoas que estão acima da média:
    nome = Vanessa; sexo = F; idade = 24;
    nome = Laura; sexo = F; idade = 32;
    nome = Mariana; sexo = F; idade = 27;
<< ENCERRADO >>
```

## Exercício 095 – Aprimorando os Dicionários

**Aprimore o desafio 93 para que ele funcione com vários jogadores, incluindo um sistema de visualização de detalhes do aproveitamento de cada jogador.**

Código:

```

ex095.py x
1  listadejogadores = []
2  contagemdegols = dict()
3  while True:
4      contagemdegols.clear()
5      jogador = str(input("Nome do jogador: "))
6      partidas = int(input(f"Quantas partidas {contagemdegols['jogador']} jogou? "))
7      golsnaspartidas = []
8      for c in range(0, partidas):
9          golsnaspartidas.append(int(input(f"    Quantos gols na partida {c}? ")))
10     contagemdegols['gols'] = golsnaspartidas[:]
11     contagemdegols['total'] = sum(golsnaspartidas) # sum: soma tudo que estiver dentro da lista
12     listadejogadores.append(contagemdegols.copy())
13  while True:
14      continuar = str(input("Quer continuar? [S/N] ")).strip().upper()[0]
15      if continuar in "SN":
16          break
17      print("RESPOSTA INVÁLIDA! Tente novamente.")
18  if continuar == "N":
19      break
20
21  print('-' * 20)
22  print('cod ', end='')
23  for i in contagemdegols.keys():
24      print(f'{i:<15}', end=' ')
25  print()
26  print('-' * 40)
27  for k, v in enumerate(listadejogadores):
28      print(f'{k}'.rjust(3), end=' ')
29      for d in v.values():
30          print(f'{d}'.ljust(15), end=' ')
31      print()
32  while True:
33      print('-' * 40)
34      dados = int(input("Mostrar dados de qual jogador? (999 para parar) "))
35      if dados == 999:
36          print('<< VOLTE SEMPRE >>')
37          break
38      if len(listadejogadores)-1 < dados or dados < 0:
39          print("NENHUM JOGADOR POSSUI ESSE CÓDIGO. Tente outro número.")
40      else:
41          print(f' == LEVANTAMENTO DO JOGADOR {listadejogadores[dados]["jogador"]}:')
42          for i, v in enumerate(listadejogadores[dados]['gols']):
43              print(f'    No jogo {i+1} fez {v} gols.')

```

Run:

```
Run: ex095 x
"C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\Exercicios/ex095.py"
Nome do jogador: Victor
Quantas partidas Victor jogou? 4
    Quantos gols na partida 0? 2
    Quantos gols na partida 1? 1
    Quantos gols na partida 2? 3
    Quantos gols na partida 3? 3
Quer continuar? [S/N] s
Nome do jogador: Fabiano
Quantas partidas Fabiano jogou? 1
    Quantos gols na partida 0? 5
Quer continuar? [S/N] s
Nome do jogador: Rodolfo
Quantas partidas Rodolfo jogou? 3
    Quantos gols na partida 0? 2
    Quantos gols na partida 1? 1
    Quantos gols na partida 2? 1
Quer continuar? [S/N] n
=====
cod jogador      gols      total
-----
  0 Victor        [2, 1, 3, 3]   9
  1 Fabiano       [5]          5
  2 Rodolfo       [2, 1, 1]     4
=====
```

```
=====
cod jogador      gols      total
-----
  0 Victor        [2, 1, 3, 3]   9
  1 Fabiano       [5]          5
  2 Rodolfo       [2, 1, 1]    4
-----
Mostrar dados de qual jogador? (999 para parar) 0
== LEVANTAMENTO DO JOGADOR Victor:
  No jogo 1 fez 2 gols.
  No jogo 2 fez 1 gols.
  No jogo 3 fez 3 gols.
  No jogo 4 fez 3 gols.
-----
Mostrar dados de qual jogador? (999 para parar) 1
== LEVANTAMENTO DO JOGADOR Fabiano:
  No jogo 1 fez 5 gols.
-----
Mostrar dados de qual jogador? (999 para parar) 99
NENHUM JOGADOR POSSUI ESSE CÓDIGO. Tente outro número.
-----
Mostrar dados de qual jogador? (999 para parar) 999
<< VOLTE SEMPRE >>
```

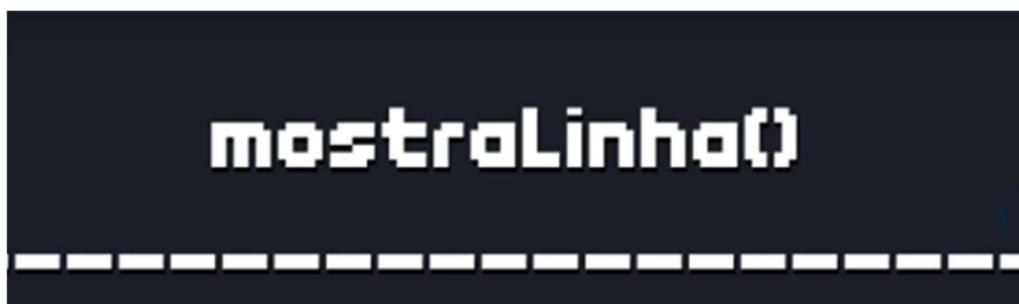
## Funções em Python

Há trechos dos nossos códigos que executamos com uma certa frequência no nosso dia a dia. Devido a essa necessidade de sempre escrever o mesmo código toda vez, criaram as funções/rotinas, que uma vez criada basta ser chamada para o código que ela será executada.

Exemplo de funções utilizadas nos códigos até o momento:

**print()** **len()**  
**input()** **int()**  
**float()**

Se você tem a rotina de criar uma linha em todo programa elaborado por você, basta criar uma função chamada mostraLinha( ), por exemplo:



Assim, toda vez que necessitar criar uma linha, basta chamar essa função.



Assim é o código sem a função. Nada prático e repetitivo

```
print('-----')
print('      SISTEMA DE ALUNOS      ')
print('-----')
print('-----')
print('      CADASTRO DE FUNCIONÁRIOS    ')
print('-----')
print('-----')
print('      ERRO DO SISTEMA      ')
print('-----')
```

Já com a função, ele fica mais limpo e funcional.

```
def mostraLinha():
    print('-----')

mostraLinha()
print('      SISTEMA DE ALUNOS      ')
mostraLinha()
mostraLinha()
print('      CADASTRO DE FUNCIONÁRIOS      ')
mostraLinha()
mostraLinha()
print('      ERRO DO SISTEMA      ')
mostraLinha()
```

Nota-se que a palavrinha mágica para invocar essas funções é a palavra def. Sempre será utilizada quando uma rotina precisar ser criada.

Mas essa é uma função muito básica para ser criada. Basta analisar o código e verá que outras partes do código se repetem. Apenas a mensagem central que é alterada. Então, basta criar uma rotina que atenda toda essa necessidade:

```
def mensagem(msg):
    print('-----')
    print(msg)
    print('-----')

mensagem('SISTEMA DE ALUNOS')
```

Nesse caso, o que está entre os parênteses é chamado de parâmetros (partes criadas no código para referenciar dentro das funções). Nesse caso, 'SISTEMA DE ALUNOS' é o parâmetro real (código principal) e o msg é o parâmetro formal (dentro da função).

Por exemplo, aqui as palavras “MESQUITA”, “PROGRAMAÇÃO” E “TECNOLOGIA” passaram para dentro da função como parâmetros virando o “txt” do código:

The screenshot shows a PyCharm interface. On the left, the code editor displays 'aula20b.py' with the following content:

```
1 def titulo(txt):
2     print('-'*30)
3     print(txt)
4     print('-'*30)
5
6
7     titulo(' MESQUITA ')
8     titulo(' PROGRAMAÇÃO ')
9     titulo(' TECNOLOGIA ')
```

On the right, the terminal window shows the output of running the script:

```
MESQUITA
-----
PROGRAMAÇÃO
-----
TECNOLOGIA
```

Exemplo matemático da utilidade de uma função:

The screenshot shows a PyCharm interface. On the left, the code editor displays 'ambiente.py' with the following content:

```
1 def soma(a, b):
2     s = a + b
3     print(s)
4
5
6 # Programa Principal
7 soma(4, 5)
8 soma(8, 9)
9 soma(2, 1)
```

On the right, the terminal window shows the output of running the script:

```
9
17
3
Process finished with exit code 0
```

Muitas vezes, não sabemos quantos parâmetros serão passados para dentro de uma função, seja lá o motivo disso. Ainda assim, temos uma solução para isso – o desempacotamento:

The screenshot shows a terminal window with the following text displayed:

```
def contador(*núm):
    pass

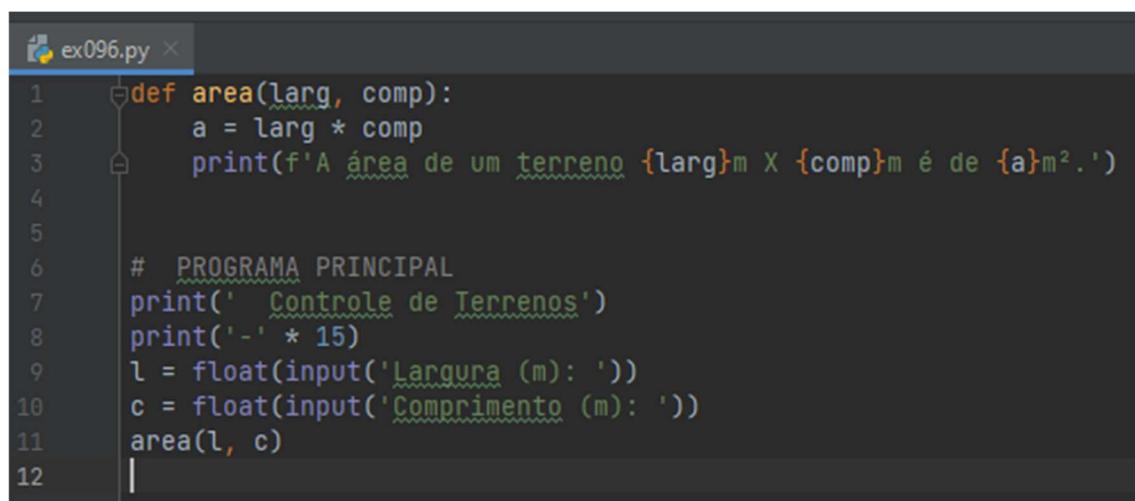
contador( 5,7,3,1,4 )
contador( 8,4,7 )
```

Ele é simbolizado por esse asterisco em azul e serve para contar a quantidade de parâmetros e anexá-los à num.

## Exercício 096 – Função que Calcula Área

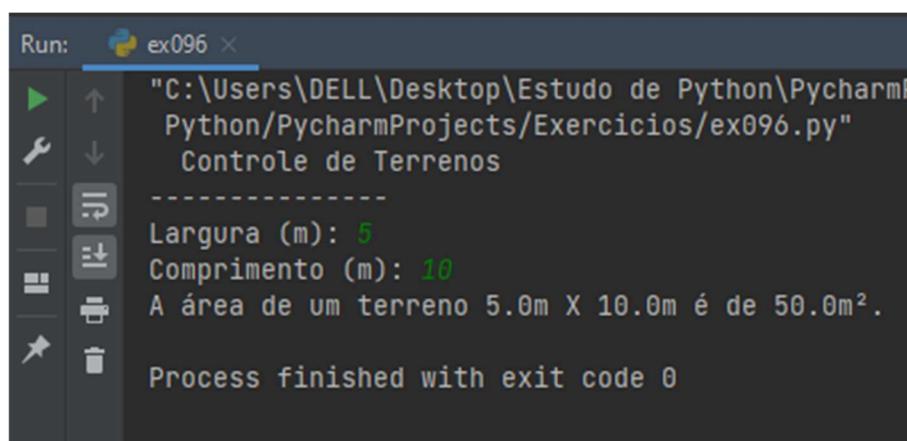
**Faça um programa que tenha uma função chamada área(), que receba as dimensões de um terreno retangular (largura e comprimento) e mostre a área do terreno.**

Código:



```
ex096.py ×
1 def area(larg, comp):
2     a = larg * comp
3     print(f'A área de um terreno {larg}m X {comp}m é de {a}m².')
4
5
6 # PROGRAMA PRINCIPAL
7 print(' Controle de Terrenos')
8 print('-' * 15)
9 l = float(input('Largura (m): '))
10 c = float(input('Comprimento (m): '))
11 area(l, c)
12 |
```

Run:



```
Run: ex096 ×
C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\PycharmProjects\Exercicios/ex096.py
Controle de Terrenos
-----
Largura (m): 5
Comprimento (m): 10
A área de um terreno 5.0m X 10.0m é de 50.0m².
Process finished with exit code 0
```

## Exercício 97 – Um print especial

**Faça um programa que tenha uma função chamada escreva(), que receba um texto qualquer como parâmetro e mostre uma mensagem com tamanho adaptável.**

**Ex.:**

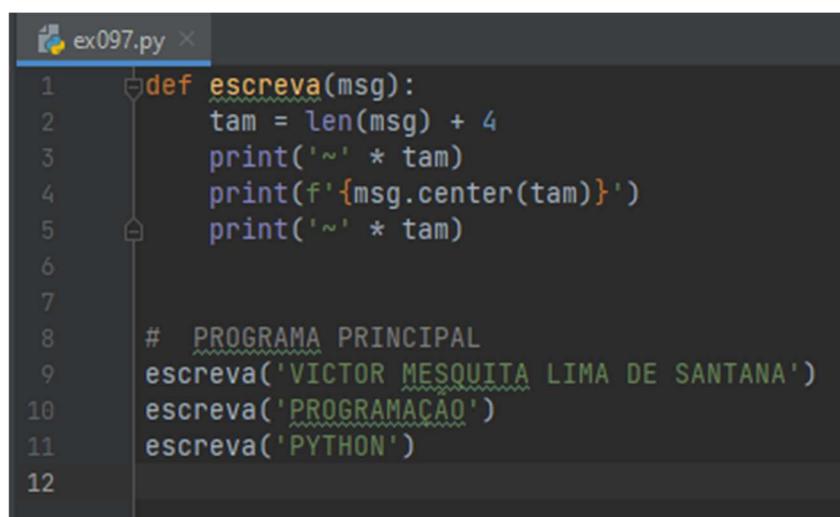
**escreva('Olá, Mundo!')** Saída:

~~~~~

**Olá, Mundo!**

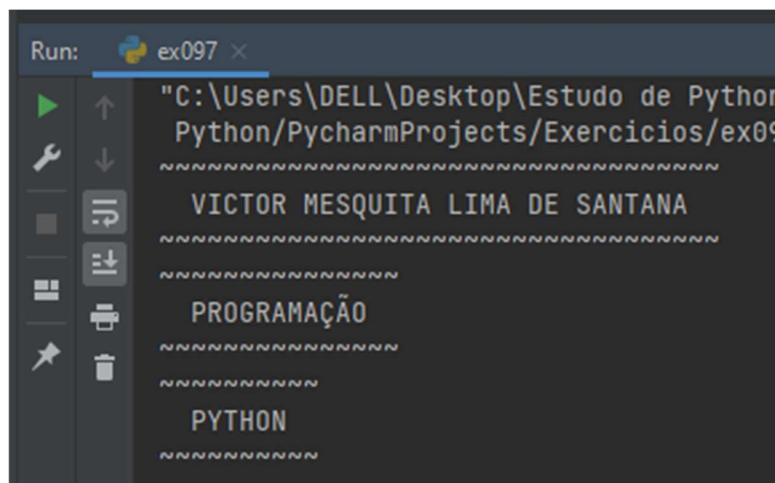
~~~~~

Código:



```
1 def escreva(msg):
2     tam = len(msg) + 4
3     print('~' * tam)
4     print(f'{msg.center(tam)}')
5     print('~' * tam)
6
7
8     # PROGRAMA PRINCIPAL
9     escreva('VICTOR MESQUITA LIMA DE SANTANA')
10    escreva('PROGRAMAÇÃO')
11    escreva('PYTHON')
12
```

Run:



```
Run: ex097 x
C:\Users\DELL\Desktop\Estudo de Python
Python\PycharmProjects\Exercicios/ex097.py
~~~~~
VICTOR MESQUITA LIMA DE SANTANA
~~~~~
~~~~~
PROGRAMAÇÃO
~~~~~
~~~~~
PYTHON
~~~~~
```

## Exercício 98 – Função de Contador

**Faça um programa que tenha uma função chamada contador(), que receba três parâmetros: início, fim e passo. Seu programa tem que realizar três**

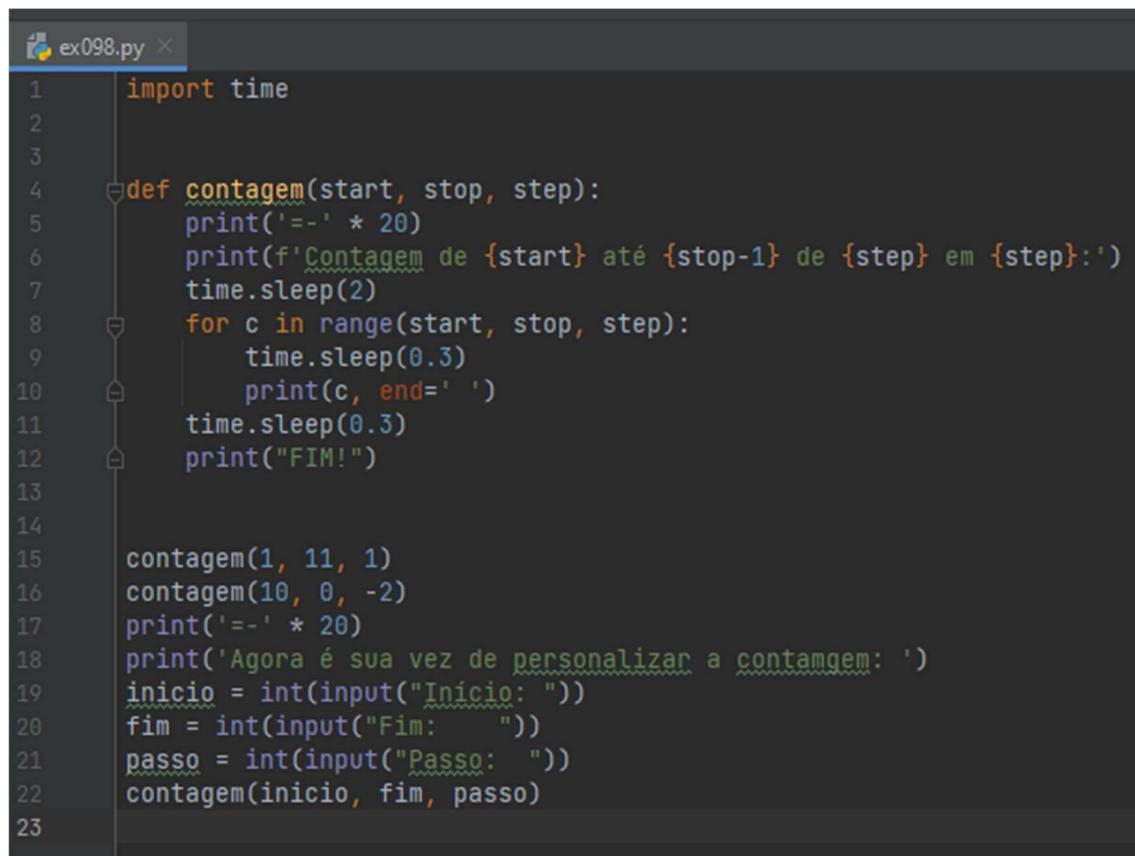
**contagens através da função criada:**

a) de 1 até 10, de 1 em 1;

b) de 10 até 0, de 2 em 2;

c) uma contagem personalizada.

Código:



```
1 import time
2
3
4 def contagem(start, stop, step):
5     print('=-' * 20)
6     print(f'Contagem de {start} até {stop-1} de {step} em {step}:')
7     time.sleep(2)
8     for c in range(start, stop, step):
9         time.sleep(0.3)
10        print(c, end=' ')
11        time.sleep(0.3)
12    print("FIM!")
13
14
15 contagem(1, 11, 1)
16 contagem(10, 0, -2)
17 print('=-' * 20)
18 print('Agora é sua vez de personalizar a contagem: ')
19 inicio = int(input("Início: "))
20 fim = int(input("Fim: "))
21 passo = int(input("Passo: "))
22 contagem(inicio, fim, passo)
23
```

Run:

The screenshot shows the PyCharm interface with the 'Run' tab selected. The title bar says 'Run: ex098'. The run configuration dropdown shows the path: "C:\Users\DELL\Desktop\Estudo de Python\Pycharm Python\PycharmProjects\Exercicios/ex098.py". The output window displays the following text:

```
"C:\Users\DELL\Desktop\Estudo de Python\Pycharm Python\PycharmProjects\Exercicios/ex098.py"
=====
Contagem de 1 até 10 de 1 em 1:
1 2 3 4 5 6 7 8 9 10 FIM!
=====
Contagem de 10 até -1 de -2 em -2:
10 8 6 4 2 FIM!
=====
Agora é sua vez de personalizar a contagem:
Início: 28
Fim: -7
Passo: -4
=====
Contagem de 28 até -8 de -4 em -4:
28 24 20 16 12 8 4 0 -4 FIM!
```

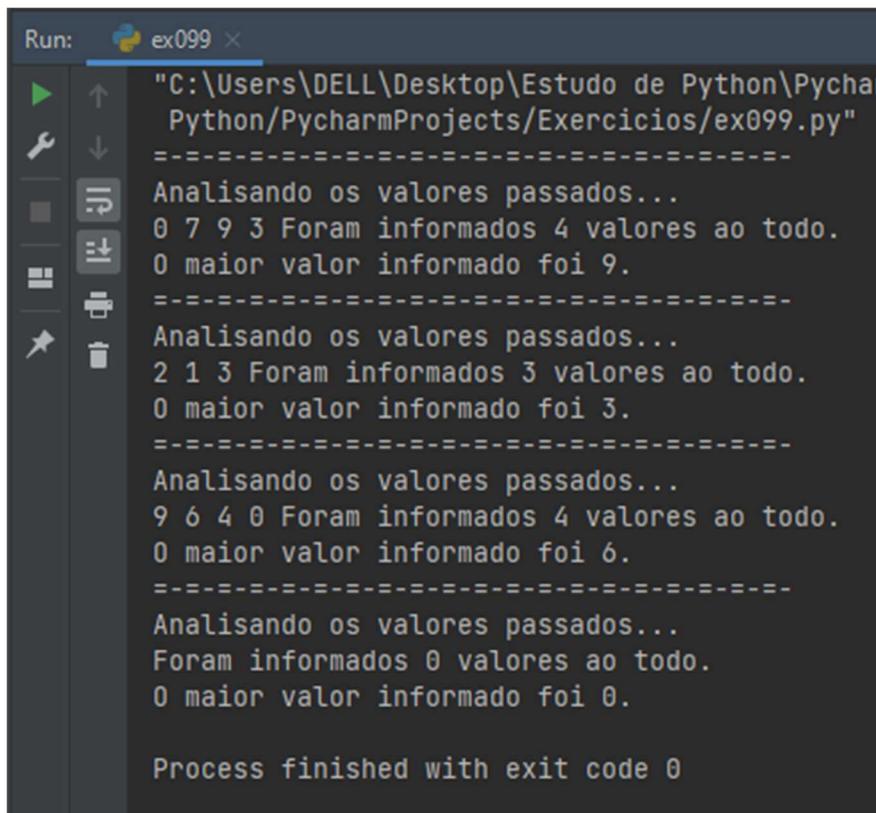
## Exercício 99 – Função que Descobre o Maior

**Faça um programa que tenha uma função chamada maior(), que receba vários parâmetros com valores inteiros. Seu programa tem que analisar todos os valores e dizer qual deles é o maior.**

Código:

```
 ex099.py ×
1   from time import sleep
2
3
4   def maior(*x):
5       cont = maior = 0
6       print("-" * 20)
7       print(f'Analisando os valores passados...')
8       sleep(1)
9       for c in x:
10           if cont == 0:
11               maior = cont
12           elif c > maior:
13               maior = c
14           print(c, end=' ')
15           cont += 1
16           sleep(0.5)
17       print(f'Foram informados {cont} valores ao todo.')
18       print(f'O maior valor informado foi {maior}.')
19
20
21 # Programa Principal
22 maior(0, 7, 9, 3)
23 maior(2, 1, 3)
24 maior(9, 6, 4, 0)
25 maior()
26
```

Run:



The screenshot shows the PyCharm interface with the 'Run' tab selected. The title bar says 'Run: ex099'. The main area displays the execution of a Python script named 'ex099.py'. The output shows four separate runs of the program, each analyzing different sets of input values and printing the count of values, the maximum value, and a separator line. The final line of output indicates the process finished with an exit code of 0.

```
"C:\Users\DELL\Desktop\Estudo de Python\PycharmPython\PycharmProjects\Exercicios/ex099.py"
=====
Analisando os valores passados...
0 7 9 3 Foram informados 4 valores ao todo.
O maior valor informado foi 9.
=====
Analisando os valores passados...
2 1 3 Foram informados 3 valores ao todo.
O maior valor informado foi 3.
=====
Analisando os valores passados...
9 6 4 0 Foram informados 4 valores ao todo.
O maior valor informado foi 6.
=====
Analisando os valores passados...
Foram informados 0 valores ao todo.
O maior valor informado foi 0.

Process finished with exit code 0
```

## Exercício 100 – Funções para Sortear e Somar

**Faça um programa que tenha uma lista chamada números e duas funções chamadas sorteia() e somaPar(). A primeira função vai sortear 5 números e vai colocá-los dentro da lista e a segunda função vai mostrar a soma entre todos os valores pares sorteados pela função anterior.**

Código:

```
ex100.py x
1   from time import sleep
2   from random import randint
3
4
5   def sorteia():
6       print('Sorteando os valores da lista: ', end='')
7       sleep(1)
8       for c in range(0, 5):
9           num.append(randint(0, 9))
10          print(num[c], end=' ')
11          sleep(0.5)
12      print('PRONTO!')
13
14
15  def somaPar():
16      somador = 0
17      for valor in num:
18          if valor % 2 == 0:
19              somador += valor
20      print(f'Somando os valores pares de {num}, temos {somador}.')
21
22
23  num = []
24  sorteia()
25  somaPar()
26
```

Run:

```
Run: ex100 x
C:\Users\DELL\Desktop\Estudo de Python\PycharmProjects\Python\PycharmProjects\Exercicios\ex100.py"
Sorteando os valores da lista: 5 7 7 1 7 PRONTO!
Somando os valores pares de [5, 7, 7, 1, 7], temos 0.
```

## Ajuda Interativa (Interactive Help)

A ajuda interativa é um manual do Python que busca orientar o programador sobre outras funções, métodos e bibliotecas. A função que chama essa ajuda interativa é essa:



Serve tanto no arquivo .py em si como também no console do Python. Se quisermos saber um pouco mais sobre a função print, ou sobre a função len, ou sobre a função input, por exemplo, podemos digitar isso tudo no console que ele retornará um informativo sobre cada uma delas:

```
help> print
Help on built-in function print in module builtins:

print(*args, **kwargs)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
        file: a file-like object (stream); defaults to the current sys.stdout.
        sep:   string inserted between values, default a space.
        end:   string appended after the last value, default a newline.
        flush: whether to forcibly flush the stream.
```

```
??
o. help> len
+ Help on built-in function len in module builtins:

len(obj, /)
    Return the number of items in a container.

help> input
Help on built-in function input in module builtins:

input(prompt=None, /)
    Read a string from standard input.  The trailing newline is stripped.

    The prompt string, if given, is printed to standard output without a
    trailing newline before reading input.

    If the user hits EOF (*nix: Ctrl-D, Windows: Ctrl-Z+Return), raise EOFError.
    On *nix systems, readline is used if available.
```

## Docstrings

Como o próprio nome sugere, são strings de documentação. O help é um manual para funções, métodos e bibliotecas muito utilizadas em Python. Quando se torna necessário criar uma função, que por sua vez ainda não tem um manual próprio, se utiliza as docstrings para criar o manual delas.

Para criá-las utiliza-se um conjunto 3 aspas duplas ("") para abrir e outro para fechar.

```
....  
→ Faz uma contagem e mostra na tela.  
:param i: início da contagem  
:param f: fim da contagem  
:param p: passo da contagem  
:return: sem retorno  
....
```

Função sem docstrings – apenas com help(nomedaFunção):

A screenshot of the PyCharm IDE interface. In the center, there is a terminal window titled 'Run: ambiente'. It displays the command '/Users/guanabara/PycharmProjects/Python/venv/bin' followed by 'Help on function contador in module \_\_main\_\_:' and the definition of the 'contador(i, f, p)' function.

```
Run: ambiente  
/Users/guanabara/PycharmProjects/Python/venv/bin  
Help on function contador in module __main__:  
    contador(i, f, p)
```

Função com docstrings – instruções criadas manualmente:

A screenshot of the PyCharm IDE interface. In the center, there is a terminal window titled 'Run: ambiente'. It displays the command '/Users/guanabara/PycharmProjects/Python/venv/bin' followed by 'Help on function contador in module \_\_main\_\_:' and the definition of the 'contador(i, f, p)' function, which includes a detailed docstring.

```
Run: ambiente  
/Users/guanabara/PycharmProjects/Python/venv/bin  
Help on function contador in module __main__:  
    contador(i, f, p)  
        → Faz uma contagem e mostra na tela.  
        :param i: início da contagem  
        :param f: fim da contagem  
        :param p: passo da contagem  
        :return: sem retorno
```

Agora sempre que utilizarem help(nomedaFunção), esse texto que aparecerá como suporte.

## Parâmetro Opcional

Quando não se há certeza da entrada de algum parâmetro dentro da função, geralmente é utilizado o conceito de parâmetro opcional. São parâmetros que, quando não estabelecidos, ainda permitem com que a função ocorra normalmente. Exemplo:

A screenshot of the PyCharm IDE interface. It shows a Python script with a function named 'somar' that has three parameters: 'a', 'b', and 'c'. The 'c' parameter is marked with a yellow dotted arrow indicating it is optional. Below the function definition, two calls to the 'somar' function are shown: 'somar(3, 2, 5)' and 'somar(8, 4)'. The 'somar(8, 4)' call uses the default value for 'c' (0).

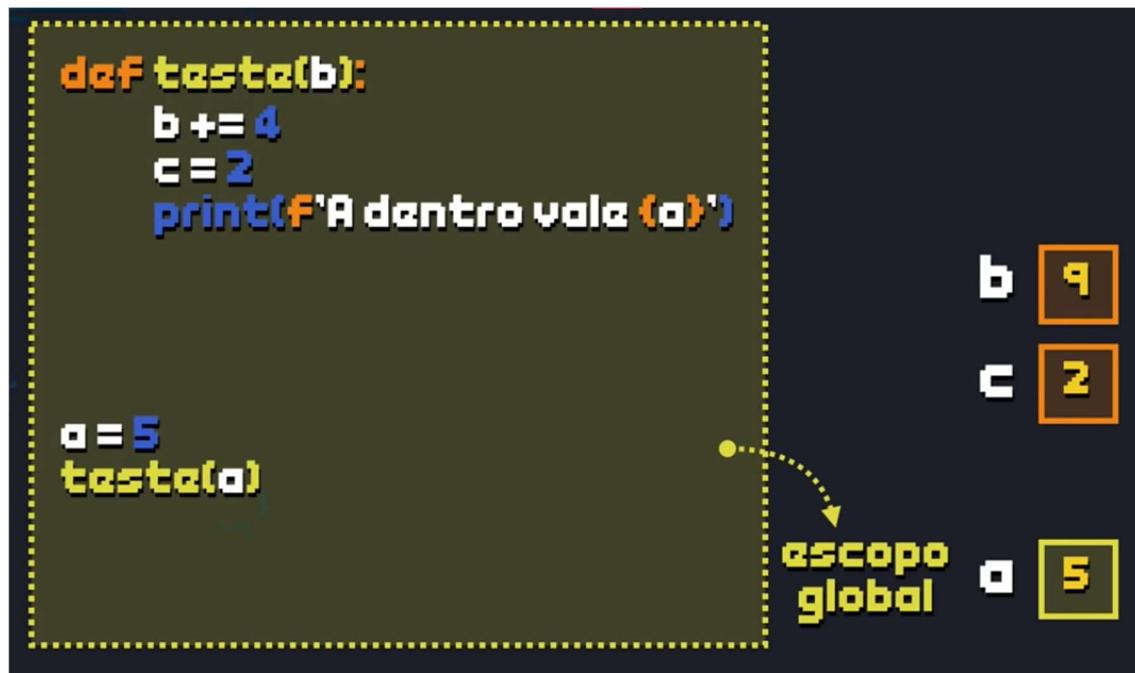
```
def somar(a, b, c=0):  
    s = a + b + c  
    print(f'A soma vale {s}')  
  
somar(3, 2, 5)  
somar(8, 4)
```

Caso 'c' não seja declarado, a função o assumirá como zero e continuará. Caso seja declarado, o valor do parâmetro real sobreporá o que está no parâmetro formal e também gerando continuidade ao código.

## Escopo de Variáveis

Dentro da programação, escopo de variáveis, ou escopo de declarações, tem como significado o local que uma variável vai existir e o local que ela não vai mais existir.

Escopo global – funciona para todo o código. Ex.:



Escopo local – funciona em uma parte do código, como em uma função ou laço de repetição. Ex.:



Caso uma variável tente ser chamada fora do seu escopo, o código daria erro, como nesse exemplo:

```
print(f'A Fora vale {a}')
print(f'B Fora vale {b}')
print(f'C Fora vale {c}')
```

Em casos em que há uma variável global e outra local, porém ambas com o mesmo nome, a local que será utilizada como base. Caso precise utilizar a variável global, existe o método global para isso:

```
global a
```

Caso ela seja alterada dentro de uma função, será alterada por todo o código também.

## Retorno de Valores

Funções podem retornar valores ou não. Esses valores podem ser anexados a outras variáveis, utilizados em outras funções, ..., coisa que antes não era possível. Para isso, utiliza-se o método return:

```
def somar(a=0, b=0, c=0):
    s = a + b + c
    return s

r1 = somar(3, 2, 5)
r2 = somar(1, 7)
r3 = somar(4)
print(f'Meus cálculos deram {r1}, {r2} e {r3}.')
```

## Exercício 101 – Funções para Votação

**Crie um programa que tenha uma função chamada voto() que vai receber como parâmetro o ano de nascimento de uma pessoa, retornando um valor literal indicando se uma pessoa tem voto NEGADO, OPCIONAL e OBRIGATÓRIO nas eleições.**

Código:

```
1  def voto(ano):
2      from datetime import datetime
3      idade = datetime.today().year - ano
4      if idade < 16:
5          return f'Com {idade} anos: NÃO VOTA.'
6      elif 16 <= idade < 18 or idade >= 65:
7          return f'Com {idade} anos: VOTO OPCIONAL.'
8      elif 18 <= idade < 65:
9          return f'Com {idade} anos: VOTO OBRIGATÓRIO.'
10
11
12 # PROGRAMA PRINCIPAL
13 voto(int(input("Em que ano você nasceu ? ")))
14
```

Run:

```
Run: ex101
C:\Users\DELL\AppData\Local\Programs\Python\Python39\ex101
Em que ano você nasceu ? 2002
Com 20 anos: VOTO OBRIGATÓRIO.
Process finished with exit code 0
```

## Exercício 102 – Função para Fatorial

**Crie um programa que tenha uma função fatorial() que receba dois parâmetros: o primeiro que indique o número a calcular e outro chamado show, que será um valor lógico (opcional) indicando se será mostrado ou não na tela o processo de cálculo do fatorial.**

Código:

```
1 def factorial(num, show=False):
2     """
3         -> Calcula o Fatorial de um número.
4         :param num: O número a ser calculado.
5         :param show: Mostrar ou não o cálculo [opcional].
6         :return: O valor do Fatorial de um número num.
7     """
8     f = 1
9     for c in range(num, 0, -1):
10        if show:
11            if c > 1:
12                print(c, end=' x ')
13            else:
14                print(c, end=' = ')
15        f *= c
16    return f
17
18
19 # PROGRAMA PRINCIPAL
20 help(factorial)
21 print(factorial(5, show=True))
```

Run:

```
Run: ex102
C:\Users\DELL\AppData\Local\Programs\Python\Python310\p
Help on function factorial in module __main__:

factorial(num, show=False)
    -> Calcula o Fatorial de um número.
    :param num: O número a ser calculado.
    :param show: Mostrar ou não o cálculo [opcional].
    :return: O valor do Fatorial de um número num.

5 x 4 x 3 x 2 x 1 = 120

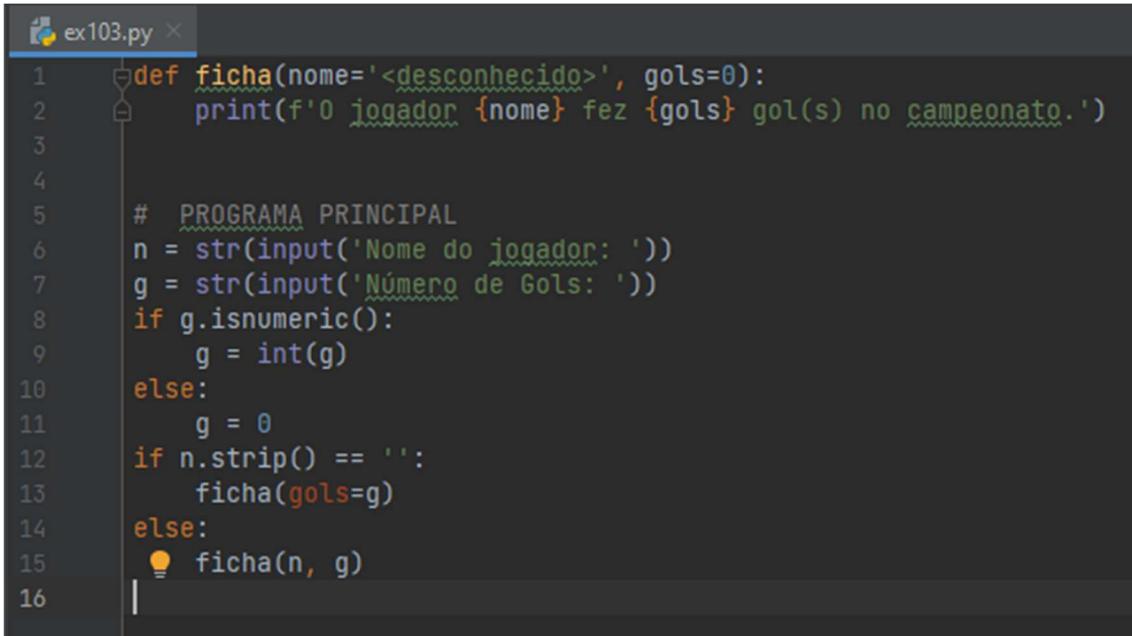
Process finished with exit code 0
```

## Exercício 103 – Ficha do Jogador

Faça um programa que tenha uma função chamada `ficha()`, que receba dois parâmetros opcionais: o nome de um jogador e quantos gols ele marcou. O

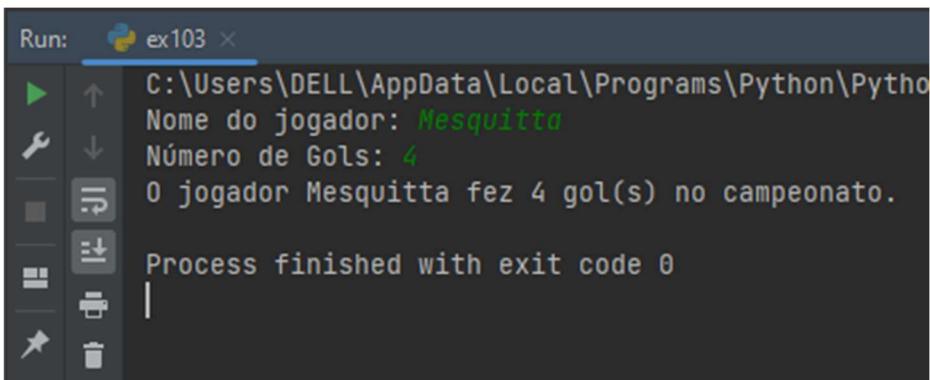
**programa deverá ser capaz de mostrar a ficha do jogador, mesmo que algum dado não tenha sido informado corretamente.**

Código:



```
1 def ficha(nome='<desconhecido>', gols=0):
2     print(f'O jogador {nome} fez {gols} gol(s) no campeonato.')
3
4
5 # PROGRAMA PRINCIPAL
6 n = str(input('Nome do jogador: '))
7 g = str(input('Número de Gols: '))
8 if g.isnumeric():
9     g = int(g)
10 else:
11     g = 0
12 if n.strip() == '':
13     ficha(gols=g)
14 else:
15     ficha(n, g)
16
```

Run:



```
Run: ex103
C:\Users\DELL\AppData\Local\Programs\Python\Python38-32\python.exe "C:/Users/DELL/AppData/Local/Programs/Python/Python38-32/ex103.py"
Nome do jogador: Mesquitta
Número de Gols: 4
O jogador Mesquitta fez 4 gol(s) no campeonato.

Process finished with exit code 0
```

## Exercício 104 – Validando entrada de Dados em Python

**Crie um programa que tenha a função `leialInt()`, que vai funcionar de forma semelhante à função `input()` do Python, só que fazendo a validação para aceitar apenas um valor numérico. Ex: `n = leialInt('Digite um n: ')`.**

Código:

```
ex104.py x
1  def leiaInt(msg):
2      while True:
3          valor = str(input(msg)).strip()
4          if valor.isnumeric():
5              valor = int(valor)
6              break
7          else:
8              print('\033[1;31mERRO! Digite um número inteiro válido.\033[m')
9      return valor
10
11
12 # PROGRAMA PRINCIPAL
13 n = leiaInt('Digite um número: ')
14 print(f'Você acabou de digitar o numero {n}.')
15
```

Run:

```
Run: ex104 x
C:\Users\DELL\AppData\Local\Programs\Python\3.9\python.exe C:/Users/DELL/AppData/Local/Programs/Python/3.9/ex104.py
Digite um número:
ERRO! Digite um número inteiro válido.
Digite um número: três
ERRO! Digite um número inteiro válido.
Digite um número: 3
Você acabou de digitar o numero 3.

Process finished with exit code 0
```

## Exercício 105 – Analisando e Gerando Dicionários

Faça um programa que tenha uma função notas() que pode receber várias notas de alunos e vai retornar um dicionário com as seguintes informações:

- Quantidade de notas
- A maior nota;
- A menor nota;
- A média da turma;
- A situação (opcional).

Código:

```
1 def notas(*num, sit=False):
2     """
3         -> Função para analisar notas e situações obtidas de vários alunos:
4         :param num: uma ou mais notas dos alunos.
5         :param sit: valor opcional, indicando se deve ou não adicionar a situação.
6         :return: dicionário com várias informações sobre a turma.
7     """
8     notasturma = dict()
9     notasturma['total'] = len(num)
10    notasturma['maior'] = max(num)
11    notasturma['menor'] = min(num)
12    notasturma['media'] = sum(num)/len(num)
13    if sit:
14        if notasturma['media'] < 5:
15            notasturma['situação'] = 'RUIM'
16        if 5 <= notasturma['media'] < 7.5:
17            notasturma['situação'] = 'RAZOÁVEL'
18        if 7.5 <= notasturma['media'] < 9:
19            notasturma['situação'] = 'BOM'
20        if 9 <= notasturma['media'] < 10:
21            notasturma['situação'] = 'EXCEPCIONAL'
22    return notasturma
23
24
25 # PROGRAMA PRINCIPAL
26 print(notas(6, 7.5, 8, 10, 3.25, 5.75, sit=True))
27 help(notas)
28
```

Run:

```
Run: ex105
C:\Users\DELL\AppData\Local\Programs\Python\Python310\python.exe C:/Users/DELL/P
{'total': 6, 'maior': 10, 'menor': 3.25, 'media': 6.75, 'situação': 'RAZOÁVEL'}
Help on function notas in module __main__:

notas(*num, sit=False)
    -> Função para analisar notas e situações obtidas de vários alunos:
    :param num: uma ou mais notas dos alunos.
    :param sit: valor opcional, indicando se deve ou não adicionar a situação.
    :return: dicionário com várias informações sobre a turma.

Process finished with exit code 0
```

## Exercício 106 – Interactive Helping System in Python

Faça um minissistema que utilize o Interactive Help do Python. O usuário vai digitar o comando e o manual vai aparecer. Quando o usuário digitar a palavra 'FIM', o programa se encerrará. Importante: use cores.

Código:

```
 1      c = ('\033[m',           # 0 - sem cor
 2          '\033[30;41m',       # 1 - vermelho
 3          '\033[42m',         # 2 - verde
 4          '\033[30;43m',       # 3 - amarelo
 5          '\033[30;44m',       # 4 - azul
 6          '\033[30;45m',       # 5 - roxo
 7          '\033[7;30m')        # 6 - branco
 8
 9  def ajuda(comando):
10      titulo(f'Acessando o manual do comando {comando}', 4)
11      print(c[5])
12      help(comando)
13      print(c[0], end='')
14
15  def titulo(msg, cor=0):
16      print(c[cor])
17      print('~' * (len(msg)+4))
18      print(f' {msg}')
19      print('~' * (len(msg)+4))
20      print(c[0], end='')
21
22  # PROGRAMA PRINCIPAL
23  com = ''
24  while True:
25      titulo('SISTEMA DE AJUDA PyHELP', 2)
26      com = str(input('Função ou biblioteca > '))
27      if com.upper() == 'FIM':
28          titulo('ATÉ LOGO!', 1)
29          break
30      else:
31          ajuda(com)
32
```

Run:

The screenshot shows a terminal window titled "Run: ex106". The command entered was "python -m pyhelp len". The output is as follows:

```
C:\Users\DELL\AppData\Local\Programs\Python\Python310\python
=====
SISTEMA DE AJUDA PyHELP
=====
Função ou biblioteca > len
=====
Acessando o manual do comando len
=====
Help on built-in function len in module builtins:
len(obj, /)
    Return the number of items in a container.

=====
SISTEMA DE AJUDA PyHELP
=====
Função ou biblioteca > fim
=====
ATÉ LOGO!
=====
```

Process finished with exit code 0

## Aula 22 – Módulos e Pacotes

### Modularização

Ato de criar módulos, dividir um programa grande por assuntos e em pequenos pedaços para facilitar a manutenção e legibilidade de todo o sistema.

Por exemplo, se você tem um arquivo que tem muitas funções de cálculos e quer separá-las do código principal para uma melhor organização, isso é possível:

Esse é o código com tudo implementado nele.

```
def factorial(n):
    f=1
    for c in range(1,n+1):
        f*=c
    return f

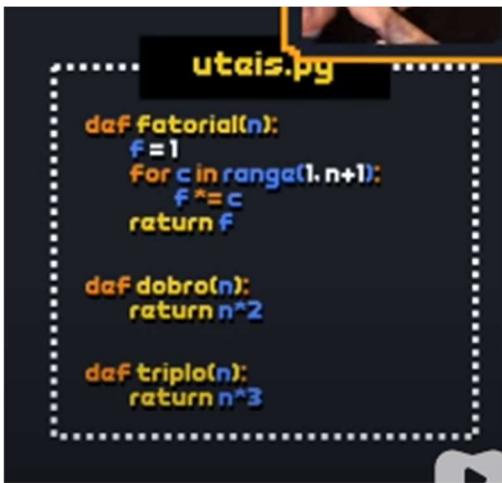
def dobro(n):
    return n*2

def triplo(n):
    return n*3

num=int(input("Digite um valor"))
fat=factorial(num)
print(f'O Fatorial de {num} é {fat}')
```

Já aqui, é o código principal em um arquivo e as funções em um outro (chamado de uteis.py).

```
num=int(input("Digite um valor"))
fat=factorial(num)
print(f'O Fatorial de {num} é {fat}')
```



```
... uteis.py ...
def factorial(n):
    f=1
    for c in range(1,n+1):
        f*=c
    return f

def dobro(n):
    return n*2

def triplo(n):
    return n*3
```

Todas elas ainda são possíveis de serem acessadas no arquivo principal. Basta importar o módulo (nome do arquivo aonde estão as defs) que será utilizada – que no caso se chama uteis.

```
1 import uteis # As defs estão no arquivo uteis.py
2
3 n = int(input('Digite um valor: '))
4 fat = uteis.fatorial(n)
5 print(f'O fatorial de {n} é {fat}.')
6 print(f'O dobro de {n} é {uteis.dobro(n)}.')
7 print(f'O triplo de {n} é {uteis.triplo(n)}.')
8
```

Assim como nos outros casos, utilizou-se o método do módulo (que recebe o mesmo nome que ela) mais o nome da função que pretende utilizar.

Também pode ser importada que nem os outros módulos – utilizando o `from`:

```
from uteis import factorial
from math import sqrt
from datetime import datetime
from random import randint
```

Porém, quando se há duas funções com o mesmo nome, porém de módulos distintos, dá conflito no sistema e ele acaba selecionando a última importada. Por conta disso, é altamente recomendável que se importe o módulo inteiro e utilize os métodos precedidos com nomedomódulo e um ponto final.

O resultado do programa ficou assim:

```
Run: aula22 ×
C:\Users\DELL\AppData\Local\Programs
Digite um valor: 5
O fatorial de 5 é 120.
O dobro de 5 é 10.
O triplo de 5 é 15.
Process finished with exit code 0
```

## Pacotes

Mesmo com o surgimento dos módulos, ainda houve a possibilidade de acontecer com eles o mesmo que aconteceu com os programas principais – eles ficarem enormes com longas e longas linhas de código.

```
uteis.py
def xyz():      def mmn():
...
def kxy():      def mnw():
...
def kkk():      def ouy():
...
def kmn():      def yuk():
...
def klk():      def wpi():
...
def jhf():      def fpr():
...
def xu0():      def dsa():
...
def qwa():      def mbv():
...
def wrk():      def vxl():
...

```

Nesse caso todas as vantagens, como legibilidade e fácil manutenção foram descartadas. Contudo, nem tudo está perdido, pois os pacotes existem. Pacotes são pastas que dividem os módulos por assuntos, resgatando todas as vantagens que eles trazem ao sistema:

```
Pacote uteis
números
def xyz():
...
def kxy():
...
def kkk():
...
def kmn():

datas
def xu0():
...
def qwa():
...
def wrk():
...
def mmn():
...
def mnw():
...
def ouy():
...
def yuk():

cores
def wpi():
...
def fpr():
...
def dsa():
...
def mbv():
...
def vxl():

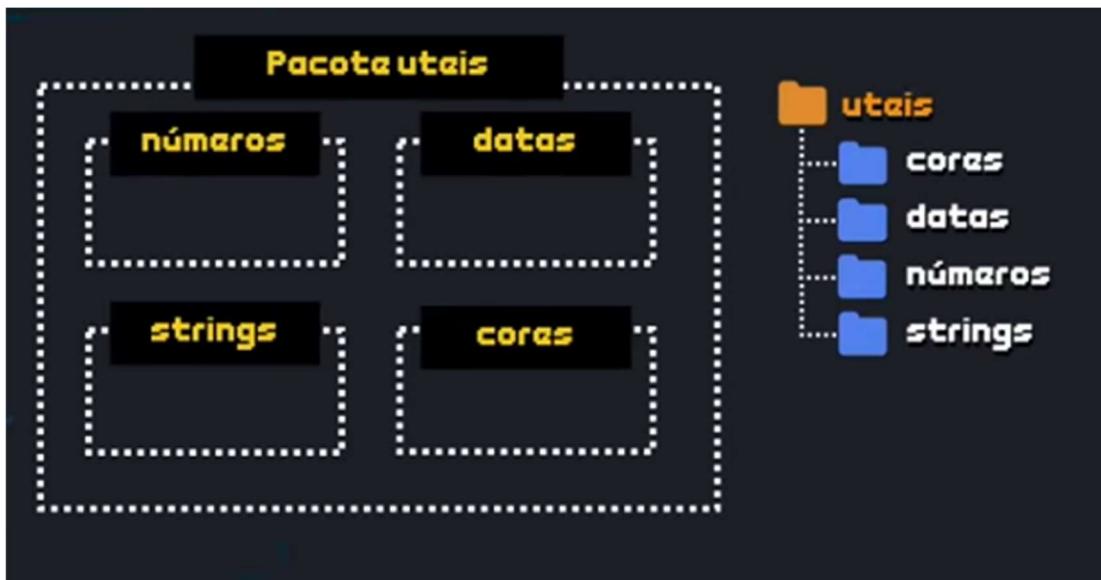
PRATO

```

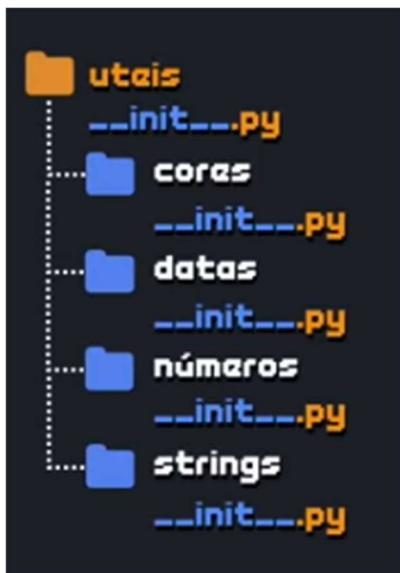
E então, as importações ficam dessa forma?

```
import uteis  
from uteis import datas  
from uteis import cores
```

Para criá-los, é bem simples. Basta criar uma pasta com o nome do pacote requerido e diversas pastas dentro dele, que serão os pacotes:



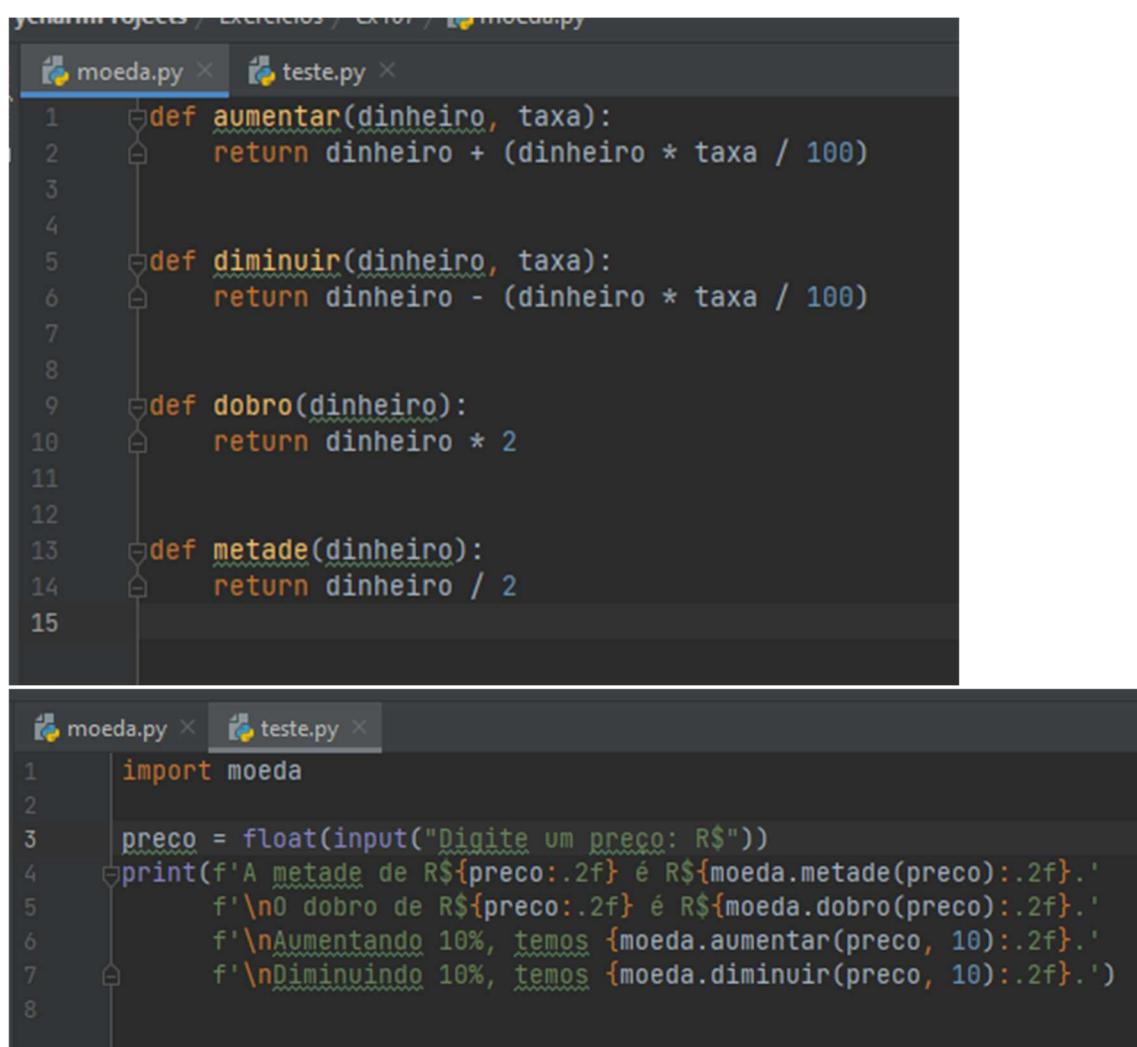
Não somente isso, também é necessário a criação de um arquivo chamado '`__init__.py`' dentro de cada pasta criada. É imprescindível pois sem isso o pacote não funcionará, e, por sua vez, os módulos também não.



## Exercício 107 – Exercitando módulos em Python

Crie um módulo chamado `moeda.py` que tenha as funções incorporadas `aumentar()`, `diminuir()`, `dobro()` e `metade()`. Faça também um programa que importe esse módulo e use algumas dessas funções.

Código:



The image shows a code editor with two files open:

- moeda.py:** Contains four functions:
  - `aumentar(dinheiro, taxa):` Returns `dinheiro + (dinheiro * taxa / 100)`.
  - `diminuir(dinheiro, taxa):` Returns `dinheiro - (dinheiro * taxa / 100)`.
  - `dobro(dinheiro):` Returns `dinheiro * 2`.
  - `metade(dinheiro):` Returns `dinheiro / 2`.
- teste.py:** Imports the `moeda` module and uses its functions to calculate half, double, and 10% increase/decrease of a given price.

```
moeda.py
1 def aumentar(dinheiro, taxa):
2     return dinheiro + (dinheiro * taxa / 100)
3
4 def diminuir(dinheiro, taxa):
5     return dinheiro - (dinheiro * taxa / 100)
6
7 def dobro(dinheiro):
8     return dinheiro * 2
9
10 def metade(dinheiro):
11     return dinheiro / 2
12
13
14
15

moeda.py
1 import moeda
2
3 preco = float(input("Digite um preço: R$"))
4 print(f'A metade de R${preco:.2f} é R${moeda.metade(preco):.2f}.')
5 print(f'\nO dobro de R${preco:.2f} é R${moeda.dobro(preco):.2f}.')
6 print(f'\nAumentando 10%, temos {moeda.aumentar(preco, 10):.2f}.')
7 print(f'\nDiminuindo 10%, temos {moeda.diminuir(preco, 10):.2f}.')
```

Run:

```
Run: teste ×
C:\Users\DELL\AppData\Local\Programs\
Digite um preço: R$10000
A metade de R$10000.00 é R$5000.00.
O dobro de R$10000.00 é R$20000.00.
Aumentando 10%, temos 11000.00.
Diminuindo 10%, temos 9000.00.

Process finished with exit code 0
```

## Exercício 108 – Formatando Moedas em Python

**Adapte o código do desafio #107, criando uma função adicional chamada moeda() que consiga mostrar os números como um valor monetário formatado.**

Código:

```
moeda.py x teste.py x
1 def aumentar(dinheiro=0, taxa=0):
2     return dinheiro + (dinheiro * taxa / 100)
3
4
5 def diminuir(dinheiro=0, taxa=0):
6     return dinheiro - (dinheiro * taxa / 100)
7
8
9 def dobro(dinheiro=0):
10    return dinheiro * 2
11
12
13 def metade(dinheiro=0):
14    return dinheiro / 2
15
16
17 def moeda(dinheiro=0, moeda='R$'):
18    return f'{moeda}{dinheiro:.2f}'.replace('.', ',')
19
```

```
moeda.py x teste.py x
1 import moeda
2
3 preco = float(input("Digite um preço: R$"))
4 print(f'A metade de {moeda.moeda(preco)} é {moeda.moeda(moeda.metade(preco))}.')
5     f'\nO dobro de {moeda.moeda(preco)} é {moeda.moeda(moeda.dobro(preco))}.'
6     f'\nAumentando 10%, temos {moeda.moeda(moeda.aumentar(preco, 10))}.'
7     f'\nDiminuindo 10%, temos {moeda.moeda(moeda.diminuir(preco, 10))}.')
8
```

Run:

```
Run: teste (1) x
C:\Users\DELL\AppData\Local\Programs\Py
Digite um preço: R$15047,65
A metade de R$15047,65 é R$ 7523,82.
O dobro de R$15047,65 é R$30095,30.
Aumentando 10%, temos R$16552,42.
Diminuindo 10%, temos R$13542,89.

Process finished with exit code 0
```

## Exercício 109 – Formatando Moedas em Python

**Modifique as funções que foram criadas no desafio 107 para que elas aceitem um parâmetro a mais, informando se o valor retornado por elas vai ser ou não formatado pela função moeda(), desenvolvida no desafio 108.**

Código:

The screenshot shows a code editor with two tabs: 'moeda.py' and 'teste.py'. The 'moeda.py' tab contains the following code:

```
1 def aumentar(dinheiro=0, taxa=0, formatacao=False):
2     res = dinheiro + (dinheiro * taxa / 100)
3     return res if formatacao is False else moeda(res)
4
5 def diminuir(dinheiro=0, taxa=0, formatacao=False):
6     res = dinheiro - (dinheiro * taxa / 100)
7     return res if not formatacao else moeda(res)
8
9 def dobro(dinheiro=0, formatacao=False):
10    res = dinheiro * 2
11    return res if formatacao is False else moeda(res)
12
13 def metade(dinheiro=0, formatacao=False):
14    res = dinheiro / 2
15    return res if formatacao is False else moeda(res)
16
17 def moeda(dinheiro=0, moeda='R$'):
18     return f'{moeda}{dinheiro:.2f}'.replace('.', ',')
```

The 'teste.py' tab contains the following code:

```
1 import moeda
2
3 preco = float(input("Digite um preço: R$"))
4 print(f'A metade de {moeda.moeda(preco)} é {moeda.metade(preco, True)}.\nO dobro de {moeda.moeda(preco)} é {moeda.dobro(preco, True)}.\nAumentando 10%, temos {moeda.aumentar(preco, 10, True)}.\nDiminuindo 10%, temos {moeda.diminuir(preco, 10, True)}.)
```

Run:

```
Run:  teste (2) ×
C:\Users\DELL\AppData\Local\Programs\P
Digite um preço: R$15000
A metade de R$15000,00 é R$7500,00.
O dobro de R$15000,00 é R$30000,00.
Aumentando 10%, temos R$16500,00.
Diminuindo 10%, temos R$13500,00.
```

## Exercício 110 – Reduzindo Ainda Mais seu Programa

**Adicione o módulo moeda.py criado nos desafios anteriores, uma função chamada resumo(), que mostre na tela algumas informações geradas pelas funções que já temos no módulo criado até aqui.**

Código:

```
moeda.py x teste.py x
1 def aumentar(dinheiro=0, taxa=0, formatacao=False):
2     res = dinheiro + (dinheiro * taxa / 100)
3     return moeda(res)
4
5 def diminuir(dinheiro=0, taxa=0, formatacao=False):
6     res = dinheiro - (dinheiro * taxa / 100)
7     return moeda(res)
8
9
10 def dobro(dinheiro=0, formatacao=False):
11     res = dinheiro * 2
12     return moeda(res)
13
14
15 def metade(dinheiro=0, formatacao=False):
16     res = dinheiro / 2
17     return moeda(res)
18
19 def moeda(dinheiro=0, moeda='R$'):
20     return f'{moeda}{dinheiro:.2f}'.replace('.', ',')
21
22 def resumo(dinheiro=0, aumento=0, reducao=0):
23     print('-' * 20)
24     print('RESUMO DO VALOR'.center(40))
25     print('-' * 20)
26     print(f'Preço analisado: \t{moeda(dinheiro)}')
27     print(f'Dobro do preço: \t{dobro(dinheiro, True)}')
28     print(f'Metade do preço: \t{metade(dinheiro, True)}')
29     print(f'{aumento}% de aumento: \t{aumentar(dinheiro, aumento, True)}')
30     print(f'{reducao}% de redução: \t{diminuir(dinheiro, reducao, True)}')
31
32
```

```
moeda.py x teste.py x
1 import moeda
2
3 preco = float(input("Digite um preço: R$"))
4 moeda.resumo(preco, 100, 30)
5
```

Run:

```
Run:  teste (3) ×
C:\Users\DELL\AppData\Local\Programs\Python\Digite um preço: R$5500
=====
RESUMO DO VALOR
=====
Preço analisado:    R$5500,00
Dobro do preço:    R$11000,00
Metade do preço:   R$2750,00
100% de aumento:   R$11000,00
30% de aumento:    R$3850,00
=====

Process finished with exit code 0
```

## Exercício 111 – Transformando Módulos em Pacotes

**Crie um pacote chamado utilidadesCeV que tenha dois módulos internos chamados moeda e dado. Transfira todas as funções utilizadas nos desafios 107, 108 e 109 para o primeiro pacote e mantenha tudo funcionando.**

Código:

The screenshot shows the PyCharm IDE interface. The Project tool window on the left displays a hierarchical view of the 'PycharmProjects' folder, which contains 'aulas', 'Brincando de Python', 'Exercicios', and several subfolders like 'ex107', 'ex108', 'ex109', 'ex110', and 'ex111'. Inside 'ex111', there is a 'utilidadeCeV' folder containing 'moeda' and two Python files: '\_\_init\_\_.py' and 'teste.py'. The code editor window on the right shows the content of the '\_\_init\_\_.py' file:

```
from Exercicios.ex111.utilidadeCeV import moeda
preco = float(input("Digite um preço: R$"))
moeda.resumo(preco, 100, 30)
```

Run:

The Run tool window shows the execution of the 'teste' script. The terminal pane displays the following output:

```
C:\Users\DELL\AppData\Local\Programs\Python
Digite um preço: R$5
=====
RESUMO DO VALOR
=====
Preço analisado: R$5,00
Dobro do preço: R$10,00
Metade do preço: R$2,50
100% de aumento: R$10,00
30% de aumento: R$3,50
=====

Process finished with exit code 0
```

## Exercício 112 – Entrada de Dados Monetários

**Dentro do pacote utilidadesCeV que criamos no desafio 111, temos um módulo chamado dado. Crie uma função chamada leiaDinheiro() que seja capaz de funcionar como a função input(), mas com uma validação de dados para aceitar apenas valores que sejam monetários.**

Código:

Code editor showing three files:

- moeda\_init\_.py** (Top Tab):

```

1 def aumentar(dinheiro=0, taxa=0, formatacao=False):
2     res = dinheiro + (dinheiro * taxa / 100)
3     return moeda(res)
4
5 def diminuir(dinheiro=0, taxa=0, formatacao=False):
6     res = dinheiro - (dinheiro * taxa / 100)
7     return moeda(res)
8
9 def dobro(dinheiro=0, formatacao=False):
10    res = dinheiro * 2
11    return moeda(res)
12
13 def metade(dinheiro=0, formatacao=False):
14    res = dinheiro / 2
15    return moeda(res)
16
17 def moeda(dinheiro=0, moeda='R$'):
18     return f'{moeda}{dinheiro:.2f}'.replace('.', ',')
19
20 def resumo(dinheiro=0, aumento=0, reducao=0):
21     print('-' * 20)
22     print('RESUMO DO VALOR'.center(40))
23     print('-' * 20)
24     print(f'Preço analisado: \t{moeda(dinheiro)}')
25     print(f'Dobro do preço: \t{dobro(dinheiro, True)}')
26     print(f'Metade do preço: \t{metade(dinheiro, True)}')
27     print(f'{aumento}% de aumento: \t{aumentar(dinheiro, aumento, True)}')
28     print(f'{reducao}% de aumento: \t{diminuir(dinheiro, reducao, True)}')
29
30
31

```
- \_init\_.py** (Second Tab):

```

1 def leiaDinheiro(msg):
2     while True:
3         res = input(msg).strip().replace(',', '.')
4         if res.isalpha() or res == '':
5             print(f'\033[1;31mERRO! "{res}" é uma resposta inválida!\033[m')
6         else:
7             return float(res)
8
9

```
- teste.py** (Bottom Tab):

```

1 from Exercicios.ex112.utilidadeCeV import moeda, dado
2
3 preco = dado.leiaDinheiro("Digite um preço: R$")
4 moeda.resumo(preco, 100, 30)
5

```

Run:

The screenshot shows a Python terminal window with the following output:

```
Run: teste (1) ×
C:\Users\DELL\AppData\Local\Programs\Python\...
Digite um preço: R$mesquitta
ERRO! "" é uma resposta inválida!
Digite um preço: R$mesquitta
ERRO! "mesquitta" é uma resposta inválida!
Digite um preço: R$5,98
=====
RESUMO DO VALOR
=====
Preço analisado:    R$5,98
Dobro do preço:    R$11,96
Metade do preço:   R$2,99
100% de aumento:   R$11,96
30% de aumento:    R$4,19
=====
Process finished with exit code 0
```

## Tratamento de Erros e Exceções

### Erros = Sintáticos

São erros na forma de escrever o código, trocar uma letra por outra, esquecer uma vírgula, um parêntese, e coisas do tipo – ex.: print('olá').

### Exceções = Semânticas

São erros de exceção no código. Ele funciona naturalmente, mas se algo em específico acontecer ele simplesmente para tudo por um erro. Ex.: print(8/0) - ZeroDivisionError: division by zero.

Exceções mais famosas

```
NameError
ValueError
ZeroDivisionError
TypeError
IndexError
KeyError
EOFError
KeyboardInterrupt
OSError
MemoryError
ConnectionError
RuntimeError
```

Por conta da ocorrência dessas exceções, utiliza-se a estrutura 'try' para tentar tratá-las. Ela é composta de quatro cláusulas:

```
try:
    operação
except:
    falhou
else:
    deu certo
finally:
    certo/falha
```

Try: é a primeira coisa que código deve tentar fazer.

Except: serve para não mostrar mais a mensagem de erro do programa e fazer uma determinada ação.

Else: essa é opcional, serve para mandar o código fazer algo caso não dê algum erro.

Finally: também opcional, mostra o que deve acontecer depois de try, independente se falhar ou não o código.

Exemplo de caso aonde se utiliza todas as cláusulas de uma só vez:

```
aula23.py
try:
    a = int(input("Digite um número: "))
    b = int(input("Digite outro número: "))
    r = a/b
except:
    print("Infelizmente deu ruim ;(")
else:
    print(f"o resultado deu {r:.1f}")
finally:
    print("Volte sempre! Muito obrigado!")
```

Porém, essa é uma estrutura básica, podendo ser expandida quando necessária. Se o seu código pode dar diversos tipos de exceções diferentes, você personaliza o que deve ser feito em cada um deles:



```
aula23.py
1 try:
2     a = int(input("Digite um número: "))
3     b = int(input("Digite outro número: "))
4     r = a/b
5 except (ValueError, TypeError):
6     print("Tivemos um problema com os tipos de dados fornecidos.")
7 except ZeroDivisionError:
8     print("Impossível dividir por zero.")
9 except KeyboardInterrupt:
10    print("O usuário preferiu não informar os dados.")
11 except Exception as erro:
12    print(f'O erro encontrado foi {erro.__cause__}')
13 else:
14    print(f"o resultado deu {r:.1f}")
15 finally:
16    print("Volte sempre! Muito obrigado!")
```

## Exercício 113 – Funções Aprofundadas em Python

**Reescreva a função leiaInt() que fizemos no desafio 104, incluindo agora a possibilidade da digitação de um número de tipo inválido. Aproveite e crie também uma função leiaFloat() com a mesma funcionalidade.**

Código:

```
ex113.py ×
1 def leiaInt(msg):
2     while True:
3         try:
4             i = int(input(msg))
5         except (ValueError, TypeError):
6             print("\033[1;31mERRO: por favor, digite um número INTEIRO válido.\033[m")
7         except (KeyboardInterrupt):
8             print("\033[1;31mO usuário preferiu não digitar esse número.\033[m")
9             break
10        else:
11            return i
12
13
14 def leiaFloat(msg):
15     while True:
16         try:
17             r = float(input(msg))
18         except (ValueError, TypeError):
19             print("\033[1;31mERRO: por favor, digite um número REAL válido.\033[m")
20         except (KeyboardInterrupt):
21             print("\033[1;31mO usuário preferiu não digitar esse número.\033[m")
22             break
23        else:
24            return r
25
26
27 # Programa principal
28 int = leiaInt("Digite um número Inteiro: ")
29 real = leiaFloat("Digite um número Real: ")
30 print(f"O numero inteiro digitado foi {int} e o real foi {real}.")
```

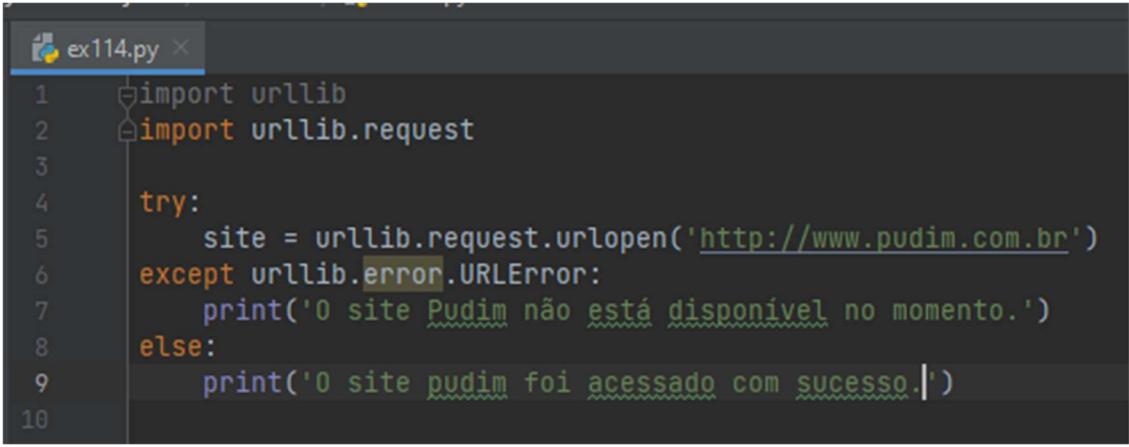
Run:

```
Run: ex113 ×
C:\Users\DELL\AppData\Local\Programs\Python\Python310\python.exe C:/...
Digite um número Inteiro: dsads
ERRO: por favor, digite um número INTEIRO válido.
Digite um número Inteiro:
ERRO: por favor, digite um número INTEIRO válido.
Digite um número Inteiro: 9
Digite um número Real: dsadsa
ERRO: por favor, digite um número REAL válido.
Digite um número Real: três
ERRO: por favor, digite um número REAL válido.
Digite um número Real: O usuário preferiu não digitar esse número.
O numero inteiro digitado foi 9 e o real foi None.
```

## Exercício 114 – Site está acessível?

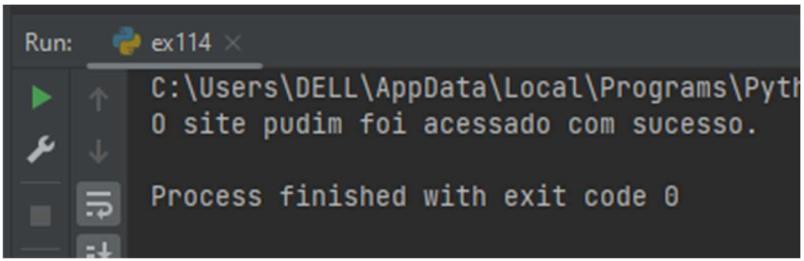
**Crie um código em Python que teste se o site pudim está acessível pelo computador usado.**

Código:



```
1 import urllib
2 import urllib.request
3
4 try:
5     site = urllib.request.urlopen('http://www.pudim.com.br')
6 except urllib.error.URLError:
7     print('O site Pudim não está disponível no momento.')
8 else:
9     print('O site pudim foi acessado com sucesso!')
```

Run:



```
Run: ex114
C:\Users\DELL\AppData\Local\Programs\Python\Python37-32\python.exe ex114
O site pudim foi acessado com sucesso.
Process finished with exit code 0
```

## Exercício 115a – Criando um Menu

**Vamos criar um menu em Python, usando modularização.**

Código:

```

__init__.py x menu.py x
1 def leiaInt(msg):
2     while True:
3         try:
4             resp = int(input(msg))
5         except (ValueError, TypeError):
6             print("\033[1;31mERRO: por favor, digite um número INTEIRO válido.\033[m")
7         except (KeyboardInterrupt):
8             print("\033[1;31mO usuário preferiu não digitar esse número.\033[m")
9             break
10        else:
11            if resp not in [1, 2, 3]:
12                print('\033[1;31mERRO! Digite uma opção válida.\033[m')
13            else:
14                return resp
15
16
17 def linha(tam=40):
18     return '-' * tam
19
20
21 def cabecalho(txt):
22     print(linha())
23     print(txt.center(40))
24     print(linha())
25
26
27 def menu(lista):
28     cabecalho('MENU PRINCIPAL')
29     for pos, opcao in enumerate(lista):
30         print(f'\033[1;33m{pos+1}\033[m - \033[1;34m{opcao}\033[m')
31     print(linha())
32     opc = leiaInt('\033[1;32mSua opção:\033[m ')
33     return opc
34

```

---

```

__init__.py x menu.py x
1 from Exercicios.ex115a.defs import *
2 from time import sleep
3
4 while True:
5     opcao = menu(['Ver pessoas cadastradas', 'Cadastrar nova Pessoa', 'Sair do Sistema'])
6     if opcao == 1:
7         cabecalho(f'OPÇÃO {opcao}')
8     elif opcao == 2:
9         cabecalho(f'OPÇÃO {opcao}')
10    elif opcao == 3:
11        cabecalho('Saindo do sistema... Até logo!')
12        break
13    else:
14        print('\033[1;31mERRO! Digite uma opção válida!\033[m')
15    sleep(2)
16

```

Run:

```
Run: menu x
C:\Users\DELL\AppData\Local\Programs\Python\Python3
-----
        MENU PRINCIPAL
-----
1 - Ver pessoas cadastradas
2 - Cadastrar nova Pessoa
3 - Sair do Sistema
-----
Sua opção: 0
ERRO: por favor, digite um número INTEIRO válido.
Sua opção: 4
ERRO! Digite uma opção válida.
Sua opção: 1
-----
OPÇÃO 1
-----
        MENU PRINCIPAL
-----
1 - Ver pessoas cadastradas
2 - Cadastrar nova Pessoa
3 - Sair do Sistema
-----
Sua opção: 3
-----
Saindo do sistema... Até logo!
```

## Exercício 115b – Arquivos com Python

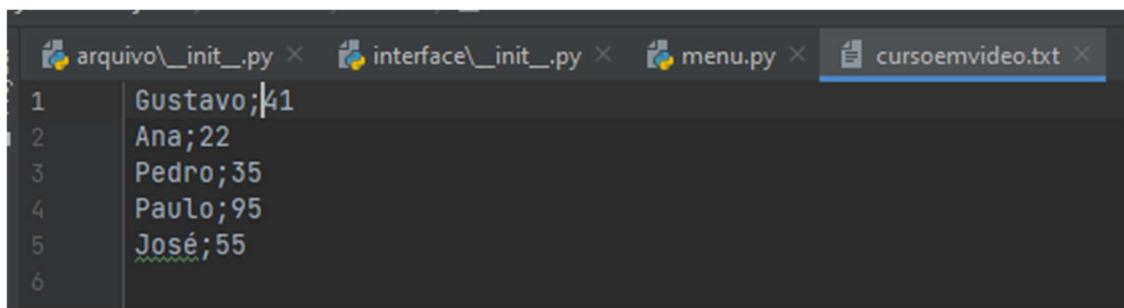
Vamos ver como fazer acesso a arquivos usando o Python.

Código:

```
arquivo\__init__.py x menu.py x interface\__init__.py x cursoemvideo.txt x
1  from Exercicios.ex115a.lib.interface import cabecalho
2
3
4  def arquivoExiste(nome):
5      try:
6          a = open(nome, 'rt')
7          a.close()
8      except FileNotFoundError:
9          return False
10     else:
11         return True
12
13
14 def criarArquivo(nome):
15     try:
16         a = open(nome, 'wt+')
17         a.close()
18     except:
19         print('Houve um erro na criação do arquivo!')
20     else:
21         print(f'Arquivo {nome} criado com sucesso!')
22
23
24 def lerArquivo(nome):
25     try:
26         a = open(nome, 'rt')
27     except:
28         print('\033[1;31mErro ao ler o arquivo!\033[m')
29     else:
30         cabecalho('PESSOAS CADASTRADAS')
31         print(a.read())
32
```

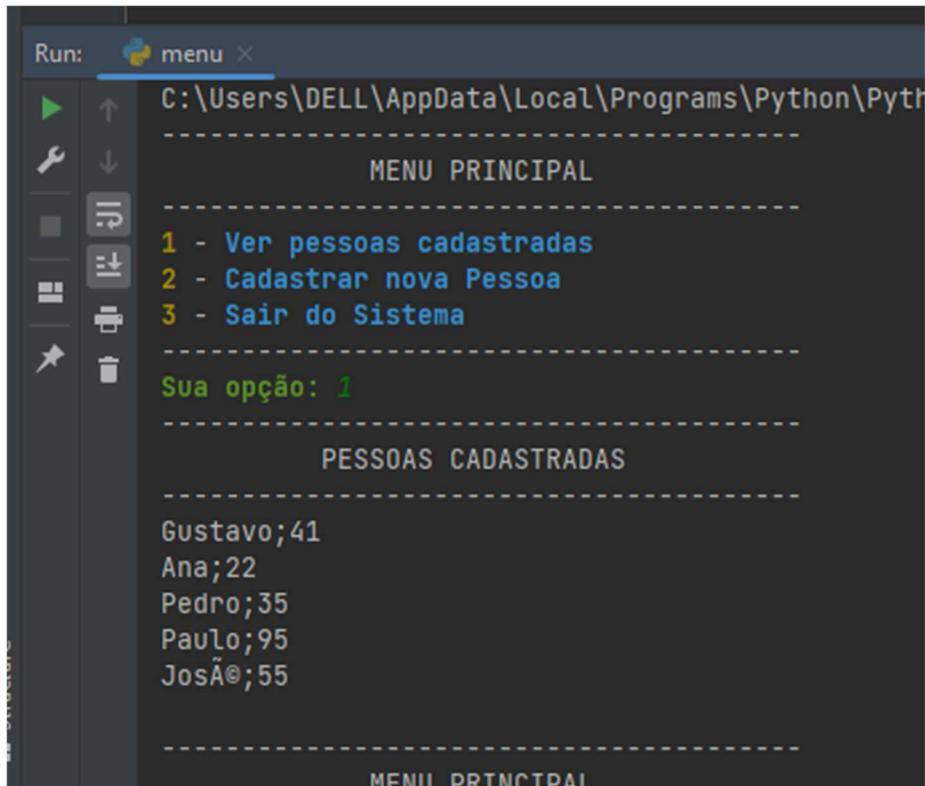
```
arquivo_init_.py x interface_init_.py x menu.py x cursoemvideo.txt x
1 def leiaInt(msg):
2     while True:
3         try:
4             resp = int(input(msg))
5         except (ValueError, TypeError):
6             print("\033[1;31mERRO: por favor, digite um número INTEIRO válido.\033[m")
7         except (KeyboardInterrupt):
8             print("\033[1;31mO usuário preferiu não digitar esse número.\033[m")
9             break
10        else:
11            if resp not in [1, 2, 3]:
12                print('\033[1;31mERRO! Digite uma opção válida.\033[m')
13            else:
14                return resp
15
16 def linha(tam=40):
17     return ' ' * tam
18
19 def cabecalho(txt):
20     print(linha())
21     print(txt.center(40))
22     print(linha())
23
24
25 def menu(lista):
26     cabecalho('MENU PRINCIPAL')
27     for pos, opcao in enumerate(lista):
28         print(f'\033[1;33m{pos+1}\033[m - \033[1;34m{opcao}\033[m')
29     print(linha())
30     opc = leiaInt('\033[1;32mSua opção:\033[m ')
31     return opc
32
```

```
arquivo_init_.py x interface_init_.py x menu.py x cursoemvideo.txt x
1 from Exercicios.ex115a.lib.interface import *
2 from Exercicios.ex115a.lib.arquivo import *
3 from time import sleep
4
5 arq = 'cursoemvideo.txt'
6
7 if not arquivoExiste(arq):
8     criarArquivo(arq)
9
10 while True:
11     opcao = menu(['Ver pessoas cadastradas', 'Cadastrar nova Pessoa', 'Sair do Sistema'])
12     if opcao == 1:
13         # Opcão de listar o conteúdo de um arquivo.
14         lerArquivo(arq)
15     elif opcao == 2:
16         cabecalho(f'OPÇÃO {opcao}')
17     elif opcao == 3:
18         cabecalho('Saindo do sistema... Até logo!')
19         break
20     else:
21         print('\033[1;31mERRO! Digite uma opção válida!\033[m')
22     sleep(2)
```



```
1 Gustavo;41
2 Ana;22
3 Pedro;35
4 Paulo;95
5 José;55
```

Run:



```
Run: menu x
C:\Users\DELL\AppData\Local\Programs\Python\Python37-32\python.exe menu.py
-----
        MENU PRINCIPAL
-----
1 - Ver pessoas cadastradas
2 - Cadastrar nova Pessoa
3 - Sair do Sistema
-----
Sua opção: 1
-----
        PESSOAS CADASTRADAS
-----
Gustavo;41
Ana;22
Pedro;35
Paulo;95
José;55
-----
        MENU PRINCIPAL
```

## Exercício 115c – Finalizando o Projeto

**Vamos finalizar o projeto de acesso a arquivos em Python.**

Código:

The image shows a screenshot of a Python code editor with two tabs open: `interface\__init__.py` and `arquivo\__init__.py`. The `interface\__init__.py` tab is active.

`menu.py`

```
1 def leiaInt(msg):
2     while True:
3         try:
4             resp = int(input(msg))
5         except (ValueError, TypeError):
6             print("\033[1;31mERRO: por favor, digite um número INTEIRO válido.\033[m")
7         except (KeyboardInterrupt):
8             print("\033[1;31mO usuário preferiu não digitar esse número.\033[m")
9             return 0
10        else:
11            return resp
12
13 def linha(tam=40):
14     return '-' * tam
15
16 def cabecalho(txt):
17     print(linha())
18     print(txt.center(40))
19     print(linha())
20
21
22 def menu(lista):
23     cabecalho('MENU PRINCIPAL')
24     for pos, opcao in enumerate(lista):
25         print(f'\033[1;33m{pos+1}\033[m - \033[1;34m{opcao}\033[m')
26     print(linha())
27     opc = leiaInt('\033[1;32mSua opção:\033[m ')
28     return opc
29
```

`menu.py`

```
1 from Exercicios.ex115a.lib.interface import cabecalho
2
3
4 def arquivoExiste(nome):
5     try:
6         a = open(nome, 'rt')
7         a.close()
8     except FileNotFoundError:
9         return False
10    else:
11        return True
12
13
14 def criarArquivo(nome):
15     try:
16         a = open(nome, 'wt+')
17         a.close()
18     except:
19         print('Houve um erro na criação do arquivo!')
20     else:
21         print(f'Arquivo {nome} criado com sucesso!')
```

```

23
24     def lerArquivo(nome):
25         try:
26             a = open(nome, 'rt')
27         except:
28             print('\033[1;31mErro ao ler o arquivo!\033[m')
29         else:
30             cabecalho('PESSOAS CADASTRADAS')
31             for linha in a:
32                 dado = linha.split(';')
33                 dado[1] = dado[1].replace('\n', '')
34                 print(f'{dado[0]}:{<30}{dado[1]}:{>3} anos')
35
36     def cadastrar(arq, nome='<desconhecido>', idade=0):
37         try:
38             a = open(arq, 'at+')
39         except:
40             print('Houve um erro na abertura do arquivo!')
41         else:
42             try:
43                 a.write(f'{nome}; {idade}\n')
44             except:
45                 print('Houve um erro na hora de escrever os dados!')
46             else:
47                 print(f'Novo registro de {nome} adicionado.')
48             a.close()
49

```

```

menu.py x interface\__init__.py x arquivo\__init__.py x
1  from Exercicios.ex115a.lib.interface import *
2  from Exercicios.ex115a.lib.arquivo import *
3  from time import sleep
4
5  arq = 'cursoemvideo.txt'
6
7  if not arquivoExiste(arq):
8      criarArquivo(arq)
9
10 while True:
11     opcao = menu(['Ver pessoas cadastradas', 'Cadastrar nova Pessoa', 'Sair do Sistema'])
12     if opcao == 1:
13         # Opcão de listar o conteúdo de um arquivo.
14         lerArquivo(arq)
15     elif opcao == 2:
16         cabecalho('NOVO CADASTRO')
17         nome = str(input("Nome: "))
18         idade = leiaInt('Idade: ')
19         cadastrar(arq, nome, idade)
20     elif opcao == 3:
21         cabecalho('Saindo do sistema... Até logo!')
22         break
23     else:
24         print('\033[1;31mERRO! Digite uma opção válida!\033[m')
25     sleep(2)
26

```

Run:

```
Run:  menu x
C:\Users\DELL\AppData\Local\Programs\Python\Python37\python.exe menu.py
-----
                MENU PRINCIPAL
-----
1 - Ver pessoas cadastradas
2 - Cadastrar nova Pessoa
3 - Sair do Sistema
-----
Sua opção: 2
-----
                NOVO CADASTRO
-----
Nome: Sabrina
Idade: 25
Novo registro de Sabrina adicionado.
-----
                MENU PRINCIPAL
-----
1 - Ver pessoas cadastradas
2 - Cadastrar nova Pessoa
3 - Sair do Sistema
-----
Sua opção: 1
-----
                PESSOAS CADASTRADAS
-----
Mesquitta           20 anos
Alvaro              2 anos
Sabrina          25 anos
|
```