

**SCRIPT EM – PYTHON –**

**Mundo 1, 2 e 3 do Curso em Vídeo**

## Sumário

Sobre o Python.....	15
Instalando o Interpretador Python .....	15
Python zen – by Tim Peters.....	17
Abrindo o Idle.....	17
Instrução de Print.....	17
Mostrando String/Mensagem .....	17
Mostrando Números .....	17
Mensagem com Números .....	17
Prática .....	18
Variáveis.....	18
Input.....	19
Criando scripts reutilizáveis .....	19
Instalando o PyCharm e QPython3 .....	20
Create new Project/Repositório.....	20
Para criar um arquivo .....	20
Tipos Primitivos e Saída de Dados.....	21
Inteiros .....	21
Float .....	21
Booleano .....	21
String .....	21
Double .....	21
Print.....	21
Print format.....	21
Na mesma linha .....	21
Quebrando Linha.....	22
Type.....	22
Print Format avançado.....	23
Print com IF- ELSE.....	24
Métodos:.....	25
isNumeric() .....	25
isAlpha() .....	25
isAlNum().....	25

isUpper().....	26
Operadores Aritméticos.....	27
Ordem de Precedência.....	27
Power().....	27
Raíz Quadrada/Cúbica.....	27
Curiosidades.....	28
Alinhamento .....	28
Formatação com alinhamento de espaços de 20 espaços .....	28
Alinhamento a direita em 20 espaços .....	28
Alinhamento a esquerda com 20 espaços.....	29
Alinhamento Centralizado.....	30
Alinhamento Centralizado, com informações entre os espaços .....	30
Formatando para ponto flutuante .....	31
Tabulação no Print .....	31
Módulos Built-In.....	32
Import .....	32
Import total da biblioteca .....	32
Import de único elemento da biblioteca .....	32
Bibliotecas Padrão do Python.....	32
Na prática .....	32
Atenção na Importação “seletiva” .....	34
Como verificar as bibliotecas padrões do Python .....	34
Importando uma biblioteca .....	35
Random .....	35
Randint .....	35
Time.....	36
Sleep .....	36
PyPI .....	37
Importando a biblioteca de Emojí .....	38
Lista.....	40
Choice(String).....	41
Manipulando Strings.....	42
Operações de Fatiamento de Strings .....	42

Fatiamento de String .....	42
Fatiamento de String em intervalos(range) .....	43
Fatiamento de String no limite .....	43
Fatiamento de String Intercalando .....	43
Fatiamento de String com contagem sem Início com Término .....	43
Fatiamento de String com contagem com Início sem Término .....	44
Fatiamento de String sem Fim e com Intervalo.....	45
Operações de Análise de Strings .....	45
Comprimento (length).....	45
Contar (count) .....	45
Contagem com Fatiamento .....	46
Procure (find).....	46
In.....	46
Operações de Transformações de Strings.....	47
Replace .....	47
Upper() .....	47
Lower().....	47
Capitalize() .....	48
Title() .....	48
Strip() .....	48
Operação de Divisão de uma String .....	49
Split() .....	49
Operação de Junção de uma String.....	50
Join() .....	50
Observações sobre Strings .....	50
Erro de atribuição de string .....	51
Replace na raíz da String.....	52
Split.....	53
Estruturas Condicionais – IF/ ELSE .....	54
If - Else .....	54
Condicional If-Else Simplificada.....	54
Condições Aninhadas - Elif .....	56
If/Else dentro de uma Função (Avançado) .....	58

Estrutura Condicional de Laço de Repetição FOR .....	59
Laço de Repetição For .....	59
Laço For Aninhado.....	59
Laço For com Contagem Regressiva/ Negativa .....	61
Iteração – Intervalo da Repetição.....	61
Recebendo um Input como parâmetro para o For.....	62
Leitura Única de Dados dentro do FOR .....	63
Estrutura Condicional de Laço de Repetição While .....	64
Loop e While Infinitos .....	65
Flag – Condição de Parada com While .....	66
Exemplo 2 de Flag.....	67
Contando números Pares e ímpares .....	68
Not in em While .....	69
Maior e Menor dentro com While .....	70
Interrompendo Repetições no While com Break .....	71
Loop Infinito.....	71
Corrigindo o problema do Flag.....	72
PEP – Python Enhancement Proposal .....	74
F-STRINGS.....	74
Print nas versões do Python .....	75
Variáveis Compostas - Tuplas.....	76
Fatiamento das Tuplas .....	76
Fatiamento com Início e com Fim.....	76
Fatiamento com Início e sem Fim .....	77
Fatiamento com Início e retroagindo. ....	77
Fatiamento com Length.....	77
Laço FOR com TUPLAS.....	77
Imutáveis .....	77
Iniciando as Tuplas .....	78
Mostrando apenas um elemento da Tupla .....	78
Mostrar a partir de um elemento da Tupla.....	79
Mostrando Intervalo da Tupla.....	79
Mostrando do Início ao Fim da Tupla.....	80

Mostrando o inverso na tupla .....	80
Tuplas Imutáveis.....	80
Mostrando Tuplas com For .....	81
Mostrando Tupla com Length .....	82
Mostrando Tuplas com For + Length .....	83
Mostrando Tupla com For através do Contador .....	84
Mostrando na Tupla, o Contador .....	85
Alinhamento na Tupla - À Esquerda .....	87
Alinhamento na Tupla – À Direita .....	88
Alinhamento na Tupla – Centralizado .....	88
Sorted.....	89
Soma em Tupla.....	89
Length na Tupla Somada .....	90
Método Count na Tupla .....	91
Propriedade/ Método Index na Tupla .....	92
Tratamento de Erro de Index.....	93
Elementos Diversos nas Tuplas .....	94
Adicionando valores na tupla .....	94
Apagar Tuplas com DEL.....	95
Variáveis Compostas - Listas .....	96
O problema das Tuplas.....	96
Lista admite substituições .....	96
Método Append .....	97
Método Insert .....	97
Apagar elementos na Lista .....	97
Método Del.....	97
Método Pop.....	98
Método Remove .....	98
Identificando Erros nas Exclusões .....	99
Método List .....	99
Criando lista através do Range .....	99
Método Sort() e Reverse ().....	100
Tamanho da Lista .....	100

Prática .....	101
Adicionando elementos de forma errada.....	102
Adicionando elementos da forma correta.....	102
Colocando números em ordem crescente .....	102
Em ordem reversa .....	103
Tamanho da Lista.....	103
Inserindo com Insert.....	104
Removendo com o pop() .....	104
Removendo a primeira ocorrência de valor com o Remove .....	105
Erro com Remove não presente na lista.....	106
Corrigindo o Problema de Elemento não listado.....	106
Imprimindo uma lista com For.....	107
Mostrando Chaves e Valores com Índices com o For .....	108
Lendo valores pelo Teclado e Adicionando a Lista + Enumerate.....	108
Atribuição de lista .....	109
CUIDADO! .....	109
Fazendo uma cópia de uma lista na outra.....	110
Variáveis Compostas - Listas – Parte 2 .....	111
Prática .....	112
Criando Lista com Lista .....	112
Erro ao fazer append com outra lista .....	112
Append com outra lista na forma correta .....	113
Print da Lista com For .....	113
Print formatando mostrando atributos das pessoas .....	115
Solicitar input do Usuário para criar a Lista .....	116
Maior de Idade e Menor de Idade com a lista, exemplo prático.....	118
Estrutura de Dados Composta – Dicionário .....	120
Identificando um Dicionário .....	120
Declarando o Dicionário .....	120
Adicionando elementos ao Dicionário criado .....	121
Removendo elemento do Dicionário.....	121
Criando um Dicionário para Guardar Elementos.....	121
Values/Valores .....	121

Keys/Chaves .....	121
Items/Item.....	121
Laço For no Dicionário.....	122
Dicionário dentro de Listas.....	122
Parte Prática.....	123
Declarando o Dicionário pronto .....	123
Erros .....	123
Print Formatado e Referência.....	124
Print nas Keys/Chaves.....	124
Print nos Values/Valores.....	124
Print nos Itens.....	125
Imprimindo as Chaves, Valores e Itens:.....	125
Usando o Del .....	126
Novas atribuições as chaves .....	126
Adicionando chaves e elementos na lista .....	127
Dicionários dentro de Lista .....	127
Imprimindo Listas Compostas com campos de Chaves .....	128
Declarando Dicionário + Leitura de Inputs .....	129
Com Laço For – Erro e Solução .....	129
Método Copy .....	131
Impressão da Lista com For .....	132
Colocando um dicionário em ordem. ....	134
Funções.....	137
Def.....	137
Estética do Python no Def .....	139
Def com Parâmetros .....	139
Na prática .....	141
Código Repetitivo e não DRY .....	141
Erro nos Parâmetros.....	142
“Invertendo” os Parâmetros.....	143
Erro na “inversão” dos parâmetros .....	144
Empacotar funções em Tuplas .....	145
Length para saber o tamanho do Pacote.....	146

Ponto de Atenção das Tuplas nos Pacotes de Funções .....	146
Empacotamento de Funções nas Listas .....	146
Desempacotamento.....	148
Funções – Parte 2.....	149
Interactive Help.....	149
Com Doc .....	151
Quit.....	152
Docstring .....	153
Iniciando uma Docstring.....	155
Parâmetros Opcionais .....	156
Erro com Parâmetros Opcionais (Mais parâmetros) .....	159
Escopo de Variáveis.....	160
Escopo Global .....	160
Escopo Local .....	161
Escopo Global e Escopo Local.....	163
Um ponto de Atenção sobre Local e Global .....	164
Variável Global aplicada a Função (na local).....	166
Escopo de Biblioteca Global e Local .....	167
Retorno de valores .....	168
Utilizando o Return.....	169
Na prática .....	171
Retorno com valor Boolean .....	172
Módulos .....	173
Vínculo entre Programa Principal e os Módulos .....	176
Importação do Módulo geral.....	176
Importação de Funções específicas do Módulo .....	177
ATENÇÃO!.....	177
Vantagens.....	178
Pacotes.....	179
Importação dos Pacotes .....	181
Como criar Pacotes .....	181
Sintaxe especial para nome de arquivo dentro de Pacotes .....	182
Criando o Pacote no PyCharm.....	183

Tratamento de Erros e Exceções.....	185
Erros de Sintaxe.....	185
Erro de NameError .....	185
Exceção de NameError .....	185
Exceção de ValueError .....	186
Exceção ZeroDivisionError.....	187
Exceção de TypeError.....	188
Exceção de IndexError (Lista) – KeyError(Dicionário).....	189
Exceção de ModuleNotFoundError .....	189
Alguns exemplos de exceções no Python.....	191
Tratamento das Exceções.....	191
Try e Except.....	191
Cores no Terminal.....	194
Padrão ANSI.....	200
Código ANSI para Cores .....	200
Aplicando Cores as Variáveis e Máscaras .....	202
Aplicando Cores com Máscara no Format() .....	202
Aplicando Cores com uma Lista de Cores.....	203
Exercícios .....	204
Desafios Iniciais .....	204
Exercício 00.....	207
Exercício 01.....	207
Exercício 02.....	207
Exercício 05.....	208
Exercício 06.....	209
Exercício 07.....	209
Exercício 08.....	210
Exercício 09.....	211
Exercício 09b.....	212
Exercício 10.....	212
Exercício 11.....	213
Exercício 12.....	213
Exercício 13.....	214

Exercício 14.....	214
Exercício 15.....	215
Exercícios de Importações de Módulos.....	215
Exercício 16.....	215
Exercício 17.....	216
Exercício 18.....	216
Exercício 19.....	217
Exercício 20.....	218
Exercício 21.....	219
Exercícios de Manipulações de Strings.....	220
Exercício 022.....	220
Exercício 023.....	221
Exercício 024.....	224
Exercício 025.....	224
Exercício 026.....	225
Exercício 027.....	226
Exercícios de Estrutura Condicional (If-Else) .....	227
Exercício 028.....	227
Exercício 029.....	228
Exercício 030.....	229
Exercício 031.....	229
Exercício 032.....	230
Exercício 033.....	230
Exercício 034.....	231
Exercício 035.....	231
Exercícios de Estruturas Aninhadas.....	232
Exercício 036.....	232
Exercício 037.....	233
Exercício 038.....	234
Exercício 039 – Serviço Militar.....	235

```
Qual o seu sexo?  
Escolha as opções  
[1]Masculino  
[2]Feminino  
Selecione: 1  
Ano de Nascimento: 1994  
Você tem 26 anos de idade.  
Você já passou do tempo de se alistar.  
Você já se alistou/deveria ter se alistado no ano de 2012
```

Exercício 040.....	236
Exercício 041.....	238
Exercício 042.....	239
Exercício 043.....	240
Exercício 044.....	241
Exercício 045.....	242
Exercícios de Estrutura de Repetição For .....	242
Exercício 046.....	242
Exercício 047.....	243
Exercício 048.....	243
Exercício 049.....	244
Exercício 050.....	244
Exercício 051.....	245
Exercício 052.....	246
Exercício 053.....	247
Exercício 054 – Maioridade 21 .....	248
Exercício 055 – Maior e Menor com For .....	249
Exercício 056 – Características com For .....	250
Exercícios de Estrutura de Repetição While .....	251
Exercício 057.....	251
Exercício 058.....	252
Exercício 059.....	253
Exercício 060.....	254
Exercício 061.....	255
Exercício 062.....	256
Exercício 063.....	257
Exercício 064.....	258
Exercício 066.....	259

Exercício 065.....	260
Exercício 067.....	261
Exercício 068.....	262
Exercício 069.....	264
Exercício 070.....	266
Exercício 071.....	268
Exercícios de Estrutura de Tuplas.....	269
Exercício 072.....	269
Exercício 073.....	270
Exercício 074.....	271
Exercício 075.....	272
Exercício 076.....	273
Exercício 077.....	274
Exercícios de Estrutura de Listas .....	275
Exercício 078.....	275
Exercício 079.....	276
Exercício 080.....	277
Exercício 081.....	278
Exercício 082.....	280
Exercício 083.....	281
Exercícios de Estrutura de Listas – Parte 2 .....	282
Exercício 084.....	282
Exercício 085.....	283
Exercício 086.....	284
Exercício 087.....	285
Exercício 088.....	287
Exercício 089.....	288
Exercícios de Estrutura de Dicionários .....	289
Exercício 090.....	289
Exercício 091.....	290
Exercício 092.....	292
Exercício 093.....	293
Exercício 094 – Exercício mais Completo e Complexo de Dicionário e Lista .....	294

Exercício 095.....	298
Exercícios de Funções – Parte 1 .....	303
Exercício 096.....	303
Exercício 097.....	304
Exercício 098.....	305
Exercício 099.....	307
Exercício 0100.....	309
Exercícios de Funções – Parte 2 .....	312
Exercício 0101.....	312
Exercício 0102.....	313
Exercício 0103 – Erro de Traceback.....	315
Exercício 0104.....	317
Exercício 0105.....	318
Exercício 0106.....	322
Exercícios de Módulos e Pacotes .....	323
Exercício 0107.....	323
Exercício 0108.....	325
Exercício 0109.....	327
Exercício 0110.....	329
Exercício 0111.....	330
Exercício 0112.....	331
Exercício 0107.....	333
Exercício 0107.....	333
Exercício 0107.....	334
Exercício de Cores no Terminal .....	345

## Sobre o Python

Um brasileiro precisa se comunicar com um grego, ele poderia usar um dicionário e isso daria certo, mas levaria horas.

Mas no lugar, poderíamos usar uma intérprete. Que traduzirá o português para o grego e vice-versa.

Agora troca a pessoa grega por um computador que entende apenas 0 e 1. No lugar da intérprete, utilizamos o INTERPRETADOR que é onde o Python é.

As linguagens de programação são intérprete, eles interpretam linguagens tanto de máquina quanto a humana.



## Instalando o Interpretador Python

Nos sistemas operacionais Mac e Linux, o Python já vem instalando. Em Windows é necessário fazer a instalação. Para verificar se o Python já está instalado, utilize o comando:

```
python --version
```

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [versão 10.0.19041.685]
(c) 2020 Microsoft Corporation. Todos os direitos reservados.

C:\Users\lucas>python --version
Python não encontrado; execute sem argumentos para instalar na Microsoft Store
Figuras > Gerenciar Aliases de Execução do Aplicativo.
```

Entre no site do [python.org](https://www.python.org).

Todo arquivo de python tem a extensão .py.



Baixe a versão mais recente e marque a opção de ADD PATH. Assim ele já vai começar a rodar no prompt e adicionar ele no Path do Windows.

Não esquecer de remover o limite no final da instalação.

Após a instalação, pode digitar: "Python" no cmd que ele irá executar o interpretador dentro do cmd.

```
C:\WINDOWS\system32\cmd.exe - PYTHON
Microsoft Windows [versão 10.0.19041.685]
(c) 2020 Microsoft Corporation. Todos os direitos reservados.

C:\Users\lucas>PYTHON
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> python --version
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'python' is not defined
>>> print('Olá, Mundo!')
Olá, Mundo!
>>>
```

## Python zen – by Tim Peters

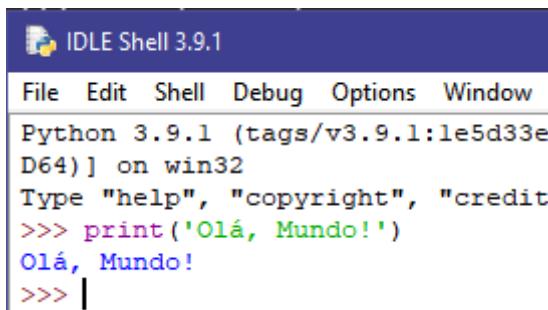
```
Ola, Mundo!
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

---

## Abrindo o Idle

Digitar o IDLE no Menu iniciar que abrirão IDLE SHELL que serve como uma IDE simples.



---

## Instrução de Print

### Mostrando String/Mensagem

Através da função: print('digite aqui')

### Mostrando Números

Através da função: print(7 + 4) = 11

### Mensagem com Números

Através do comando: print('7' + '4') = 74

## Prática

Abrir o Idle:

```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21)
D64) ] on win32
Type "help", "copyright", "credits" or "license()" for more information
>>> print('Olá, Mundo!')
Olá, Mundo!
>>> print(7+4)
11
>>> print('7'+'4')
74
>>> 7+4
11
>>> '7'+'4'
'74'
>>> print('Olá' + 5)
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    print('Olá' + 5)
TypeError: can only concatenate str (not "int") to str
>>> print ('Olá', 5)
Olá 5
>>> |
```

---

## Variáveis

Em python, toda variável é um objeto. Um objeto é algo muito maior que uma variável. E toda variável recebe algo, algum valor (=). As variáveis são espaços dentro da memória que eu posso armazenar dados.

nome recebe lucas --> **nome** = 'Lucas'

idade recebe 26 --> **idade** = 26

peso recebe --> **peso** = 82

**print** (**nome**, **idade**, **peso**) -> Lucas 26 82

Em Python, não se declara variável, se INICIA uma variável.

---

## Input

O `input` serve para receber dados dos usuários, são entradas de dados. E sempre que eu realizar um `Input`, qualquer que seja o dado digitado, será considerado um `String`, mesmo que eu tenha digitado um número. Para fazer conversões, futuramente, é necessário usar `int(input())`.

nome recebe o resultado do input de qual é o seu nome --> `nome = input` ('Qual é o seu nome?')

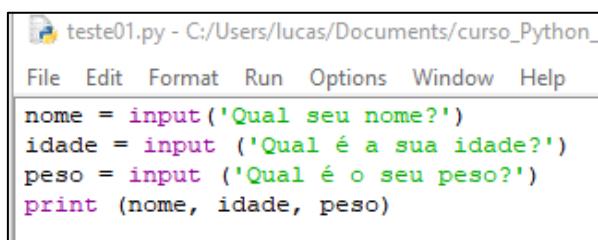
idade recebe o resultado do input de qual é a sua idade --> `idade = input` ('Qual a sua idade?')

peso recebe o resultado do input de qual é o seu peso --> `peso = input` ('Qual é o seu peso?')

```
>>> nome = input('Qual seu nome?')
Qual seu nome?Lucas
>>> idade = input('Quantos anos voce tem?')
Quantos anos voce tem?26
>>> peso = input ('Qual seu peso?')
Qual seu peso?82
>>> print(nome,idade,peso)
Lucas 26 82
>>> |
```

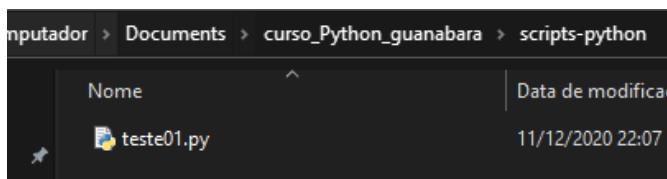
## Criando scripts reutilizáveis

Vá ao desktop > novo pasta > scripts-python. Volta no IDLE e New File e vamos criar comandos.

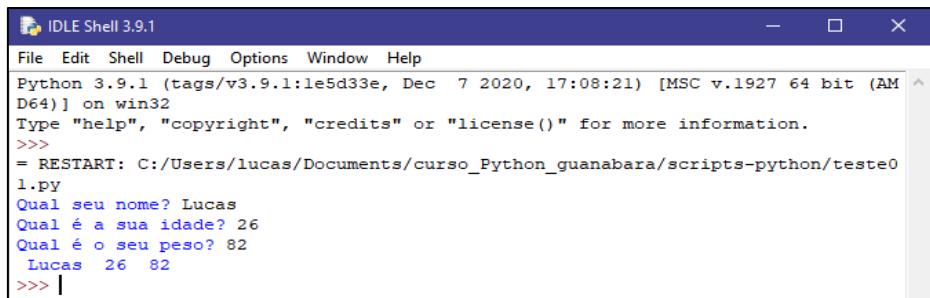


```
teste01.py - C:/Users/lucas/Documents/curso_Python_
File Edit Format Run Options Window Help
nome = input('Qual seu nome?')
idade = input ('Qual é a sua idade?')
peso = input ('Qual é o seu peso?')
print (nome, idade, peso)
```

E salvamos, vale lembrar que na verdade eu não estou mais no modo interativo, eu estou no "Modo de criação de scripts". Portanto, nenhum comando será executado. Mas foi salvo.



Clicar em Run > Run Module nesse arquivo.



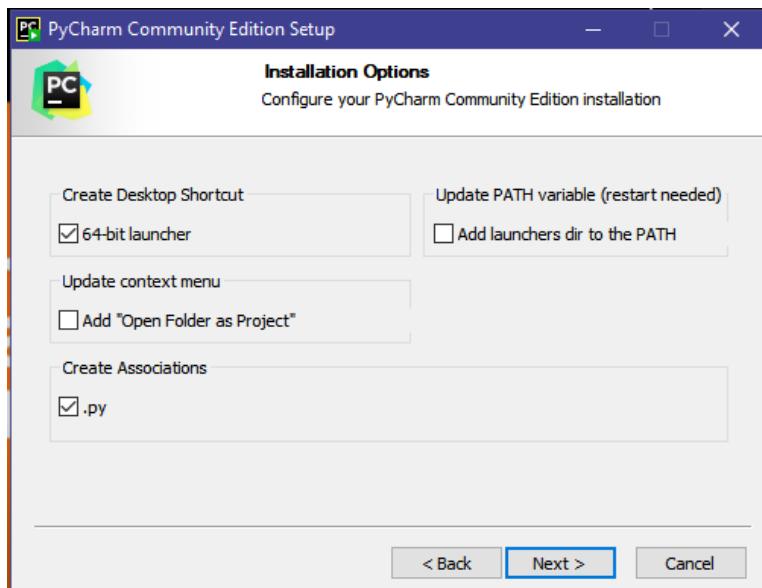
```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/lucas/Documents/curso_Python_guanabara/scripts-python/teste0
1.py
Qual seu nome? Lucas
Qual é a sua idade? 26
Qual é o seu peso? 82
Lucas 26 82
>>> |
```

## Instalando o PyCharm e QPython3

É preciso ter instalado o Python pelo menos a versão 3 para que o PyCharm funcione.

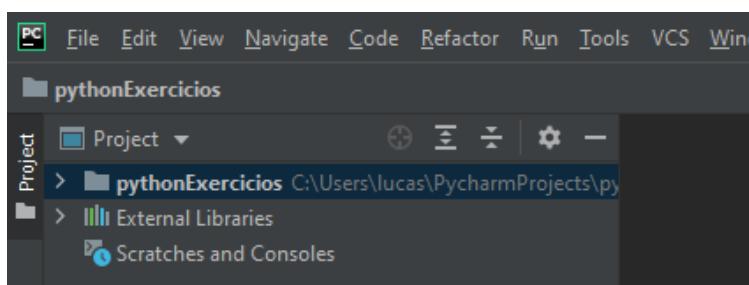
<https://www.jetbrains.com/pt-br/pycharm/download/>

Para configurar o ambiente:



## Create new Project/Repositório

Digite o nome do projeto e Create. Criará um repositório! Não é um arquivo e sim um repositório!



Para criar um arquivo

New > Python File

ObsCuidado porque precisa ser criado como python file!

# Tipos Primitivos e Saída de Dados

## Inteiros

Valores inteiros, negativos. Por exemplo: 7 / 4 / 9875 / -13

## Float

Números com casas decimais, números flutuantes, são os números reais. Por exemplo:  
4.5 / 0.075 / -15,223 / PI (3,1415) / 7.0

## Booleano

São valores lógicos, utilizar sempre True e False, sempre capitalized.

## String

São palavras, que devem estar entre aspas, simples ou duplas. Por exemplo: 'Olá', '7.0', '' (uma String vazia).

## Double

Não é considerado um tipo primitivo em Python.

---

## Print

Existe o método “comum” do print:

```
print('A soma vale ', soma)
```

## Print format

E existe o método, contendo uma máscara {}, nova sintaxe, a partir do Python 3.

```
print('A soma vale {}'.format(soma))
```

## Na mesma linha

Como se nota, pode-se usar o end= '' quando se tem um print em uma linha e deseja unir junto com o próximo print em outra linha.

The screenshot shows a PyCharm interface with a code editor and a terminal window. The code editor contains the following Python script:

```
divrest = n1 % n2
print('A soma é {}'.format(soma), end=' ')
print('A subtração é {}'.format(sub))
print('A divisão é {:.3f}'.format(div))

# aula07a
```

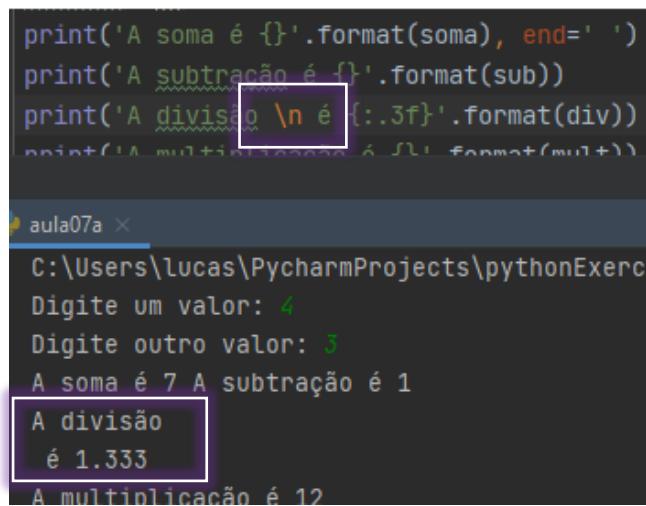
The terminal window titled "aula07a" shows the execution of the script:

```
C:\Users\lucas\PycharmProjects\pythonExerci
Digite um valor: 4
Digite outro valor: 3
A soma é 7 A subtração é 1
A divisão é 1.333
```

A red box highlights the comma and space in the first print statement: ', end=' ' '. This is the separator used to print multiple values on the same line without a new line character.

## Quebrando Linha

No meio do código podemos quebrar a linha também, basta utilizar o '\n' (contra-barra n)



```
print('A soma é {}'.format(soma), end=' ')
print('A subtração é {}'.format(sub))
print('A divisão \n é {:.3f}'.format(div))
print('A multiplicação é {}'.format(mult))
```

aula07a ×

C:\Users\lucas\PycharmProjects\pythonExercícios>python aula07a.py

Digite um valor: 4

Digite outro valor: 3

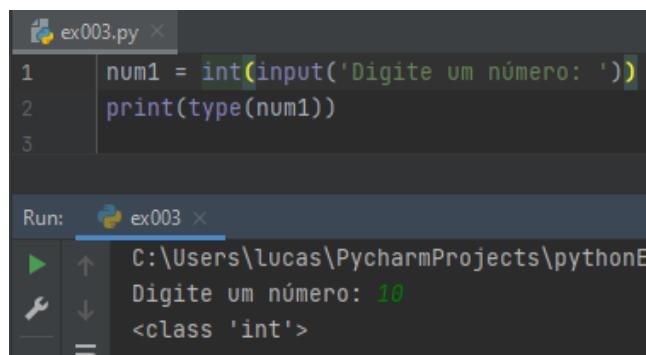
A soma é 7 A subtração é 1

A divisão  
é 1.333

A multiplicação é 12

## Type

Pode utilizar o método Type para verificar qual é o tipo de dado que está armazenado na variável.



```
1 num1 = int(input('Digite um número: '))
2 print(type(num1))
3
```

Run: ex003 ×

C:\Users\lucas\PycharmProjects\pythonExercícios>python ex003.py

Digite um número: 10

<class 'int'>

## Print Format avançado

A maneira de mostrar variáveis e o resultado dela pode ser feito dessa forma:

The screenshot shows the PyCharm interface. The code editor window has a tab labeled 'ex003.py'. The code itself is:

```
1 num1 = int(input('Digite o primeiro número: '))
2 num2 = int(input('Digite o segundo número: '))
3 soma = num1 + num2
4
5 print('A soma entre ', num1, ' e o número ', num2, ' é de ', soma)
```

Below the code editor is the 'Run' tool window, which has a tab labeled 'ex003'. It shows the following interaction:

```
Run: ex003 ×
C:\Users\lucas\PycharmProjects\pythonExercicios\venv\Scripts\python.exe ex003.py
Digite o primeiro número: 5
Digite o segundo número: 10
A soma entre 5 e o número 10 é de 15
```

Mas pode ser feito muito simples:

```
print('A soma entre {} e {} vale {}'.format(num1, num2, soma))
```

The screenshot shows the PyCharm interface again. The code editor window has a tab labeled 'ex003.py'. The code is now:

```
1 num1 = int(input('Digite o primeiro número: '))
2 num2 = int(input('Digite o segundo número: '))
3 soma = num1 + num2
4
5 #print('A soma entre ', num1, ' e o número ', num2, ' é de ', soma)
6 print('A soma entre {} e {} vale {}'.format(num1, num2, soma))
```

The run output is identical to the previous one:

```
Run: ex003 ×
C:\Users\lucas\PycharmProjects\pythonExercicios\venv\Scripts\python.exe ex003.py
Digite o primeiro número: 5
Digite o segundo número: 10
A soma entre 5 e 10 vale 15
```

### Atenção:

Existe um novo método do print com 'f-strings', verificar no capítulo: PEP > F-STRINGS

## Print com IF- ELSE

É possível fazer uma condição if-else dentro de um print, onde algo somente algo será printado na tela se atender uma requisição do if-else dentro do print.

The screenshot shows a PyCharm interface with two tabs: 'ex060b' and 'ex060c'. The 'ex060c' tab is active, displaying Python code. The code calculates the factorial of a number entered by the user. It uses a while loop to iterate from the input number down to 1, printing each factor followed by a multiplication sign ('x') or an equals sign ('='). The terminal window below shows the execution of the script 'ex060c' and its output for the number 10.

```
numero = int(input('Digite um número para calcularmos o fatorial: '))
c = numero
fatorial = 0
proximo = numero

while c > 0:
    print('{} '.format(c), end=' ')
    print('x ' if c > 1 else ' = ', end=' ')
    c -= 1

while c > 0
```

ex060b × ex060c ×

C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/U  
Digite um número para calcularmos o fatorial: 10  
10 x 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1 =  
FIM

Métodos:

### isNumeric()

Verifica se o que foi digitado é numérico ou não

```
aula06b.py ×
1 n = input('Digite algo: ')
2 print(n.isnumeric())
```

Run: aula06b ×  
C:\Users\lucas\PycharmPro  
Digite algo: oi  
False

### isAlpha()

Verifica se algo é alfabético, não string, mas sim alfabético.

```
aula06b.py ×
1 n = input('Digite algo: ')
2 print(n.isalpha())
```

Run: aula06b ×  
C:\Users\lucas\PycharmPro  
Digite algo: oi  
True

```
aula06b.py ×
1 n = input('Digite algo: ')
2 print(n.isalpha())
```

Run: aula06b ×  
C:\Users\lucas\PycharmPro  
Digite algo: 123  
False

### isAlNum()

```
aula06b.py ×
1 n = input('Digite algo: ')
2 print(n.isalnum())
```

Run: aula06b ×  
C:\Users\lucas\PycharmPro  
Digite algo: aaa  
True

```
aula06b.py ×
1 n = input('Digite algo: ')
2 print(n.isalnum())
```

Run: aula06b ×  
C:\Users\lucas\PycharmPro  
Digite algo: 3333aaaa  
True

```
aula06b.py ×
1 n = input('Digite algo: ')
2 print(n.isalnum())
```

Run: aula06b ×  
C:\Users\lucas\PycharmPro  
Digite algo: @@@@@"
False

```
aula06b.py ×
1 n = input('Digite algo: ')
2 print(n.isnumeric())
```

Run: aula06b ×  
C:\Users\lucas\PycharmPro  
Digite algo: 10  
True

Verifica se algo é  
alfanumérico.

## isUpper()

The image shows two side-by-side code editors in PyCharm. Both editors contain the same Python script:

```
aula06b.py
1 n = input('Digite algo: ')
2 print(n.isupper())
```

Below each editor is a 'Run' window. The left 'Run' window shows the output for the input 'oi', which is 'True'. The right 'Run' window shows the output for the input 'oi', which is 'False'.

Verifica se todos os valores são maiúsculos.

Existem vários e vários exemplos do “is”.

Existem o “is” para lower, se existem espaços ou não, entre vários outros.

# Operadores Aritméticos

Utilizaremos o operandus (5) e teremos o operador binário (precisa de dois operandus) e o operandus 2.

Adição (+):  $5 + 2 == 7$

Subtração (-):  $5 - 2 == 3$

Divisão (/):  $5 / 2 == 2.5$

Multiplicação (\*):  $5 * 2 == 10$

Exponenciação (\*\*):  $5 ^ 2 == 25$

Divisão Inteira (//):  $5 // 2 == 2$

Resto da Divisão (%):  $5 \% 2 == 1$  Vale lembrar que o Módulo(ou o resto da divisão, na verdade é a divisão normal e aí sim é puxado o resto dela.)

5 (5 divide 2)	2
1	2 (Divisão Inteira)
5 (5 divide 2)	2
1 (Resto da Divisão/Módulo)	2

## Ordem de Precedência

1. Parênteses
2. Exponencial
3. Multiplicação/Divisão/Divisão Inteira/Resto da Divisão
4. Soma/Subtração

```
C:\Users\lucas>python
Python 3.9.1 (tags/v3.9.1-0-2020.11.24-0814)
Type "help", "copyright" or "credits" for more information.
>>> 5+3*2
11
>>> 3*5+4^2
17
>>> 3*5+4**2
31
>>> 3*(5+4)**2
243
>>>
```

## Power()

Ele é a função interna de potência. O pow(x,y), funciona de forma que o X é o número que quero ser elevado e o Y o número elevar. Por exemplo, pow(5,2) == 25. Vale lembrar que isso faz quebrar a precedência.

## Raíz Quadrada/Cúbica

$81^{(1/2)} == 9.0$

$25^{(1/2)} == 5.0$

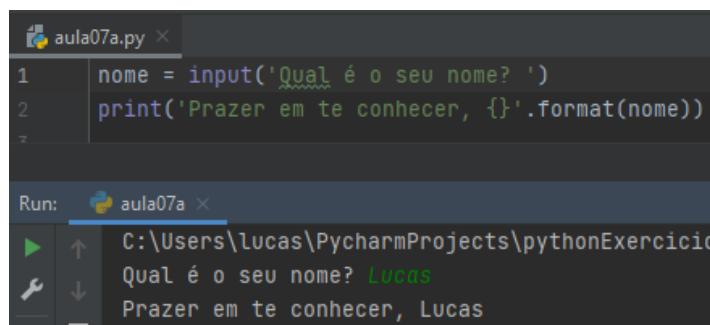
$127^{(1/3)} == 5,026$

## Curiosidades

```
>>> 'Oi' + 'Olá'  
'Oiolá'  
>>> 'oi'*5  
'oioioioiooi'  
>>> print('Oi ' *5)  
Oi Oi Oi Oi Oi
```

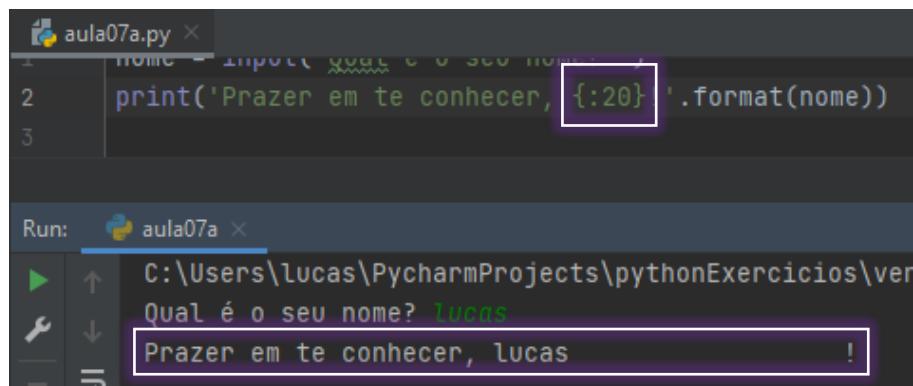
## Alinhamento

Formatação “padrão”



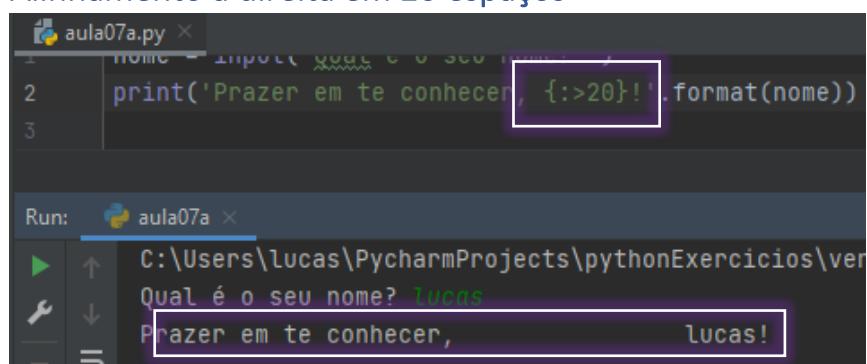
```
aula07a.py ×  
1 nome = input('Qual é o seu nome? ')  
2 print('Prazer em te conhecer, {}'.format(nome))  
  
Run: aula07a ×  
C:\Users\lucas\PycharmProjects\pythonExercicios\ver  
Qual é o seu nome? Lucas  
Prazer em te conhecer, Lucas
```

Formatação com alinhamento de espaços de 20 espaços, por exemplo. [{:qtdDeEspaço}](#)



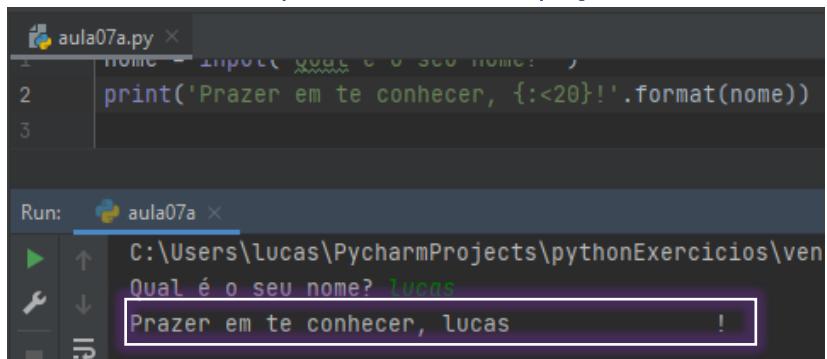
```
aula07a.py ×  
1 nome = input('Qual é o seu nome? ')  
2 print('Prazer em te conhecer, {:>20}'.format(nome))  
  
Run: aula07a ×  
C:\Users\lucas\PycharmProjects\pythonExercicios\ver  
Qual é o seu nome? Lucas  
Prazer em te conhecer, !
```

Alinhamento a direita em 20 espaços



```
aula07a.py ×  
1 nome = input('Qual é o seu nome? ')  
2 print('Prazer em te conhecer, {:<20}!'.format(nome))  
  
Run: aula07a ×  
C:\Users\lucas\PycharmProjects\pythonExercicios\ver  
Qual é o seu nome? Lucas  
Prazer em te conhecer, !
```

## Alinhamento a esquerda com 20 espaços



The screenshot shows the PyCharm IDE interface. In the top-left corner, there's a code editor window titled "aula07a.py" containing the following Python code:

```
1 nome = input('Qual é o seu nome? ')
2 print('Prazer em te conhecer, {:<20}!'.format(nome))
3
```

In the bottom-right corner, there's a terminal window titled "Run: aula07a" showing the output of the program. The user typed "Lucas" into the terminal, and the program responded with "Prazer em te conhecer, lucas !".

## Alinhamento Centralizado

The screenshot shows a PyCharm interface. In the editor, there is a file named 'aula07a.py' with the following content:

```
1 nome = input('Qual é o seu nome? ')
2 print('Prazer em te conhecer, {:^20}!'.format(nome))
3
```

In the terminal window, the command 'Run: aula07a' is selected. The output shows:

```
C:\Users\lucas\PycharmProjects\pythonExercicios\venv
Qual é o seu nome? lucas
Prazer em te conhecer,      lucas      !
```

The output is centered within a 20-character width.

## Alinhamento Centralizado, com informações entre os espaços

The screenshot shows a PyCharm interface. In the editor, there is a file named 'aula07a.py' with the following content:

```
1 nome = input('Qual é o seu nome? ')
2 print('Prazer em te conhecer, {:=>20}!'.format(nome))
3
```

In the terminal window, the command 'Run: aula07a' is selected. The output shows:

```
C:\Users\lucas\PycharmProjects\pythonExercicios\venv
Qual é o seu nome? Lucas
Prazer em te conhecer, =====Lucas=====!
```

The output is centered within a 20-character width, with the padding information '{:=>20}' highlighted in the code.

The screenshot shows a PyCharm interface. In the editor, there is a file named 'aula07a.py' with the following content:

```
8 divint = n1 // n2
9 divrest = n1 % n2
10 print('A soma é {}'.format(soma))
11 print('A subtração é {}'.format(sub))
12 print('A divisão é {:.3f}'.format(div)) (A divisão é {:.3f})
13 print('A multiplicação é {}'.format(mult))
14 print('A expo é {}'.format(expo))
15 print('A divisão inteira é {}'.format(divint))
16 print('O resto da divisão é {}'.format(divrest))
17
18
```

In the terminal window, the command 'Run: aula07a' is selected. The output shows:

```
C:\Users\lucas\PycharmProjects\pythonExercicios\venv
Digite um valor: 4
Digite outro valor: 3
A soma é 7
A subtração é 1
A divisão é 1.333
A multiplicação é 12
A expo é 64
A divisão inteira é 1
O resto da divisão é 1
```

The output shows various arithmetic operations and their results, with the division result '{:.3f}' highlighted in the code.

## Formatando para ponto flutuante

Quando se utilizar da formatação, podemos utilizar o `(:.xf)`, onde 'x' vale o número que quero de casas decimais após a vírgula. Nota-se na página 14.

## Tabulação no Print

Basta colocar um “`\t`” antes do início do que precisa ser tabulado.

```
print(f'Preço Analisado: \t{monetFormat(preco)}')
print(f'Dobro do Preço: \t{dobro(preco, True)}')
print(f'Metade do Preço: \t{metade(preco, True)}')
print(f'{aumento}% de aumento: \t{aumentar(preco, a)}')
print(f'{desconto}% de desconto: \t{diminuir(preco, d)}')
mostrarLinha()

resumo()
# main x

Digite um preço: R$ 500
-----
RESUMO DO VALOR
-----
Preço Analisado:      R$500,00
Dobro do Preço:       R$1000,00
Metade do Preço:      R$250,00
80% de aumento:       R$900,00
35% de desconto:      R$325,00
-----
```

# Módulos Built-In

Como estender a linguagem, como deixar o programa mais “funcional”, além do padrão da linguagem. Funciona a partir de pacotes, de programa pré-definidos. O programa vem com o básico para poder funcionar, os pacotes são recursos extras. O corpo humano vem do jeito que ele é. Para manter a energia durante o dia precisa se alimentar, com doces, comidas salgadas, e líquidos. Essas adições são chamadas de bibliotecas. Bibliotecas de comidas por exemplo e eu faço importações dessas bibliotecas. É adicionar coisas para ter mais funcionalidades. Um carro por exemplo vem puro, com rodas, motor, portas, mas eu queira colocar um teto solar, um painel ou um rádio mais bonito e melhor.

## Import

É necessário utilizar o import. Quando eu dou um “**import bebidas**” eu posso utilizar todas as bebidas possíveis.

### Import total da biblioteca

**Import bebidas** - > café, suco, refri, leite

**Import doces** -> pudim, chocolate, sorvete, bolo

Dessa forma, eu faço a importação de todos os elementos na biblioteca.

### Import de único elemento da biblioteca

Porém, se eu quiser importar somente algum elemento da biblioteca, eu preciso utilizar uma forma diferente do import. Após o import, aperte CTRL + Espaço para abrir a lista de importações.

**from doces import pudim** (e assim eu só importarei um único elemento da biblioteca)

## Bibliotecas Padrão do Python

O Python já vem com o padrão da biblioteca “Math” que possuí fórmulas matemáticas integradas.

Biblioteca Math:

- ceil (arredondamento para cima)
- floor (arredondamento para baixo)
- trunc (trunca um valor após a vírgula)
- pow (exponenciação)
- sqrt (raíz quadrada)
- factorial (acha o fator)

Importação total: **import math**

Importação de elementos: **from math import sqrt, ceil**

## Na prática

Quando eu for utilizar no código, eu preciso importar em cada arquivo .py

Quando eu crio a variável, eu chamo a biblioteca e coloco um ponto, quando eu faço essa adição do ponto, ele trás para mim todas as funções que estão presentes na biblioteca que eu importei.

```
import math

num = int(input('Digite um número: '))
raiz = math.
```

```
import math

num = int(input('Digite um número: '))
raiz = math.sqrt(num)
print('A raiz do {} é igual a {}'.format(num, raiz))

aula08a ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
Digite um número: 81
A raiz do 81 é igual a 9.0
```

Posso usar para arredondar para cima uma raiz também, por exemplo, a raiz de 29 é 5.385164807134504. Eu posso arredondar para cima ou para baixo no próprio format.

```
import math

num = int(input('Digite um número: '))
raiz = math.sqrt(num)
print('A raiz do {} é igual a {}'.format(num, math.ceil(raiz)))

aula08a ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
Digite um número: 29
A raiz do 29 é igual a 6
```

```
import math

num = int(input('Digite um número: '))
raiz = math.sqrt(num)
print('A raiz do {} é igual a {:.2f}'.format(num, math.ceil(raiz)))
print('A raiz do {} é igual a {:.2f}'.format(num, math.floor(raiz)))
```

```
aula08a x
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
Digite um número: 29
A raiz do 29 é igual a 6.00
A raiz do 29 é igual a 5.00
```

### Atenção na Importação “seletiva”

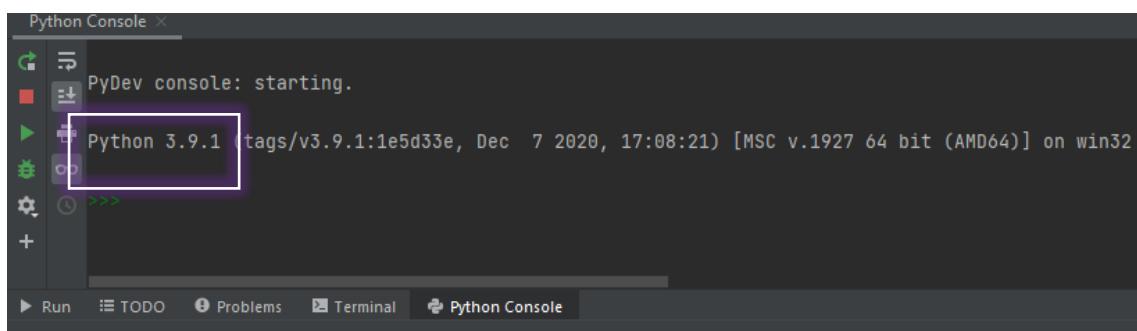
Ao realizar a importação selecionando elementos na biblioteca, não se deve utilizar o math. Antes do elemento a ser selecionado!

```
#import math
from math import sqrt, ceil, floor

num = int(input('Digite um número: '))
raiz = sqrt(num)
print('A raiz do {} é igual a {:.2f}'.format(num, ceil(raiz)))
print('A raiz do {} é igual a {:.2f}'.format(num, floor(raiz)))
```

## Como verificar as bibliotecas padrões do Python

Primeiro, é necessário clicar no Python Console e verificar qual a versão instalada do Python.



E procurar pela documentação do python na <https://docs.python.org/3/> > <https://docs.python.org/3/library/index.html>

# Bibliotecas

Ao longo desse capítulo, vamos nos deparar com bibliotecas simples e outras mais avançadas, caso não entenda alguma, é porque o assunto será abordado mais para frente. Portanto, atente-se a somente aquelas que serão necessárias em alguns exercícios e o momento exato que estiver necessitando delas.

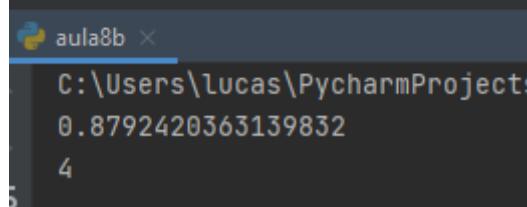
## Random

Elá gera número aleatórios, através do random.random() e também posso adicionar uma função do random que aceita gerar números entre um intervalo de inteiros.

### Randint

```
import random

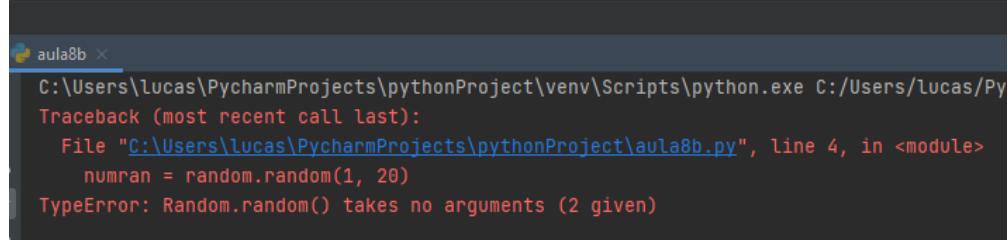
num = random.random()
# numran = random.random(1, 20)
numint = random.randint(1, 10)
print(num)
print(numint)
```

Possíveis erros:

```
import random

num = random.random()
# numran = random.random(1, 20) esse vai dar erro porque não aceita argumentos
numint = random.randint(1, 10)
print(num)
print(numint)
```

## Time

Essa biblioteca, no seu geral, faz com que o programa “durma” durante algum tempo, e assim, de a sensação de “pensamento”. Através do sleep.

### Sleep

Através da importação da biblioteca TIME. Após a importação, pode utilizar somente uma linha de código com

sleep (valorEmSegundos)

```
import random
from time import sleep

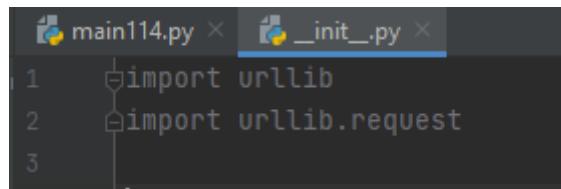
numint = random.randint(0, 5) #Cria um número aleatório entre 0 e 5
#print(numint)
print('---' * 20)
numUser = int(input('Digite um número: '))
print('---' * 20)
print('Processando os dados...')
sleep(2)
```

---

## UrlLib

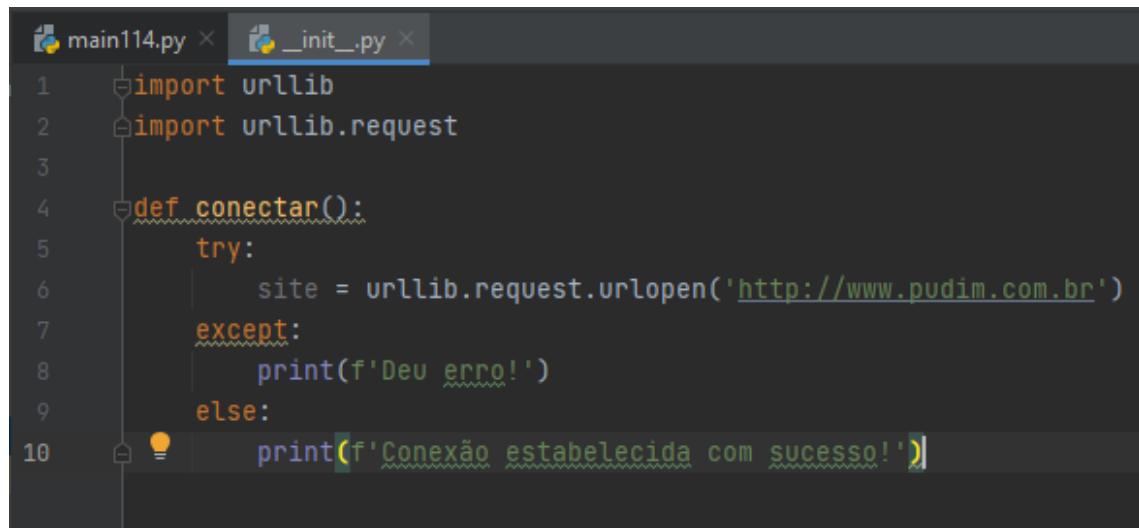
Essa biblioteca será muito útil no final do curso, no exercício 114 em Tratamento de Funções. Ela serve para verificar e estabelecer uma conexão entre o programa e algum dado site na internet.

Utilizaremos o módulo interno chamado “REQUEST”

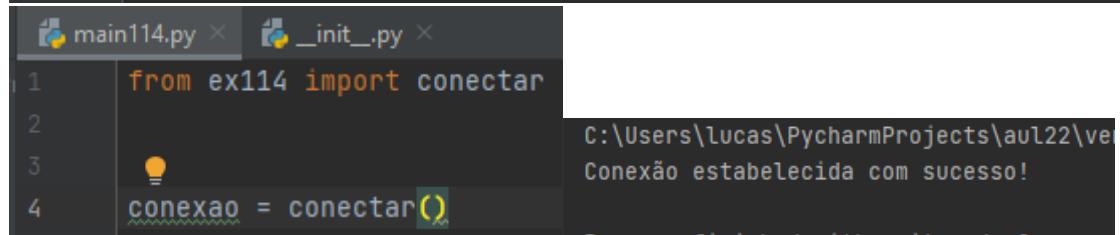


```
main114.py × _init_.py ×
1 import urllib
2 import urllib.request
3
```

Primeira tentativa de conexão e com o programa simples



```
main114.py × _init_.py ×
1 import urllib
2 import urllib.request
3
4 def conectar():
5     try:
6         site = urllib.request.urlopen('http://www.pudim.com.br')
7     except:
8         print(f'Deu erro!')
9     else:
10        print(f'Conexão estabelecida com sucesso!')
```



```
main114.py × _init_.py ×
1 from ex114 import conectar
2
3 conexao = conectar()
4
C:\Users\lucas\PycharmProjects\aul22\ver
Conexão estabelecida com sucesso!
Process finished with exit code 0
```



```
try:
    site = urllib.request.urlopen('http://www.pudim.br')
except:
    print(f'Deu erro!')
```

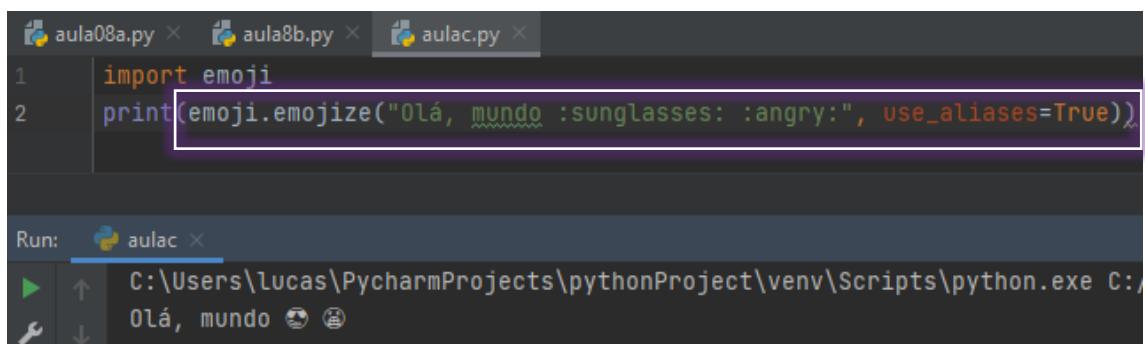
```
C:\Users\lucas\
Deu erro!
```

# PyPI

É o Python Package Index, um local onde existem diversas bibliotecas que não são built-in no Python e pode fazer o download para incluir bibliotecas no código e nos programas

## Importando a biblioteca de Emoji

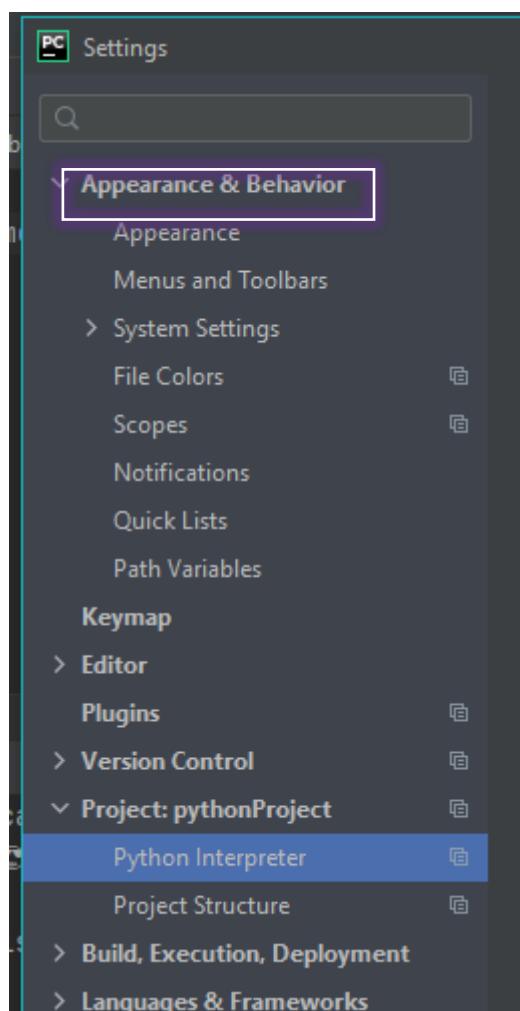
Digite a import emoji e por se tratar de uma biblioteca externa, ele vai aparecer um erro e a lâmpada vermelha, basta clicar e ele irá adicionar. Precisa fazer igual a biblioteca do random e fazer a chamada da biblioteca para utilizar. Muito importante não esquecer disso.



The screenshot shows the PyCharm interface. At the top, there are three tabs: 'aula08a.py', 'aula8b.py', and 'aulac.py'. The 'aulac.py' tab is active. The code in 'aulac.py' is:

```
1 import emoji
2 print(emoji.emojize("Olá, mundo :sunglasses: :angry:", use_aliases=True))
```

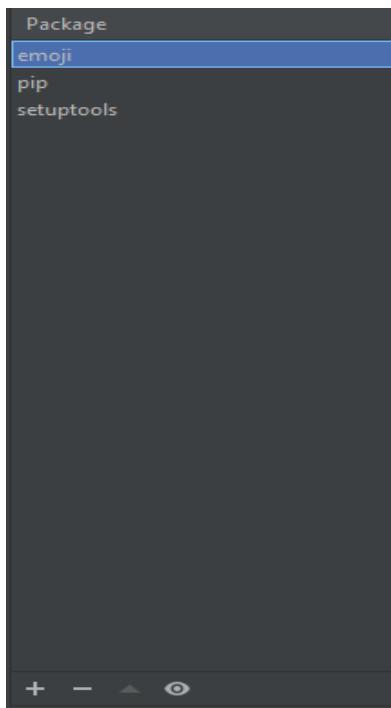
In the bottom right corner, there is a terminal window titled 'Run: aulac'. It shows the command: 'C:\Users\Lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/'. Below the command, the output is: 'Olá, mundo ☀️😡'. The status bar at the bottom indicates 'File: aulac.py'.



Verificar quais bibliotecas externas estão instaladas

File> Settings > Project > Python Interpreter

Clicar no Interpretador e eu posso clicar no Package e selecionar e a partir daí eu posso clicar no + para adicionar pacotes no projeto e para remover eu selecione o pacote a ser removido e clico no - para remover.



## Lista

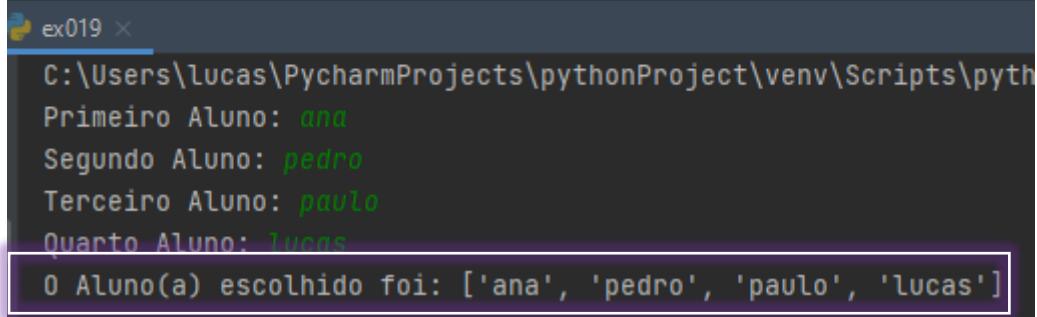
Para criar uma lista em Python é necessário atribuir a uma variável, uma lista. Por exemplo:

```
lista = [nome1, nome2, nome3, nome4]
```

Estrando, portanto, entre colchetes ([ ]).

```
n1 = str(input('Primeiro Aluno: '))
n2 = str(input('Segundo Aluno: '))
n3 = str(input('Terceiro Aluno: '))
n4 = str(input('Quarto Aluno: '))
lista = [n1, n2, n3, n4]

print('O Aluno(a) escolhido foi: {}'.format(lista))
```



```
ex019 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe ex019.py
Primeiro Aluno: ana
Segundo Aluno: pedro
Terceiro Aluno: paulo
Quarto Aluno: lucas
O Aluno(a) escolhido foi: ['ana', 'pedro', 'paulo', 'lucas']
```

Posso criar várias listas, uma de compras, por exemplo.

## Choice(String)

```
from random import choice
n1 = str(input('Primeiro Aluno: '))
n2 = str(input('Segundo Aluno: '))
n3 = str(input('Terceiro Aluno: '))
n4 = str(input('Quarto Aluno: '))
lista = [n1, n2, n3, n4]
escolhido = choice(lista)

print('A lista de alunos é: {}'.format(lista))
print('O Aluno(a) escolhido foi: {}'.format(escolhido))
```

Run: ex019

```
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\
Primeiro Aluno: laura
Segundo Aluno: tony
Terceiro Aluno: nina
Quarto Aluno: lucas
A lista de alunos é: ['laura', 'tony', 'nina', 'lucas']
O Aluno(a) escolhido foi: tony
```

Mais informações estão nos exercícios.

# Manipulando Strings

O que é uma cadeia de caracteres/cadeia de texto/string.

É uma frase.

Para Python, uma cadeia de caracteres, estará entre aspas simples ou duplas, mais frequente entre aspas simples.

Português se lê: Curso em video Python

Python lê: frase = 'Curso em video Python'

O Python, na verdade, pega cada carácter de uma frase e aloca em espaços da memória. Coloca em cada pedacinho dela, como abaixo, incluindo espaços.

frase =

C	U	R	S	O		E	M		V	I	D	E	O		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

## Operações de Fatiamento de Strings

### Fatiamento de String

Essa técnica consistente em fatiar pedaços da string.

Por exemplo, dentro de uma estrutura de dados (lista), onde eu tenho a variável frase, onde ela tem todas as strings acima.

Se eu mandar ele print (frase), ele irá imprimir toda a cadeira de carácter acima.

Porém, se eu mandar imprimir apenas `print ( frase [ 9 ] )`. O Python vai identificar e imprimirá somente a letra 9. Sim, na lista ele lerá a letra V.

C	U	R	S	O		E	M		V	I	D	E	O		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado: V

---

### Fatiamento de String em intervalos(range)

Consiste na mesma técnica do fatiamento, só que aplicando em um intervalo dado. Eu preciso selecionar até o 14, pois o último elemento que eu quero selecionar é a string 'O', portanto, eu tenho que ir sempre 1 a mais do limite que eu quero. E portanto, o espaço vazio da 14 não entrará na seleção que eu preciso, trazendo somente o 'video', incluindo o O e removendo o espaço em branco.

```
print ( frase [ 9 :14 ] )
```

C	U	R	S	O		E	M	V	I	D	E	O		P	Y	T	H	O	N	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado: VIDEO

### Fatiamento de String no limite

E se eu for além do limite? Por exemplo, pedir até o 21? Nesse caso, ele irá isolar o limite, indo somente até o 20, independente.

```
print ( frase [ 9 :21 ] )
```

C	U	R	S	O		E	M	V	I	D	E	O		P	Y	T	H	O	N	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado: VIDEO PYTHON

---

### Fatiamento de String Intercalando

E se eu quiser que imprima até o final, só que intercalando valores? Por exemplo, pulando de dois em dois?

```
print ( frase [ 9 :21 :2 ] )
```

C	U	R	S	O		E	M	V	I	D	E	O		P	Y	T	H	O	N	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
								0	1	2	1	2	1	2	1	2	1	2	1	

Resultado: VDO PTO

---

### Fatiamento de String com contagem sem Início com Término

Antes dos dois pontos (:) é onde ele vai começar e depois dos dois pontos (:) é onde ele vai terminar.

```
print frase [ :5 ]
```

C	U	R	S	O		E	M	V	I	D	E	O		P	Y	T	H	O	N	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado: CURSO

### Fatiamento de String com contagem com Início sem Término

Eu sei o início e não sei onde termina, os dois pontos (:) após o número, significa que eu não sei onde termina, mas o número antes dos dois pontos (:), eu sei onde começa.

```
print ( frase [ 15 : ] )
```

C	U	R	S	O		E	M		V	I	D	E	O		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado: PYTHON

## Fatiamento de String sem Fim e com Intervalo

Onde eu sei um início, não sei o final e quero ir pulando/intercalando na cadeia de carácter.

```
print ( frase [ 9 :: 3 ] )
```

C	U	R	S	O		E	M		V	I	D	E	O		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
									0	1	2	3	1	2	3	1	2	3	1	2

Resultado: VE PH

---

## Operações de Análise de Strings

É possível analisar a string, por exemplo, com que letra começa, qual termina, entre outras.

frase =

C	U	R	S	O		E	M		V	I	D	E	O		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

### Comprimento (length)

Se eu mandar imprimir o length da frase, obterei 21 como resposta

```
print ( len (frase) )
```

C	U	R	S	O		E	M		V	I	D	E	O		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado: 21 caracteres

---

### Contar (count)

Eu vou pedir para o programa contar quantas vezes algum carácter específico aparece, qual a incidência desse carácter na string.

```
print ( frase . count( 'o' ) )
```

C	u	r	s	o		E	m		V	i	d	e	o		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado: 3

## Contagem com Fatiamento

É um pouco mais complexo, pois, deve ler após o que eu quero contar. Por exemplo, abaixo eu desejo contar a letra 'o' minúscula dentro do intervalo do 0 ao 14.

Lembrando que é sempre um a menos de qual eu desejo saber, portanto, como eu quero que a letra 'o' de 'video' esteja inclusa, eu preciso contar até uma a mais de sua posição na 13.

```
print ( frase . count ( ' o ', 0 , 14 ) )
```

C	u	r	s	o		E	m		V	i	d	e	o		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado: 2

---

## Procure (find)

Ele procurará por alguma string definida. Por exemplo:

```
print ( frase . find ( ' deo ' ) )
```

C	U	R	S	O		E	M		V	I	D	E	O		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado: 11 (Começou na posição 11, ele retornará 11 apenas)

E se eu colocar algo que não existe?

```
print ( frase . find ( ' android ' ) )
```

O resultado será -1.

Quando se obtém um resultado de -1, significa que essa função não existe na string pedida. Uma vez que não existe a posição -1.

O **find** sempre me retornará apenas qual a posição estará algo que procurei.

---

## In

Essa função se lê, "dentro da variável da lista que é variável lista, existe a palavra/cadeia de string/carácter "Curso"?"

O "in" nunca me retornará qual a posição, ele retornará **True or False**

'Curso' **in** frase

C	U	R	S	O		E	M		V	I	D	E	O		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado: True

## Operações de Transformações de Strings

Em via de regra, uma cadeia de string, é imutável, não é possível alterar. Mas é possível alterar através métodos.

### Replace

Esse método reposiciona, substitui. Por exemplo, ele procura pelo primeiro argumento e substitui pelo segundo.

```
print ( frase . replace ( 'Python' , 'Android' ) )
```

C	U	R	S	O		E	M		V	I	D	E	O		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado:

C	U	R	S	O		E	M		V	I	D	E	O		A	N	D	R	O	I	D
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

---

### Upper()

Para cima, em maiúscula.

```
frase. upper()
```

C	u	r	s	o		E	m		V	i	d	e	o		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado:

C	U	R	S	O		E	M		V	I	D	E	O		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

---

### Lower()

O contrário do upper().

C	U	R	S	O		E	M		V	I	D	E	O		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado:

c	u	r	s	o		e	m		v	i	d	e	o		p	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

## Capitalize()

Ele pegará tudo na string e colocará em minúsculo, exceto a primeira letra da string que será mantida/convertida para maiúsculo. Não pode confundir com o Title.

C	U	R	S	O		E	M		V	I	D	E	O		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado:

C	u	r	s	o		e	m		v	i	d	e	o		p	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

---

## Title()

De forma parecida com o capitalize, ele transformará **cada** letra inicial de cada palavra e ela se tornará maiúscula. Letra inicial palavra por palavra.

frase.title()

C	u	r	s	o		E	m		V	i	d	e	o		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

---

## Strip()

Remover os espaços excedentes no início e no final da cadeia de strings. Por exemplo, uma pessoa mais idosa ao fazer um cadastro em um site, ela acaba por preencher um campo com vários espaços antes de começar a digitar algo, armazenando em espaços, valores vazios. Com o método do strip() ele removerá esses espaços no início e no fim da cadeia de caracteres. Embora os espaços já definidos não ocupem os espaços removidos pela strip. Permanecendo cada um no seu lugar de origem.

frase =

			A	P	R	E	N	D	O		P	Y	T	H	O	N		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

frase.strip()

X	X	X	A	P	R	E	N	D	O		P	Y	T	H	O	N	X	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

frase.rstrip() e se fosse frase.lstrip()

X	X	X	A	P	R	E	N	D	O		P	Y	T	H	O	N	X	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Vale lembrar que o length vai ser modificado com as utilizações da strip()

---

## Operação de Divisão de uma String

### Split()

O split pode receber argumentos também. Ele vai observar onde existem espaços na cadeia de string e “dividirá” essa string em várias strings sem espaço. Cada palavra recebe novas indexações e são colocadas em novas listas. E o melhor, cada uma delas, recebe um valor dentro dessa lista. Por exemplo, a palavra CURSO recebe a posição 0, a palavra EM recebe a posição 1, a palavra VIDEO recebe a posição 2 e a palavra PYTHON recebe a posição 3. Ele divide elementos na lista e cria.

frase =

C	U	R	S	O		E	M		V	I	D	E	O		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Resultado:

C	U	R	S	O
0	1	2	3	4

Posição 0

E	M
0	1

Posição 1

V	I	D	E	O
0	1	2	3	4

Posição 2

P	Y	T	H	O	N
0	1	2	3	4	5

Posição 3

## Operação de Junção de uma String

### Join()

O join é a forma análoga da split(), sendo que uma vez que a split quebra uma string em várias, a join reúne essa quebra

```
'-'.join(frase)
```

Se lê, junte todos aqueles elementos quebrados da frase e coloque um “-“ entre os caracteres.

C	U	R	S	O	-	E	M	-	V	I	D	E	O	-	P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Caso eu deseje um espaço em branco ao invés do traço, eu poderia deixar:

```
'.join(frase).
```

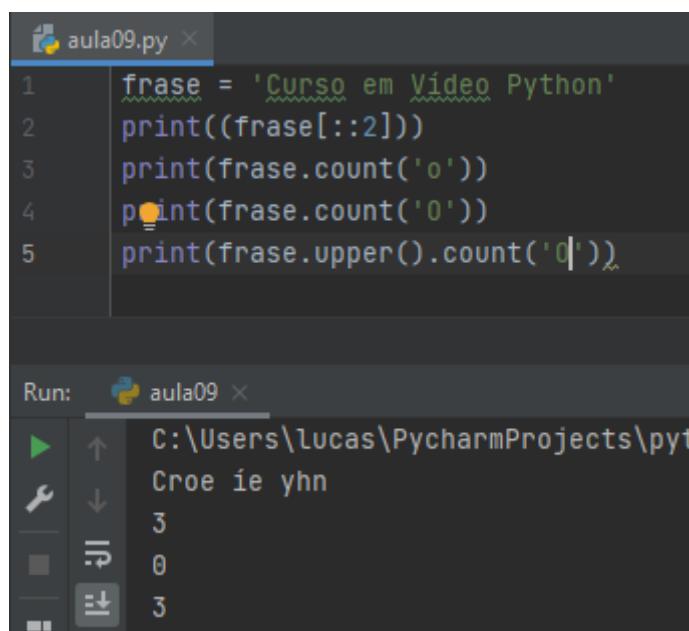
---

## Observações sobre Strings

Como em Python, tudo é objeto, eu posso utilizar(.) para acionar um método a uma variável, portanto, posso utilizar todos os métodos acima.

Note que na linha de código 4, eu mandei contar quantos O maiúsculos existiam e não existiam nenhum.

Na linha 5 eu primeiro mandei a frase para todos os elementos maiúsculos e depois pedi para contar os O's maiúsculos e ai ele encontrou 3.



The screenshot shows a PyCharm interface with two panes. The top pane contains the code for 'aula09.py':

```
1 frase = 'Curso em Vídeo Python'
2 print(frase[::-2])
3 print(frase.count('o'))
4 print(frase.count('O'))
5 print(frase.upper().count('O'))
```

The bottom pane shows the run output for 'aula09':

```
Run: aula09
C:\Users\lucas\PycharmProjects\pyt
Croe ie yhn
3
0
3
```

```
frase = 'Curso em Vídeo Python'
print(len(frase))
```

```
frase = 'Curso em Vídeo Python'
print(frase.replace('Python', 'Android'))
```

---

### Erro de atribuição de string

Sabe-se que a posição 0 dessa frase, é a letra 'C', eu não posso fazer atribuição diretamente assim, atribuindo diretamente. É preciso fazer o replace.

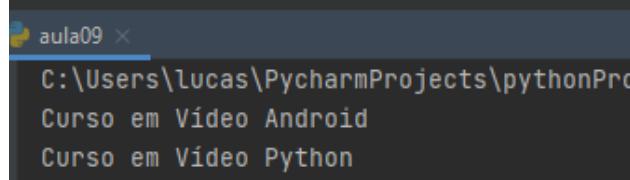
```
frase = 'Curso em Vídeo Python'
frase[0] = 'J'
print(frase)
```

```
aula09 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/P
Traceback (most recent call last):
  File "C:/Users/lucas/PycharmProjects/pythonProject/aula09.py", line 2, in <module>
    frase[0] = 'J'
TypeError: 'str' object does not support item assignment
```

## Replace na raíz da String

Abaixo, vale lembrar que uma string é imutável, veja que eu utilizando o replace, ele apenas substitui naquela instância, mas a cadeia de caracteres não foi alterada na “raiz”, continua valendo o Python.

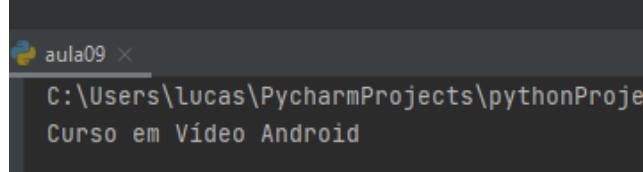
```
frase = 'Curso em Vídeo Python'
print(frase.replace('Python', 'Android'))
print(frase)
```



Para atribuir uma mudança na string, eu preciso agir:

Preciso atribuir a variável onde a string foi armazenada, esse método agindo na raiz da variável, e ai sim, ela substituirá.

```
frase = 'Curso em Vídeo Python'
frase = frase.replace('Python', 'Android')
print(frase)
```

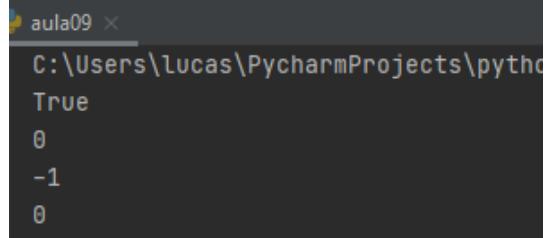


---

Note que se eu tentar achar um valor que não existe, o Curso está em maiúsculo e encontrou e o outro que procurei por curso, retornou o valor -1. Enquanto eu fizer as combinações por exemplo e chamar todo mundo da variável para minúsculo e ai sim ele

```
print(frase.find('Curso'))
print(frase.find('curso'))
print(frase.lower().find('curso'))
```

retorna a posição 0 que é onde o curso começa.

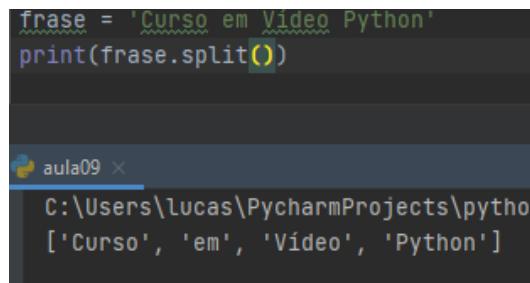


```
True
0
-1
0
```

## Split

Vale lembrar que em Python as listas são representadas por colchetes [ ' valor1, valor2 ' ]

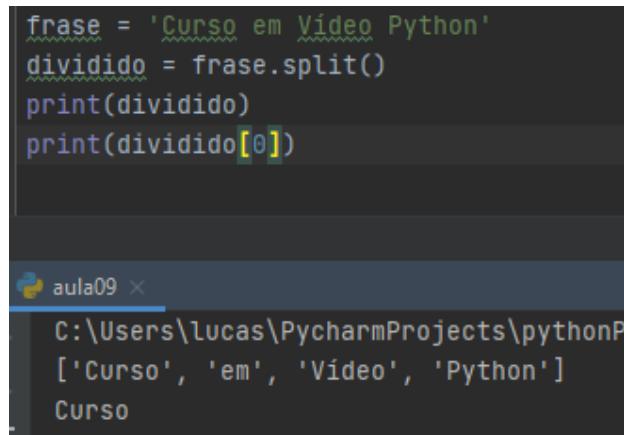
```
frase = 'Curso em Vídeo Python'
print(frase.split())
```



Veja como a frase foi quebrada, splitada. Para fazer essa quebra uma coisa nova, eu preciso atribuir uma variável (nesse caso eu criei uma var nova, mas poderia ter usado na própria frase).

E eu posso mandar imprimir essa variável splitada na posição zero, que como ela foi quebrada, o valor que se assumiu na posição zero foi o Curso.

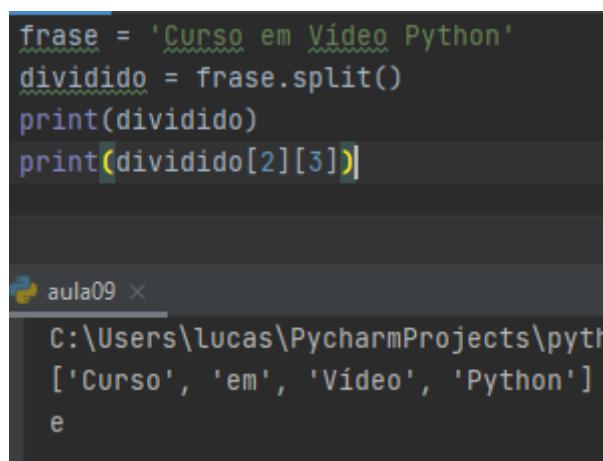
```
frase = 'Curso em Vídeo Python'
dividido = frase.split()
print(dividido)
print(dividido[0])
```



Em comparação, eu também posso pegar essa variável splitada, por exemplo, se lê: na splitada, pega a posição [2] e nessa posição, me mostra o que está na posição [3], que seria o carácter 'e'. Poderia fazer um [2][3:5] também para verificar um intervalo.

Existem inúmeras combinações possíveis.

```
frase = 'Curso em Vídeo Python'
dividido = frase.split()
print(dividido)
print(dividido[2][3])
```



## Estruturas Condicionais – IF/ ELSE

Em Python, a estrutura condicional é semelhante a todas as outras linguagens de programação, porém, existem alguns detalhes adicionais no Python que não estão presentes em outras linguagens:

### If - Else

Para que a estrutura IF seja executada, ela precisa estar:

*If + condição + sinal de dois pontos (:)*

*bloco Verdadeiro*

*else:*

*bloco Falso*

Exemplo 1:

```
if x > y:  
    bloco True (verdadeiro)  
else:  
    bloco False (falso)
```

Exemplo 2:

```
tempo = int (input ('Quanto tempo tem seu carro?'))  
if tempo <= 3:  
    print ('Seu carro é novo')  
else:  
    print ('Seu carro é velho')  
print ('—FIM—')
```

---

### Condisional If-Else Simplificada

Em Python não existe operador ternário! Portanto, pode usar em uma única linha de código, mas precisa ser uma estrutura simples.

```
tempo = int (input ('Quanto tempo tem seu carro?'))  
print ('carro novo' if tempo <= 3 else 'carro velho')  
print ('—FIM—')
```

Vale lembrar, que ele executará em condição se algo for verdadeiro, como pode ver abaixo, se o nome digitado não for ‘Lucas’, ele continuará executando, mas não executa a linha de código onde fala que o nome é bonito. Ele apenas printa o ‘Bom dia + nome’.

```
nome = str(input('Qual seu nome? '))
if nome == 'Lucas':
    print('Que nome lindo você tem!')
print('Bom dia {}!'.format(nome))

aula10 ×
C:\Users\lucas\PycharmProjects\pythonProject\src\aula10.py
Qual seu nome? Lucas
Que nome lindo você tem!
Bom dia Lucas!

C:\Users\lucas\PycharmProjects\pythonProject\src\aula10.py
Qual seu nome? Laura
Bom dia Laura!
```

Com a estrutura if – else ativa:

```
nome = str(input('Qual seu nome? '))
if nome == 'Lucas':
    print('Que nome lindo você tem!')
else:
    print('Seu nome não é tão bonito assim!')
print('Bom dia {}!'.format(nome))

aula10 ×
C:\Users\lucas\PycharmProjects\pythonProject\src\aula10.py
Qual seu nome? Tony
Seu nome não é tão bonito assim!
Bom dia Tony!
```

```
n1 = float(input('Digite a primeira nota: '))
n2 = float(input('Digite a segunda nota: '))
media = (n1 + n2)/2
print('Sua média foi de {}'.format(media))
if media >= 6.0:
    print('Sua média foi boa, parabéns!')
else:
    print('Sua nota foi baixa, estude mais.')

aula10 ×
C:\Users\lucas\PycharmProjects\pythonProject\src\aula10.py
Digite a primeira nota: 5
Digite a segunda nota: 10
Sua média foi de 7.5
Sua média foi boa, parabéns!
```

## Condições Aninhadas - Elif

Em Python e em outras linguagens de programação, podemos criar estruturas condicionais dentro de estruturas condicionais, esse método se chama “aninhar”, ou “criar condições aninhadas”.

Quando se tem duas opções, Sim ou Falso, virar o carro pra direita ou virar o carro pra esquerda, essas situações são assistidas pelo IF-ELSE. Porém, se eu precisar de uma terceira opção, um Sim dentro de outro Sim, um seguir reto com o carro ou virar à direita ou virar à esquerda. Quando se existem mais de 3 condições, o IF-ELSE já não me atende completamente.

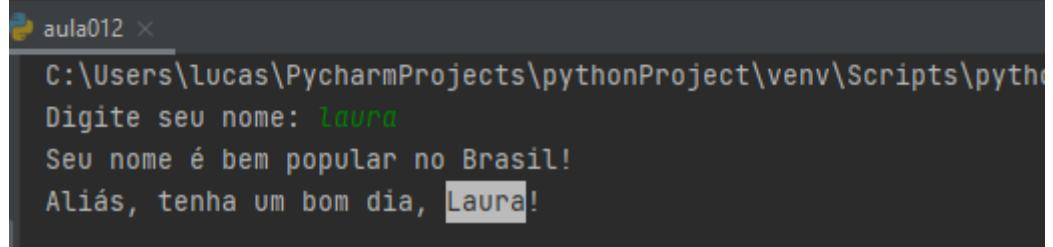
A estrutura aninhada tem o seguinte modelo:

```
If alguma condição:  
    bloco1;  
elif alguma outra condição:  
    bloco2;  
elif alguma outra condição:  
    bloco3;  
else:  
    bloco4;
```

É possível ter um “IF” sem um “ELSE”, mas nunca um “ELSE” sem pelo menos um “IF”.

Outro ponto de observação, é que é possível ter vários e quantos “ELIF” forem necessários para o programa executar.

```
nome = str(input('Digite seu nome: ')).title().strip()  
if nome == 'Lucas':  
    print('Seu nome é o mais bonito!')  
elif nome == 'Laura' or nome == 'Tony' or nome == 'Nina':  
    print('Seu nome é bem popular no Brasil!')  
else:  
    print('Seu nome é comum.')  
print('Aliás, tenha um bom dia, \033[7m{}\033[m!'.format(nome))
```



The screenshot shows the PyCharm terminal window titled 'aula012'. The command line shows the path: C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe. The user types 'Digite seu nome: Laura' and presses Enter. The program then prints three messages: 'Seu nome é bem popular no Brasil!', followed by 'Aliás, tenha um bom dia, Laura!'.

É possível utilizar o ‘in’ para que ele leia nomes ao invés de ficar buscando comparar a variável com algo.

```
nome = str(input('Digite seu nome: ')).title().strip()
if nome == 'Lucas':
    print('Seu nome é o mais bonito!')
elif nome in 'Laura Tony Nina':
    print('Seu nome é bem popular no Brasil!')
else:
    print('Seu nome é comum.')
print('Aliás, tenha um bom dia, \033[7m{}\033[m!'.format(nome))
```

```
elif nome in 'Laura Tony Nina'
```

```
aula012 ×
```

```
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\pyth
Digite seu nome: tony
Seu nome é bem popular no Brasil!
Aliás, tenha um bom dia, Tony!
```

## If/Else dentro de uma Função (Avançado)

```
def fatorial(num=1, show=False):
    num = int(input('Digite um número: '))
    i = 1
    for c in range(num, 0, -1):
        i *= c
    if show == True:
        for c in range(num, 0, -1):
            print(f'{c}{" x " if c > 1 else " = "}', end=' ')
    return i

print(fatorial(5, show=True))

fatorial() → for c in range(num, 0, -1)

```

ex102 ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\pyth  
Digite um número: 6  
6 x 5 x 4 x 3 x 2 x 1 = 720

# Estrutura Condicional de Laço de Repetição FOR

## Laço de Repetição For

O Laço de Repetição/ Iteração FOR, é um laço de variável de controle, pois dado a ele a estrutura, é necessário impor um limite de algo, por exemplo, um contador.

Por exemplo, uma linha com 10 espaços, onde o personagem está na posição 0 e precisa chegar na 10 e pegar uma maçã.

Estrutura WHILE	Estrutura FOR
Não se sabe quantas x vezes será executado	Se sabe exatamente quantas vezes
Não tem limitação	Tem limites
Enquanto (se algo True or False)	Para cada vez que algo acontecer

Laço contador No intervalo (1,10)

    darPasso()

pegarMaça #esse comando está do lado de fora do laço, só foi executado após sair do laço.

Em Python:

```
For contandor in range (1,10):
```

```
    darPasso
```

```
    pegarMaça
```

## Laço For Aninhado

É possível inserir um IF dentro de um For

Laço contador no intervalo (1,10)

    se condição:

```
        pegarMaça
```

```
    sairdoLaço
```

```
For contandor in range (0,3):
```

```
    if condição:
```

```
        pegarMaça
```

```
        darPasso
```

```
        darPulo
```

```
    sairDoLaço
```

```
print('Oi')
print('Oi')
print('Oi')
print('Oi')
print('Oi')
print('Oi')
```

Dado uma execução de um programa, quero executar um comando seis vezes. Por exemplo, mandar dar um 'Oi' seis vezes:

Mas e se eu precisasse imprimir esse programa, dez mil vezes?

Seria uma trabalheira muito grande para contar quantas vezes, para copiar e colar, parar saber se realmente foi dito dez mil vezes ‘Oi’.

Para isso, é indicado utilizar o laço de repetição e controle FOR.

Podemos, portanto, utilizar a estrutura abaixo:

Cuidado com a indentação, pois para fazer parte da estrutura do laço, precisar estar

```
for c in range(1, 6):
    print('Oi')
print('FIM')
```

```
aula013 ×
C:\Users\lucas\PycharmProjects\aula013>
Oi
Oi
Oi
Oi
Oi
FIM
```

indentado dentro desse laço, como o comando ‘Fim’ está fora da indentação, ele será executado somente quando a contagem ‘Oi’ acabarem, como mostra a figura a esquerda.

Porém, se a indentação do comando ‘Fim’ estiver dentro do laço, como mostra a figura a direita, o laço repetirá sempre ‘Oi’ e ‘Fim’ juntos.

```
for c in range(1, 6):
    print('Oi')
    print('FIM')
```

```
aula013 ×
C:\Users\lucas\PycharmProjects\aula013>
Oi
FIM
Oi
FIM
Oi
FIM
Oi
FIM
Oi
FIM
```

```
for c in range(1, 6):
    print(c)
print('FIM')
```

```
aula013 ×
C:\Users\lucas\PycharmProjects\aula013>
1
2
3
4
5
FIM
```

Vale lembrar que em Python, os laços começam na posição normal, se selecionar 0 ele vai contar a partir do zero, se selecionar 1, ele conta a partir do um.

Como nesse caso, foi pedido para que o Python contasse de 1 até 6, ele começou no 1 e parou antes do 6, o último ele “ignora”.

Se eu quisesse que o 6 aparecesse, eu deveria contar ou de 0 a 6 ou de 1 a 7.

## Laço For com Contagem Regressiva/ Negativa

Suponhamos que eu deseje contar de 6 até 0, utilizando a estrutura acima, o programa compilaria, mas não mostraria nada.

The image displays four screenshots of the PyCharm IDE interface, each showing a code editor and a terminal window. The code in the editors is as follows:

- Top Left:** `for c in range(2, 10, 2):  
 print(c)  
print('FIM')`. The terminal shows:  
C:\Users\lucas\PycharmProjects  
2  
4  
6  
8  
FIM
- Top Right:** `for c in range(6, 0, -2):  
 print(c)  
print('FIM')`. The terminal shows:  
C:\Users\lucas\PycharmProjects  
6  
4  
2  
FIM
- Bottom Left:** `for c in range(6, 0, -1):  
 print(c)  
print('FIM')`. The terminal shows:  
C:\Users\lucas\PycharmProjects  
6  
5  
4  
3  
2  
1  
FIM
- Bottom Right:** `for c in range(6, 0):  
 print(c)  
print('FIM')`. The terminal shows:  
C:\Users\lucas\PycharmProjects  
FIM

### Iteração – Intervalo da Repetição

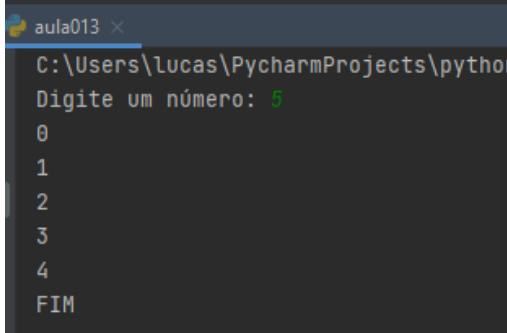
Para corrigir esse problema, precisamos dizer a linguagem que terá uma Iteração (qual a repetição, quantas vezes algo acontecerá).

Foi utilizado uma vírgula após o range, ele também pode ser considerado o intervalo que algo acontecerá. Também **positivo**, quanto o **negativo**.

## Recebendo um Input como parâmetro para o For

É possível solicitar ao usuário que ele digite um número e esse número seja utilizado como parâmetro para o laço de repetição:

```
n = int(input('Digite um número: '))
for c in range(0, n):
    print(c)
print('FIM')
```



```
C:\Users\lucas\PycharmProjects\pythonPro
Digite um número: 5
0
1
2
3
4
FIM
```

```
i = int(input('Digite um inicio: '))
f = int(input('Digite um fim: '))
p = int(input('Digite um intervalo: '))
for c in range(i, f+1, p):
    print(c)
print('FIM')

for c in range(i, f+1, p)
aula013 ×
C:\Users\lucas\PycharmProjects\pythonPro
Digite um inicio: 0
Digite um fim: 10
Digite um intervalo: 2
0
2
4
6
8
10
FIM
```

```
i = int(input('Digite um inicio: '))
f = int(input('Digite um fim: '))
p = int(input('Digite um intervalo: '))
for c in range(i, f+1, p):
    print(c)
print('FIM')

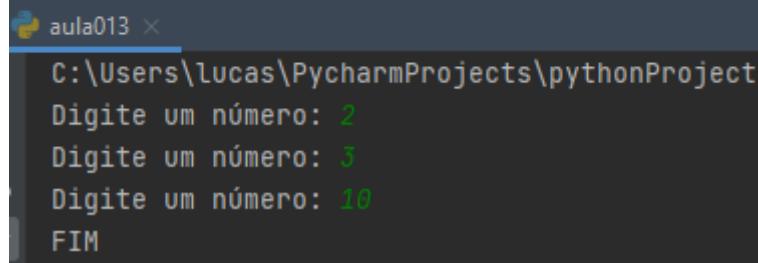
aula013 ×
C:\Users\lucas\PycharmProjects\pythonPro
Digite um inicio: 0
Digite um fim: 9
Digite um intervalo: 2
0
2
4
6
8
FIM
```

Quando você insere um limite, se o próximo intervalo dele ultrapassar esse limite, ele vai parar no anterior e não vai mostrar o próximo, como pode ser visto na imagem a direita.

## Leitura Única de Dados dentro do FOR

Para deixar o código DRY, eu posso solicitar ao usuário que ele digite um número dentro do for para que ele leia um número x vezes, sem ter a necessidade de escrever a mesma linha de código várias e várias vezes.

```
for c in range(0, 3):
    numero = int(input('Digite um número: '))
print('FIM')
```

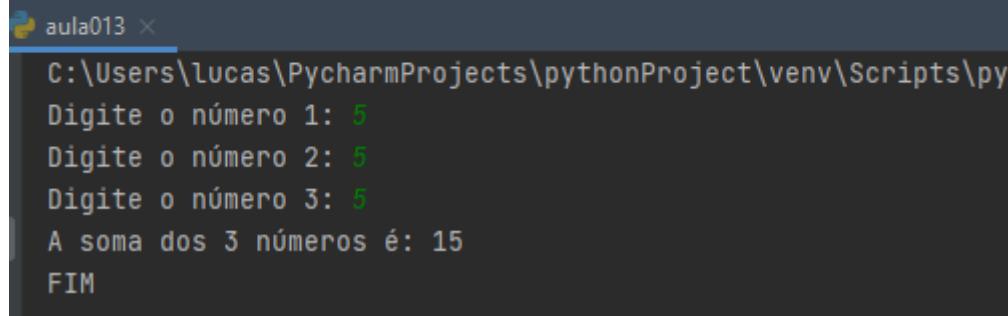


```
aula013 ×
C:\Users\lucas\PycharmProjects\pythonProject
Digite um número: 2
Digite um número: 3
Digite um número: 10
FIM
```

O programa solicita ao usuário para que ele digite um número x vezes necessárias conforme o range dado no laço. Nesse exemplo, se esse laço vai acontecer 3 vezes, então ele vai pedir o número 3 vezes. Se esse laço acontecer 10 mil vezes, ele vai pedir ao usuário que digite 10 mil vezes os números antes de sair do laço de repetição.

O exemplo abaixo mostra uma soma acumulada, dado uma variável global (fora do laço) e que ela será somada sempre dentro do laço, quando for solicitada fora do laço, ela continuará sendo apresentada com o valor que nela foi alterado.

```
soma = 0
for c in range(0, 3):
    numero = int(input('Digite o número {}: '.format(c + 1)))
    soma += numero
print('A soma dos {} números é: {}'.format(c + 1, soma))
print('FIM')
```



```
aula013 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\py
Digite o número 1: 5
Digite o número 2: 5
Digite o número 3: 5
A soma dos 3 números é: 15
FIM
```

## Estrutura Condicional de Laço de Repetição While

O WHILE como estrutura de repetição é muito utilizado para quando NÃO se sabe exatamente quantos passos, não se sabe exatamente quantas vezes algo será executado.

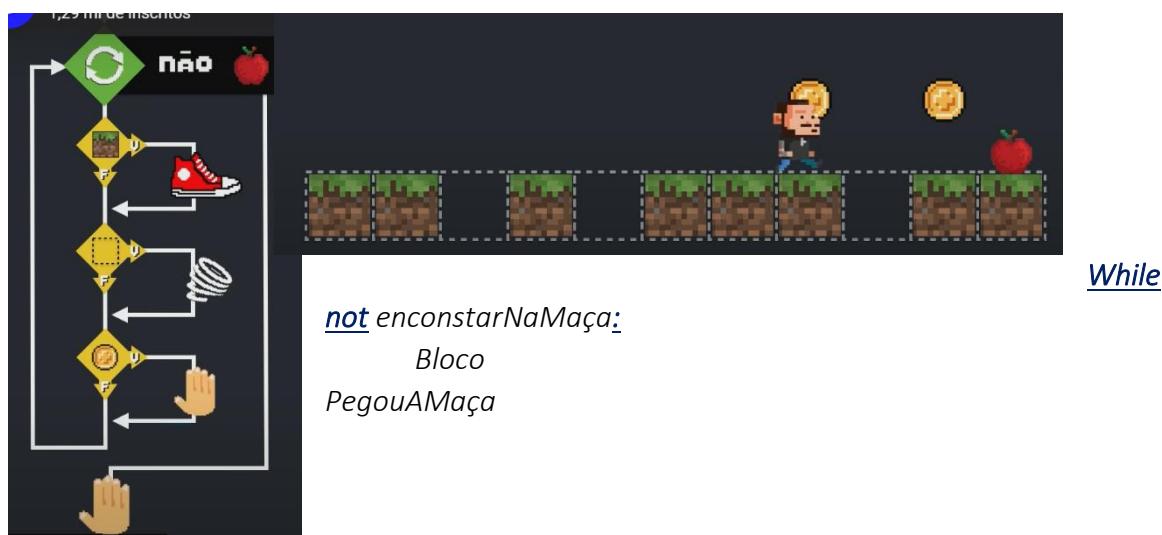
Enquanto a estrutura FOR serve justamente para o contrário, quando se sabe exatamente quantas vezes algo será executado.

Estrutura WHILE	Estrutura FOR
Não se sabe quantas x vezes será executado	Se sabe exatamente quantas vezes
Não sei o Limite ou Quando sei o Limite	Preciso saber qual o limite
Enquanto (se algo True or False)	Para cada vez que algo acontecer

Enquanto não encostarNaMaça

Bloco

PegouMaça



O bonequinho vai andar sempre conforme o laço e as perguntas feitas durante a execução do laço e os If's lá escritos. Por exemplo:

"Tem chão para andar"? Se sim, anda pra frente, se não, ele vai executar um código If de "Tem buraco?" Se sim, pula um buraco. Aí vai ter outro teste lógico, como o "Tem moeda?". Se sim, pega a moeda. O laço **While** vai terminar a sua execução quando ele encostar na maçã.

Aparentemente, utilizar o *FOR* e o *WHILE* como no exemplo abaixo pode demonstrar um mesmo resultado.

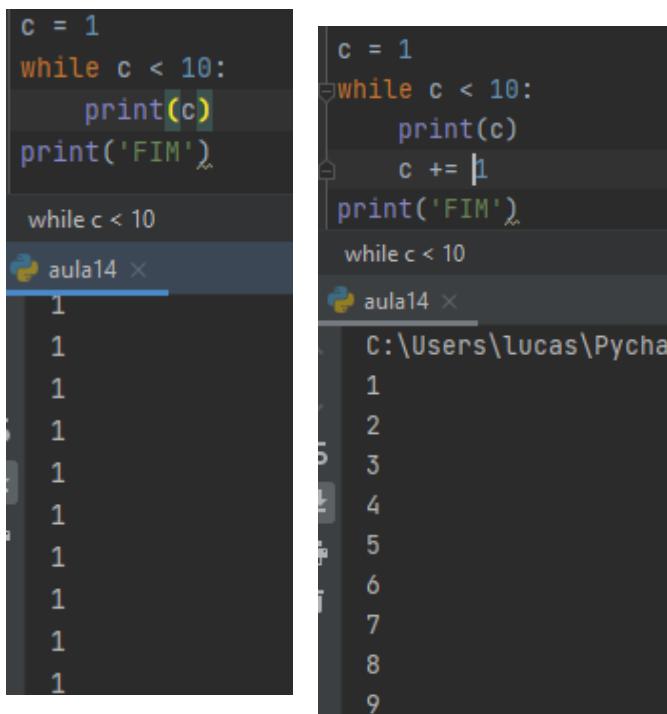
```
"""for c in range(1, 10):
    print(c)
print('FIM')"""

c = 1
```

```
while c < 10:
    print(c)
print('FIM')
```

## Loop e While Infinitos

O laço while necessita de um contador que adiciona sempre a variável procurada um valor, pois, se não, criará um loop infinito dentro do while. Como abaixo a esquerda e a solução a direita.



```
c = 1
while c < 10:
    print(c)
print('FIM')

while c < 10
```

aula14 ×

```
1
1
1
1
1
1
1
1
1
1
```

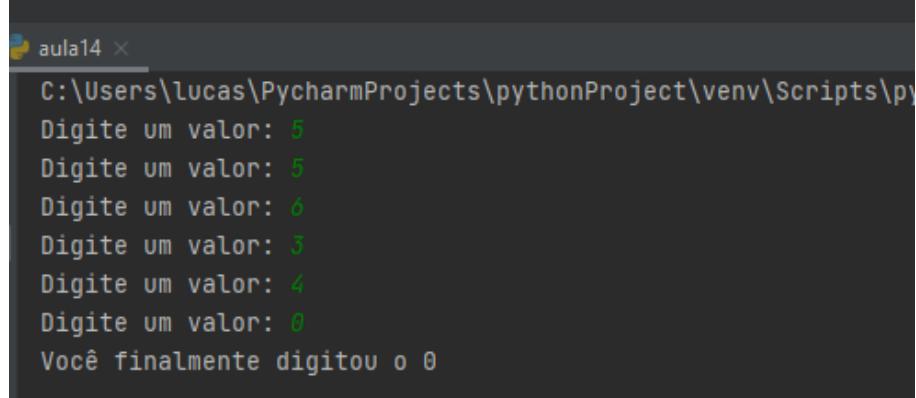
aula14 ×

```
c:\Users\lucas\Pycha
1
2
3
4
5
6
7
8
9
```

## Flag – Condição de Parada com While

Utilizando o While para interromper a execução de algo até que algum número seja finalmente inserido.

```
n = 1
while n != 0:
    n = int(input('Digite um valor: '))
print('Você finalmente digitou o {}, obrigado!'.format(n))
```

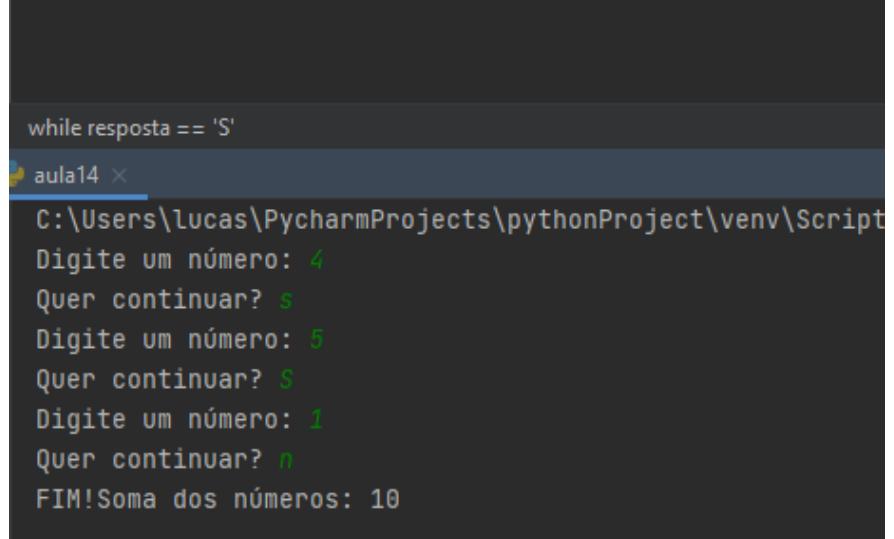
```
Digite um valor: 5
Digite um valor: 5
Digite um valor: 6
Digite um valor: 3
Digite um valor: 4
Digite um valor: 0
Você finalmente digitou o 0
```

Cuidado ao manipular esse tipo de problema, pois é necessário muitas vezes se atentar ao usuário digitando um “s” minúsculo e interrompendo a execução do programa!

É possível fazer contagem dos números.

```
soma = 0
resposta = 'S'
while resposta == 'S':
    n = int(input('Digite um número: '))
    resposta = str(input('Quer continuar?')).upper()
    soma += n
print('FIM!Soma dos números: {}'.format(soma))

while resposta == 'S'
aula14 ×
```

```
Digite um número: 4
Quer continuar? s
Digite um número: 5
Quer continuar? S
Digite um número: 1
Quer continuar? n
FIM!Soma dos números: 10
```

## Exemplo 2 de Flag

Como é mostrado, se o usuário digitar 999 ele vai encerrar o programa, só que é preciso fazer o WHILE diferente do 999. É necessário pedir o número e verificar se ele é igual a 999 antes de fazer as somas e os contadores de quantas vezes algo foi feito.

Se ele ler o número e verifica que é 999 dentro do Laço IF-ELSE, ele não vai fazer a soma e as quantidades, e vai encerrar o programa.

```
from time import sleep
numero = soma = count = 0

while numero != 999:
    numero = int(input('Digite um número: [999 Encerra o programa]: '))
    if numero != 999:
        soma += numero
        count += 1
    else:
        print('Encerrando o programa...')
        sleep(1)

print('A soma entre os números: {}'
      '\nA quantidade de números: {}'.format(soma, count))
```

```
ex064 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/
Digite um número: 10
Digite um número: 10
Digite um número: 10
Digite um número: 20
Digite um número: 999
Encerrando o programa...
A soma entre os números: 50
A quantidade de números: 4
```

## Contando números Pares e Ímpares

Esse exercício é um exercício muito interessante pelo fato que como pode notar, foi digitado 2 números pares e 2 números ímpares. Mas o 0 (zero) que foi digitado como um “finalizador” do laço, ele foi lido e analisado como número par.

```
#Calculando par e ímpar:

n = 1
par = impar = 0
while n != 0:
    n = int(input('Digite um número: '))
    if n % 2 == 0:
        par += 1
    else:
        impar += 1
print('Você digitou {} números pares e'
      '\n{} números ímpares.'.format(par, impar))
```

```
aula14 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:/Users/lucas/PycharmProjects/pythonProject/aula14.py"
Digite um número: 2
Digite um número: 3
Digite um número: 4
Digite um número: 1
Digite um número: 0
Você digitou 3 números pares e
2 números ímpares.
```

Para corrigir isso, é necessário criar um outro laço IF indicando que somente se um número for diferente de zero que ai sim, esse contador será iniciado.

```
n = 1
par = impar = 0
while n != 0:
    n = int(input('Digite um número: '))
    if n != 0:
        if n % 2 == 0:
            par += 1
        else:
            impar += 1
print('Você digitou {} números pares e'
      '\n{} números ímpares.'.format(par, impar))
```

```
aula14 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:/Users/lucas/PycharmProjects/pythonProject/aula14.py"
Digite um número: 2
Digite um número: 2
Digite um número: 3
Digite um número: 3
Digite um número: 0
Você digitou 2 números pares e
2 números ímpares.
```

## Not in em While

Em Python, a estrutura de repetição While, se desejar utilizar somente uma “condicional” como critério, por exemplo, se algo for diferente de zero, basta utilizar while numero != 0: e entrará na condição “simples” do While.

Entretanto, em Python, para utilizar uma condição mais complexa que onde vários limitadores podem iniciara condição, devemos utilizar o ‘not in’.

Caso se deseje encontrar um sexo de uma pessoa através de um valor digitado.

Errado: ~~while sexo != 'M' or 'F'~~

Correto: `while sexo not in 'MmFf'`

A forma correta em Python não admite que utilize a estrutura da condição OU/OR.

```
sexo = str(input('Digite o sexo da pessoa: ')).strip().upper()[0]

while sexo not in 'MmFm':
    sexo = str(input('Inválido! Digite o sexo da pessoa: ')).strip().upper()[0]
print('O sexo digitado foi: {}'.format(sexo))

while sexo not in 'MmFm'
ex057 x
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/luc
Digite o sexo da pessoa: p
Inválido! Digite o sexo da pessoa: p
Inválido! Digite o sexo da pessoa: y
Inválido! Digite o sexo da pessoa: v
Inválido! Digite o sexo da pessoa: r
Inválido! Digite o sexo da pessoa: m
O sexo digitado foi: M.
```

## Maior e Menor dentro com While

Para fazer uma comparação sem utilizar uma lista, que seria o melhor jeito de resolver essa solicitação, imagine que em uma primeira “rodada/passada” no laço, o número digitado vai ser único, pois se o número “4” foi digitado pela primeira vez, ele é considerado o maior e também o menor número digitado até o momento.

Ao passar dos números, deve ser feita a comparação para saber se o número digitado foi maior ou menor do que a própria variável que está gravada, se uma das condições for verdadeira, então esse novo número digitado tomará o lugar do número “4” digitado. Por exemplo:

**número = 4**

maior = 4 e menor = 4

**número = 3**

maior = 4 e menor = 3

```
continuar = 's'
contar = 0
qtdNumeros = qtd = somar = media = maior = menor = 0

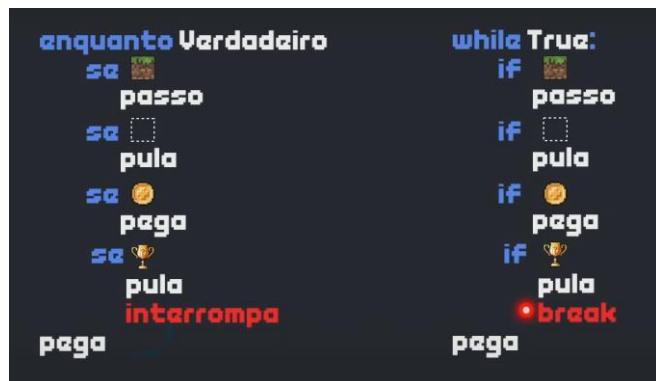
while continuar != 'N':
    contador = int(input('Quantos números deseja: '))
    while contador > contar:
        numero = int(input('Digite um número({}): '.format(contar + 1)))
        contar += 1
        somar += numero
        qtd += 1
        if qtd == 1:
            maior = menor = numero
        else:
            if numero > maior:
                maior = numero
            else:
                menor = numero
        media = somar
        qtdNumeros += contador
    print('A média entre os valores: {:.2f}'
          '\nO maior valor lido: {}'
          '\nO menor valor lido: {}'.format(media / qtdNumeros, maior, menor))
    continuar = str(input('Deseja continuar? [s/n]')).strip().upper()
while continuar != 'N'
```

## Interrompendo Repetições no While com Break

Como nas aulas, o personagem quer dar um passo pra frente pra pegar a Maçã, só que existem condicionais que impedem/ ajudam ele a realizar algo. Por exemplo, se houver um espaço ele anda, senão houver, ele pula, se houver moeda, ele pega.

Só que nesse exemplo, não existia um objeto que quando ele encontrar acabou o laço, não. O laço é eterno, veja, While TRUE, significa que vai executar para sempre. Porém, se ele encontrar um “troféu” no meio do caminho, ele vai interromper o código e vai jogar para fora do laço While e parando o programa.

Esse comando que interrompe um While “eterno” é o [Break](#).



Loop Infinito

```
contador = 0

#utilizando o while "Padrão"
"""while contador <= 10:
    print(contador, '...', end='')
    contador += 1
print('Fim.')"""

#se remover o contador e deixar o True
# algo será executado para sempre:
while True:
    print(contador, '...', end='')
    contador += 1
print('Não tem fim!')
```

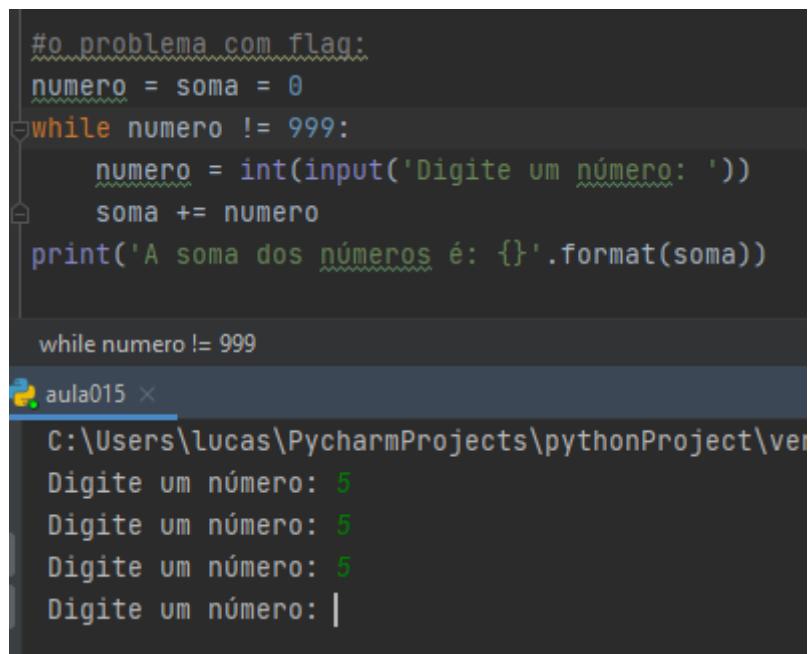
```
aula015 ×
...112997 ...112998 ...112999 ...113000 ...113001 ...113002 ...113003 ...113004 ...1130
...113012 ...113013 ...113014 ...113015 ...113016 ...113017 ...113018 ...113019 ...1130
...113027 ...113028 ...113029 ...113030 ...113031 ...113032 ...113033 ...113034 ...1130
...113042 ...113043 ...113044 ...113045 ...113046 ...113047 ...113048 ...113049 ...1130
...113057 ...113058 ...113059 ...113060 ...113061 ...113062 ...113063 ...113064 ...1130
...113072 ...113073 ...113074 ...113075 ...113076 ...113077 ...113078 ...113079 ...1130
...113087 ...113088 ...113089 ...113090 ...113091 ...113092 ...113093 ...113094 ...1130
...113102 ...113103 ...113104 ...113105 ...113106 ...113107 ...113108 ...113109 ...1131
...113117

Process finished with exit code -1
```

Veja que se o código While True estiver no código, ele vai executar uma sequência/loop infinita(o).

## Corrigindo o problema do Flag

O flag que é um “grande salvador”, onde significa que o programa não será mais executado, também pode gerar um problema.



```
#o problema com flag:
numero = soma = 0
while numero != 999:
    numero = int(input('Digite um número: '))
    soma += numero
print('A soma dos números é: {}'.format(soma))
```

```
while numero != 999
aula015 ×
C:\Users\lucas\PycharmProjects\pythonProject\ver
Digite um número: 5
Digite um número: 5
Digite um número: 5
Digite um número: |
```

É visto que esse programa está dizendo que enquanto o número for diferente de 999 ele vai pedir um número, vai somar esse número e somente quando for digitado 999, que ele interromperá o programa. Portanto, a soma que será mostrada no print deveria ser 15. Mas, não será, porque o programa pede um número e atribui a variável e utiliza a soma, somente aí, quando ele voltar ao início do laço e verificar que o número é 999, ele encontrará um problema.

Ao digitarmos o 999 para interromper o programa, ele será interrompido, somente após executar as condições de dentro do laço e então, a soma ao invés de ser 15, será:

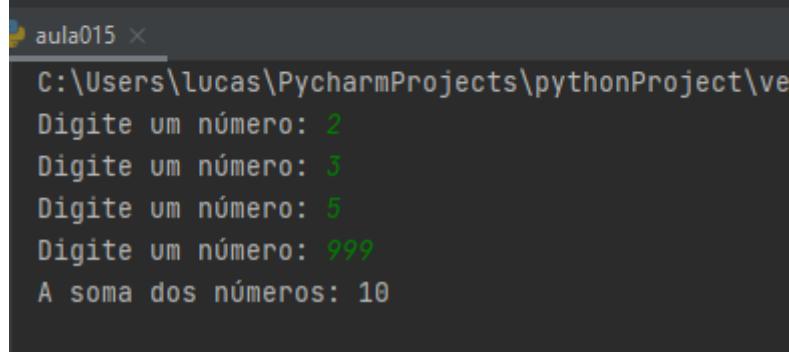
```
C:\Users\lucas\PycharmProjects\pyt
Digite um número: 5
Digite um número: 5
Digite um número: 5
Digite um número: 999
A soma dos números é: 1014
```

Como corrigir esse problema?

Como não deixar o flag nos atrapalhar e somente ser utilizado para interromper o programa.

É simples, basta antes da variável soma ser chamada/executada, eu interrompo o programa fazendo uma condição “If” para verificar se o número solicitado ao usuário será de ‘999’ e se for verdadeiro, terá um bloco Break que interromperá o programa na hora e não fará a soma do ‘999’ a variável soma.

```
#com loop infinito com o Break:  
numero = soma = 0  
while True:  
    numero = int(input('Digite um número: '))  
    if numero == 999:  
        break  
    soma += numero  
print('A soma dos números: {}'.format(soma))
```

```
aula015 ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv  
Digite um número: 2  
Digite um número: 3  
Digite um número: 5  
Digite um número: 999  
A soma dos números: 10
```

# PEP – Python Enhancement Proposal

Proposta de melhoria do Python, no vídeo que foi assistido em 2017, está na PEP 498, trata de ‘f-strings’. Estão disponíveis apenas a partir da versão 3.6 do Python.

Verifique pelo cmd:

```
C:\Users\lucas>python --version
Python 3.9.1
C:\Users\lucas>
```

## F-STRINGS

As ‘f-strings’ são utilizadas para “substituir” o .format no print, ou seja, não é mais necessário utilizar o .format para escrever algo na tela.

Podemos utilizar a sintaxe que antes das aspas simples, se insere um ‘f’ minúsculo e na máscara, podemos inserir diretamente o nome da variável que antigamente, utilizávamos no .format.

Ele serve para a interpolação ser mais produtiva e eficiente na digitação.

```
print('A soma dos números: {}'.format(soma))
```

```
print(f'A soma dos números: {soma}' )
```

```
#com loop infinito com o Break:
numero = soma = 0
while True:
    numero = int(input('Digite um número: '))
    if numero == 999:
        break
    soma += numero
#print('A soma dos números: {}'.format(soma))
print(f'A soma dos números: {soma}')
```

```
aula015 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\$
Digite um número: 5
Digite um número: 5
Digite um número: 5
Digite um número: 999
A soma dos números: 15
```

## Print nas versões do Python

Em cada versão do Python, o print foi se aperfeiçoando e melhorando, desde o Python 2 até a nova versão do Python 3.6 + onde podemos usar o ‘f-string’

```
idade = 33
nome = 'José'
print('O {} tem {} anos.')#Python 3.6+
print('O {} tem {} anos.'.format(nome, idade))#Python 3.0
print('O %s tem %d anos.' %(nome, idade))#Python 2
```

```
aula015 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\py
O José tem 33 anos.
O José tem 33 anos.
O José tem 33 anos.
```

Exemplos:

Ainda é possível fazer a formatação da máscara para mostrar as casas decimais, basta adicionar o ‘:**.2f**’ após terminar de digitar o nome da variável na máscara.

```
nome = 'Lucas'
idade = 26
salário = 15000.14
print('O {} tem {} e ganha o salário de R$ {:.2f}'.format(nome, idade, salário))
```

```
aula015 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
O Lucas tem 26 e ganha o salário de R$ 15000.14
```

Também é possível mexer e “brincar” com as strings:

Verificar o capítulo

```
nome = 'Lucas'
idade = 26
salário = 15000.14
print('O {:<20} tem {} e ganha o salário de R$ {:.2f}'.format(nome, idade, salário))
```

```
aula015 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users
O -----Lucas----- tem 26 e ganha o salário de R$ 15000.14
```

# Variáveis Compostas - Tuplas

Em Python existem 3 tipos de Variáveis compostos: Tuplas, Listas e Dicionários.

Quando se inicia uma variável em Python, ela “ocupa” um espaço na memória, por exemplo:

Lanche = hamburger

Hamburguer

Porém, se eu trocar por: Lanche = suco, o suco tomará o lugar do hamburguer e lanche “vale” suco, nesse momento.

Lanche = suco

suco

Até agora isso é uma variável simples. Mas, e se eu quiser “um Hamburguer, um suco, uma pizza e um pudim”?

Para isso, podemos utilizar variáveis compostas, ela pode guardar inúmeros dados, não só necessariamente 4 dados. Essas variáveis compostas podem ser utilizadas em Tuplas, Listas e Dicionários. Nesse capítulo, será abordado as Tuplas, nesse momento, uma variável composta que armazena vários valores/elementos é uma TUPLA. Que são identificados por índices.

Elementos	Hamburguer	Suco	Pizza	Pudim
Índice	0	1	2	3

Se executar o código: print(lanche) ele mostrará todos os elementos da tupla.

Se executar o código: print(lanche[2]), ele mostrará somente o elemento que está na posição 2, que é a pizza

## Fatiamento das Tuplas

### Fatiamento com Início e com Fim

Caso queira mostrar vários elementos da tupla, pode utilizar exemplos dos fatiamentos vistos no capítulo de fatiamento de strings.

Elementos	Hamburguer	Suco	Pizza	Pudim
Índice	0	1	2	3

Print(lanche [0:2])

Resultado: Hamburguer e Suco.

Sim, funciona exatamente igual ao fatiamento de string, o último elemento será desconsiderado. Contando somente do 0, 1 e para quando chega no dois, se eu quiser mostrar a Pizza, preciso contar até o seu próximo que é o elemento nº 3.

### Fatiamento com Início e sem Fim

Elementos	Hamburguer	Suco	Pizza	Pudim
Índice	0	1	2	3

`Print(lanche[1:])`

**Resultado:** suco, pizza, pudim

Esse comando printará algo que comece na posição 1 em diante.

Fatiamento com Início e retroagindo.

Para acessar elementos e retroagindo, literalmente para trás, usa-se o comando

Elementos	Hamburguer	Suco	Pizza	Pudim
Índice	0	1	2	3

`Print (lanche [-1])`

Resultado Pudim

`Print (lanche [-2])`

Resultado: Pizza

### Fatiamento com Length

Pode utilizar o comprimento, quantos elementos tem dentro do lanche.

Elementos	Hamburguer	Suco	Pizza	Pudim
Índice	0	1	2	3

`Len(lanche)`

Resultado: 4

### Laço FOR com TUPLAS

Elementos	Hamburguer	Suco	Pizza	Pudim
Índice	0	1	2	3

Pode utilizar o comando:

`for comida in lanche:`

`print (comida)`

Lê: Para cada comida/ elemento na variável lanche, printa a comida, printa cada elemento.

O resultado para o Python, será que para cada execução do FOR, será mostrado em cada linha um dos elementos que está na variável lanche.

### Imutáveis

As tuplas são imutáveis.

Isso significa que quando definida, ela não pode ser alterada. Por exemplo, não é possível alterar um item como o Pudim para um sorvete.

## Iniciando as Tuplas

3 tipos de inicialização de tuplas: (tupla) ou [lista] ou {dicionário}. No novo python pode tirar () .

```
lancheSimples = 'Hamburguer'
lancheSimples= 'Sucão'
lanche = ('Hamburguer', 'Sucão', 'Pizza', 'Pudim')

print(lancheSimples)
print(lanche)
```

```
aula16 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe -i "C:/Users/lucas/PycharmProjects/pythonProject/aula16.py"
Sucão
('Hamburguer', 'Sucão', 'Pizza', 'Pudim')
```

## Mostrando apenas um elemento da Tupla

Para isso, eu digo a variável e entre [aPosição], não é porque tupla se inicia com parênteses, embora no novo Python 3.5 não seja mais necessário o parêntese, para a chamada no print usamos o []:

```
lanche = ('Hamburguer', 'Sucão', 'Pizza', 'Pudim')

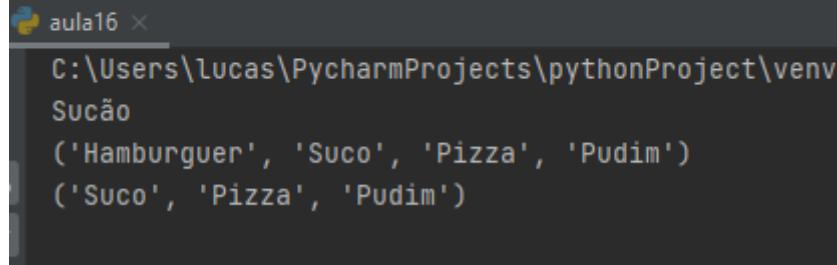
print(lancheSimples)
print(lanche)
print((lanche[1]))
```

```
aula16 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe -i "C:/Users/lucas/PycharmProjects/pythonProject/aula16.py"
Sucão
('Hamburguer', 'Sucão', 'Pizza', 'Pudim')
Sucão
```

## Mostrar a partir de um elemento da Tupla

```
print((lanche[1:]))
```



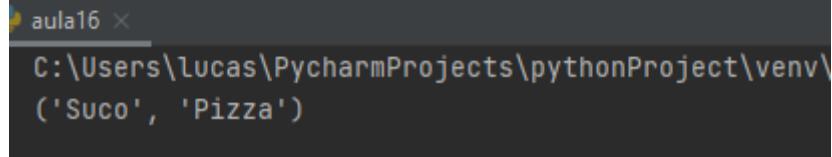
```
aula16 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\
Sucão
('Hamburguer', 'Suco', 'Pizza', 'Pudim')
('Suco', 'Pizza', 'Pudim')
```

## Mostrando Intervalo da Tupla

O comando abaixo mostrará começando pelo elemento 1 até o 3, onde o último será ignorado.

```
lanche = ('Hamburguer', 'Suco', 'Pizza', 'Pudim')

#print(lancheSimples)
#print(lanche)
print((lanche[1:3]))
#print(lanche[-1])
#print(lanche[-4])
```



```
aula16 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\
('Suco', 'Pizza')
```

## Mostrando do Início ao Fim da Tupla

Para executar o comando para mostrar algo do início a um fim dado, , deve-se usar o abaixo, lembrando que o Python ignora o último elemento, não contando o 2.

```
lanche = ('Hamburguer', 'Suco', 'Pizza', 'Pudim')

#print(lancheSimples)
#print(lanche)
#print((lanche[1:3]))
#print(lanche[-1])
#print(lanche[-4])
print(lanche[:2])
```

```
aula16 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:/Users/lucas/PycharmProjects/pythonProject/aula16.py"
('Hamburguer', 'Suco')
```

## Mostrando o inverso na tupla

Interessante que podemos pedir em forma negativa, o que vai mudar e inverter como os elementos serão mostrados na tela.

Elementos	Hamburguer	Suco	Pizza	Pudim
Índice	0	1	2	3
Negativo	-4	-3	-2	-1

```
lanche = ('Hamburguer', 'Suco', 'Pizza', 'Pudim')
💡
#print(lancheSimples)
#print(lanche)
#print((lanche[1:]))
print(lanche[-1])
print(lanche[-2])
```

```
aula16 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:/Users/lucas/PycharmProjects/pythonProject/aula16.py"
Pudim
Pizza
```

## Tuplas Imutáveis

Para verificar como é impossível mudar e modificar algum valor da tupla:

Não é possível atribuir valores a tupla após a inicialização dela. Somente no ato de iniciar uma tupla.

```
lanche = ('Hamburguer', 'Suco', 'Pizza', 'Pudim')

#print(lancheSimples)
#print(lanche)
#print(lanche[1:3])
#print(lanche[-1])
#print(lanche[-4])
#print(lanche[-2])

#Tuplas São Imutáveis:
lanche[1] = 'Refrigerante'
print(lanche[1])
```

```
aula16 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects\pythonProject\aula16.py
Traceback (most recent call last):
  File "C:/Users/lucas/PycharmProjects\pythonProject\aula16.py", line 15, in <module>
    lanche[1] = 'Refrigerante'
TypeError: 'tuple' object does not support item assignment
```

## Mostrando Tuplas com For

Demonstrando um exemplo com For, pode utilizar com While, com If-else..

```
lanche = ('Hamburguer', 'Suco', 'Pizza', 'Pudim')

for comida in lanche:
    print(f'Eu vou comer {comida}.')
print('Terminei de comer tudo.')
```

```
aula16 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects\pythonProject\aula16.py
Eu vou comer Hamburguer.
Eu vou comer Suco.
Eu vou comer Pizza.
Eu vou comer Pudim.
Terminei de comer tudo.

Process finished with exit code 0
```

## Mostrando Tupla com Length

A tupla pode ser alterada com o programa pausado. Utilizamos o len para demonstrar:

```
lanche = ('Hamburguer', 'Suco', 'Pizza', 'Pudim')

print(len(lanche))

aula16 ✘
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scr
4

lanche = ('Hamburguer', 'Suco', 'Pizza', 'Pudim', 'Batata Frita')

print(len(lanche))

aula16 ✘
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
5
```

## Mostrando Tuplas com For + Length

Como é mostrando no print abaixo, o hamburguer ocupa a posição 0, suco 1, pizza 2, pudim 3 e batata frita 4.

Por mais que o length identifique 5 elementos, vale lembrar que a posição começa no zero.

Podemos utilizar o length como um parâmetro para imprimir algo com o for.

```
lanche = ('Hamburguer', 'Suco', 'Pizza', 'Pudim', 'Batata Frita')

print(f'O tamanho da tupla de lanche: {len(lanche)}\n')

for comida in range(0, len(lanche)):
    print(f'Hoje eu comi {comida}')
print('Terminei de comer tudo e engordei.')
```

```
aula16 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
O tamanho da tupla de lanche: 5

Hoje eu comi 0
Hoje eu comi 1
Hoje eu comi 2
Hoje eu comi 3
Hoje eu comi 4
Terminei de comer tudo e engordei.
```

```
print(f'O tamanho da tupla de lanche: {len(lanche)}\n')

for comida in range(0, len(lanche)):
    print(f'Hoje eu comi {lanche[comida]}')
print('Terminei de comer tudo e engordei.')

for comida in range(0, len(lanche)):
    print(f'Hoje eu comi {lanche[comida]}')
print('Terminei de comer tudo e engordei.')

Hoje eu comi ('Hamburguer', 'Suco', 'Pizza', 'Pudim', 'Batata Frita')
Hoje eu comi ('Hamburguer', 'Suco', 'Pizza', 'Pudim', 'Batata Frita')
Hoje eu comi ('Hamburguer', 'Suco', 'Pizza', 'Pudim', 'Batata Frita')
Hoje eu comi ('Hamburguer', 'Suco', 'Pizza', 'Pudim', 'Batata Frita')
Hoje eu comi ('Hamburguer', 'Suco', 'Pizza', 'Pudim', 'Batata Frita')
Terminei de comer tudo e engordei.
```

## Mostrando Tupla com For através do Contador

Aparentemente, aquele problema acima não parece ter solução, não é? Pois se mandar imprimir a comida (que na verdade assume como um CONTADOR) ele mostra a posição de cada elemento na tupla. Se mandar imprimir o lanche (que na verdade é a TUPLA INTEIRA), ele mostrará toda a tupla em cada momento que realizar um print. Para isso, chamamos a tupla e em cada momento do contador, ele deverá realizar o print:

```
lanche = ('Hamburguer', 'Suco', 'Pizza', 'Pudim', 'Batata Frita')

print(f'O tamanho da tupla de lanche: {len(lanche)}\n')

for comida in range(0, len(lanche)):
    print(f'Hoje eu comi {lanche[comida]}')
print('Terminei de comer tudo e engordei.')

aula16 ✘
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
O tamanho da tupla de lanche: 5

Hoje eu comi Hamburguer
Hoje eu comi Suco
Hoje eu comi Pizza
Hoje eu comi Pudim
Hoje eu comi Batata Frita
Terminei de comer tudo e engordei.
```

```
lanche = ('Hamburguer', 'Suco', 'Pizza', 'Pudim', 'Batata Frita')

print(f'O tamanho da tupla de lanche: {len(lanche)}\n')

for comida in range(0, len(lanche)):
    print(f'Hoje eu comi {lanche[comida]} que é a posição {comida}')
print('Terminei de comer tudo e engordei.')

for comida in range(0, len(lanche)):
    print(f'Hoje eu comi {lanche[comida]} que é a posição {comida}')
print('Terminei de comer tudo e engordei.')

aula16 ✘
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/U...
O tamanho da tupla de lanche: 5

Hoje eu comi Hamburguer que é a posição 0
Hoje eu comi Suco que é a posição 1
Hoje eu comi Pizza que é a posição 2
Hoje eu comi Pudim que é a posição 3
Hoje eu comi Batata Frita que é a posição 4
Terminei de comer tudo e engordei.
```

## Mostrando na Tupla, o Contador

No método onde eu chamo diretamente a variável composta com o for, eu não consigo mostrar o contador.

```
# Não mostra o contador:  
for comida in lanche:  
    print(f'Eu vou comer {comida}.')  
print('Terminei de comer tudo.')  
  
#print(lancheSimples)  
#print(lanche)  
  
▶ aula16 ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts  
0 tamanho da tupla de lanche: 5  
  
Hoje eu comi Hamburguer que é a posição 0  
Hoje eu comi Suco que é a posição 1  
Hoje eu comi Pizza que é a posição 2  
Hoje eu comi Pudim que é a posição 3  
Hoje eu comi Batata Frita que é a posição 4  
Terminei de comer tudo e engordei.  
=====  
Eu vou comer Hamburguer.  
Eu vou comer Suco.  
Eu vou comer Pizza.  
Eu vou comer Pudim.  
Eu vou comer Batata Frita.  
Terminei de comer tudo.
```

Para mostrar o contador, seria necessário usar o outro método:

```
for comida in range(0, len(lanche)):  
    print(f'Hoje eu comi {lanche[comida]} que é a posição {comida}')  
print('Terminei de comer tudo e engordei.')
```

Porém, é possível ainda mostrar contador com o método que está na página anterior.

```
# Não mostra o contador:  
for pos, comida in enumerate(lanche):  
    print(f'Eu vou comer {comida} na posição {pos}.')  
print('Terminei de comer tudo.')
```

```
#print(lancheSimples)  
#print(lanche)  
#print((lanche[1:3]))  
for pos, comida in enumerate(la...
```

aula16 ×

```
Hoje eu comi Batata Frita que é a posição 4  
Terminei de comer tudo e engordei.
```

```
-----  
Eu vou comer Hambúrguer na posição 0.  
Eu vou comer Suco na posição 1.  
Eu vou comer Pizza na posição 2.  
Eu vou comer Pudim na posição 3.  
Eu vou comer Batata Frita na posição 4.  
Terminei de comer tudo.
```

## Alinhamento na Tupla - À Esquerda

Podemos alinhar o print das tuplas, para

```
produtos = ('Lápis', 1.75, 'Borracha', 2.00, 'Caderno', 15.90, 'Estojo', 25.00, 'Transferidor', 4.20,
           'Compasso', 9.99, 'Mochila', 120.32, 'Canetas', 22.30, 'Livro', 34.90)

for prod in range(0, len(produtos)):
    if prod % 2 == 0:
        print(f'{produtos[prod]:<30}')
```

for prod in range(0, len(produt... > if prod % 2 == 0  
ex072 x ex076 x  
C:/Users/lucas/PycharmProjects/pythonProject/venv/Scripts/python.exe C:/Users/lucas/PycharmProjects/py  
Lápis.....  
Borracha.....  
Caderno.....  
Estojo.....  
Transferidor.....  
Compasso.....  
Mochila.....  
Canetas.....  
Livro.....

```
for prod in range(0, len(produtos)):
    if prod % 2 == 1:
        print(f'{produtos[prod]:>30}')

for prod in range(0, len(produt... > if prod % 2 == 1  
ex072 x ex076 x  
C:/Users/lucas/PycharmProjects/pythonPr  
      1.75  
      2.0  
      15.9  
      25.0  
      4.2  
      9.99  
    120.32  
    22.3  
    34.9
```

## Alinhamento na Tupla – À Direita

The screenshot shows a Python script in PyCharm. The code uses f-strings to print product names and their prices. The `print` statement includes a format specifier `:.>50` which is highlighted with a blue rectangle. The output window shows the execution of the script, displaying the aligned product names and their prices.

```
for prod in range(0, len(produtos)):
    if prod % 2 == 0:
        print(f'{produtos[prod]:.>50}, end=' ')
    else:
        print(f'R$ {produtos[prod]:.2f}')
```

```
C:\Users\lucas\PycharmProjects\pythonProject\venv>
.....LápisR$ 1.75
.....BorrachaR$ 2.00
.....CadernoR$ 15.90
.....EstojoR$ 25.00
.....TransferidorR$ 4.20
.....CompassoR$ 9.99
.....MochilaR$ 120.32
.....CanetasR$ 22.30
.....LivroR$ 34.90
```

## Alinhamento na Tupla – Centralizado

The screenshot shows a Python script in PyCharm. The code is identical to the previous one, but the `print` statement uses a format specifier `:.^50` which is highlighted with a blue rectangle. The output window shows the execution of the script, displaying the centered product names and their prices.

```
for prod in range(0, len(produtos)):
    if prod % 2 == 0:
        print(f'{produtos[prod]:.^50}, end=' ')
    else:
        print(f'R$ {produtos[prod]:.2f}')
```

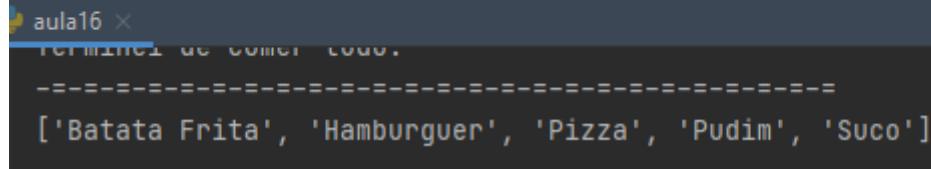
```
C:\Users\lucas\PycharmProjects\pythonProject\venv>
.....Lápis.....R$ 1.75
.....Borracha.....R$ 2.00
.....Caderno.....R$ 15.90
.....Estojo.....R$ 25.00
.....Transferidor.....R$ 4.20
.....Compasso.....R$ 9.99
.....Mochila.....R$ 120.32
.....Canetas.....R$ 22.30
.....Livro.....R$ 34.90
```

## Sorted

Para mostrar os elementos da tupla em ordem alfabética, utilizaremos o sorted.

A tupla não foi mudada! Ela apenas ordenou naquele momento na hora de imprimir!

```
print(sorted(lanche))
```

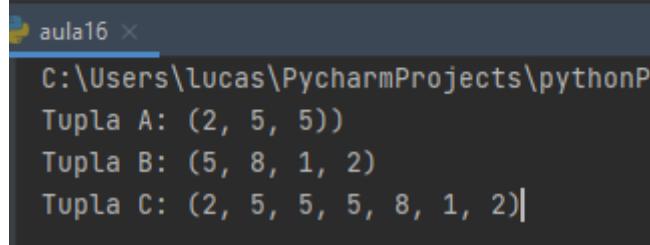


```
aula16 ✘ terminal de comandos
=====
['Batata Frita', 'Hamburguer', 'Pizza', 'Pudim', 'Suco']
```

## Soma em Tupla

Criando duas tuplas, 'a' e 'b' com valores e uma tupla 'c' que soma a + b. O resultado, não é uma soma de cada elemento em sua posição, e sim, as tuplas se unirão o final com o começo da outra.

```
a = (2, 5, 5)
b = (5, 8, 1, 2)
c = a + b
print(f'Tupla A: {a}')
print(f'Tupla B: {b}')
print(f'Tupla C: {c}')
```



```
aula16 ✘
C:\Users\lucas\PycharmProjects\pythonP
Tupla A: (2, 5, 5)
Tupla B: (5, 8, 1, 2)
Tupla C: (2, 5, 5, 5, 8, 1, 2)|
```

### Length na Tupla Somada

```
a = (2, 5, 5)
b = (5, 8, 1, 2)
c = a + b
print(f'Tupla A: {a}')
print(f'Tupla B: {b}')
print(f'Tupla C: {c}')
print(f'O tamanho da tupla somada: {len(c)}')
```

```
aula16 ×
C:\Users\lucas\PycharmProjects\pythonProject\ve
Tupla A: (2, 5, 5)
Tupla B: (5, 8, 1, 2)
Tupla C: (2, 5, 5, 5, 8, 1, 2)
O tamanho da tupla somada: 7
```

## Método Count na Tupla

Vamos utilizar, nesse exemplo, o método count para contar quantas vezes algo definido aparece na tupla. Claro, foi utilizado nesse exemplo em uma tupla somada, não é algo específico dela e sim de todas as tuplas.

```
a = (2, 5, 4)
b = (5, 8, 1, 2)
c = a + b
print(f'Tupla A: {a}')
print(f'Tupla B: {b}')
print(f'Tupla C: {c}')
print(f'O tamanho da tupla somada: {len(c)}')
print(f'O número 5 aparece {c.count(5)} vezes')
```

```
aula16 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Sc
Tupla A: (2, 5, 4)
Tupla B: (5, 8, 1, 2)
Tupla C: (2, 5, 4, 5, 8, 1, 2)
O tamanho da tupla somada: 7
O número 5 aparece 2 vezes
```

## Propriedade/ Método Index na Tupla

O index serve para identificar em qual posição algum elemento da tupla está.  
Quero saber onde está o número 8 na tupla C. Está na 4 posição.

```
a = (2, 5, 4)
b = (5, 8, 1, 2)
c = a + b
print(f'Tupla A: {a}')
print(f'Tupla B: {b}')
print(f'Tupla C: {c}')
print(f'O tamanho da tupla somada: {len(c)}')
print(f'O número 5 aparece {c.count(5)} vezes')
print(f'O número 8 está no index da posição: {c.index(8)}')
```

```
aula16 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\pyt
Tupla A: (2, 5, 4)
Tupla B: (5, 8, 1, 2)
Tupla C: (2, 5, 4, 5, 8, 1, 2)
O tamanho da tupla somada: 7
O número 5 aparece 2 vezes
O número 8 está no index da posição: 4
```

Mas, como pode ver, na Tupla C, vamos fazer o index do elemento 5. Note que ele está na posição 1 e, também na posição 3. Para isso, ele pegará a primeira ocorrência do número 5, mas podemos contornar indicando para contar a partir da posição 2 por exemplo.

```
a = (2, 5, 4)
b = (5, 8, 1, 2)
c = a + b
print(f'Tupla A: {a}')
print(f'Tupla B: {b}')
print(f'Tupla C: {c}')
print(f'O tamanho da tupla somada: {len(c)}')
print(f'O número 5 aparece {c.count(5)} vezes')
print(f'O número 8 está no index da posição: {c.index(8)}')
print(f'O número 5 está no index da posição: {c.index(5, 2)}')
```

```
aula16 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\pyt
Tupla A: (2, 5, 4)
Tupla B: (5, 8, 1, 2)
Tupla C: (2, 5, 4, 5, 8, 1, 2)
O tamanho da tupla somada: 7
O número 5 aparece 2 vezes
O número 8 está no index da posição: 4
O número 5 está no index da posição: 3
```

## Tratamento de Erro de Index

O index pode ser uma mão na roda para descobrirmos a posição de algo na tupla. Porém, com ele, temos dois grandes problemas encontrados nas aulas e exercícios:

1. Quando se mandar ele encontrar um número/string na tupla, ele vai retornar a posição exata, o que no “Mundo Real” não condiz, pois não poderia estar numa posição zero por exemplo. Em verde.
2. Caso queira encontrar o número e ele não esteja na lista, seja qual for o motivo, vai dar erro de index. Para isso, se corrige com um “if”. Em amarelo

```
tupla = (int(input('Digite o número 1: ')), int(input('Digite o número 2: ')),
         int(input('Digite o número 3: ')), int(input('Digite o número 4: ')),
         int(input('Digite o número 5: ')))

print(f'Tupla: {tupla}')
print(f'O número 9 apareceu: {tupla.count(9)} vezes')
if 3 in tupla:
    print(f'O número 3 apareceu pela primeira vez na posição {tupla.index(3) + 1}')
else:
    print('O número 3 não estava na lista!''')

par = impar = 0
for elemento in tupla:
    if elemento % 2 == 0:
        par += 1
    else:
        impar += 1

ex072 × ex075 ×
Digite o número 2: 4
Digite o número 3: 6
Digite o número 4: 9
Digite o número 5: 1
Tupla: (1, 4, 6, 9, 1)
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\pythonProject\ex075.py", line 9, in <module>
    print(f'O número 3 apareceu pela primeira vez na posição {tupla.index(3) + 1}')
ValueError: tuple.index(x): x not in tuple
0 número 9 apareceu: 1 vezes
```

## Correção do problema 2:

Poderíamos acrescentar o “else” embora não seja necessário.

```
if 3 in tupla:
    print(f'O número 3 apareceu pela primeira vez na posição {tupla.index(3) + 1}')
else:
    print('O número 3 não estava na lista!''')

ex072 × ex075 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas
Digite o número 1: 1
Digite o número 2: 4
Digite o número 3: 5
Digite o número 4: 4
Digite o número 5: 9
Tupla: (1, 4, 5, 4, 9)
0 número 9 apareceu: 1 vezes
0 total de pares: 2
0 total de ímpares: 3
```

## Elementos Diversos nas Tuplas

Em algumas linguagens de programação, não é possível misturar **tipos** diferentes como string com int, com float. **Em Python é possível.**

Em outras linguagens de programação:

```
numero = [1, 2, 3, 4]
```

Em Python:

```
pessoa = ('Lucas', 26, 'M', 83.2)
print(pessoa)

aula16 ×
C:\Users\lucas\PycharmProjects\python
('Lucas', 26, 'M', 83.2)
```

## Adicionando valores na tupla

Como a tupla não pode ser adicionada mesmo com um laço de repetição, porque mesmo que utilize um laço **For** ou **While**, a variável composta vai somente adquirir um último elemento. Muito cuidado com isso!!!!

```
from random import randint

tupla = (randint(1, 10), randint(1, 10), randint(1, 10),
         randint(1, 10), randint(1, 10))

#com for
print('Os valores da Lista com For rodando: ', end=' ')
for c in tupla:
    print(f'{c} ', end='')

#Sem for:
print(f'\nA lista vale: {tupla}')
print(f'O maior elemento da lista: {max(tupla)}')
print(f'O menor elemento da lista: {min(tupla)}')
```

```
ex072 × ex074 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Script
Os valores da Lista com For rodando: 1 8 7 10 7
A lista vale: (1, 8, 7, 10, 7)
O maior elemento da lista: 10
O menor elemento da lista: 1
```

## Apagar Tuplas com DEL

Para deletar e excluir uma variável composta/ tupla, podemos usar o comando `del` = nomeDaVariável. Se mandar imprimir essa variável, encontraremos um erro.

```
pessoa = ('Lucas', 26, 'M', 83.2)
del(pessoa)
print(pessoa)

aula16 ✘
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/Pycha
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\pythonProject\aula16.py", line 49, in <module>
    print(pessoa)
NameError: name 'pessoa' is not defined
```

Não é possível deletar algum elemento específico, não posso deletar um item da tupla.  
Mas posso deletar uma tupla inteira.

# Variáveis Compostas - Listas

## O problema das Tuplas

As listas são semelhantes as tuplas, porém, com algumas diferenças. Elas também são variáveis compostas. Lembrando que as tuplas são imutáveis, uma vez definidas, não se pode alocar mais nenhum valor.

Por exemplo, em Tuplas:

lanche = ('pizza', 'hamburguer', 'pudim', 'refrigerante')

Pizza	Hamburguer	Pudim	Refrigerante
0	1	2	3

Se por algum motivo eu queira realizar:

lanche[2] = 'sorvete'

Eu gostaria que isso acontecesse:

Pizza	Hamburguer	Sorvete	Refrigerante
0	1	2	3

Mas, as tuplas não funcionam assim.

Ficando, portanto, do modo como foram definidas anteriormente:

Pizza	Hamburguer	<del>Sorvete</del>	Pudim	Refrigerante
0	1	<del>2</del>	2	3

Para resolver esse “problema” onde todos os elementos foram definidos e caso precise substituir, a solução está em **LISTA**.

## Lista admite substituições

Diferente das tuplas que se iniciam com ‘( )’, as listas se iniciam com ‘[ ]’. As listas são **mutáveis!**

De forma similar as tuplas, para definir a lista:

Lanche = ['pizza', 'hamburguer', 'pudim', 'refrigerante']

Pizza	Hamburguer	Pudim	Refrigerante
0	1	2	3

Veja como é exatamente igual ao iniciar uma tupla.

Portanto, se:

lista[Posição] = ‘novoElemento’

>>>> lanche[2] = 'sorvete':

Pizza	Hamburguer	<b>Sorvete</b>	Refrigerante
0	1	<b>2</b>	3

## Método Append

As listas são realmente mais úteis que as tuplas, onde é possível alterar ou substituir um elemento da lista em qualquer momento da execução do programa.

Porém, para se adicionar um elemento a uma lista já pré-estabelecida, é necessário utilizar o método “**append**”, por exemplo:

```
nomeDaLista.append('elemento')
>>> lanche.append('biscoito')
```

Pizza	Hamburguer	Sorvete	Refrigerante	Biscoito
0	1	2	3	4

Sempre que a inclusão ocorrer dessa forma, o elemento adicionado será adicionado no final da lista.

## Método Insert

Se há necessidade de inserir um elemento em uma posição ‘x’ da lista, é necessário utilizar um método diferente do append. Deve-se utilizar o método “**insert**”, por exemplo:

```
nomeDaLista.insert(índice, 'elemento')
>>> lanche.insert(0, 'hot-dog')
```

Hot-Dog	Pizza	Hamburguer	Sorvete	Refrigerante	Biscoito
0	1	2	3	4	5

Nesse caso, se reparar com os exemplos anteriores, quando se adiciona dessa forma, o Python se encarrega de “empurrar” os elementos das listas para as demais posições.

Essas são as formas de declarações de listas. Há um método chamado ‘List’ que cria uma lista. Ver mais à frente.

## Apagar elementos na Lista

### Método Del

Para apagar um elemento, utilizamos o método mais convencional que é o método “**del**”, incluindo no parâmetro, o índice a ser eliminado. por exemplo:

```
del nomeDaLista [índice]
>>> del lanche [3]
```

Hot-Dog	Pizza	Hamburguer	Sorvete	Refrigerante	Biscoito
0	1	2	3	4	5

Hot-Dog	Pizza	Hamburguer	Refrigerante	Biscoito
0	1	2	3	4

## Método Pop

Outro método que permite deletar um elemento da lista, é o “**pop**”, normalmente, o método pop é utilizado para eliminar o último elemento da lista, mas pode passar como parâmetro o índice a ser eliminado, por exemplo:

**nomeDaLista.pop(indice)**

>>>> lanche.pop(3)

Hot-Dog	Pizza	Hamburguer	Sorvete	Refrigerante	Biscoito
0	1	2	<del>3</del>	4	5

Hot-Dog	Pizza	Hamburguer	Refrigerante	Biscoito
0	1	2	3	4

*Utilizando somente o método pop() sem passar nenhum índice ao parâmetro, ele eliminará o último elemento/índice da lista.*

>>> lanche.pop()

Hot-Dog	Pizza	Hamburguer	Refrigerante	<del>Biscoito</del>
0	1	2	3	<del>4</del>

Resultado:

Hot-Dog	Pizza	Hamburguer	Refrigerante
0	1	2	3

## Método Remove

Para eliminar um elemento onde não se elimina pelo índice, e sim pelo valor, isso é conceito muito importante, pois os demais anteriores, era necessário saber o índice do elemento. Para o remove, é o valor, “por extenso”,

Mas, ele vai procurar a **PRIMEIRA OCORRÊNCIA** desse valor, então, se houver mais um sorvete em alguma posição mais pra cima, esse método Remove só removerá a primeira ocorrência do sorvet. Por exemplo:

**nomeDaLista.remove('valorDoElemento')**

>>>> lanche.remove('sorvete')

Hot-Dog	Pizza	Hamburguer	<del>Sorvete</del>	Refrigerante	Biscoito
0	1	2	<del>3</del>	4	5

Hot-Dog	Pizza	Hamburguer	Refrigerante	Biscoito
0	1	2	3	4

## Identificando Erros nas Exclusões

Caso em algum dos comandos se executar por exemplo, um remove de algum valor/índice que não existe, o programa retornará com um erro. Para isso, podemos aplicar um teste lógico(if-else) para verificar a existência do elemento na lista. Por exemplo:

```
lanche.remove('pizza')
```

Hot-Dog	Hamburguer	Sorvete	Refrigerante	Biscoito
0	1	2	3	4

Nessa lista, o elemento pizza não existe. Vai gerar um erro. A não ser que tenha um teste lógico impedindo esse erro, utilizando o poderoso operador 'in' para verificação.

Resolução:

```
If 'pizza' in lanche:  
    lanche.remove('pizza')
```

Portanto, se na lista houver pizza, ele remove, se não houver, não será gerado nenhum erro.

Hot-Dog	Pizza	Hamburguer	Sorvete	Refrigerante	Biscoito
0	1	2	3	4	5

## Método List

A possibilidade de criar uma lista utilizando o método 'List'

### Criando lista através do Range

É possível criar uma lista utilizando orange do for, por exemplo:

```
variável = list(range(índiceInicial, índiceFinal))
```

```
>>> variável = list (range(4, 11))
```

```
Variável =
```

Elemento	4	5	6	7	8	9	10
Índice/Pos	0	1	2	3	4	5	6

O Range criou uma lista onde contempla desde o número 4 até o 11, onde o 11 é desconsiderado. Mas vale lembrar que não é a posição, e sim os elementos, os índices continuarão normalmente.

Pode utilizar também, com intervalos:

```
>>> variável = list (range(4, 11, 2)) – Intervalo de 2
```

```
Variável =
```

Elemento	4	6	8	10
Índice/Pos	0	1	2	3

## Método Sort() e Reverse ()

Se, dado uma lista desordenada, é possível organizar em ordem crescente, por exemplo:

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

Elemento	8	2	5	4	9	3	0
Índice/Pos	0	1	2	3	4	5	6

valores.sort()

Elemento	0	2	3	4	5	8	9
Índice/Pós	0	1	2	3	4	5	6

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

Elemento	9	8	5	4	3	2	0
Índice/Pós	0	1	2	3	4	5	6

## Tamanho da Lista

Para se descobrir um tamanho de uma determinada Lista, pode-se utilizar o comando 'Length', muito útil para realizar laços, por exemplo:

len(valores) = 7

Elemento	9	8	5	4	3	2	0
Índice/Pós	0	1	2	3	4	5	6

## Prática

Mostrando o erro da Tupla e como Listas corrigem:

Com Tupla:

```
num = (2, 5, 9, 1)
print(num)
print(num[0])
num[0] = 1

aula17 ✘
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas
(2, 5, 9, 1)
2
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\pythonProject\aula17.py", line 4, in <module>
    num[0] = 1
TypeError: 'tuple' object does not support item assignment

Process finished with exit code 1
```

Com Lista:

```
num = [2, 5, 9, 1]
print(num)
print(num[0])
num[0] = 500
print(num)

aula17 ✘
C:\Users\lucas\PycharmProjects\pythonProject\aula17.py
[2, 5, 9, 1]
2
[500, 5, 9, 1]

Process finished with exit code 0
```

### Adicionando elementos de forma errada

```
num = [2, 5, 9, 1]
num[4] = 7
```

aula17 x  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/python  
Traceback (most recent call last):  
 File "C:/Users/lucas/PycharmProjects/pythonProject/aula17.py", line 13, in <module>  
 num[4] = 7  
IndexError: list assignment index out of range

### Adicionando elementos da forma correta

```
num = [2, 5, 9, 1]
num.append(7)
print(num)
```

aula17 x  
C:\Users\lucas\PycharmProjects\python  
[2, 5, 9, 1, 7]

Process finished with exit code 0

### Colocando números em ordem crescente

```
num = [2, 5, 9, 1]
num.append(7)
num.sort()
print(num)
```

aula17 x  
C:\Users\lucas\PycharmProjects\python  
[1, 2, 5, 7, 9]

Process finished with exit code 0

Em ordem reversa

```
num = [2, 5, 9, 1]
num.append(7)
num.sort(reverse=True)
print(num)
```

```
aula17 ×
C:\Users\lucas\PycharmProjects\python
[9, 7, 5, 2, 1]

Process finished with exit code 0
```

Tamanho da Lista

```
num = [2, 5, 9, 1]
num.append(7)
num.sort(reverse=True)
print(num)

print(f'A minha lista tem {len(num)} elementos.')
```

```
aula17 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Script
[9, 7, 5, 2, 1]
A minha lista tem 5 elementos.

Process finished with exit code 0
```

### Inserindo com Insert

```
num = [2, 5, 9, 1]
num.append(7)
num.sort(reverse=True)
num.insert(2, 0) #adiciona na segunda posição, o número 0
print(num)

print(f'A minha lista tem {len(num)} elementos.')
```

```
aula17 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\
[9, 7, 0, 5, 2, 1]
A minha lista tem 6 elementos.

Process finished with exit code 0
```

### Removendo com o pop()

```
num = [2, 5, 9, 1]
num.append(7)
num.sort(reverse=True)
num.insert(2, 0) #adiciona na segunda posição, o número 0
num.pop()#vai remover o último elemento da lista
print(num)
```

```
aula17 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\
[9, 7, 0, 5, 2]
A minha lista tem 5 elementos.

Process finished with exit code 0
```

Pela ordem, o nº 0 ocupa a segunda posição:

```
num = [2, 5, 9, 1]
num.append(7)
num.sort(reverse=True)
num.insert(2, 0) #adiciona na segunda posição o número 0
num.pop()#vai remover o último elemento da lista
num.pop(2)
print(num)

aula17 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts
[9, 7, 5, 2]
A minha lista tem 4 elementos.
```

### Removendo a primeira ocorrência de valor com o Remove

Pela ordem, ele foi adicionando na posição 2, onde já havia um valor e também na lista existente. Porém, se futuramente vier um remove(2), ele irá remover o primeiro valor que existe com essa condição.

```
num = [2, 5, 9, 1]
num.append(7)
num.sort(reverse=True)
num.insert(2, 2)#insira o num 2 na posição 2
print(num)

print(f'A minha lista tem {len(num)} elementos.')

aula17 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts
[9, 7, 2, 5, 2, 1]
A minha lista tem 6 elementos.
```

```
num = [2, 5, 9, 1]
num.append(7)
num.sort(reverse=True)
num.insert(2, 2)#insira o num 2 na posição 2
num.remove(2)
print(num)

aula17 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts
[9, 7, 5, 2, 1]
A minha lista tem 5 elementos.
```

## Erro com Remove não presente na lista

Tentando remover algum elemento que não está presente na lista:

```
num = [2, 5, 9, 1]
num.append(7)
num.sort(reverse=True)
num.insert(2, 2)#insira o num_2 na posição_2
num.remove(4)
print(num)

aula17 ×
C:\Users\Lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/Py
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\pythonProject\aula17.py", line 16, in <module>
    num.remove(4)
ValueError: list.remove(x): x not in list
```

## Corrigindo o Problema de Elemento não listado

```
num = [2, 5, 9, 1]
num.append(7)
num.sort(reverse=True)
num.insert(2, 2)#insira o num_2 na posição_2
if 4 in num:
    num.remove(4)
else:
    print('Não achei o número 4.')
print(num)

aula17 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scr
Não achei o número 4.
[9, 7, 2, 5, 2, 1]
A minha lista tem 6 elementos.
```

```
num = [2, 5, 9, 1]
num.append(7)
num.sort(reverse=True)
num.insert(2, 2)#insira o num 2 na posição 2
if 5 in num:
    num.remove(5)
else:
    print('Não achei o número 4.')
print(num)

if 5 in num
▶ aula17 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Sc
[9, 7, 2, 2, 1]
A minha lista tem 5 elementos.
```

## Imprimindo uma lista com For

```
valores = []
valores.append(5)
valores.append(9)
valores.append(4)

for v in valores:
    print(f'{v}...', end='')

▶ aula17 ×
C:\Users\lucas\PycharmProjects\pythonProject\ven
5...9...4...
Process finished with exit code 0
```

## Mostrando Chaves e Valores com Índices com o For

```
valores = []
valores.append(5)
valores.append(9)
valores.append(4)

for c, v in enumerate(valores):
    print(f'Na posição {c} encontrei o valor: {v}...')
print('Final da Lista')

for c, v in enumerate(valores)
    aula17 ×

C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe -u "C:/Users/lucas/PycharmProjects/pythonProject/venv/Scripts/aula17.py"
Na posição 0 encontrei o valor: 5...
Na posição 1 encontrei o valor: 9...
Na posição 2 encontrei o valor: 4...
Final da Lista
```

## Lendo valores pelo Teclado e Adicionando a Lista + Enumerate

```
valores = list()

for cont in range(0, 5):
    valores.append(int(input('Digite um valor: ')))
for c, v in enumerate(valores):
    print(f'Na posição {c} encontrei o valor: {v}...')
print('Final da Lista')

aula17 ×

C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe -u "C:/Users/lucas/PycharmProjects/pythonProject/venv/Scripts/aula17.py"
Digite um valor: 5
Digite um valor: 5
Digite um valor: 6
Digite um valor: 3
Digite um valor: 4
Na posição 0 encontrei o valor: 5...
Na posição 1 encontrei o valor: 5...
Na posição 2 encontrei o valor: 6...
Na posição 3 encontrei o valor: 3...
Na posição 4 encontrei o valor: 4...
Final da Lista
```

## Atribuição de lista

```
a = [2, 3, 4, 7]
b = a

print(a)
print(b)
```

```
aula17 ×
C:\Users\lucas\PycharmProjects\pyt
[2, 3, 4, 7]
[2, 3, 4, 7]
```

### CUIDADO!

Ao fazer uma igualdade, ao igualar duas listas, como no caso de `b = a`, tudo que se alterar na lista se alterará na outra. Se mexer na lista B, vai mexer na Lista A também.

```
a = [2, 3, 4, 7]
b = a

print(f'A lista A vale: {a}')
print(f'A lista B vale: {b}')
```

```
aula17 ×
C:\Users\lucas\PycharmProjects\pyt
A lista A vale: [2, 3, 4, 7]
A lista B vale: [2, 3, 4, 7]
```

```
a = [2, 3, 4, 7]
b = a
b[2] = 8 #espera-se que somente B recebe 8, mas não. A recebe também.
print(f'A lista A vale: {a}')
print(f'A lista B vale: {b}')
```

```
aula17 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/U
A lista A vale: [2, 3, 8, 7]
A lista B vale: [2, 3, 8, 7]
```

## Fazendo uma cópia de uma lista na outra.

Quando ao fazer a igualdade na lista, acaba-se por criar um vínculo entre as duas listas, mas é possível fazer uma “igualdade” de cópia, se um valor numa lista for alterado, a outra não será alterada.

É necessário ao invés do `b = a`, se faz o uso de `b = a[:]`, onde o `[:]` significa que uma cópia de a será criada em b.

```
a = [2, 3, 4, 7]
b = a[:] #o [:] vai criar uma cópia dos seus valores, não ligação
b[2] = 8 #espera-se que somente B recebe 8, como assim foi copiado, somente B altera]
print(f'A lista A vale: {a}')
print(f'A lista B vale: {b}')


aula17 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/Py
A lista A vale: [2, 3, 4, 7]
A lista B vale: [2, 3, 8, 7]
```

## Variáveis Compostas - Listas – Parte 2

É possível manipular as listas de diversas formas. Uma delas é fazer uma lista receber uma outra lista. Um append dentro de uma lista., por exemplo:

dados =

Pedro	25
0	1

Podemos interpretar que na lista dados, o índice 0 é 'Pedro' e o índice 1 é '25'.

Porém, podemos criar uma lista chamada: pessoas = [ ] ou pessoas = list ( )

E executar o comando de append nessa lista pessoas, só que fazer o append de dados:

**pessoas.append(dados[:])** → Esse comando adicionará a lista dados na lista pessoas.

*Obs: '[:] significa um fatiamento completo da estrutura da lista.*

Portanto, essa lista pessoas, assumirá na posição/ índice 0 → Pedro e 25.

pessoas =

Tony	3
0	1
0	

Portanto, Pedro e 25 onde as suas posições em dados são correspondentes a 0 e 1, quando são adicionados à lista de pessoas, assumem o índice 0.

pessoas =

Tony      3 0          1 0	Laura    24 0          1 1	Lucas    26 0          1 2
----------------------------------	----------------------------------	----------------------------------

Podemos criar a tabela acima, com uma estrutura direta da lista:

pessoas = [[‘Tony’,3], [‘Laura’,24], [‘Lucas’,26]]

Por exemplo, se o comando:

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

```
>>>> ‘Tony’
```

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

```
>>>> 19
```

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

```
>>>> ‘Lucas’
```

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

```
>>>>[‘Laura’, 24]
```

O retorno será exatamente tudo que está contido na lista naquela posição, incluindo os colchetes.

Surge uma necessidade de: sabe-se que o índice zero é um “nome” e o índice 1 é uma “Idade”, por que não se utiliza? Porque a lista não atende essa questão, esse tratamento será realizado em **Dicionários**.

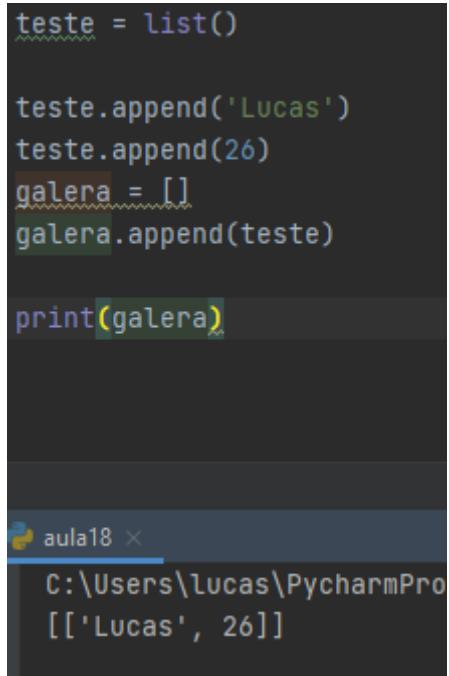
## Prática

### Criando Lista com Lista

```
teste = list()

teste.append('Lucas')
teste.append(26)
galera = []
galera.append(teste)

print(galera)
```



The screenshot shows the PyCharm interface with a code editor containing the provided Python code. Below it is a terminal window titled 'aula18' showing the execution of the code and its output: '[[ 'Lucas', 26]]'.

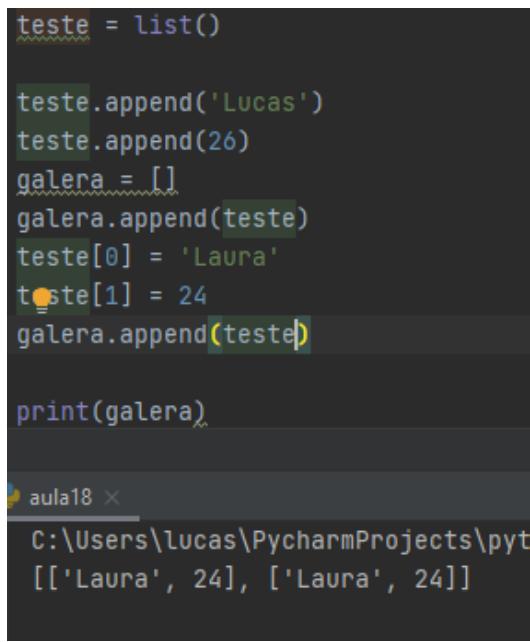
### Erro ao fazer append com outra lista

Se tentarmos fazer uma substituição de quem já estava na lista diretamente. O Python criará uma vínculo entre as duas listas, ocasionando o erro abaixo, porque ao fazer a substituição, tá trocando lá atrás na origem da lista.

```
teste = list()

teste.append('Lucas')
teste.append(26)
galera = []
galera.append(teste)
teste[0] = 'Laura'
teste[1] = 24
galera.append(teste)

print(galera)
```



The screenshot shows the PyCharm interface with the modified code. The terminal window shows the output: '[['Laura', 24], ['Laura', 24]]', indicating that both original lists were modified.

Append com outra lista na forma correta

```
galera = [['Laura', 24], ['Lucas', 26], ['Tony', 3], ['Nina', 10]]
print(galera)
print(galera[0])
print(galera[0][0])
```

```
aula18 ✘
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
[['Laura', 24], ['Lucas', 26], ['Tony', 3], ['Nina', 10]]
['Laura', 24]
Laura
```

Print da Lista com For

```
galera = [['Laura', 24], ['Lucas', 26], ['Tony', 3], ['Nina', 10]]

for pessoa in galera:
    print(pessoa)#print de todos

print('=-' * 20)

for pessoa in galera:
    print(pessoa[0])#print só os nomes

'''print(galera)
for pessoa in galera
aula18 ✘
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
['Laura', 24]
['Lucas', 26]
['Tony', 3]
['Nina', 10]
=====
Laura
Lucas
Tony
Nina
```

```
galera = [['Lucas', 26], ['Laura', 24], ['Tony', 3], ['Nina', 10]]  
  
for pessoa in galera:    pessoa: ['Laura', 24]  
    print(pessoa)
```

```
> 1 galera = {list: 4} [['Lucas', 26], ['Laura', 24], ['Tony', 3], ['Nina', 10]]  
> 2 pessoa = {list: 2} ['Laura', 24]  
> 3 Special Variables
```

```
C:/Users/lucas/PycharmProjects/pythonProject/aula18b.py  
Connected to pydev debugger (build 203.5981.165)  
['Lucas', 26]  
['Laura', 24]
```

Para (for) cada pessoa na (in) galera: print pessoa.

O que isso significa? Que para cada elemento Pessoa na lista, ele vai printar toda aquele elemento. Só que na lista “grande/composta” o elemento Pessoa, é todo o elemento, ou seja, se o elemento zero é Lucas, 26. Ele vai trazer todo o elemento zero, e por aí em diante.

Print formatando mostrando atributos das pessoas

Se le no printf: A pessoa tem dados na posição zero tem dados da pessoa na posição um  
anos de idade

```
galera = [['Laura', 24], ['Lucas', 26], ['Tony', 3], ['Nina', 10]]  
  
for pessoas in galera:  
    print(f'{pessoas[0]} tem {pessoas[1]} anos de idade.')
```

```
aula18 ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python  
Laura tem 24 anos de idade.  
Lucas tem 26 anos de idade.  
Tony tem 3 anos de idade.  
Nina tem 10 anos de idade.
```

### Solicitar input do Usuário para criar a Lista

```
#Criando Lista com Input do Usuário:

galera = []
dado = list()

for c in range(0, 3):
    dado.append(str(input('Digite Nome: ')))
    dado.append(int(input('Digite Idade: ')))
```

Primeiro cria-se duas listas, uma que armazenará os dados futuramente e outra que receberá os dados momentaneamente. Dado isso, utilizamos um laço For ou While (depende da situação) que fará a pergunta ao usuário sobre o nome e a idade. Lembre-se de armazenar, ou seja, o append será na lista momentanea. Após isso:

```
#Criando Lista com Input do Usuário:

galera = []
dado = list()

for c in range(0, 3):
    dado.append(str(input('Digite Nome: ')))
    dado.append(int(input('Digite Idade: ')))
    galera.append(dado[:]) #cria uma cópia sem vínculo com dado
    dado.clear() #limpa a lista de dados para não armazenar dados desnecessários.
print(galera)
```

```
aula18 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas
Digite Nome: Laura
Digite Idade: 24
Digite Nome: Lucas
Digite Idade: 26
Digite Nome: Tony
Digite Idade: 3
[['Laura', 24], ['Lucas', 26], ['Tony', 3]]
```

Lembre-se da importância do ‘[:]’ ao fazer cópia de uma lista para outra, o fatiamento inteiro da lista é necessário. Se não, se fizermos um clear() de dados sem termos feito a cópia, listas vazias serão geradas, mesmo se der input dos dados corretamente.

Primeiro eu crio duas listas, uma temporária e outra que armazenará de fato a “informação da galera”. Para isso, eu vou ler inputs do usuário através de um FOR que terá um contador que lerá 3 inputs do usuário. A lista temporária armazenará os dados dos nomes e idades, após cada inserção essa lista será gerada uma cópia na permanente e aí sim, ela será limpa para que possa armazenar novos dados novamente.

```
galera = list()
dados = []

for contador in range(0, 3):
    dados.append(str(input('Nome: ')))
    dados.append(int(input('Idade: ')))
    galera.append(dados[:])
    dados.clear()
```

Para que os dados sejam mostrados na tela de forma organizada, ao invés de simplesmente expostos na estrutura da lista, podemos imprimir através de um laço FOR, qual assumirá os mesmos princípios da página 103.

```
for pessoas in galera:
    print(f'{pessoas[0]} tem {pessoas[1]} anos de idade.')
```

Resultando em:

```
Nome: Lucas
Idade: 26
Nome: Laura
Idade: 24
Nome: Tony
Idade: 3
Lucas tem 26 anos de idade.
Laura tem 24 anos de idade.
Tony tem 3 anos de idade.
```

## Maior de Idade e Menor de Idade com a lista, exemplo prático

Podemos criar um laço que contará quantas pessoas são maiores quantas não são.

Lemos da seguinte forma: para cada pessoa na galera: se a posição[1] dessa pessoa (que é a idade) for maior ou igual a 21, ela é maior de idade.

```
#Criando Lista com Input do Usuário:  
galera = []  
dado = list()  
maiorIdade = menorIdade = 0  
  
for c in range(0, 3):  
    dado.append(str(input('Digite Nome: ')))  
    dado.append(int(input('Digite Idade: ')))  
    galera.append(dado[:]) #cria uma cópia sem vínculo com dado  
    dado.clear() #limpa a lista de dados para não armazenar dados desnecessários.  
  
for pessoas in galera:  
    if pessoas[1] >= 21:  
        print(f'{pessoas[0]}, é maior de idade')  
        maiorIdade += 1  
    else:  
        print(f'{pessoas[0]}, não é maior de idade')  
        menorIdade += 1  
  
print(f'0 n° de maiores: {maiorIdade}'  
      f'\n0 n° de menores: {menorIdade}')
```

```
Digite Nome: Lucas  
Digite Idade: 15  
Digite Nome: Laura  
Digite Idade: 14  
Digite Nome: Tony  
Digite Idade: 26  
Lucas,não é maior de idade  
Laura,não é maior de idade  
Tony, é maior de idade  
0 n° de maiores: 1  
0 n° de menores: 2
```

```
Process finished with exit code 0
```

Seguindo a lógica das duas listas(temporária e permanente) de dados, continuamos solicitando 3 inputs ao usuário e limpando a lista temporária.

```
galera = list()
dados = []

for contador in range(0, 3):
    dados.append(str(input('Nome: ')))
    dados.append(int(input('Idade: ')))
    galera.append(dados[:])
    dados.clear()
```

Podemos analisar, que para cada vez que o elemento foi adicionado a lista permanente, ele foi criado como um exemplo, a posição: Lucas e 26 e por aí em diante.

```
01 contador = {int} 2
> 2: dados = {list: 0} []
> 2: galera = {list: 3} [['Lucas', 26], ['Laura', 24], ['Tony', 3]]
> 3: Special Variables
```

Podemos, portanto, fazer um laço FOR que lerá e fará uma condição lógica para saber qual pessoa é maior de idade.

Para isso, precisamos fazer a leitura: “Para cada pessoa na lista galera”, “se a posição da pessoa[1], sabendo que pessoa assume cada elemento por exemplo a posição zero (lucas, 26), ou seja, se a pessoa por pessoa[1] que assume o número 26 é maior que 21, então:

Print aquela pessoa na posição zero-zero, que é o nome, e possui aquela pessoa na posição zero-um, que é a idade.

```
for pessoa in galera:
    if pessoa[1] >= 21:
        print(f'{pessoa[0]} é maior de idade. Possui idade de {pessoa[1]} anos de idade.')
```

```
01 contador = {int} 2
> 2: dados = {list: 0} []
> 2: galera = {list: 3} [['Lucas', 26], ['Laura', 24], ['Tony', 3]]
> 2: pessoa = {list: 2} ['Laura', 24]
> 3: Special Variables
```

Perceba como ele aloca para cada pessoa é o elemento da lista.

# Estrutura de Dados Composta – Dicionário

Uma estrutura de dados composta, é o **dicionário**, similar as **tuplas** e as **listas**. Formando as 3 estruturas de dados compostas.

Nas ‘Listas’ podíamos declarar os valores como por exemplo:

Dados = ['Pedro', 25]

E se mandássemos imprimir dados [0] = Pedro e dados [1] = 25.

Não seria mais interessante e eficaz ao invés de utilizar um índice, utilizarmos por exemplo, nome para a posição zero e idade para a posição um? Utilizaremos os dicionários, pois poderemos ter índices literais, índices personalizáveis.

## Identificando um Dicionário

Vale lembrar a identificação:

Tuplas = () – parênteses

Listas = [] - colchetes

Dicionários = {} – chaves

## Declarando o Dicionário

Podemos seguir com o exemplo anterior ainda:

dados = dict()

ou

dados = {'nome': 'Pedro', 'idade': 25 }

Sendo ‘Pedro’ como o **valor** e o ‘nome’ como o **identificador do índice/elemento**.

Ou seja:

Dados = {

Valor →	Pedro	25
Índice →	nome	idade

}

Não tenho mais, ex: dados[0] ou dados[1]

Podemos utilizar:

print(dados['nome']) = Pedro

print(dados['idade']) = 25

## Adicionando elementos ao Dicionário criado

Por exemplo, caso seja necessário criar mais um índice, como o sexo do usuário.

Ressaltar que o “append” **não** funciona em um dicionário.

dados[‘sexo’] = ‘M’

Valor →	Pedro	25	‘M’
Índice →	nome	idade	sexo

## Removendo elemento do Dicionário

Para remover um elemento é um simples. Basta utilizar o comando “del”:

del dados[‘idade’]

Valor →	Pedro	25	‘M’
Índice →	nome	idade	sexo

## Criando um Dicionário para Guardar Elementos

Vamos criar um dicionário para criar uma lista/nome de filmes:

filmes = {‘título’: ‘Star Wars’,

‘ano’: 1977,

‘diretor’: ‘George Lucas’

}

Valor →	‘Star Wars’	1977	‘George Lucas’
Índice/elementos →	Título	Ano	diretor

Esses elementos em Python são chamados de **‘Chaves’** ou **‘Keys’**. Vamos entender:

### Values/Valores

Imprimirá somente os valores que as chaves deram.

print(filme.values()) → ‘Star Wars’, 1977, ‘George Lucas’

### Keys/Chaves

Imprimirá somente as chaves.

print(filme.keys()) → ‘título’, ‘ano’, ‘diretor’

### Items/Item

Imprimirá todos os valores e chaves do dicionário

print(filme.items()) → ‘título’: ‘Star Wars’, ‘ano’: 1977, ‘diretor’: ‘George Lucas’

## Laço For no Dicionário

Podemos utilizar as chaves, valores dentro do laço for:

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'
```

Para cada key/chave e values/valores no filme.items(): printe a chave e o valor.

Valor →	'Star Wars'	1977	'George Lucas'
Índice/elementos →	Título	Ano	diretor

## Dicionário dentro de Listas

Suponhamos que nós temos uma locadora. Dentro dessa locadora há filmes.

Locadora = [

Star Wars	1977	George Lucas	Avengers	2012	Joss Whedon	Matrix	1999	Wachowski
título	ano	diretor	título	ano	diretor	título	ano	diretor
0		1			2			

]

Tenho uma lista locadora = [] que tem 3 elementos e que esses elementos tem dicionários. Nesse exemplo, a lista ela é identificada pelos números e os dicionários identificados pelos chaves literais, textos (embora possamos colocar números nos dicionários também).

```
print(locadora[0]['ano'] = 1977
print[locadora[2]['título'] = 'Matrix'
```

## Parte Prática

Declarando o Dicionário pronto

```
pessoas = {'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}  
print(pessoas)
```

```
ex088b × ex089b × aula19 ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts  
{'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}
```

Erros

```
pessoas = {'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}  
  
print(pessoas[0])  
  
ex088b × ex089b × aula19 ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/Py  
Traceback (most recent call last):  
  File "C:\Users\lucas\PycharmProjects\pythonProject\aula19.py", line 3, in <module>  
    print(pessoas[0])  
KeyError: 0
```

Esse erro acima apareceu porque não existe ‘pessoa[0]’, afinal, não é uma lista, para imprimir ‘Lucas’ é necessário informar a key/chave:

```
pessoas = {'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}  
print(pessoas['nome'])  
  
ex088b × ex089b × aula19 ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts  
Lucas
```

## Print Formatado e Referência

Muito cuidado ao imprimir um comando formatado. Para declarar o dicionário utilizar as chaves {}, mas para referenciar num print é o colchetes [ ] e entre aspas duplas.

```
pessoas = {'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}
```

```
print(f'{pessoas["nome"]} tem {pessoas["idade"]} anos.')
```

```
ex088b × ex089b × aula19 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\py
Lucas tem 26 anos.
```

## Print nas Keys/Chaves

```
pessoas = {'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}
```

```
#print(f'{pessoas["nome"]} tem {pessoas["idade"]} anos.')
```

```
print(pessoas.keys())
```

```
ex088b × ex089b × aula19 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\py
dict_keys(['nome', 'idade', 'sexo'])

Process finished with exit code 0
```

## Print nos Values/Valores

```
pessoas = {'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}
```

```
#print(f'{pessoas["nome"]} tem {pessoas["idade"]} anos.')
```

```
#print(pessoas.keys())
print(pessoas.values())
```

```
ex088b × ex089b × aula19 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\py
dict_values(['Lucas', 26, 'M'])
```

## Print nos Itens

Percebe que os itens são uma lista e dentro dessa lista de elementos, são compostas de 3 tuplas com 2 valores em cada.

```
pessoas = {'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}
```

```
#print(f'{pessoas["nome"]} tem {pessoas["idade"]} anos.')
```

```
#print(pessoas.keys())
#print(pessoas.values())
print(pessoas.items())
```

```
ex088b × ex089b × aula19 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.
dict_items([('nome', 'Lucas'), ('idade', 26), ('sexo', 'M')])
```

## Imprimindo as Chaves, Valores e Itens:

```
pessoas = {'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}
```

```
for k in pessoas.keys():
    print(k)
```

```
ex088b × ex089b × aula19 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts
nome
idade
sexo
```

```
pessoas = {'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}
```

```
for k in pessoas.values():
    print(k)
```

```
ex088b × ex089b × aula19 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Script
Lucas
26
M
```

Entretanto, para mostrar os itens, é preciso dois elementos no laço for:

```
pessoas = {'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}
```

```
for k, v in pessoas.items():
    print(f'{k} = {v}')
```

```
for k, v in pessoas.items()
ex088b × ex089b × aula19 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Script
0 nome = Lucas
0 idade = 26
0 sexo = M
```

```
pessoas = {'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}
```

```
del pessoas['sexo']
```

```
for k, v in pessoas.items():
    print(f'{k} = {v}')
```

```
ex088b × ex089b × aula19 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Script
0 nome = Lucas
0 idade = 26
```

## Usando o Del

Novas atribuições as chaves

```
pessoas = {'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}
```

```
#del_pessoas['sexo']
pessoas['nome'] = 'Laura'
```

```
for k, v in pessoas.items():
    print(f'{k} = {v}')
```

```
ex088b × ex089b × aula19 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Script
0 nome = Laura
0 idade = 26
0 sexo = M
```

## Adicionando chaves e elementos na lista

```
pessoas = {'nome': 'Lucas', 'idade': 26, 'sexo': 'M'}

#del pessoas['sexo']
pessoas['nome'] = 'Laura'
pessoas['peso'] = 45

for k, v in pessoas.items():
    print(f'O {k} = {v}')
```

```
ex088b × ex089b × aula19 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/pythonProject/aula19.py
O nome = Laura
O idade = 26
O sexo = M
O peso = 45
```

## Dicionários dentro de Lista

```
#Dicionário dentro de lista:

brasil = []#lista
estado1 = {'uf': 'Rio de Janeiro', 'sigla': 'RJ'}#dicionario
estado2 = {'uf': 'São Paulo', 'sigla': 'SP'}#dicionario

brasil.append(estado1)
brasil.append(estado2)

print('Estado 1: {}'.format(estado1))
print('Estado 2: {}'.format(estado2))
print('Lista Brasil Inteira: {}'.format(brasil))
print('Brasil o dicionário 1: {}'.format(brasil[0]))
print('Brasil o dicionário 2: {}'.format(brasil[1]))
```

```
ex088b × ex089b × aula19 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/pythonProject/aula19.py
Estado 1: {'uf': 'Rio de Janeiro', 'sigla': 'RJ'}
Estado 2: {'uf': 'São Paulo', 'sigla': 'SP'}
Lista Brasil Inteira: [{'uf': 'Rio de Janeiro', 'sigla': 'RJ'}, {'uf': 'São Paulo', 'sigla': 'SP'}]
Brasil o dicionário 1: {'uf': 'Rio de Janeiro', 'sigla': 'RJ'}
Brasil o dicionário 2: {'uf': 'São Paulo', 'sigla': 'SP'}
```

## Imprimindo Listas Compostas com campos de Chaves

```
#Dicionário dentro de lista:  
  
brasil = []#lista  
estado1 = {'uf': 'Rio de Janeiro', 'sigla': 'RJ'}#dicionario  
estado2 = {'uf': 'São Paulo', 'sigla': 'SP'}#dicionario  
  
brasil.append(estado1)  
brasil.append(estado2)  
  
print(brasil)  
print(brasil[0]['uf'])  
print(brasil[1]['sigla'])
```

```
ex088b × ex089b × aula19 ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/luc  
[{'uf': 'Rio de Janeiro', 'sigla': 'RJ'}, {'uf': 'São Paulo', 'sigla': 'SP'}]  
Rio de Janeiro  
SP
```

## Declarando Dicionário + Leitura de Inputs

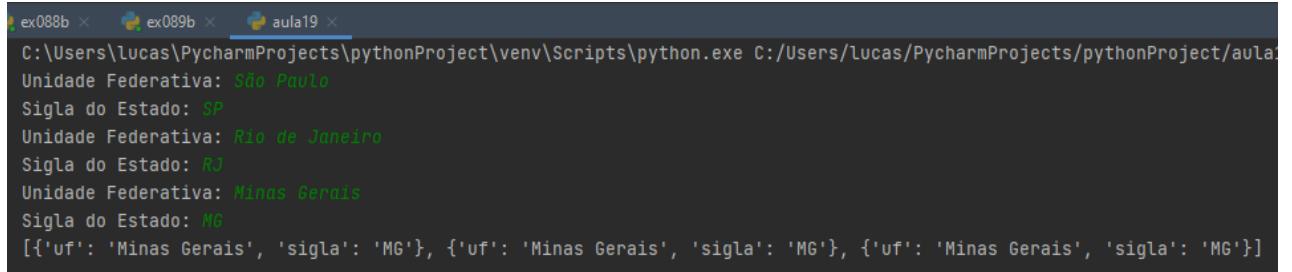
### Com Laço For – Erro e Solução

Repare que ao criar o laço for para fazer a leitura, criamos as chaves e adicionamos os valores, temos e na hora de fazer os dicionários serem adicionados a lista Brasil, ocorre um erro pois somente o último valor foi atribuído a lista do Brasil.

```
estado = dict() #estados serão dicionários
brasil = list() #brasil será uma lista composta por estados

#desejo ler 3 estados:
for contador in range(0, 3):
    estado['uf'] = str(input('Unidade Federativa: '))
    estado['sigla'] = str(input('Sigla do Estado: '))
    #essa leitura acima será realizada 3 vezes.
    #Ainda é preciso adicionar os estados a lista Brasil.
    brasil.append(estado)

print(brasil)
```



```
ex088b x ex089b x aula19 x
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Lucas/PycharmProjects/pythonProject/aula19.py
Unidade Federativa: São Paulo
Sigla do Estado: SP
Unidade Federativa: Rio de Janeiro
Sigla do Estado: RJ
Unidade Federativa: Minas Gerais
Sigla do Estado: MG
[{'uf': 'Minas Gerais', 'sigla': 'MG'}, {'uf': 'Minas Gerais', 'sigla': 'MG'}, {'uf': 'Minas Gerais', 'sigla': 'MG'}]
```

Outro Erro: Não podemos fazer o fatiamento inteiro, que é fazer aquela cópia de lista pra outra lista.

```
estado = dict() #estados serão dicionários
brasil = list() #brasil será uma lista composta por estados

#desejo ler 3 estados:
for contador in range(0, 3):
    estado['uf'] = str(input('Unidade Federativa: '))
    estado['sigla'] = str(input('Sigla do Estado: '))
    #essa leitura acima será realizada 3 vezes.
    #Ainda é preciso adicionar os estados a lista Brasil.
    brasil.append(estado[:]) # Erro: TypeError: unhashable type: 'slice'

print(brasil)

for contador in range(0, 3)
```

ex088b × ex089b × aula19 ×

C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\p  
Unidade Federativa: São Paulo  
Sigla do Estado: SP  
Traceback (most recent call last):  
File "C:\Users\lucas\PycharmProjects\pythonProject\aula19"  
 brasil.append(estado[:])  
TypeError: unhashable type: 'slice'

## Método Copy

O método copy é o responsável por fazer essa cópia e não permanecer o último valor do dicionário na lista:

```
estado = dict() #estados serão dicionários
brasil = list() #brasil será uma lista composta por estados

#desejo ler 3 estados:
for contador in range(0, 3):
    estado['uf'] = str(input('Unidade Federativa: '))
    estado['sigla'] = str(input('Sigla do Estado: '))
    #essa leitura acima será realizada 3 vezes.
    #Ainda é preciso adicionar os estados a lista Brasil.
    brasil.append(estado.copy())

print(brasil)
for contador in range(0, 3):
    Unidade Federativa: São Paulo
    Sigla do Estado: SP
    Unidade Federativa: Rio de Janeiro
    Sigla do Estado: RJ
    Unidade Federativa: Minas Gerais
    Sigla do Estado: MG
[{'uf': 'São Paulo', 'sigla': 'SP'}, {'uf': 'Rio de Janeiro', 'sigla': 'RJ'}, {'uf': 'Minas Gerais', 'sigla': 'MG'}]
```

## Impressão da Lista com For

```
estado = dict() #estados serão dicionários
brasil = list() #brasil será uma lista composta por estados

#desejo ler 3 estados:
for contador in range(0, 3):
    estado['uf'] = str(input('Unidade Federativa: '))
    estado['sigla'] = str(input('Sigla do Estado: '))
    #essa leitura acima será realizada 3 vezes.
    #Ainda é preciso adicionar os estados a lista Brasil.
    brasil.append(estado.copy())
    #Fim da Alimentação

for est in brasil:
    print(est)
#fim da impressão dos dados
```

```
Unidade Federativa: São Paulo
Sigla do Estado: SP
Unidade Federativa: Rio de Janeiro
Sigla do Estado: RJ
Unidade Federativa: Minas Gerais
Sigla do Estado: MG
{'uf': 'São Paulo', 'sigla': 'SP'}
{'uf': 'Rio de Janeiro', 'sigla': 'RJ'}
{'uf': 'Minas Gerais', 'sigla': 'MG'}
```

Ainda no for sobre a Impressão da lista com dicionário:

Com chaves e os seus respectivos valores é preciso usar o “items”.

```
|for est in brasil:  
|    for k, v in est.items():  
|        print(f'O campo {k} tem valor {v}')  
|    #fim da impressão dos dados
```

```
Unidade Federativa: Rio de Janeiro  
Sigla do Estado: RJ  
Unidade Federativa: Minas Gerais  
Sigla do Estado: MG  
O campo uf tem valor São Paulo  
O campo sigla tem valor SP  
O campo uf tem valor Rio de Janeiro  
O campo sigla tem valor RJ  
O campo uf tem valor Minas Gerais  
O campo sigla tem valor MG
```

Somente valores podemos utilizar apenas um elemento no for e apenas o values.

```
|for est in brasil:  
|    for v in est.values():  
|        print(v)  
|    #fim da impressão dos dados
```

```
Sampa  
SP  
Rio  
RJ  
Minas  
MG
```

```
|for est in brasil:  
|    for v in est.values():  
|        print(v, end=', ')  
|    print()  
|    #fim da impressão dos dados
```

```
Unidade Federativa: Sampa  
Sigla do Estado: SP  
Unidade Federativa: Rio  
Sigla do Estado: RJ  
Unidade Federativa: Minas  
Sigla do Estado: MG  
Sampa, SP,  
Rio, RJ,  
Minas, MG,
```

Colocando um dicionário em ordem.

```
jogador = {'jogador 1': randint(1, 6),
            'jogador 2': randint(1, 6),
            'jogador 3': randint(1, 6),
            'jogador 4': randint(1, 6)
        }

print(jogador)
print('Valores sorteados: ')
for k, v in jogador.items():
    print(f'O {k} tirou o número: {v}')
    sleep(1)

{'jogador 1': 1, 'jogador 2': 5, 'jogador 3': 6, 'jogador 4': 4}
Valores sorteados:
O jogador 1 tirou o número: 1
O jogador 2 tirou o número: 5
O jogador 3 tirou o número: 6
O jogador 4 tirou o número: 4
```

Caso queira saber quem foi o jogador que tirou o maior número nos dados, precisamos criar um outro dicionário para colocar esse em ordem. E também será necessário importar o operador itemgetter.

```
from random import randint
from time import sleep
from operator import itemgetter

jogador = {'jogador 1': randint(1, 6),
            'jogador 2': randint(1, 6),
            'jogador 3': randint(1, 6),
            'jogador 4': randint(1, 6)
        }

#dicionário ranking para criar ordem em jogadores:
ranking = {}
```

Após a importação do operador e a criação da lista secundária que servirá de ranking para ordenar a principal, vamos colocar a ordem agora:

O operador itemgetter vai selecionar a posição (1) para ordenar. Pois se seu quisesse a posição zero e ordenasse pela chave(que no caso é o nome do jogador) eu teria que colocar que é itemgetter(0), mas nesse caso vai ser ordenado pelo valor do dado que está na (1).

```
print(jogador)
print('Valores sorteados: ')
for k, v in jogador.items():
    print(f'O {k} tirou o número: {v}')
    sleep(1)

ranking = sorted(jogador.items(), key=itemgetter(1))
```

Repare, porém, que o resultado dessa forma, vai colocar o valor maior por último. O que de certa forma é o correto na ordem, mas para esse exemplo, precisamos declarar o vencedor que é quem tirou o maior número nos dados.

```
0 jogador 2 tirou o número: 5
0 jogador 3 tirou o número: 3
0 jogador 4 tirou o número: 1

[('jogador 1', 1), ('jogador 4', 1), ('jogador 3', 3), ('jogador 2', 5)]

Process finished with exit code 0
```

Para isso, basta adicionar o “`reverse=True`”

```
print('='* 20)
ranking = sorted(jogador.items(), key=itemgetter(1), reverse=True)
print(ranking)
```

E aí o vencedor, quem tirou o maior valor nos dados, é alinhado em primeiro lugar:

```
0 jogador 3 tirou o numero: 2
0 jogador 4 tirou o número: 1

[('jogador 2', 6), ('jogador 1', 2), ('jogador 3', 2), ('jogador 4', 1)]
```

Mas, repare que na verdade, o resultado, é uma LISTA, não um dicionário, tanto é que o ranking poderia ser iniciado como `ranking = []` que é uma lista ao invés de ser iniciado como um dicionário `ranking = {}`

Para imprimir esse resultado:

```
print('=-'* 25)
#percorre a lista ranking, o enumerate serve para o índice
#o valor serve para encontrar os valores de cada posição na lista do ranking
for indice, valor in enumerate(ranking):
    print(f'º {indice + 1}° lugar foi para: {valor[0]} que tirou {valor[1]} nos dados')
```

```
=====
0 1º lugar foi para: jogador 3 que tirou 6 nos dados
0 2º lugar foi para: jogador 1 que tirou 3 nos dados
0 3º lugar foi para: jogador 2 que tirou 2 nos dados
0 4º lugar foi para: jogador 4 que tirou 1 nos dados
```

```
Process finished with exit code 0
```

## Funções

Uma função é uma rotina, algo que é realizado constantemente. Exemplos de funções ‘built-in’: input(), len(), int(), float(). Ela é uma função pré-existente, não precisa ser escrita, mas, algumas não nos satisfaz totalmente. Por exemplo, em muitos exercícios anteriores, imprimíamos uma linha na tela para sinalizar uma mudança ou alguma ordem. Com o print('-' \* 30). Podemos criar uma função com o nome: mostraLinha():.

Por exemplo, abaixo, podemos analisar que em várias linhas, existem inúmeras repetições, o código não está DRY. O print('') acontece 6 vezes iguais, como otimizar para que seja escrito apenas uma única vez?

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

Figura 0-1. Curso em Vídeo Aula 20# Funções

### Def

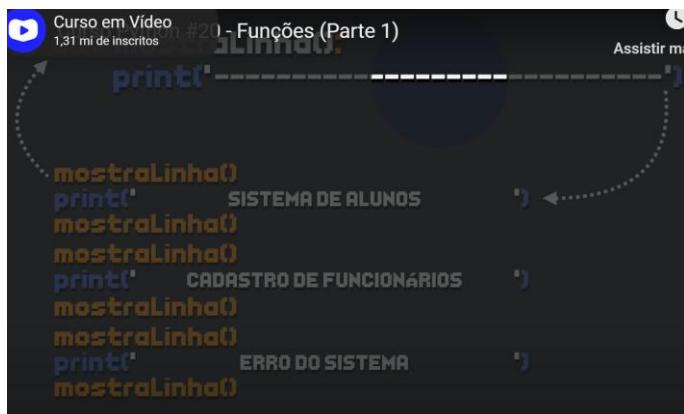
Def é a palavra utilizada para criar uma função personalizada. Para otimizar, utilizamos a definição, o def. Todas as funções em Python são identificadas pelo '( )' após o final do nome, com um deslocamento igual a um laço de repetição.

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

Figura 0-2. Curso em Vídeo Aula 20# Funções

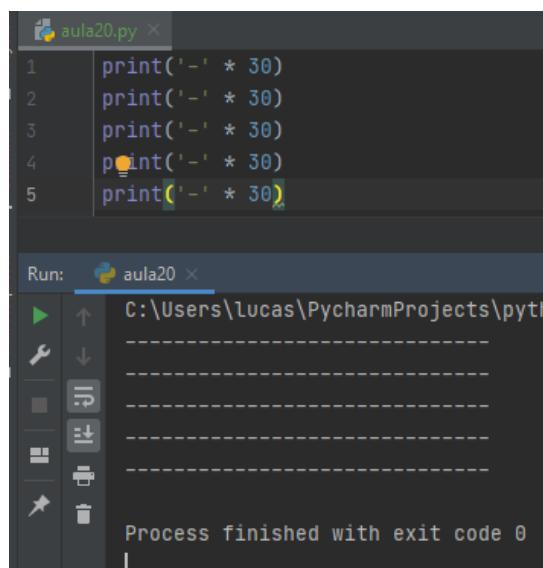
```
mostraLinha()
print('      SISTEMA DE ALUNOS      ')
mostraLinha()
mostraLinha()
print('      CADASTRO DE FUNCIONÁRIOS      ')
mostraLinha()
mostraLinha()
print('      ERRO DO SISTEMA      ')
mostraLinha()
```

Toda vez que for necessário ter uma linha separando algum código, basta chamarmos a função ‘mostraLinha():’ que foi definida lá em cima. E assim o código ficará muito mais limpo.



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

Isso é um código muito ruim:



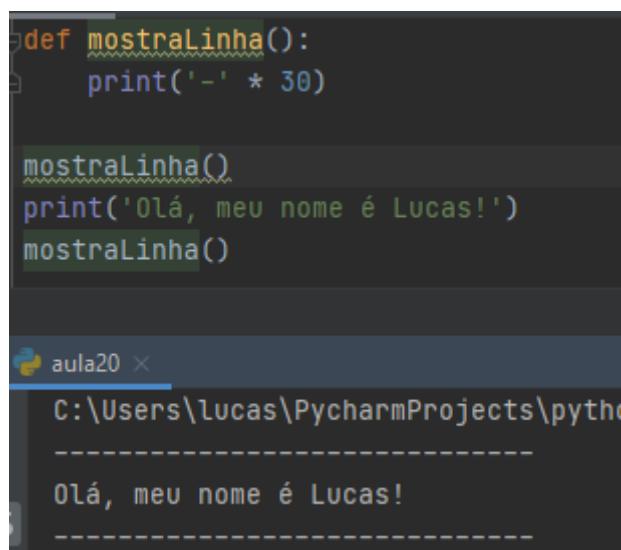
```
aula20.py
```

```
1 print('-' * 30)
2 print('-' * 30)
3 print('-' * 30)
4 print('-' * 30)
5 print('-' * 30)
```

Run: aula20

```
C:\Users\lucas\PycharmProjects\pytho
-----
-----
-----
-----
-----
Process finished with exit code 0
```

Esse é um exemplo criando uma função da forma correta:



```
def mostraLinha():
    print('-' * 30)

mostraLinha()
print('Olá, meu nome é Lucas!')
mostraLinha()
```

aula20

```
C:\Users\lucas\PycharmProjects\pytho
-----
-----
Olá, meu nome é Lucas!
-----
```

## Estética do Python no Def

Algo interessante é que seguindo a estética do Python, após a definição da função, o Python pede pelo menos 2 linhas que separam as definições do Programa Principal.

### Def com Parâmetros

O def é uma função muito poderosa no Python, pois com ela é possível aceitar parâmetros também. Repare que como abaixo, os códigos se repetem com frequência, alterando apenas o “miolo” que é a linha do meio.

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

Na parte das definições das funções, é possível passar um parâmetro de uma função definida por exemplo:

```
'def mensagem(parâmetro MSG aqui):'
    print()
    print(parâmetro que MSG aqui)'
```

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

E quando o código for escrito durante o Programa Principal, basta apenas acionar o parâmetro que foi passado dentro da função mensagem:

```
mensagem( 'SISTEMA DE ALUNOS' )
```

Em palavras, eu defini a função mensagem que receberá um parâmetro msg. E dentro dessa função, definirei também qual será a rotina que ela fará, como por exemplo, imprimir linhas na tela, mas em uma linha específica, ela receberá o próprio parâmetro e esse parâmetro será invocado somente quando o código principal acontecer.

The diagram illustrates the execution flow. It starts with the main code block at the bottom, which contains a call to the `mensagem` function with the argument `'Sistema de Alunos'`. A yellow arrow points from this call up to the `mensagem` function definition in the code editor. Inside the function definition, another yellow arrow points from the parameter `msg` to the line `print(msg)`, indicating that the parameter is being used within the function body. The code editor shows the function definition and the main call.

```
def mensagem(msg):
    print('-' * 30)
    print(msg)
    print('-' * 30)

# Código Principal rolando:
mensagem('Sistema de Alunos')
```

The screenshot shows the PyCharm interface with a terminal window titled "aula20". The terminal displays the output of the Python script. It starts with the path "C:\Users\lucas\PycharmProjects\python", followed by a dashed line, then the text "Sistema de Alunos", another dashed line, and finally "Process finished with exit code 0". Below the terminal, the Python code is shown again, with the first two lines executed and the third line partially visible.

```
# Código Principal rolando:
mensagem('Sistema de Alunos')
mensagem('Posso chamar o parâmetro quantas vezes eu quiser')
```

```
aula20 ×
C:\Users\lucas\PycharmProjects\python
-----
Sistema de Alunos
-----
Process finished with exit code 0
```

## Na prática

### Código Repetitivo e não DRY

Repare quantas vezes foi necessário reescrever essas variáveis que tem “elementos em comuns”.

```
a = 5
b = 4
s = a + b
print(s)
a = 9
b = 8
s = a + b
print(s)|
```

```
a = 2
b = 1
s = a + b
print(s)
```

```
aula20 ×
C:\Users\lucas\PycharmProject
9
17
3
```

Como fazer um jeito mais fácil, como abaixo, por exemplo, como seria a ‘def’?

```
soma(4, 5)
soma(9, 8)
soma(2, 1)
```

Como fazer esses valores serem parâmetros para uma função criada? Lê: defina uma função soma que receberá dois valores/**parâmetros**, sendo ‘a’ e ‘b’, e dentro dessa função, aplique a ‘soma’ a soma de ‘a’ + ‘b’.

```
def soma(a, b):
    soma = a + b
    print(soma)

#Programa Principal
soma(5, 5)
soma(10, 8)
soma(20, 1)
```

```
aula20 ×
C:\Users\lucas\PycharmProject
10
18
21
```

## Erro nos Parâmetros

Como a função soma acima necessita de dois parâmetros para fazer a operação aritmética de soma, caso eu informe somente um único parâmetro, o Python apontará logo o erro ao rodar o programa, veja:

The screenshot shows a PyCharm interface. In the code editor, there is a Python script named 'aula20.py' with the following content:

```
def soma(a, b):
    soma = a + b
    print(soma)

#Programa Principal
soma(5, 5)
soma(10, 8)
soma(20, 1)
soma(4) #passando apenas 1 parâmetro.
```

In the terminal window, titled 'aula20', the output is:

```
10
18
21
Traceback (most recent call last):
  File "C:/Users/lucas/PycharmProjects/pythonProject/aula20.py", line 30, in <module>
    soma(4) #passando apenas 1 parâmetro.
TypeError: soma() missing 1 required positional argument: 'b'
```

Para corrigir, poderia adicionar um soma (4, 1) e aí estaria correto.

## “Invertendo” os Parâmetros

Podemos “Inverter” a ordem dos parâmetros, desde que eles sejam devidamente explícitos.

```
def soma(a, b):
    print(f'A soma de A = {a} e a soma de B = {b}')
    soma = a + b
    print(f'A soma de A + B = {soma}')

#Programa Principal
soma(5, 5)
soma(10, 8)
soma(20, 1)
#soma(4) #passando apenas 1 parâmetro.'''
soma(a=4, b=5)
soma(b=10, a=20)
```

```
aula20 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
A soma de A = 4 e a soma de B = 5
A soma de A + B = 9
A soma de A = 20 e a soma de B = 10
A soma de A + B = 30
```

## Erro na “inversão” dos parâmetros

O Python não é inteligente o suficiente para que se um valor seja invertido e seja explicitado mas o outro não, isso gerará um erro, é necessário explicitar os dois, caso ocorra a inversão.

```
def soma(a, b):
    print(f'A soma de A = {a} e a soma de B = {b}')
    soma = a + b
    print(f'A soma de A + B = {soma}')

#Programa Principal
soma(5, 5)
soma(10, 8)
soma(20, 1)
#soma(4) #passando apenas 1 parâmetro.'''
#soma(a=4, b=5)
soma(b=10, 20)

aula20 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:/Users/lucas/PycharmProjects/pythonProject/aula20.py"
soma(b=10, 20)
^
SyntaxError: positional argument follows keyword argument
```

## Empacotar funções em Tuplas

Suponha que uma função chamada contador() que está passando 5 números e numa próxima invocação, ela passa 3 números. Em muitas linguagens isso não é possível, mas em Python, como uma linguagem moderna, ela permite.

```
def contador(* num):
    print(f'{num}')
```

Para isso, utilizamos dentro da definição da função, no parâmetro um asterisco (\*), por exemplo abaixo, o (\*) significa que o usuário irá passar vários parâmetros, não se sabe ao certo quantos, mas que serão vários e todos os parâmetros que o usuário digitar, coloca tudo isso dentro do 'núm'. O resultado será uma TUPLA, sendo assim, podemos realizar manipulação, como se manipula uma tupla.

```
def contador(* num):
    print(f'{num}')

contador(2, 1, 7)
contador(5, 4, 1, 3, 10, 11, 15)
contador(1)

contador()
aula20
```

C:\Users\lucas\PycharmProjects\pythonProject\

0 números são: (2, 1, 7).  
0 números são: (5, 4, 1, 3, 10, 11, 15).  
0 números são: (1,).

Process finished with exit code 0

```
def contador(* num):
    for valor in num:
        print(f'{valor}', end=' ')
    print('FIM')

contador(2, 1, 7)
contador(5, 4, 1, 3, 10, 11, 15)
contador() > for valor in num
aula20
```

C:\Users\lucas\PycharmProjects\pythonProject\

2 1 7 FIM  
5 4 1 3 10 11 15 FIM  
1 FIM

## Length para saber o tamanho do Pacote

Para podermos saber o tamanho do pacote que foi gerado, podemos por exemplo, criar uma variável que receberá o len(desse parâmetro que receberá os parâmetros).

```
def contador(* num):
    tamanho = len(num)
    print(f'Recebi os números, sendo {num}, e são ao todos {tamanho} números.')

contador(2, 1, 7)
contador(5, 4, 1, 3, 10, 11, 15)
contador(1)

contador()
aula20 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/pythonProject/aula20.py
Recebi os números, sendo (2, 1, 7), e são ao todos 3 números.
Recebi os números, sendo (5, 4, 1, 3, 10, 11, 15), e são ao todos 7 números.
Recebi os números, sendo (1,), e são ao todos 1 números.
```

## Ponto de Atenção das Tuplas nos Pacotes de Funções

Algo a ressaltar, é que como até então, todos os dados foram gerados em forma de tuplas, vale lembrar que elas são imutáveis, ou seja, os valores ali presentes não podem ser alterados, mas há uma forma de alterar.

## Empacotamento de Funções nas Listas

Vamos pegar como exemplo que uma determinada lista criada, precisa ter os números dos seus índices dobrados, por exemplo um '7' vira '14', um '2' vira '4' e por aí vai.

```
valores = [7, 2, 5, 0, 4]
print(valores)
```

valores				
7	2	5	0	4
0	1	2	3	4

Para isso, vamos criar uma função chamado “dobra()”.

```
valores = [7, 2, 5, 0, 4]
dobra(valores)
print(valores)
```

Vamos analisar a função dobra() abaixo.

Foi definida uma função dobra que recebeu o parâmetro lista(uma variável lst), definiu-se uma variável = 0 para controle do While. Enquanto a posição for menor do que o tamanho da lista, então, a lista em sua posição atual, (por exemplo a lista começa na posição zero, então seguindo o exemplo acima) a lista pegará o '7' e multiplicará por 2. Dado isso, voltamos ao contador para somar e sair do Loop quando a posição não for maior que o tamanho da lista (quando ela já for totalmente percorrida).

```
CURSO PYTHON #20 - Funções (Parte 1)
def dobrar(lst):
    pos = 0
    while pos < len(lst):
        lst[pos] *= 2
        pos += 1
```

```
def dobrar(lst):
    pos = 0
    while pos < len(lst):
        lst[pos] *= 2
        pos += 1
    print(valores)

valores = [6, 3, 9, 1, 0, 2]
dobra(valores)

aula20 ✘
C:\Users\lucas\PycharmProjects\python
[12, 6, 18, 2, 0, 4]

Process finished with exit code 0
```

## Desempacotamento

Desempacotamento é quando o parâmetro passado para a função pode ter várias conformes ao que a necessidade da chamada dessa função. Por exemplo, uma hora eu chamo a função com 2 parâmetros, numa hora com 1, em outra com 20 parâmetros. Desempacotamento é quando o número de parâmetros é (\*) variado.

```
def soma (* valores):
    soma = 0
    for numero in valores:
        soma += numero
    print(f'Somando os valores {valores}, teremos a soma de {soma}')
```

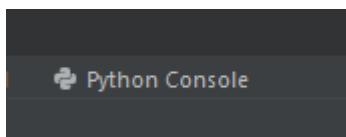
## Funções – Parte 2

Tópicos que serão abordados nesse capítulo: Interactive Help, Docstrings, Argumentos opcionais, Escopo de Variáveis, Funções com retorno de variáveis.

### Interactive Help

É uma função/método interna, é uma função/método “built-in”. A sua invocação é através da: `help()` e para sair do prompt do help, basta digitar “`quit`”.

Procure por Console Python



Ao digitar a função no console, o prompt é alterado:

```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)
>>> help()

Welcome to Python 3.9's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.9/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

>?
```

Essa função tem a função de dizer, informar qual e como as funções que estão na biblioteca do Python são e o que fazem. Vamos perguntar ao Python help o que é a função “`print()`”.

```
>? 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.
```

Vamos pedir ajuda para sabermos o que é a função len():

```
>? len
Help on built-in function len in module builtins:

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

Veja que diz que a função len() retorna o número de itens dentro do container.

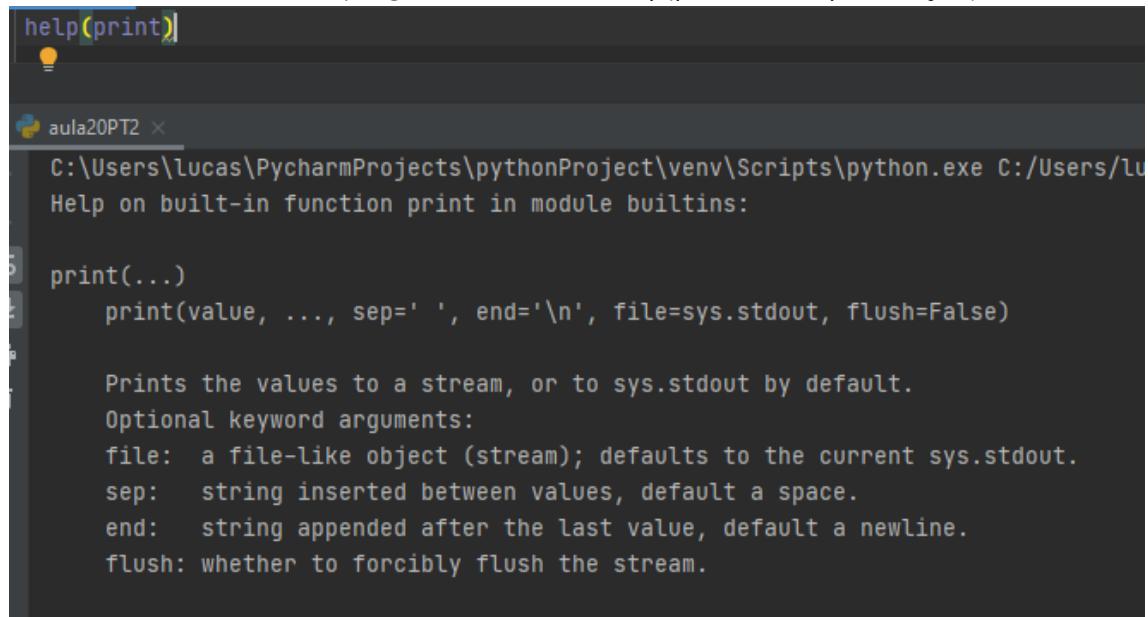
E não é só de função! Podemos perguntar sobre as bibliotecas, como a **Datetime** por exemplo:

```
builtins.object
    date
        datetime
    time
    timedelta
    tzinfo
        timezone

    class date(builtins.object)
        | date(year, month, day) --> date object
        |
        | Methods defined here:
        |
        |     __add__(self, value, /)
        |         Return self+value.

help>
```

Outra forma de acionar o Interactive Help sem ser pelo console do Python é através da chamada diretamente no programa usando o **help(parâmetro qual função)**.



```
help(print)

aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas
Help on built-in function print in module builtins:

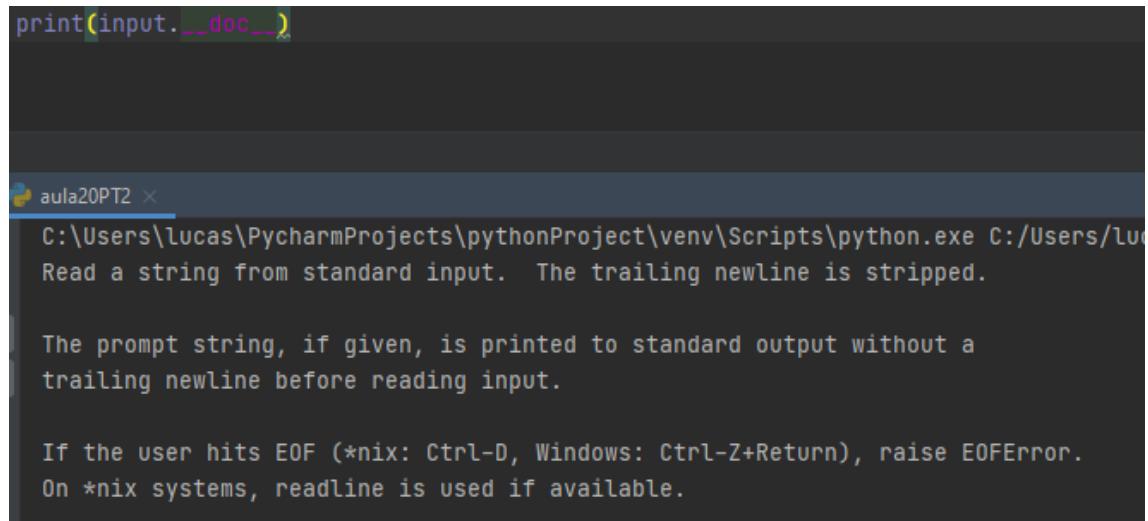
print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    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.
```

### Com Doc

Utilizando o **print(nomeDaFuncao.\_\_doc\_\_)**

Não necessariamente as informações que serão recebidas pelo doc são iguais as oferecidas pelo help().



```
print(input.__doc__)

aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas
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.
```

Quit

Saindo da Interactive Help

```
help> quit
```

```
You are now leaving help and returning to the Python interpreter.  
If you want to ask for help on a particular object directly from the  
interpreter, you can type "help(object)". Executing "help('string')"  
has the same effect as typing a particular string at the help> prompt.
```

## Docstring

Uma docstring, é uma string de documentação, igual a que foi vista nas páginas acima. Ou seja, é a documentação das funções criadas, tanto diretamente pelo Python, quanto por usuários comuns que desejam utilizar suas funções em bibliotecas. Quando invocamos a `help()`, o retorno é uma docstring, que é um manual do comando que eu quero entender através do `help`.

Por exemplo, podemos criar/definir uma função chamada contador que recebe 3 parâmetros. Fica muito claro para o desenvolvedor desse método qual é o retorno. Nesse quadro pontilhado fica disponível. Os parâmetros formais são os parâmetros que estão no ato da declaração da função. Enquanto os parâmetros reais, são os parâmetros que estão na invocação da função no decorrer do código.

```
def contador(i, f, p):
    c = i
    while c <= f:
        print(f'{c}', end=' ')
        c += p
    print('FIM')

contador(2, 10, 2)
```

É claro para o desenvolvedor o que essa função faz (descrita no quadro pontilhado laranja). Porém, e para quem não a escreveu e desenvolveu? Como saber o que os parâmetros recebem? O que ela e qual o seu retorno? Se o dev digitar o comando `'help(contador)'`, por causa do docstring, ele será capaz de ter todas essas informações.

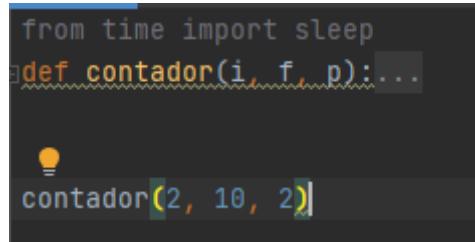
Note que podemos ver o que essa função faz, como ela foi definida, mas ainda assim, também, seria confuso sendo que não foi o dev que desenvolveu.

```
def contador(i, f, p):
    c = i
    while f >= c:
        print(f'{c} ', end='')
        c += p
    print('FIM')

|
contador(2, 10, 2)

aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonProject\
2 4 6 8 10 FIM
Process finished with exit code 0
```

Imagine que estamos utilizando biblioteca de terceiros.



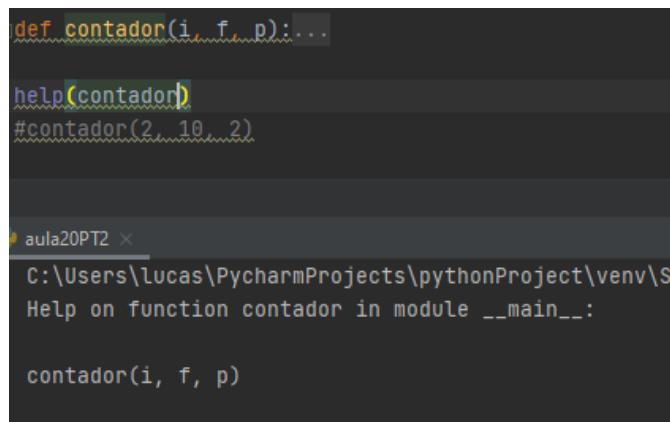
```
from time import sleep
def contador(i, f, p):...
```

A screenshot of a Python code editor (PyCharm) showing code completion. A tooltip with a lightbulb icon appears over the argument 'p' in the 'contador' function call. The tooltip displays the function signature 'def contador(i, f, p):...'.

```
contador(2, 10, 2)
```

Não é possível mais identificar como a função foi estruturada. Outra coisa, também não é possível como o método “sleep()” foi implementado na biblioteca Time, mas podemos utilizar o “interactive help” para entendermos, assim como, futuramente, utilizaremos para entender o “contador()”. Nos próximos capítulos mostraremos como importar a função contador através de uma biblioteca.

Se utilizarmos o “help(contador)”, nesse exato momento, não adiantará nada, o resultado não será informativo o suficiente. Só conseguimos saber que a função recebe 3 parâmetros.



```
def contador(i, f, p):...
```

```
help(contador)
#contador(2, 10, 2)
```

```
aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\S
Help on function contador in module __main__:

contador(i, f, p)
```

A screenshot of a Python code editor (PyCharm) showing the output of the 'help(contador)' command. The terminal window shows the following:

```
Help on function contador in module __main__:
```

```
contador(i, f, p)
```

## Iniciando uma Docstring

Para iniciar uma docstring, é necessário iniciar logo após a definição da função, através de aspas duplas 3 vezes.

```
def contador(i, f, p):
    """
    → Faz uma contagem e mostra na tela
    :param i: inicio da contagem
    :param f: fim da contagem
    :param p: passo da contagem
    :return: sem retorno
    """
    c = i
    while c <= f:
        print(f'{c}', end=' ')
        c += p
    print('FIM!')
```

O mais legal ainda sobre o Python é que ele oferece o que cada parâmetro recebe, o retorno...

The screenshot shows a PyCharm interface with a code editor and a terminal window. In the code editor, there is a Python function named `contador` with a multi-line docstring. Below the editor, the terminal window displays the output of the `help(contador)` command, which shows the function signature, its docstring, and the parameter descriptions from the docstring.

```
def contador(i, f, p):
    """
    → Faz uma contagem e mostra na tela
    :param i: inicio da contagem
    :param f: fim da contagem
    :param p: passo(salto) da contagem
    :return: sem retorno
    """
    c = i
    while f >= c:
        print(f'{c}', end=' ')
        c += p
    print('FIM!')

help(contador)
#contador(2, 10, 2)

contador()
aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/U
Help on function contador in module __main__:

contador(i, f, p)
    → Faz uma contagem e mostra o resultado na tela
    :param i: Início da contagem
    :param f: Fim da contagem
    :param p: Passo(salto) da contagem
    :return: Sem retorno
    Função criada por Lucas Galves assistindo Curso em Vídeo - Aula 21
```

## Parâmetros Opcionais

Se lembra sobre o desempacotamento dos parâmetros? Que caso fosse necessário somar 2 números uma hora, depois somar 8 números, no momento da declaração da função, utilizávamos o (\* número) para indicar que receberia diversos parâmetros conforme a necessidade. Pois os “Parâmetros Opcionais” são similares.

```
def somar(a, b, c):
    s = a + b + c
    print(f'A soma vale {s}')

somar(3, 2, 5)
somar(8, 4)
```

A primeira invocação da função “somar” não vai causar problema pois mandou 3 parâmetros para uma função que exige 3 parâmetros, porém, logo abaixo, chamando a função “somar” novamente, como só mandou 2 parâmetros, vai causar problema, caso no momento da declaração/definição da função, ela tenha sido feita obrigatoriamente exigindo 3 parâmetros(exceto pelo desempacotamento de funções).

Agora, podemos utilizar o conceito de parâmetro opcional para a variável “c” que está como parâmetro. Podemos defini-la como “c = 0” (C recebe zero). Ou seja, caso a variável “c” não seja enviada na invocação no decorrer do código, “c” vai assumir o valor zero.

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

somar(3, 2, 5)
somar(8, 4)
```

E nada impede que todos os outros parâmetros recebam parâmetros opcionais, a fim de evitar qualquer erro, por exemplo:

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

```
somar(3, 2, 5)
somar(8, 4)
```

Nesse caso, uma chamada de “somar()” sem parâmetro, funciona e vai retornar um valor igual a zero.

```
def somar(a, b, c):
    """
    Faz a soma de três valores e mostra o resultado na tela.
    :param a: Primeiro valor
    :param b: Segundo valor
    :param c: Terceiro valor
    Criado por Lucas Pereira Galves
    """
    soma = a + b + c
    print(f'O valor da soma é: {soma}')

somar(3, 2, 5)

somar()
```

aula20PT2 ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\py  
0 valor da soma é: 10

Note que se não passarmos um parâmetro opcional ou um desempacotamento de função, na invocação do método com somente dois parâmetros, causará erro.

```
def somar(a, b, c):
    """
    Faz a soma de três valores e mostra o resultado na tela.
    :param a: Primeiro valor
    :param b: Segundo valor
    :param c: Terceiro valor
    Criado por Lucas Pereira Galves
    """
    soma = a + b + c
    print(f'O valor da soma é: {soma}')

somar(3, 2, 5)
somar(3, 2)
```

aula20PT2 ×

```
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.
O valor da soma é: 10
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\pythonProject\aula20PT2.py", line 1, in <module>
    somar(3, 2)
TypeError: somar() missing 1 required positional argument: 'c'
```

```
def somar(a = 0, b = 0, c = 0):
    """
    Faz a soma de três valores e mostra o resultado na tela.
    :param a: Primeiro valor
    :param b: Segundo valor
    :param c: Terceiro valor
    Criado por Lucas Pereira Galves
    """
    soma = a + b + c
    print(f'O valor da soma é: {soma}')

somar(3, 2, 5)
somar(3, 2)

somar()
```

aula20PT2 ×

```
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\pyt
O valor da soma é: 10
O valor da soma é: 5
```

## Erro com Parâmetros Opcionais (Mais parâmetros)

Mas, tem um grande problema. Veja que essa solução serve somente se o usuário digitar menos parâmetros que o necessário na função. Mas e se o usuário digitar mais parâmetros que a função exige? Resposta: Vai causar um erro, e o único jeito de corrigir é utilizando o conceito de parâmetros múltiplos/desempacotamento, por exemplo: `def somar (* valores):`

The screenshot shows the PyCharm interface. In the top editor window, the code for the `somar` function is displayed. It includes docstrings and three calls to the function with different argument lists. In the bottom "Run" tab, the output of the first two calls is shown as expected (0 and 5). The third call, `somar(3, 2, 5, 10)`, results in a `TypeError`: `somar() takes from 0 to 3 positional arguments but 4 were given`.

```
def somar(* valores):
    """
    Faz a soma de três valores e mostra o resultado na tela.
    :param a: Primeiro valor
    :param b: Segundo valor
    :param c: Terceiro valor
    Criado por Lucas Pereira Galves
    """
    soma = 0
    print(f'O valor da soma é: {soma}')

somar(3, 2, 5)
somar(3, 2)
somar(3, 2, 5, 10)

Run: aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/Pychar
0 valor da soma é: 10
0 valor da soma é: 5
Traceback (most recent call last):
File "C:/Users/lucas/PycharmProjects/pythonProject/aula20PT2.py", line 35, in <module>
    somar(3, 2, 5, 10)
TypeError: somar() takes from 0 to 3 positional arguments but 4 were given
```

Note que na linha 35, o erro acontece, pois foram enviados muitos parâmetros a mais que a função exige e nem o parâmetro opcional consegue resolver (mesmo que seja um único valor a mais passado). Foi necessário criar um laço de repetição, muito mais complexo.

The screenshot shows the PyCharm interface again. The code has been modified to use a `for` loop instead of multiple parameters. The function now correctly handles any number of arguments. The execution in the "Run" tab shows the function working as intended for all four test cases.

```
def somar(* valores):
    """
    Faz a soma de três valores e mostra o resultado na tela.
    :param a: Primeiro valor
    :param b: Segundo valor
    :param c: Terceiro valor
    Criado por Lucas Pereira Galves
    """
    soma = 0
    for numero in valores:
        soma += numero
    print(f'O valor da soma é: {soma}')

somar(3, 2, 5)
somar(3, 2)
somar(3, 2, 5, 10)

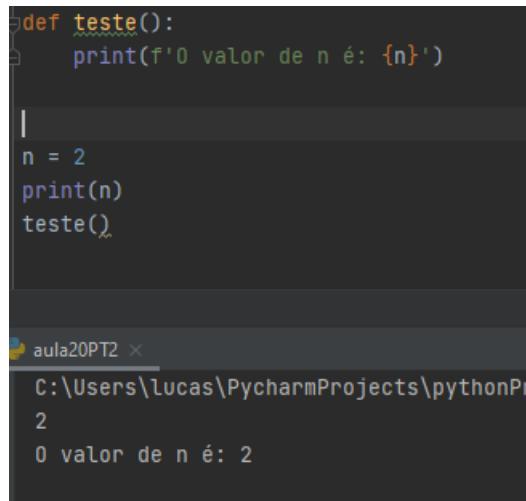
somar()

Run: aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/Pychar
0 valor da soma é: 10
0 valor da soma é: 5
0 valor da soma é: 20
```

## Escopo de Variáveis

### Escopo Global

Quando uma variável é definida, dependendo do momento que ela é definida dentro do código e do programa principal, essa definição pode interferir qual valor ela apresentará no programa. No caso abaixo, como a variável n foi iniciada antes da chamada da função “teste()”, quando teste foi invocado, ele atribui “n” o valor 2.

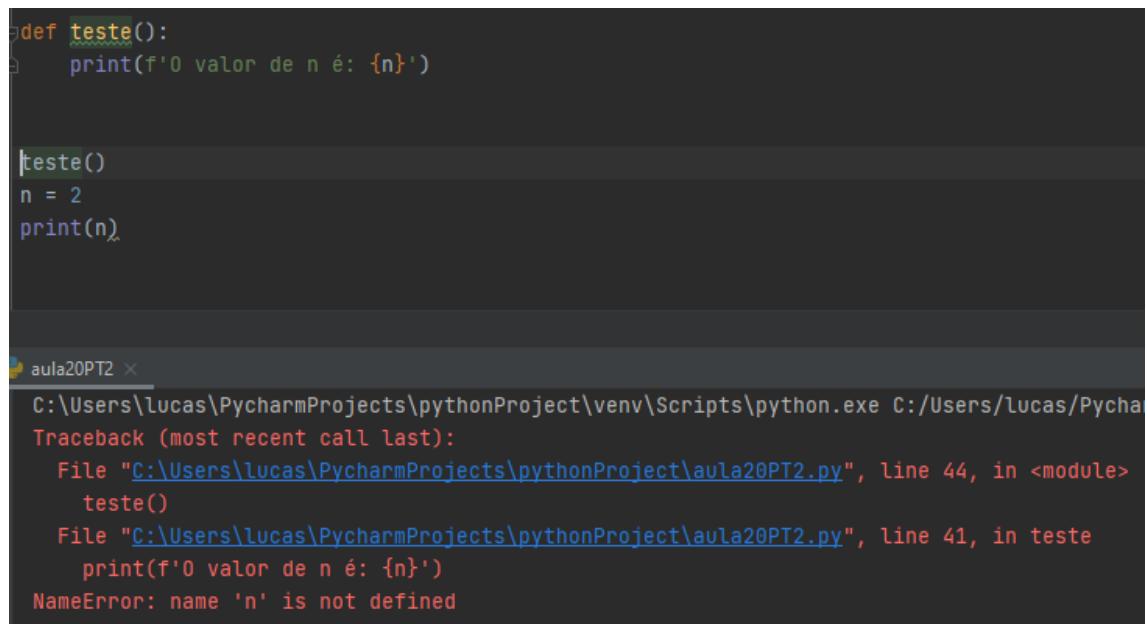


```
def teste():
    print(f'O valor de n é: {n}')

n = 2
print(n)
teste()
```

```
aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonPr
2
0 valor de n é: 2
```

Em contrapartida, nesse cenário, não atribuiu nenhum valor e ainda por cima como a variável não foi iniciada, a função “teste()” apresenta um erro:



```
def teste():
    print(f'O valor de n é: {n}')

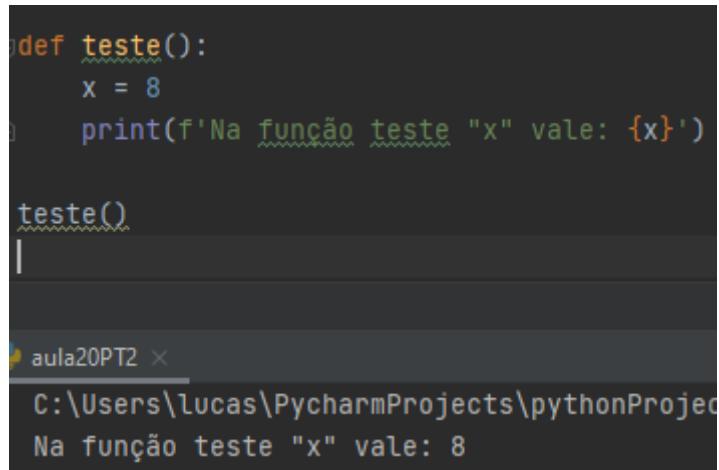
teste()
n = 2
print(n)
```

```
aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/Pycha
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\pythonProject\aula20PT2.py", line 44, in <module>
    teste()
  File "C:\Users\lucas\PycharmProjects\pythonProject\aula20PT2.py", line 41, in teste
    print(f'O valor de n é: {n}')
NameError: name 'n' is not defined
```

## Escopo Local

Se uma variável foi criada dentro de uma função, ela vai ter somente o escopo dentro daquela função, se durante a execução do programa principal ela tentar ser usada, vai apresentar um erro também, note:

Dessa forma, “x” somente foi declarado dentro da função “teste()”.

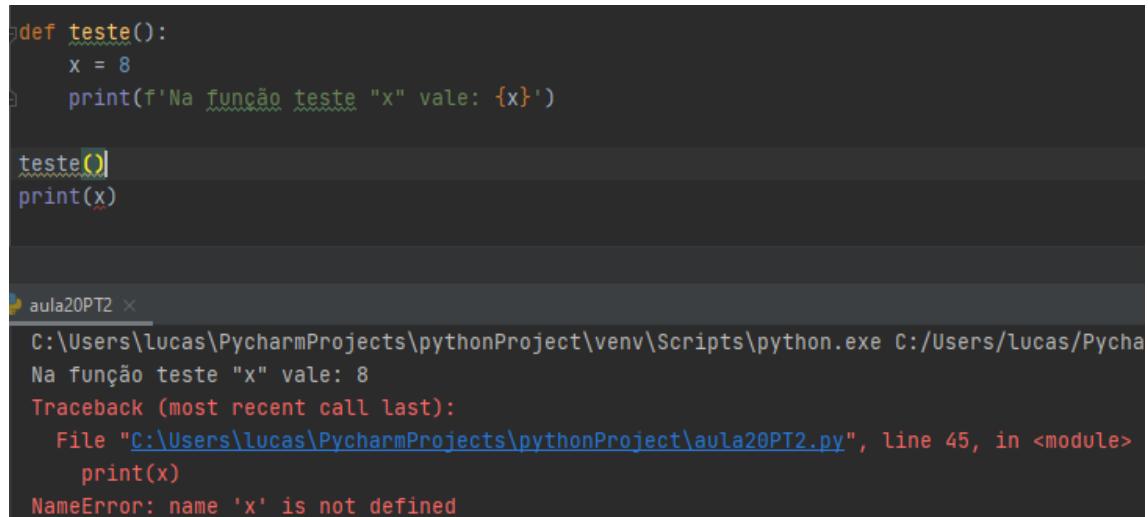


```
def teste():
    x = 8
    print(f'Na função teste "x" vale: {x}')

teste()
|
```

aula20PT2 ×  
C:\Users\lucas\PycharmProjects\pythonProject  
Na função teste "x" vale: 8

Se ele tentar ser impresso no programa principal, sem que a função “teste()” seja invocada, vai causar o erro:

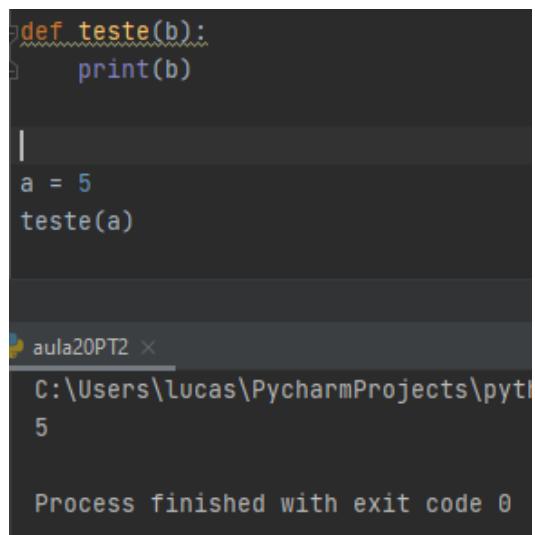


```
def teste():
    x = 8
    print(f'Na função teste "x" vale: {x}')

teste()
print(x)

aula20PT2 ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/Pycha
Na função teste "x" vale: 8
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\pythonProject\aula20PT2.py", line 45, in <module>
    print(x)
NameError: name 'x' is not defined
```

Apenas para complementar os estudos, caso crie uma função, por exemplo: teste(b), e durante o programa principal na hora de passar o parâmetro, não necessariamente vai passar um “b” é apenas para a passagem do parâmetro um valor.



The screenshot shows a PyCharm interface. The code editor contains the following Python script:

```
def teste(b):
    print(b)

|
a = 5
teste(a)
```

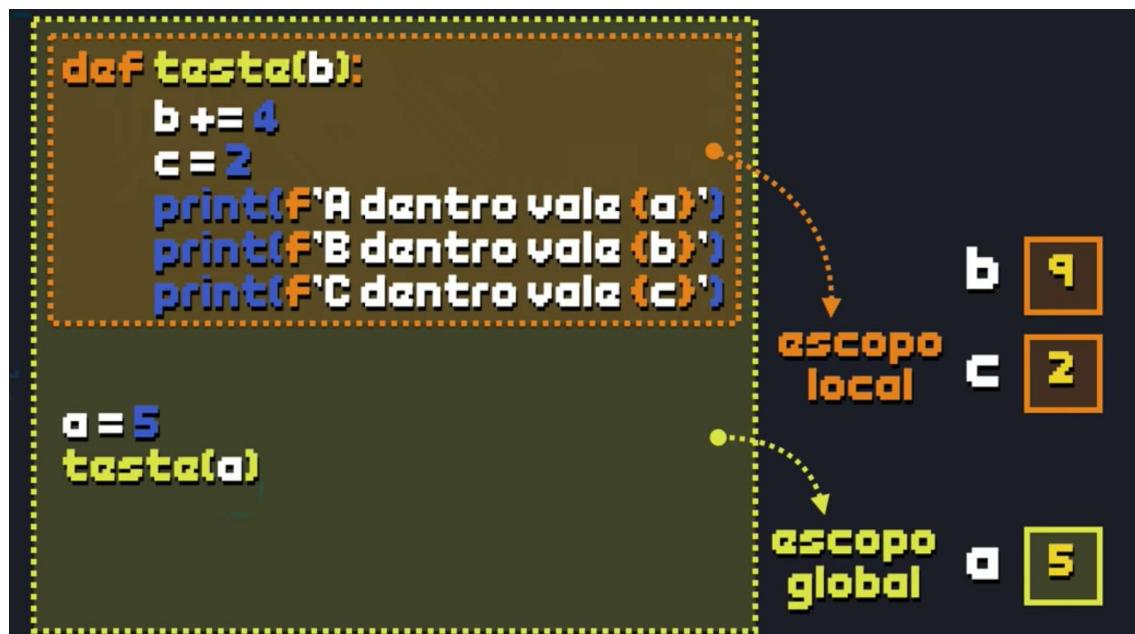
The terminal window below it shows the output of running the script:

```
aula20PT2 ×
C:\Users\Lucas\PycharmProjects\pyth
5

Process finished with exit code 0
```

## Escopo Global e Escopo Local

No primeiro cenário, ao invocar a função teste(), todos os valores seriam printados na tela, pois existe variável local e global. A mostraria 5, B mostraria 9 e C mostraria 2.



Porém, ao tentarmos chamar as variáveis B e C no programa principal, elas não serão printadas, pois são variáveis locais (da função teste()) e portanto, não podem ser chamadas como se fossem globais.



## Um ponto de Atenção sobre Local e Global

Todo problema começa num momento que se queira criar uma variável “A” que já existe no escopo global, só que dentro da função teste(), se tornando uma variável local. Nesse caso, existirão duas variáveis “A” no programa, uma com o escopo global e outra com o escopo local.

Dentro da função, se “A” for iniciada dentro da função, ela se tornará de escopo local, não afetando a variável que tem o escopo global.



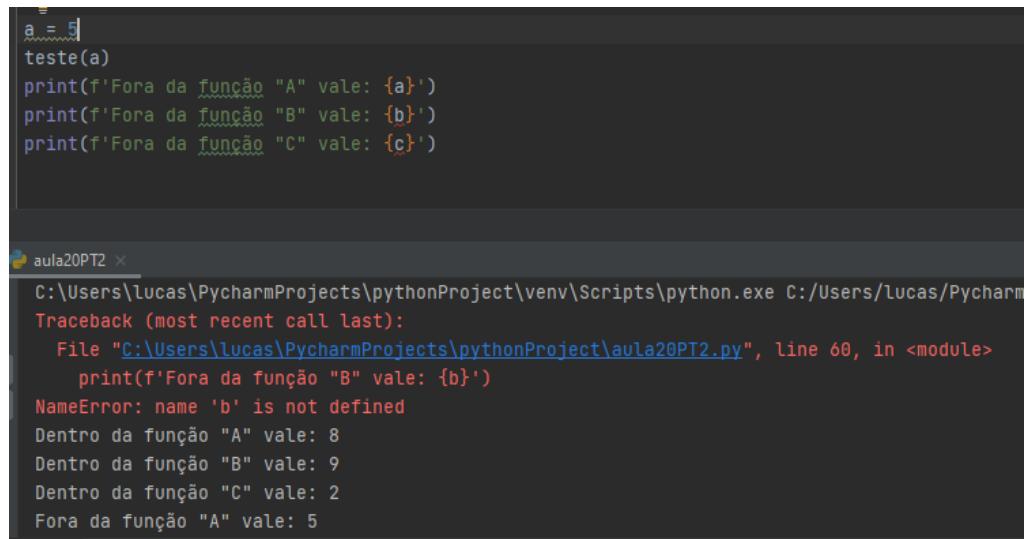
```
def teste(b):
    a = 8
    b += 4
    c = 2
    print(f'Dentro da função "A" vale: {a}')
    print(f'Dentro da função "B" vale: {b}')
    print(f'Dentro da função "C" vale: {c}')

a = 5
teste(a)
print(f'Fora da função "A" vale: {a}')

Process finished with exit code 0
```

aula20PT2 ×  
C:\Users\lucas\PycharmProjects\pythonProject\ve  
Dentro da função "A" vale: 8  
Dentro da função "B" vale: 9  
Dentro da função "C" vale: 2  
Fora da função "A" vale: 5

E se tentar printar as variáveis “B” e “C” que foram definidas somente no escopo local, vai causar erro:



The screenshot shows a PyCharm interface. In the top editor pane, there is a single line of code: `a = 5`. Below it, another editor pane contains the following Python script:

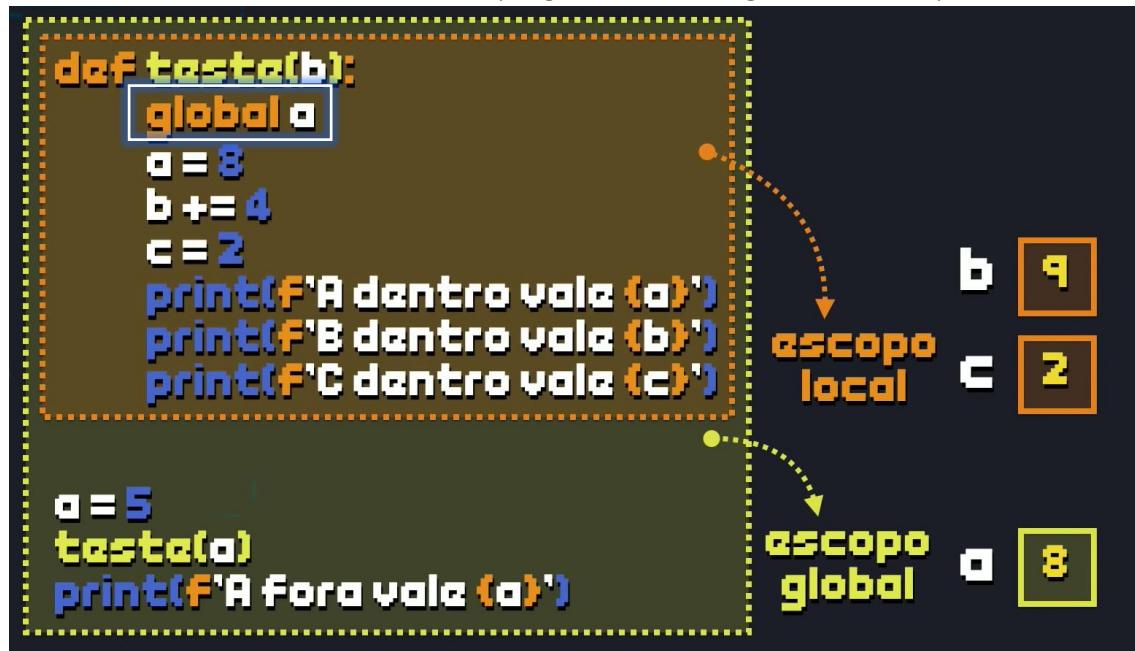
```
a = 5
teste(a)
print(f'Fora da função "A" vale: {a}')
print(f'Fora da função "B" vale: {b}')
print(f'Fora da função "C" vale: {c}')
```

In the bottom terminal window, the script is run, and the output is:

```
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/pythonProject/aula20PT2.py
Traceback (most recent call last):
  File "C:/Users/lucas/PycharmProjects/pythonProject/aula20PT2.py", line 60, in <module>
    print(f'Fora da função "B" vale: {b}')
NameError: name 'b' is not defined
Dentro da função "A" vale: 8
Dentro da função "B" vale: 9
Dentro da função "C" vale: 2
Fora da função "A" vale: 5
```

### Variável Global aplicada a Função (na local)

Para fazer que uma variável global seja aplicada a variável local, podemos utilizar o “global” dentro da função. Isso significa que qualquer alteração da variável “a” na função vai impactar em todas as chamadas de “A” no escopo global. Ou seja “a” não vale mais 5 em nenhum momento do programa, nem no global, ela sempre valerá 8



```
def teste(b):
    global a
    a = 8
    b += 4
    c = 2
    print(f'Dentro da função "A" vale: {a}')
    print(f'Dentro da função "B" vale: {b}')
    print(f'Dentro da função "C" vale: {c}')

a = 5
teste(a)
print(f'Fora da função "A" vale: {a}')
```

```
aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonProject\
Dentro da função "A" vale: 8
Dentro da função "B" vale: 9
Dentro da função "C" vale: 2
Fora da função "A" vale: 8
```

## **Escopo de Biblioteca Global e Local**

Também existe a possibilidade do tratamento de bibliotecas em escopo global e local.  
Somente pelas explicações nos exercícios,

VOLTAR MAIS TARDE AQUI E COLOCAR A EXPLICAÇÃO PROVENINENTE DOS EXERCÍCIOS.

## Retorno de valores

As funções/métodos em Python, podem retornar dentro delas e podem não retornar. Utilizando a palavra reservada “return”.

Nos códigos abaixo, podemos notar que existe um print, uma impressão na tela.

Eles escrevem o resultado da mesma maneira, porque a função foi chamada várias vezes e tem dentro dela a função de printar.

The screenshot shows a PyCharm interface with a code editor and a terminal window. The code editor contains a Python script named 'aula20PT2.py'. It defines a function 'somar' with three parameters (a, b, c) all set to 0 by default. Inside the function, it calculates the sum of a, b, and c and prints the result using an f-string. Below the function definition, there are three calls to 'somar': 'somar(3, 2, 5)', 'somar(2, 2)', and 'somar(6)'. The terminal window shows the output of these calls: 'A soma vale: 10.', 'A soma vale: 4.', and 'A soma vale: 6.' respectively.

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

somar(3, 2, 5)
somar(2, 2)
somar(6)

aula20PT2 ×
C:\Users\lucas\PycharmProjects\python
A soma vale: 10.
A soma vale: 4.
A soma vale: 6.
```

Note que para cada vez que a função é chamada, ela printa “A soma vale: tanto”. Mas, e se o desejo for “As somam valem: tantos”?

## Utilizando o Return

A ideia de se utilizar o return no lugar do print, é que quando utilizamos o return, ele apenas retorna um valor, sem mostrar nada na tela. Portanto, eu posso utilizar esse retorno da forma que mais eu quiser que seja impressa na tela. Por exemplo, se a função somar() somente retornar um valor que foi a soma em forma de variável 's'. Isso significa que, se atribuir uma variável r1 ou r2 ou r3, e cada uma delas realizar a chamada da função somar(), eles vão receber apenas o retorno do valor da soma e nunca um print direto da função. Podendo assim, utilizarmos um print formatado conforme a nossa vontade.

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

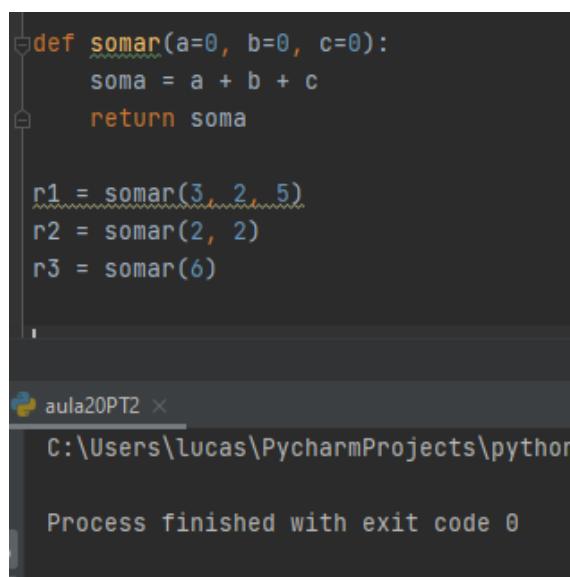
resp = somar(3, 2, 5)
```

```
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}.')
```

Veja abaixo que não teve nenhuma informação printada na tela



The screenshot shows a PyCharm interface with a code editor and a terminal window. The code in the editor is:

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

r1 = somar(3, 2, 5)
r2 = somar(2, 2)
r3 = somar(6)

.

aula20PT2 ×
C:\Users\lucas\PycharmProjects\python

Process finished with exit code 0
```

Se desejar que os valores das variáveis r1, r2 e r3 sejam mostrados na tela, ai sim, é necessário realizar um print.

The screenshot shows a PyCharm interface with a code editor and a terminal window. The code editor contains the following Python script:

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

r1 = somar(3, 2, 5)
r2 = somar(2, 2)
r3 = somar(6)

print(f'Os valores dos retornos são: {r1}, {r2}, {r3}')
```

The terminal window shows the output of running the script:

```
aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\py
Os valores dos retornos são: 10, 4, 6

Process finished with exit code 0
```

## Na prática

```
#0 factorial precisa receber um parâmetro de um número,
# e se não receber, será o valor 1 e se tornou um
# parâmetro opcional(para não valer zero)
def factorial(num=1):
    f = 1 #Local
    for c in range(num, 0, -1):
        f *= c
    return f

n = int(input('Digite um número: '))
print(f'O factorial de {n} é igual a {factorial(n)}')

factorial()
aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python
Digite um número: 5
O factorial de 5 é igual a 120

Process finished with exit code 0
```

```
def factorial(num=1):
    f = 1 #Local
    for c in range(num, 0, -1):
        f *= c
    return f

f1 = factorial(5)
f2 = factorial(4)
f3 = factorial()

print(f'Os retornos são: {f1}, {f2}, {f3}.')
```

```
aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python
Os retornos são: 120, 24, 1.

Process finished with exit code 0
```

## Retorno com valor Boolean

Podemos retornar qualquer tipo no Python, strings, inteiros, floats, booleanos, listas, dicionários, tuplas...

```
def par(valor=0):
    if valor % 2 == 0:
        return True
    else:
        return False

num = int(input('Digite um número: '))
if par(num):
    print(f'O número {num} é par.')
else:
    print(f'O número {num} é ímpar.')

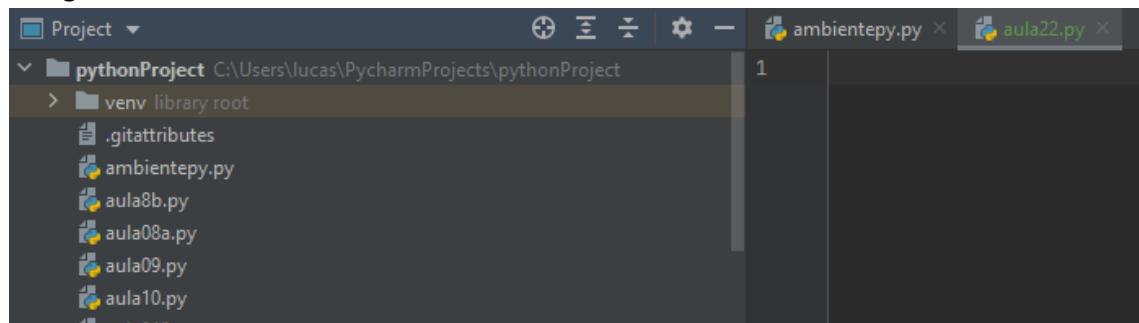
if par(num)
aula20PT2 ×
C:\Users\lucas\PycharmProjects\pythonProj
Digite um número: 6
O número 6 é par.
```

## Módulos

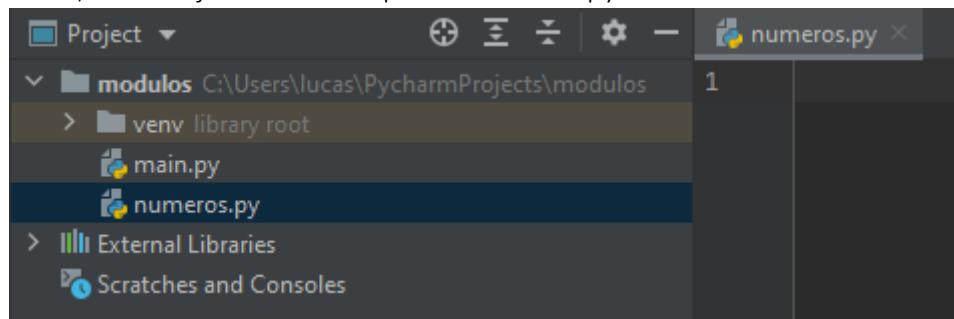
Modularização é construir módulos. Sistemas cada vez maiores necessitam de maior espaço de leitura, comando, entendimento. É dividir um programa em partes/módulos. Aumentando o foco pois aumenta a legibilidade. Facilitar a manutenção dos programas e módulos.

Para os próximos capítulos, criamos, um novo projeto, pois como será necessário criar as ligações entre as pastas, deixamos os exercícios até o momento em um Projeto e agora Módulos, será um outro Projeto.

Antigo:



Novo, criando já um novo arquivo "números.py"



Ao tentarmos utilizar, por exemplo, uma função que calcula o fatorial de um número, nesse novo projeto, repare que teremos um erro, pois, a função ainda não foi definida, até sem surpresa.

```
1 num = int(input('Digite um número: '))
2 fat = fatorial(num)
3
4 print(f'O fatorial de {num} é {fat}')
```

The code in the 'numeros.py' editor shows a syntax error. The variable 'fatorial' is underlined with a red squiggle, indicating it is undefined. The code attempts to call a function 'fatorial' on line 2, which is not yet defined.

Para isso, criaremos a função para calcular o fatorial

Portanto, agora com a função definida, a função vai trazer um resultado.

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

num = int(input('Digite um número: '))
fat = fatorial(num)

print(f'O fatorial de {num} é {fat}')
```

numeros x  
C:\Users\lucas\PycharmProjects\modulos\venv\Scripts\pyth  
Digite um número: 5  
O fatorial de 5 é 120

Mas e quando o programa começa a tomar outras proporções? E começa a crescer de tamanho e código?

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

def dobro(num):
    return num * 2

def triplo(num):
    return num * 3

num = int(input('Digite um número: '))
fat = fatorial(num)
print(f'O fatorial de {num} é {fat}')
print(f'O dobro de {num} é {dobra(num)}')
print(f'O triplo de {num} é {triplo(num)}')
```

numeros x  
C:\Users\lucas\PycharmProjects\modulos\venv\Scripts  
Digite um número: 5  
O fatorial de 5 é 120  
O dobro de 5 é 10  
O triplo de 5 é 15

Para isso, é possível separar do programa principal. Vamos pegar aquele bloco de funções e “levar/arrastar” para uma pasta/programa/arquivo novo que receberá um nome de “uteis.py”. Separando o módulo do programa principal. Como fazer isso na prática?

```

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

```

Criar, um novo arquivo no Pycharm e vamos remover do programa principal, todas aquelas funções e levaremos para esse novo arquivo. Portanto, teremos o programa principal rodando o “Numeros.py” os “uteis.py” que será o módulo.

```

Project: numeros
modulos C:\Users\lucas\PycharmProj
  v  venv library root
    main.py
    numeros.py
    uteis.py
  External Libraries
  Scratches and Consoles

numeros.py
1 def factorial(n):
2     f = 1
3     for c in range(1, n+1):
4         f *= c
5     return f
6
7 def dobro(num):
8     return num * 2
9
10 def triplo(num):
11     return num * 3

```

Porém, vamos ter um problema, note que agora, dentro do programa principal, não será possível executar a função do factorial, pois ainda não foi definida.

```

Project: numeros
modulos C:\Users\lucas\PycharmProj
  v  venv library root
    main.py
    numeros.py
    uteis.py
  External Libraries
  Scratches and Consoles

numeros.py
1
2
3
4 num = int(input('Digite um número: '))
5 fat = factorial(num)
6 print(f'O factorial de {num} é {fat}')
7 print(f'O dobro de {num} é {dobro(num)}')
8 print(f'O triplo de {num} é {triplo(num)}')

Run: numeros
C:\Users\lucas\PycharmProjects\modulos\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/modulos/numeros.py
Digite um número:
Traceback (most recent call last):
  File "C:/Users/lucas/PycharmProjects/modulos/numeros.py", line 4, in <module>
    num = int(input('Digite um número: '))
ValueError: invalid literal for int() with base 10: ''

```

## Vínculo entre Programa Principal e os Módulos

Para o Python, todo arquivo com a extensão “.py” pode ser um módulo, desde que nele contenha funções definidas.

### Importação do Módulo geral

Vamos usar o comando “IMPORT” para fazer a importação. E o mais importante, precisamos fazer a invocação/chamada da função

Como importamos diretamente o módulo, se quisermos achar o dobro de um número pela função desse, precisamos: **“chamar a módulo . (ponto) a função contida no módulo e passar o parâmetro”**.

```
numeros.py
import uteis

num = int(input('Digite um número: '))
fat = uteis.fatorial(num)
print(f'O factorial de {num} é {fat}')
print(f'O dobro de {num} é {uteis.dobro(num)}')
print(f'O triplo de {num} é {uteis.triplo(num)}')

Run: numeros
C:\Users\lucas\PycharmProjects\modulos\venv\Scripts
Digite um número: 5
O factorial de 5 é 120
O dobro de 5 é 10
O triplo de 5 é 15
```

## Importação de Funções específicas do Módulo

Diferente da importação geral, podemos importar somente algumas funções específicas, através do “`From móduloX Import FunçãoEspecíficaDesejada`”

Nesse caso, não é necessário fazer o “módulo.(ponto)função(parâmetro).

```
from uteis import dobro

n = int(input('Digite um número: '))
fat = uteis.fatorial(num)
print(f'O fatorial de {num} é {fat}')
print(f'O dobro de {num} é {dobro(num)}')
print(f'O triplo de {num} é {uteis.triplo(num)}')

Run:  numeros
C:\Users\lucas\PycharmProjects\modulos\venv\Script
Digite um número: 5
O dobro de 5 é 10
Process finished with exit code 0
```

## ATENÇÃO!

Segundo a documentação, a melhor forma de realizar a importação é pelo método que realiza a importação geral, pois é possível que duas funções com o mesmo nome sejam criadas em módulos diferentes, causando um problema de incompatibilidade. Sendo assim, a melhor importação que causará menos possíveis erros é na chamada completa da função, conforme na página 172 desse manual.

“Chamar a biblioteca/módulo . (ponto) a função contida na biblioteca e passar o parâmetro”.

## Vantagens

Quais são as melhores vantagens da modularização?

- ✓ Organização do Código (Mais limpo)
- ✓ Facilidade na Manutenção (Basta apenas ir diretamente no módulo e função desejada e todos os programas se beneficiarão dessa manutenção)
- ✓ Ocultação do código detalhado (Sem necessidade do dev entender como o código foi feito na sua raiz)
- ✓ Reutilização em outros projetos

Vai chegar um determinado momento, que somente esses módulos não serão mais “úteis” e o suficiente para que o programa seja elaborado. Precisaremos de Pacotes.

## Pacotes

Algumas linguagens de programação chamam de “*Bibliotecas*”, o Python chama de “*Pacotes*”.

Pegaremos o código que utilizamos nas páginas anteriores de Módulo.

No começo, resolvemos o problema que não deveríamos definir funções no programa principal. Para isso criamos a `uteis.py` e isso resolveu o problema.

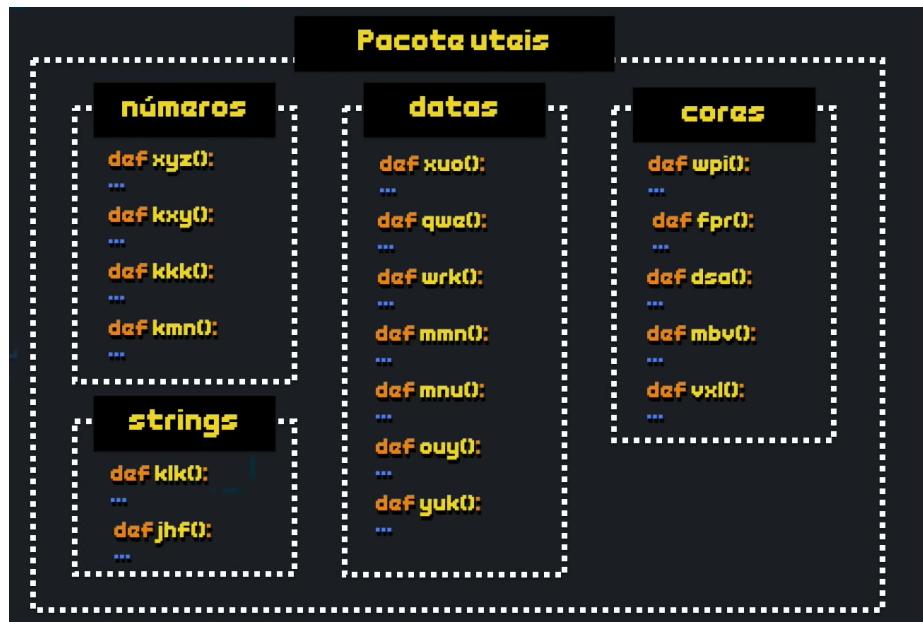
The screenshot shows a terminal window with a black background and white text. At the top, it says "uteis.py". Below that is a dashed-line box containing several function definitions: `def xyz():`, `def kxy():`, `def kkk():`, `def kmn():`, and `def klk():`. Below the box, the word "import" is written in orange, followed by "uteis" in yellow.

Imagina que temos o módulo “`uteis.py`” com muitas, mas muitas funções ao logo do tempo. Por exemplo, vamos criando várias e várias funções, guardando tudo dentro de um **único módulo**. As vantagens que os módulos proporcionavam, deixaram de existir, uma vez que com muitas linhas de código, não era mais tão legível, rápido acesso, manutenção (imagina função matemática misturada com função que corta *string* e outras coisas).

The screenshot shows a terminal window with a black background and white text. At the top, it says "uteis.py". Below that is a dashed-line box containing many function definitions, including pairs like `def xyz(): def mmn():`, `def kxy(): def mnu():`, `def kkk(): def ouy():`, `def kmn(): def yuk():`, `def klk(): def wpi():`, `def jhf(): def fpr():`, `def xu0(): def dsa():`, `def qwe(): def mbv():`, and `def wrk(): def vxl():`.

Para resolver esse problema, vamos criar Pacotes. Um Pacote nada mais é do que uma pasta/ um pacote que contém módulos de diversos assuntos. Isso é, dentro de um pacote chamado de uteis, podemos ter vários módulos com inúmeros assuntos.

Sendo então, um Pacote chamado Uteis com seus módulos (Números, Tratamento de Strings, Datas, Cores) e cada módulo contém a sua função.



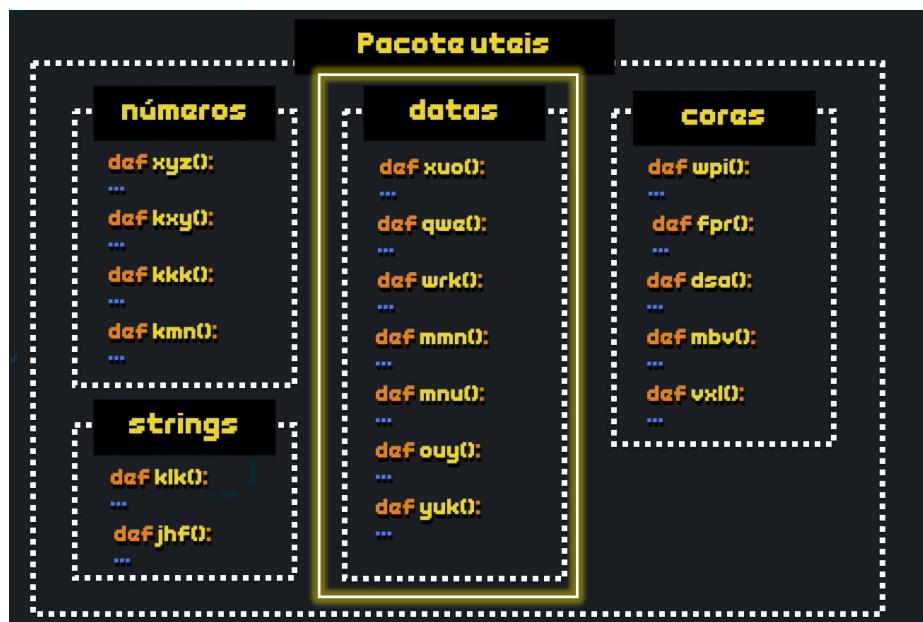
## Importação dos Pacotes

Para realizar as importações dos pacotes, é similar as importações dos módulos. Podemos importar o pacote inteiro através do “IMPORT nomeDoPacote” ou através de chamadas de módulos específicos como: “FROM nomeDoPacote IMPORT nomeDaArea/Módulo”.

Por exemplo:

```
FROM uteis IMPORT datas
```

Dentro de Pacotes, importará somente as funções dentro de Datas



## Como criar Pacotes

Até aí também, nenhuma surpresa, certo? Mas, como criar os pacotes? Para o Python, todo arquivo extensão .py pode ser considerado um possível módulo. E dentro de um projeto Pyhton, todas as pastas são consideradas um pacote. Para isso, vamos criar uma pasta chamada UTEIS que conterá os arquivos .py de Cores, Datas, Números, Strings... Pacotes somente devem ser utilizados quando eles forem “Maiores que o grande”, para projetos gigantescos!

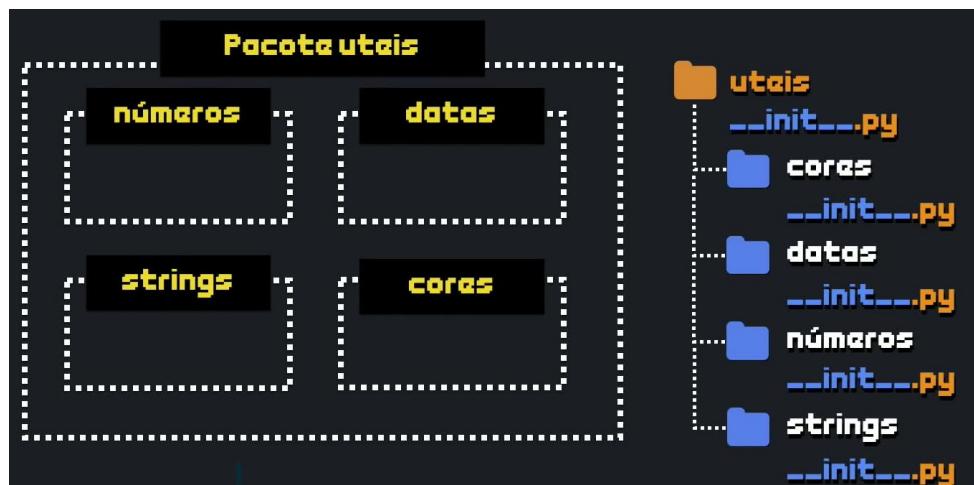


## Sintaxe especial para nome de arquivo dentro de Pacotes

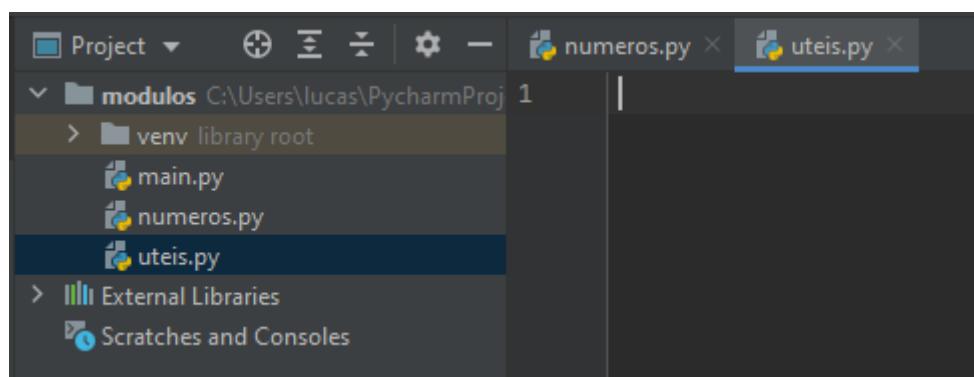
Existe uma sintaxe especial para o nome de arquivos dentro de pacotes, em especial, um arquivo necessário que deve conter dentro de cada uma das pastas do pacote:

`__init__.py`

Ao utilizar o pyCharm não precisamos nos preocupar pois esse arquivo é criado automaticamente, mas, se não estiverem os utilizando o PC, precisamos criar esse arquivo sempre que criarmos os módulos. Inclusive o próprio pacote pode ter esse arquivo.



No PyCharm, vamos voltar aquele nosso código no módulo uteis.py e vamos recortar para não perdemos o conteúdo e vamos deletar o arquivo uteis.py:



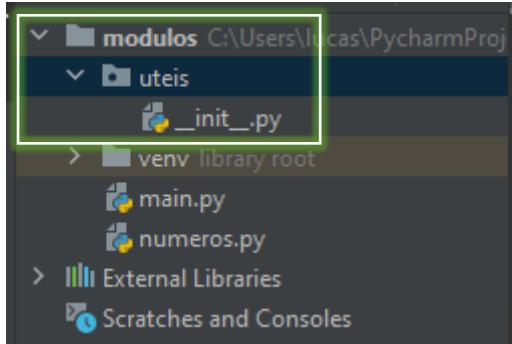
Na hora após fazermos isso, note que o programa principal já deu erro:

```
1 import uteis
2
3
4 num = int(input('Digite um número: '))
5 fat = uteis.fatorial(num)
```

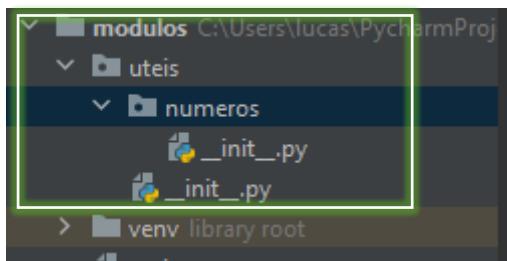
## Criando o Pacote no PyCharm

Vamos no nome do projeto, e ao invés de clicarmos no “new > python.file” vamos em: “new > Python Package”. Vale lembrar que dentro de um pacote podemos ter outros pacotes.

Após digitarmos o nome do pacote, ele já criará o pacote e o arquivo “Init”:



Dentro do pacote Uteis, teremos um outro pacote que criaremos que se chamará “Numeros”. E por aí vamos criando mais e mais pacotes para separarmos e dividirmos os nossos problemas.



E dentro agora de uteis > números > \_\_init\_\_.py, nele nós colocaremos aquelas funções que removemos um tempo atrás quando uteis ainda era um módulo:

A screenshot of the PyCharm code editor. The file \_\_init\_\_.py is open, showing three functions: fatorial, dobro, and triplo. The code is as follows:

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

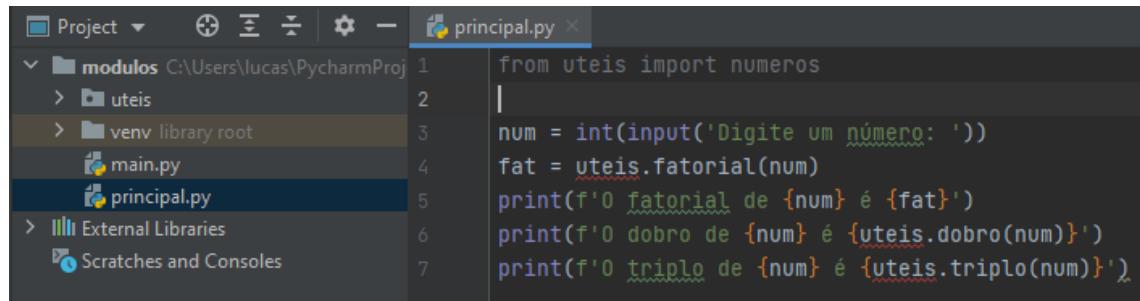
def dobro(num):
    return num * 2

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

The 'modulos' folder and its subfolders ('uteis', 'numeros') are visible in the project tree on the left. An arrow points from the project tree to the \_\_init\_\_.py file in the code editor.

E no programa principal, agora, vamos utilizar a importação específica do pacote Uteis, apenas os números:

Repare que utilizamos o “**FROM nomeDoPacote IMPORT nomeDoPacoteModulo**”

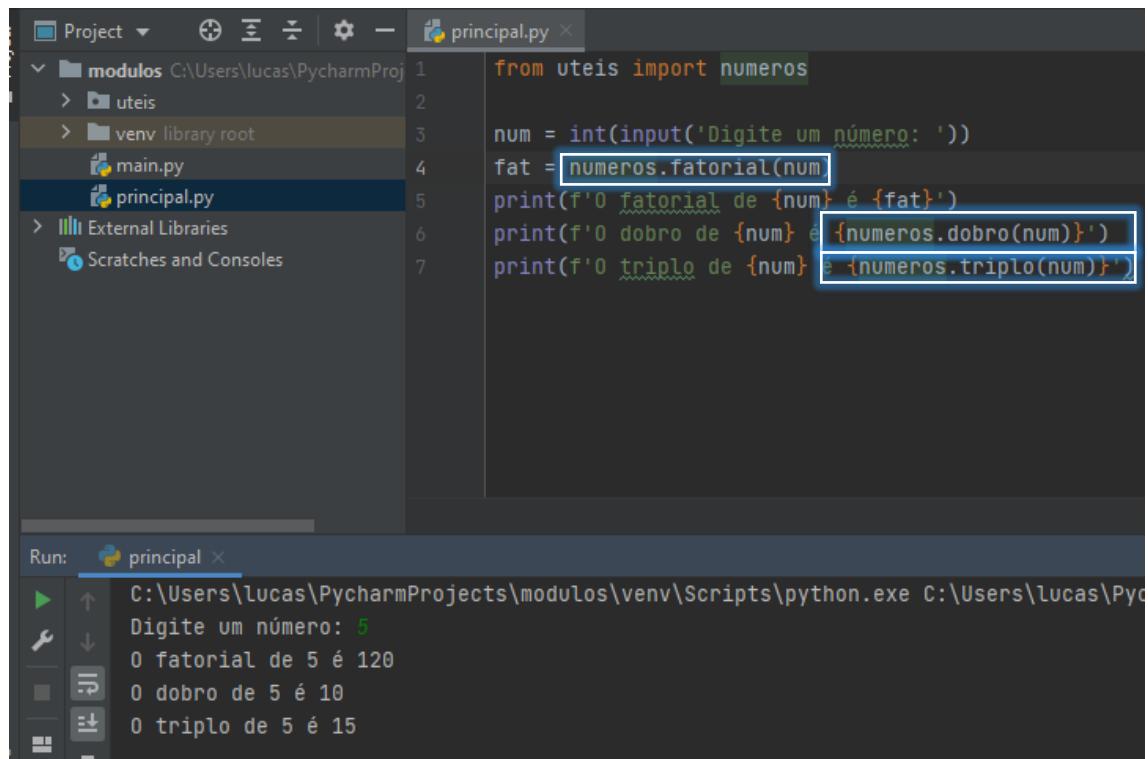


```
from uteis import numeros

num = int(input('Digite um número: '))
fat = numeros.fatorial(num)
print(f'O fatorial de {num} é {fat}')
print(f'O dobro de {num} é {numeros.dobro(num)}')
print(f'O triplo de {num} é {numeros.triplo(num)}')
```

Embora, ainda assim, dentro do programa principal, note que teremos um erro, ao chamarmos as funções uteis.dobro, uteis.fatorial. Por quê? Porque agora uteis é um pacote.

Para essas funções funcionarem, eu preciso substituir o nome que elas eram invocadas por um antigo módulo(que agora se tornou um pacote) e aí chamar o pacote certo para ela e tratá-lo como um módulo.



```
from uteis import numeros

num = int(input('Digite um número: '))
fat = numeros.fatorial(num)
print(f'O fatorial de {num} é {fat}')
print(f'O dobro de {num} é {numeros.dobro(num)}')
print(f'O triplo de {num} é {numeros.triplo(num)}')
```

Run: principal

```
C:\Users\lucas\PycharmProjects\modulos\venv\Scripts\python.exe C:\Users\lucas\Py
Digite um número: 5
O fatorial de 5 é 120
O dobro de 5 é 10
O triplo de 5 é 15
```

# Tratamento de Erros e Exceções

Erros são comuns, existem alguns bem simples e outros bem complexos.

## Erros de Sintaxe

Problemas de Sintaxe. Esses erros são fáceis de tratar. Não são exatamente esses erros que esse capítulo vai abordar.



## Erro de NameError

Porém, além do erro de sintaxe, tem um erro de **NameError**. É um erro de referência. O valor da variável "X" não foi iniciada. Ou seja, a variável X não existe. Na verdade, esse "erro" se chama: Exceção.

## Exceção de NameError

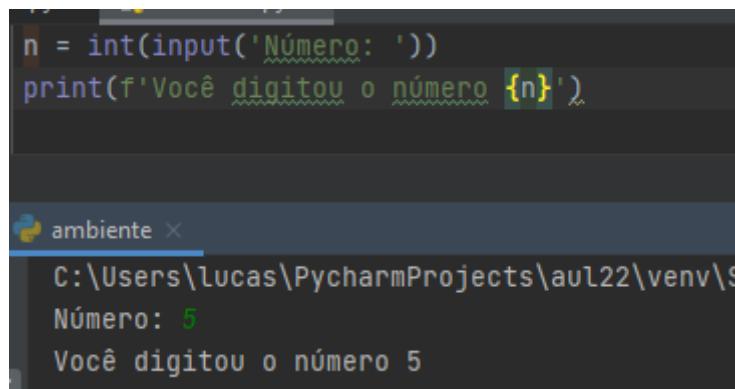
Na verdade, o erro que foi citado acima, se chama EXCEÇÃO. Como o que ocorre abaixo.

```
print(x)

ambiente ×
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/aul22/ambiente.py
Traceback (most recent call last):
  File "C:/Users/lucas/PycharmProjects/aul22/ambiente.py", line 1, in <module>
    print(x)
NameError: name 'x' is not defined
```

## Exceção de ValueError

É fácil notar que as linhas de código abaixo estão corretas. Afinal, nenhum erro ou exceção aconteceram. Porém...

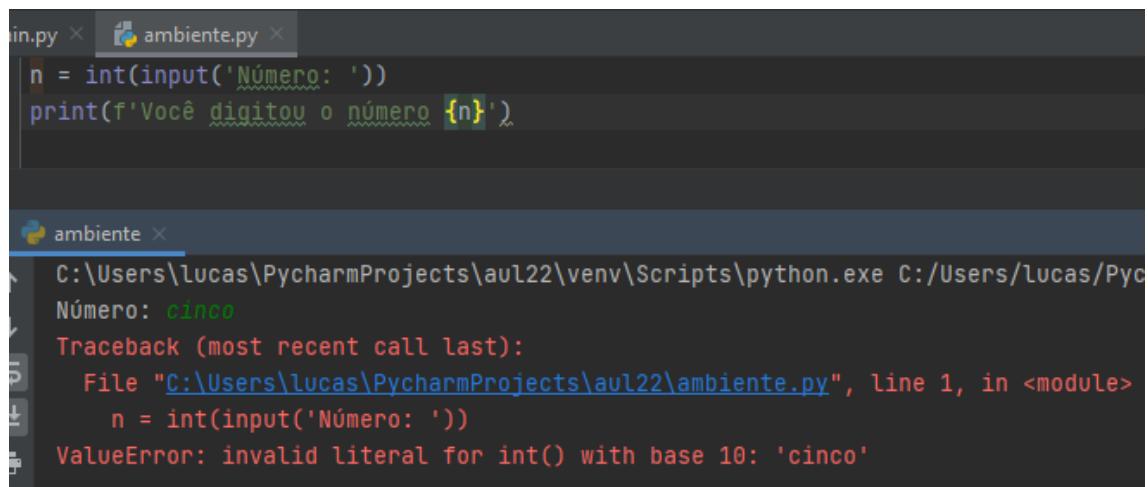


A screenshot of the PyCharm IDE interface. The top bar shows the file 'ambiente.py'. Below it, the terminal window displays the command 'C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/aul22/ambiente.py' followed by the output: 'Número: 5' and 'Você digitou o número 5'. The code in the editor is:

```
n = int(input('Número: '))
print(f'Você digitou o número {n}')
```

E se o usuário digitar ao invés de "5", resolver digitar por extenso "cinco"?

Gera um erro de **ValueError**. A palavra cinco não pode ser convertida para ser um número inteiro.



A screenshot of the PyCharm IDE interface. The terminal window shows the command 'C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/aul22/ambiente.py' and the input 'Número: cinco'. It then displays a traceback and the error message 'ValueError: invalid literal for int() with base 10: 'cinco''. The code in the editor is identical to the previous screenshot.

```
n = int(input('Número: '))
print(f'Você digitou o número {n}')
```

## Exceção de AttributeError

Essa exceção é disparada ao tentar utilizar um atributo a um determinado objeto/variável que não deveria receber tal atributo. Por exemplo, um objeto float não pode receber um .isalpha(), vai causar um erro de atributo, pois esse atributo não deveria ser empregado a um float e sim a um objeto string, por exemplo.

The screenshot shows the PyCharm IDE interface. The code editor contains Python code for validating user input. The terminal window below it shows the execution of the script and the resulting error message.

```
try:
    valido = False
    while valido is False:
        numero = float(input('Digite um valor Real: '))
        if numero is not numero.isalpha() or numero.strip() == '':
            valido = True
            print(f'O valor digitado foi {numero}')
    except ValueError:
        print('Erro! Valor não digitado não é válido!')
    except Exception as error:
        print(f'O erro ocorrido foi {error}')
leiaFloat() > try > while valido is False
```

main ×

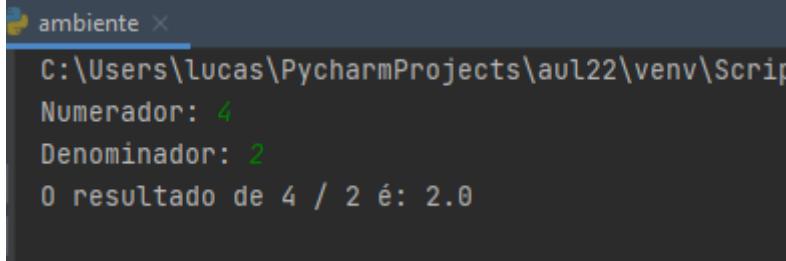
```
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/aul22/main.py
Digite um valor Real: 5
O erro ocorrido foi <class 'AttributeError'>

Process finished with exit code 0
```

## Exceção ZeroDivisionError

Veja, fazendo uma fórmula simples, de um número dividido pelo outro. E alocando o resultado em uma variável e mandando o print para mostrar o resultado. Porém...

```
a = int(input('Numerador: '))
b = int(input('Denominador: '))
r = a / b
print(f'O resultado de {a} / {b} é: {r}')
```

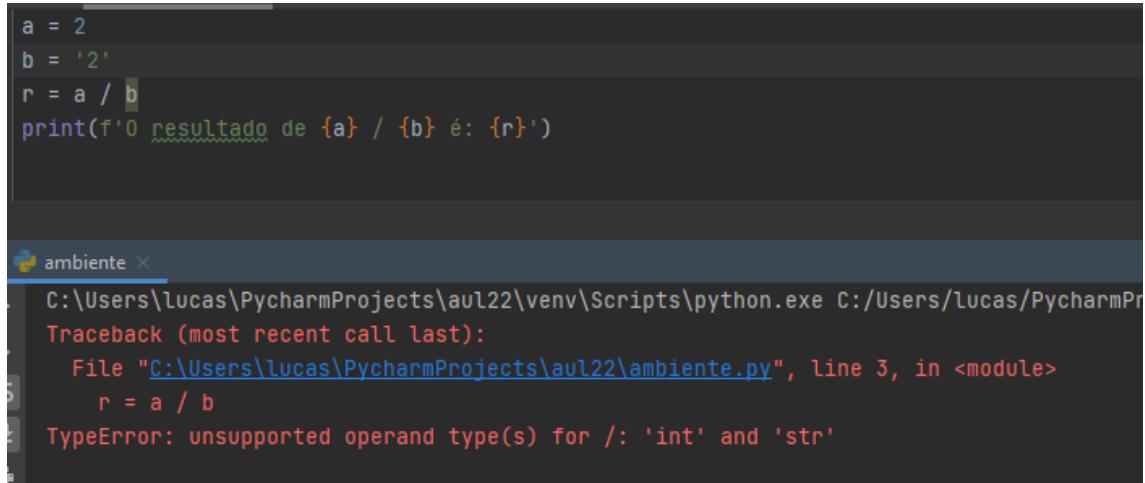


```
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/aul22/ambiente.py
Numerador: 4
Denominador: 2
O resultado de 4 / 2 é: 2.0
```

E se o usuário no denominador realizar o input de zero(0)? Na matemática, um número dividido por zero não pode acontecer. Isso causa um erro.

## Exceção de TypeError

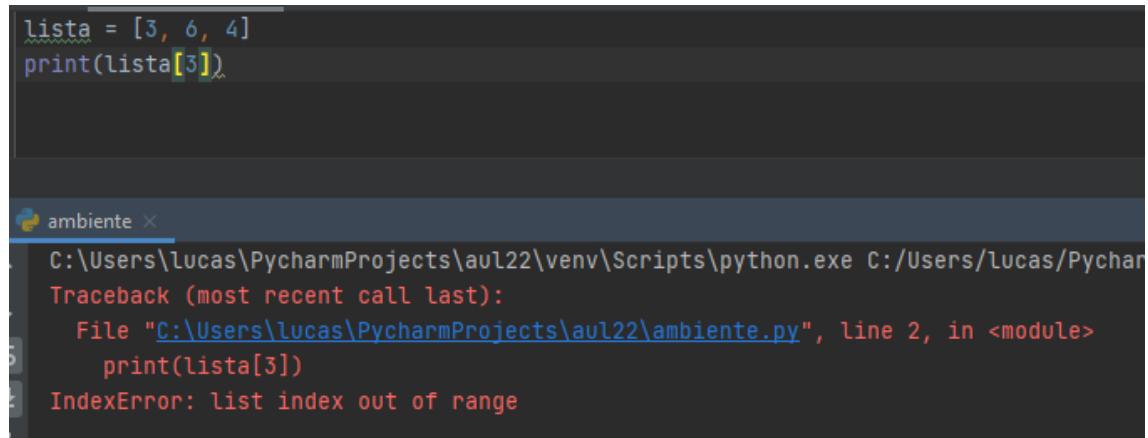
Em algumas linguagens de programação, as linhas de código abaixo poderiam acontecer, mas no Python, não. Pois, o que está dentro de aspas simples, é considerado string. E inteiro dividido por string ocasiona um erro de tipo.



```
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/aul22/ambiente.py
Traceback (most recent call last):
  File "C:/Users/lucas/PycharmProjects/aul22/ambiente.py", line 3, in <module>
    r = a / b
TypeError: unsupported operand type(s) for /: 'int' and 'str'
```

## Exceção de IndexError (Lista) – KeyError(Dicionário)

Em uma lista, existem as posições, o primeiro elemento da lista, é no índice zero. Portanto, se tentar imprimir uma posição que não está contida na lista, ocasionará uma exceção de IndexError – List Index Out Of Range.



A screenshot of the PyCharm IDE interface. The code editor shows the following Python code:

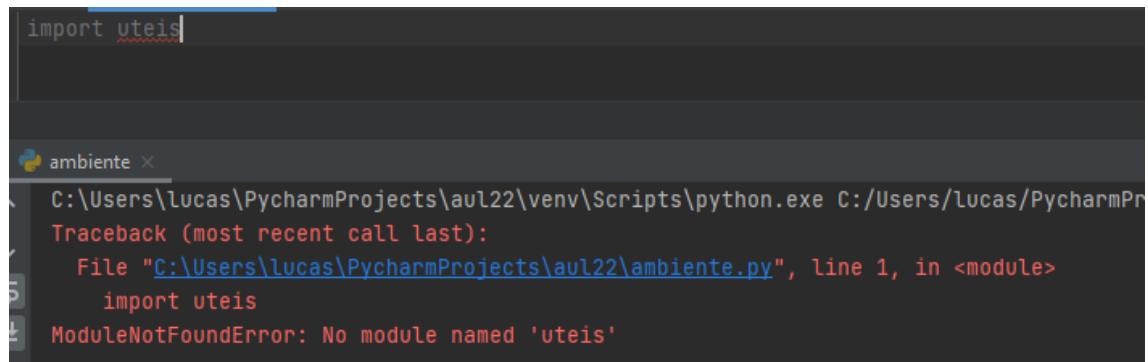
```
lista = [3, 6, 4]
print(lista[3])
```

The output window below shows the execution results and an error message:

```
ambiente ×
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/Pychar
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\aul22\ambiente.py", line 2, in <module>
    print(lista[3])
IndexError: list index out of range
```

## Exceção de ModuleNotFoundError

Veja que também um erro na hora de realizar a importação de um módulo no Python irá disparar uma exceção que diz que é impossível importar pois não há nenhum módulo com esse nome.



A screenshot of the PyCharm IDE interface. The code editor shows the following Python code:

```
import uteis
```

The output window below shows the execution results and an error message:

```
ambiente ×
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/Pychar
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\aul22\ambiente.py", line 1, in <module>
    import uteis
ModuleNotFoundError: No module named 'uteis'
```

## Exceção de UrlLib Error

Essa é uma exceção que vai verificar um erro que ocorreu ao tentar estabelecer, conectar com algum site da internet.

```
import urllib
import urllib.request

def conectar():
    try:
        site = urllib.request.urlopen('http://www.pudir')
    except Exception as error:
        print(f'Ocorreu um erro! Error: {error.__class__}')
    else:
        print(f'Conexão estabelecida com sucesso!')
```

```
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe
Ocorreu um erro! Error: <class 'urllib.error.URLError'>
```

```
try:
    site = urllib.request.urlopen('http://www.pudir')
except urllib.error.URLError:
    print(f'O site não está acessível no momento.')
except Exception as error:
    print(f'Ocorreu um erro! Error: {error.__class__}')
else:
    print(f'Conexão estabelecida com sucesso!')
```

Para corrigir esse erro, basta informar a url correta quando solicitado.

Para mais informações sobre essa biblioteca, verificar o exercício 114.

## Alguns exemplos de exceções no Python

Existem muitas mais, mas essa são uns exemplos:

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

## Tratamento das Exceções

Existe um comando que é o responsável por realizar o tratamento dessas exceções. Até agora, sempre que algum erro acontecia, íamos lá e corrigíamos até ele não apresentar, sempre mandávamos no código. Agora, se algum erro acontecer, o programa vai “tentar tomar uma outra ação”.

Esse comando é através do

### Try e Except

O Try é a tentativa de realizar a operação e o Except é algo se acontecer a falha.

```
try:  
    operação  
except:  
    falhou
```

## Na prática

Se lembra do erro que gerava uma exceção da divisão por zero?

```
a = int(input('Numerador: '))
b = int(input('Denominador: '))
r = a / b
print(f'O resultado de {a} / {b} é: {r}')

ambiente ×
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/PycharmP
Numerador: 4
Denominador: 0
Traceback (most recent call last):
  File "C:/Users/lucas/PycharmProjects\aul22\ambiente.py", line 3, in <module>
    r = a / b
ZeroDivisionError: division by zero
```

Vamos tratar esse agora. Como pode ver abaixo.

```
try:
    a = int(input('Numerador: '))
    b = int(input('Denominador: '))
    r = a / b
except:
    print('Infelizmente ocorreu um problema :(')
print(f'O resultado de {a} / {b} é: {r}')
```

Ou seja, o programa tentará (try) gerar algo. Caso alguma exceção seja disparada o (except) acontecerá e mostrará algo na tela, retornará algo (a ser definido pelo dev) impedindo que o erro da exceção da divisão por zero apareça na tela:

O Try funcionando:

```
Numerador: 4
Denominador: 2
O resultado de 4 / 2 é: 2.0

Process finished with exit code 0
```

O except agindo:

```
C:\Users\lucas\PycharmProjects\aul22\ve
Numerador: 4
Denominador: 0
Infelizmente ocorreu um problema :(
```

Maravilha! Funcionou! Não, não é bem assim.

Na verdade, tratamos o erro da divisão por zero, mas como a variável “r” não foi “iniciada”, tivemos um disparo da exceção de **NameError**:

Em verde podemos ver que o tratamento da divisão por zero foi resolvido. Mas, estamos com outro problema, uma vez que o programa não conseguiu localizar o {r}

```
try:
    a = int(input('Numerador: '))
    b = int(input('Denominador: '))
    r = a / b
except:
    print('Infelizmente ocorreu um problema :(')

print(f'O resultado de {a} / {b} é: {r}')

except
ambiente ×
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/Pychar
Numerador: 4
Denominador: 0
Infelizmente ocorreu um problema :(
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\aul22\ambiente.py", line 8, in <module>
    print(f'O resultado de {a} / {b} é: {r}')
NameError: name 'r' is not defined
```

Esse erro na verdade acontece porque, na verdade, a variável “r” só seria criada CASO o TRY desse certo. Porém, como o TRY deu errado e houve um EXCEPT. É de se entender que na verdade todo aquele bloco que está em branco brilhante, na verdade, não aconteceu. Ou seja, a mensagem da exceção está ok, mas, quando chegou a hora de printar o “r”. Não foi possível, porque as variáveis “a” e “b” foram criadas normalmente, mas não chegou a ocorrer a divisão. E por isso, o “r” não foi criado.

## Clausula Else

Então, na verdade, o bloco do TRY tentou realizar uma operação. Essa operação disparou uma exceção de divisão por zero. O bloco Except agiu mostrando o erro. Mas, existia uma linha mostrando o print desse resultado, que não aconteceu!



Portanto, se essa operação não aconteceu, o “r” não foi iniciado e a linha que deseja o print não aconteceu! Mas, mesmo assim, lá está ela escrita no programa principal. Portanto, somente vamos mostrar algo no bloco ELSE, se o bloco do TRY realmente aconteceu conforme o esperado, ou seja, se a operação foi possível, então o bloco EXCEPT não ocorreu e então aí sim, precisamos informar o que acontecerá, que será definido no bloco do ELSE

```
try:
    a = int(input('Numerador: '))
    b = int(input('Denominador: '))
    r = a / b
except:
    print('Infelizmente ocorreu um problema :(')
else:
    print(f'O resultado de {a} / {b} é: {r}')
else
```

ambiente ×  
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts  
Numerador: 8  
Denominador: 4  
O resultado de 8 / 4 é: 2.0

E então, portanto, caso o bloco TRY tente realizar a operação e não consiga, o bloco do Except irá agir e lançar alguma mensagem, retorno ou o que for definido. Mas o bloco ELSE não acontecerá e o programa irá parar na mensagem que veio da exceção.

Como a divisão por zero não é possível o bloco parou no EXCEPT, e o ELSE não foi executado.

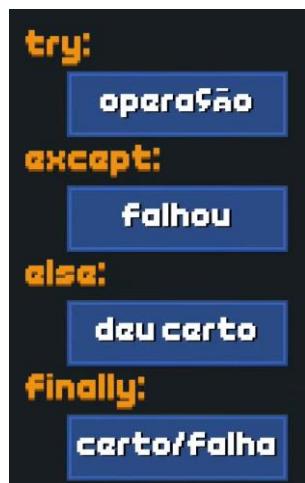
```
except:
    print('Infelizmente ocorreu um problema :(')
else:
    print(f'O resultado de {a} / {b} é: {r}')
else
```

ambiente ×  
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\py  
Numerador: 5  
Denominador: 0  
Infelizmente ocorreu um problema :(

## Clausula Finally

Achou que tinha acabado? Achou errado.

Existe além da “clausula” *ELSE*, existe também a “clausula” *FINALLY*. Lembrando que essas duas clausulas são na maioria dos casos, opcionais. Mas nesse exercício, estamos utilizando. O bloco então, tem a seguinte estrutura final:



O finally significa que, “dando certo ou dando errado”, algo vai acontecer. Se o bloco TRY lançar ok e o ELSE executar, o FINALLY vai acontecer. Se o bloco TRY tentar lançar, disparou uma exceção e o EXCEPT tratou, o FINALLY vai acontecer. Veja:

O bloco TRY aconteceu perfeitamente, o bloco EXCEPT não tratou a exceção porque não precisou, o bloco ELSE aconteceu porque o bloco TRY estava certo. E o bloco FINALLY vai acontecer.

```
except:  
    print('Infelizmente ocorreu um problema :(')  
else:  
    print(f'O resultado de {a} / {b} é: {r}')  
finally:  
    print(f'Volte sempre! Muito obrigado')  
  
finally  
ambiente x  
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts  
Numerador: 4  
Denominador: 2  
O resultado de 4 / 2 é: 2.0  
Volte sempre! Muito obrigado  
  
Process finished with exit code 0
```

Mas e se tivesse dado erro? Se o bloco TRY tivesse disparado uma exceção a ser tratada pelo EXCEPT? O ELSE não aconteceria, mas o FINALLY, sim.

Dando certo, ou dando erro, o FINALLY executará algo dentro dele.

```
except:
    print('Infelizmente ocorreu um problema :(')
else:
    print(f'O resultado de {a} / {b} é: {r}')
finally:
    print('Volte sempre! Muito obrigado')

finally
ambiente ×
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts
Numerador: 4
Denominador: 0
Infelizmente ocorreu um problema :(
Volte sempre! Muito obrigado

Process finished with exit code 0
|
```

```
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts
Numerador: 4
Denominador: zero
Infelizmente ocorreu um problema :(
Volte sempre! Muito obrigado

Process finished with exit code 0
|
```

E se eu quiser deixar algo mais específico? Porque apenas aquela frase do tratamento dizendo que um erro aconteceu, é muito genérica.

### Except Exception As (Descobre o Erro)

Vamos utilizar o **EXCEPT EXCEPTION AS**. Para mostrar o que aconteceu com os erros.

```
except Exception as erro:
    print(f'Problema encontrado foi: {erro.__class__}')
else:
    print(f'O resultado de {a} / {b} é: {r}'
```

Através da classe EXCEPTION, ai criamos uma variável. Dentro do print, chamaremos a {variável.\_\_\_} várias funcionalidades aqui \_\_\_

```
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts
Numerador: 4
Denominador: zero
Problema encontrado foi: <class 'ValueError'>
Volte sempre! Muito obrigado
```

Repare que o erro acima quis dizer que foi um erro ao tentar dividir um número inteiro por uma string. E se tentar dividir um número por zero? Lembra que vai ter tratamento de exceção.

Vai acontecer a exceção de ZeroDivisionError

```
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe
Numerador: 4
Denominador: 0
Problema encontrado foi: <class 'ZeroDivisionError'>
Volte sempre! Muito obrigado

Process finished with exit code 0
|
```

Claro, esse erro não deveria ser mostrado ao usuário. Apenas para o programador enquanto nesse tempo de desenvolvimento estiver buscando as saídas e receberá uma mensagem personalizada.

E tem mais...

### Try com muitos Execpts

Um mesmo bloco TRY é possível de gerar várias exceções, conforme os erros que queira tratar.

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

Podemos tratar um erro genérico, um erro específico e vários erros ao mesmo tempo, conforme na página 195.

## Tratando apenas um Erro

Basta chamar o erro que pode acontecer:

```
except (ValueError):
    print(f'Tivemos um problema com o tipo de dado digitado.')
else:
    print(f'O resultado de {a} / {b} é: {r}')
finally:
    print('Volte sempre! Muito obrigado')
```

```
except (ValueError)
```

ambiente ×

```
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Us
Numerador: 4
Denominador: zero
Tivemos um problema com o tipo de dado digitado.
Volte sempre! Muito obrigado
```

## Tratando vários Erros

Apenas necessário colocar uma vírgula

```
except (ValueError, TypeError):
    print(f'Tivemos um problema com o tipo de dado digitado.')
else:
    except (ValueError, TypeError)
ambiente ×
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Us
Numerador: 4
Denominador: zero
Tivemos um problema com o tipo de dado digitado.
Volte sempre! Muito obrigado
```

```
except (ValueError, TypeError):
    print(f'Tivemos um problema com o tipo de dado digitado.')
except ZeroDivisionError: #erro na divisão se for zero.
```

```
    print('Não é possível dividir um número por zero.')
except KeyboardInterrupt: #deu enter sem apertar nada
    print('Usuário não digitou nenhum dado')
else:
    print(f'O resultado de {a} / {b} é: {r}')
```

```
except ZeroDivisionError
ambiente ×
```

```
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Us
Numerador:
Tivemos um problema com o tipo de dado digitado.
Volte sempre! Muito obrigado
```

E podemos adicionar um erro genérico também, que se nenhum dos erros acima for identificado, então aí ele lança o erro genérico e a partir daí vai montando mais exceções personalizadas. Conforme for identificando erros no código, aí vai criando os exceções personalizados.

```
except (ValueError, TypeError):
    print(f'Já vemos um problema com o tipo de dado digitado')
except ZeroDivisionError:
    print('Não é possível dividir um número por zero.')
except KeyboardInterrupt:
    print('Usuário não digitou nenhum dado')
except Exception as erro:
    print(f'O erro encontrado foi: {erro.__class__}')
else:
    except Exception as erro
```

Mas o código está ficando grande. Utilizar pacotes!

# Cores no Terminal

## Padrão ANSI

Padrão de Formalização Internacional Escape Sequence.

Tudo no padrão ASNI começa com um contra barra e o código (/código)

## Código ANSI para Cores

Para Python, o melhor em funcionamento é o 033, exemplo:

\033[aquiVocêColocaOCódigoEFechaComo'M'NoFinalm, exemplo:

3 valores: Sendo um para:

**Style** (se Negrito, Itálico, Sublinhado) – **Text** (Qual cor do texto) – **Background** (Qual cor do fundo)

\033[0;33;44m

0: No style significa que vai manter o padrão de estilo do terminal.

33: Texto amarelo

44: Background Azul

Para o Python, os melhores códigos para os valores, seguem abaixo. Vale lembrar que para cada linguagem e terminal, os códigos podem ser diferentes.

Style	Descr	Text	Descr	Background	Descr
0	None	30	White	40	White
1	Bold	31	Red	41	Red
4	Underline	32	Verde	42	Verde
7	Negative	33	Amarelo	43	Amarelo
-	-	34	Azul	44	Azul
-	-	35	Magenta	45	Magenta
-	-	36	Cyan	46	Cyan
-	-	37	Cinza	47	Cinza



Para colocar mais cores, estilos, é necessário importar uma biblioteca.

O texto começa com 30 e o background começa com o 40.

\033[0;30;41m

\033[4;33;44m

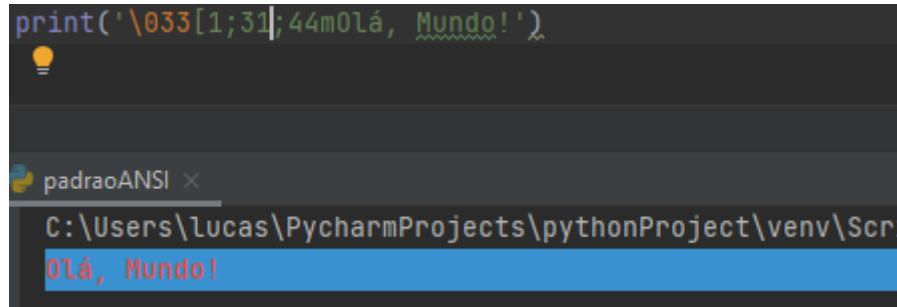
\033[1;35;43m

\033[30;42m

\033[m → Configuração Padrão do Terminal

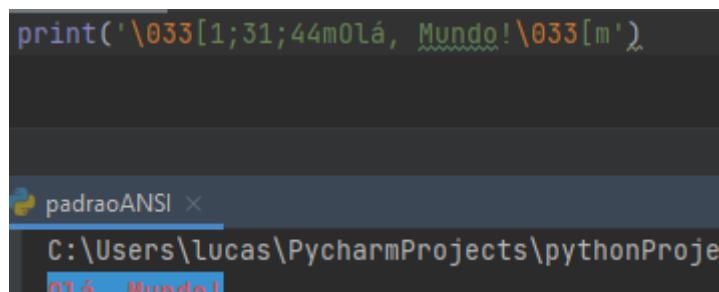
\033[7;30m → Inverter com negativo

```
print('\033[1;31;44mOlá, Mundo!')
```



Mas, por exemplo, perceba que essa faixa vai até o final da linha. Como corrigir? Basta colocar novamente a formatação após o final da linha do código.

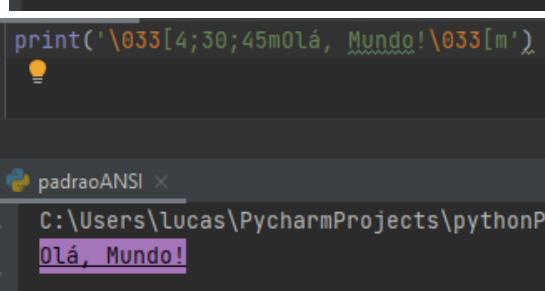
```
print('\033[1;31;44mOlá, Mundo!\033[m')
```



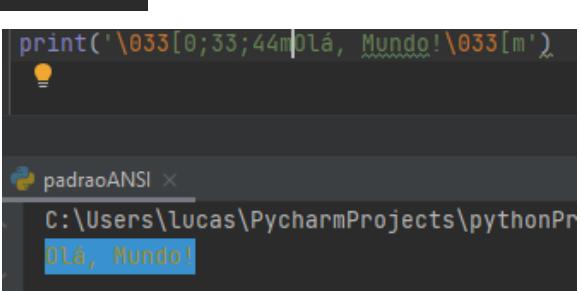
```
print('\033[4;30;45mOlá, Mundo!\033[m')
```



```
print('\033[0;33;44mOlá, Mundo!\033[m')
```



```
print('\033[7mOlá, Mundo!\033[m')
```



<sup>1</sup>Mais estilos através de métodos e bibliotecas.

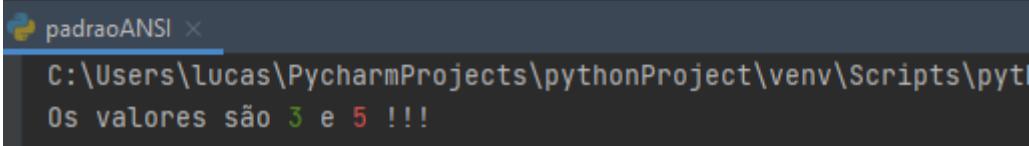
---

<sup>1</sup> Procurar pelo método Colorized

## Aplicando Cores as Variáveis e Máscaras

Note que eu estou utilizando no print, antes de abrir a máscara e depois da máscara eu fecho a formatação do texto.

```
a = 3
b = 5
print('Os valores são \033[32m{}\033[m e \033[31m{}\033[m !!!'
      .format(a, b))
```

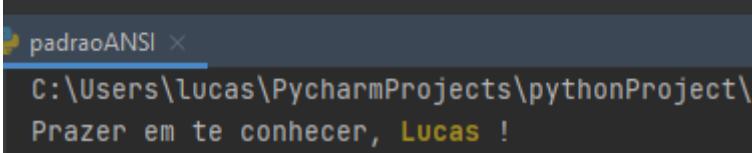


## Aplicando Cores com Máscara no Format()

Para utilizar esse método de adicionar cores, é um pouco mais avançado, basta adicionar 3 máscaras, por exemplo, sendo a primeira máscara a formatação inicial, a segunda máscara a variável e a terceira máscara vai ser o fechamento da formatação.

Lembrar que é necessário estar entre aspas simples dentro do format, se não, vai apresentar erro.

```
nome = 'Lucas'
print('Prazer em te conhecer, {}{}{} !'
      .format('\033[1;33m', nome, '\033[m'))
```



## Aplicando Cores com uma Lista de Cores

Esse método é muito mais avançado, pois cria-se uma lista que será chamada dentro do print. Usa uma coleção, que é um dicionário.

Repare que foi criada uma variável que utiliza uma lista para ser preenchida, muita atenção na sintaxe dessa lista que será ensinada mais na frente da apostila e do curso.

Após essa criação, basta chamar a lista no formato do print através da chamada do nome da variável seguida pela ['nomeDeUmaPosiçãoDaLista']

```
nome = 'Lucas'
cores = {'limpa': '\033[m',
         'azul': '\033[34m',
         'amarelo': '\033[33m',
         'pretoebranco': '\033[7m'}

print('Prazer em te conhecer, {}{}{} !'
      .format(cores['pretoebranco'], nome, cores['limpa']))

'pretoebranco'
padraoANSI ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\p
Prazer em te conhecer, Lucas !
```

# Exercícios

## Desafios Iniciais

Lendo dois números reais dados pelo usuário e retornando na tela:

```
numero1 = float(input('Digite o valor do número 1#: '))
numero2 = float(input('Digite o valor do número 2#: '))
soma = numero1 + numero2

print('A soma entre o número {} e o número {} é o valor de:
{}'.format(numero1,numero2,soma))
numero1 = float(input('Digite o valor do número 1#: '))
```

```
numero2 = float(input('Digite o valor do número 2#: '))
```

```
soma = numero1 + numero2
```

```
print('A soma entre o número {} e o número {} é o valor de:
{}'.format(numero1,numero2,soma))
```

```
algo = input('Digite qualquer coisa: ')

print('Os tipos primitivos abaixo são: ')
print(type(algo))
print('O que você digitou é alfabético? {}'.format(algo.isalpha()))
print('O que você digitou é númerico? {}'.format(algo.isnumeric()))
print('O que você digitou é alfanúmerico? {}'.format(algo.isalnum()))
print('O que você digitou está em Maiúsculo? {}'.format(algo.isupper()))
print('O que você digitou está em Minúsculo? {}'.format(algo.islower()))
algo = input('Digite qualquer coisa: ')
```

```
print('Os tipos primitivos abaixo são: ')
```

```
print(type(algo))
```

```
print('O que você digitou é alfabético? {}'.format(algo.isalpha()))
```

```
print('O que você digitou é númerico? {}'.format(algo.isnumeric()))
```

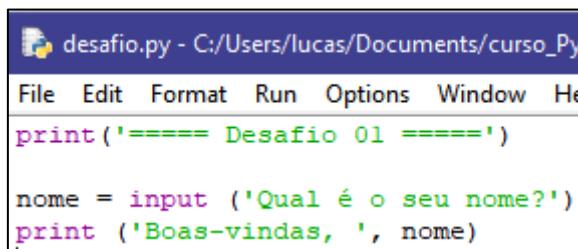
```
print('O que você digitou é alfanúmerico? {}'.format(algo.isalnum()))
```

```
print('O que você digitou está em Maiúsculo? {}'.format(algo.isupper()))
```

```
print('O que você digitou está em Minúsculo? {}'.format(algo.islower()))
```

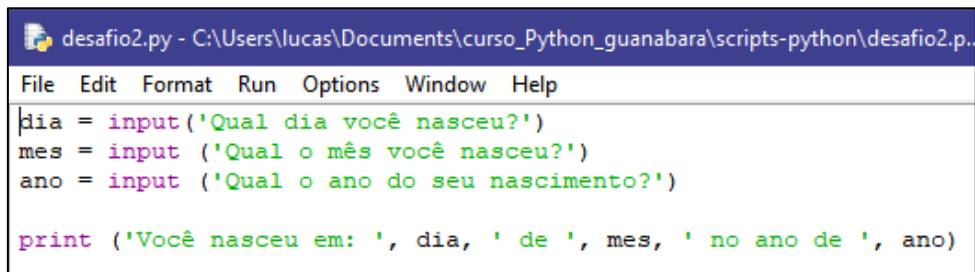
## Desafios Iniciais

1. Crie um script Python que leia o nome de uma pessoa e mostre uma mensagem de boas-vindas de acordo com o valor digitado.



```
desafio.py - C:/Users/lucas/Documents/curso_Py
File Edit Format Run Options Window Help
print('===== Desafio 01 =====')
nome = input ('Qual é o seu nome?')
print ('Boas-vindas, ', nome)
```

2. Crie um script em Python que leia o dia, o mês e o ano de nascimento de uma pessoa e mostre uma mensagem com a data formatada.



```
desafio2.py - C:\Users\lucas\Documents\curso_Python_guanabara\scripts-python\desafio2.p.
File Edit Format Run Options Window Help
dia = input('Qual dia você nasceu?')
mes = input ('Qual o mês você nasceu?')
ano = input ('Qual o ano do seu nascimento?')

print ('Você nasceu em: ', dia, ' de ', mes, ' no ano de ', ano)
```

3. Crie um script em Python que leia os dois números e mostre a soma entre eles.

```
desafio3.py - C:/Users/lucas/Documents/curso_Python_guanabara/sci  
File Edit Format Run Options Window Help  
num1 = int(input('Digite o primeiro número: '))  
num2 = int(input('Digite o segundo número: '))  
soma = num1 + num2  
  
print('A soma entre os dois números é: ', soma)
```

---

## Exercício 00 – Soma de Números

Crie um programa que leia dois números e mostre a soma entre eles.

The screenshot shows the PyCharm interface with the file 'aula06b.py' open. The code defines two float variables, n1 and n2, which are input from the user. Their sum is calculated and printed. Below the editor is the 'Run' tool window, which shows the command to run 'aula06b', the user's inputs (5 and 10), and the resulting output (A soma entre 5.0 e 10.0 é: 15.0).

```
aula06b.py
1 n1 = float(input('Digite o número 1: '))
2 n2 = float(input('Digite o número 2: '))
3 soma = n1 + n2
4
5 print('A soma entre {} e {} é: {}'.format(n1, n2, soma))
6
```

Run: aula06b ×  
C:\Users\lucas\PycharmProjects\pythonExercicios\venv\Scripts\python.exe aula06b.py  
Digite o número 1: 5  
Digite o número 2: 10  
A soma entre 5.0 e 10.0 é: 15.0

## Exercício 01 – Print('Hello World!')

Crie um programa que escreva ‘Olá Mundo’ na tela.

The screenshot shows the PyCharm interface with the file 'ex001.py' open. It contains a single print statement with the message 'Olá, Mundo!'. Below the editor is the 'Run' tool window, which shows the command to run 'ex001', the output 'Olá, Mundo!', and the message 'Process finished with exit code 0'.

```
ex001.py
1 msg = 'Olá, Mundo!'
2 print(msg)
```

Run: ex001 ×  
C:\Users\lucas\PycharmProjects\pythonExercicios\venv\Scripts\python.exe ex001.py  
Olá, Mundo!  
Process finished with exit code 0

## Exercício 02 – Input

Faça um programa que leia o nome de um usuário e mostre uma mensagem de boas-vindas.

The screenshot shows the PyCharm interface with the file 'ex002.py' open. It uses input to get a name from the user and then prints a welcome message. Below the editor is the 'Run' tool window, which shows the command to run 'ex002', the user input 'lucas', and the output 'Prazer em te conhecer, lucas'.

```
pythonExercicios > ex002.py
ex001.py × ex002.py ×
1 nome = input('Digite seu nome: ')
2 print('Prazer em te conhecer, ', nome)
3
```

Run: ex002 ×  
C:\Users\lucas\PycharmProjects\pythonExercicios\venv\Scripts\python.exe ex002.py  
Digite seu nome: lucas  
Prazer em te conhecer, lucas

```
name = input('Digite aqui o seu nome: ')
print('Prazer em te conhecer, {}!'.format(name))
```

Run: ex002

```
C:\Users\lucas\PycharmProjects\pythonExercicios
Digite aqui o seu nome: LUCAS
Prazer em te conhecer, Lucas!
```

### Exercício 05 – Input + Funções

Faça um programa que leia algo pelo teclado e mostre na tela o seu tipo primitivo e todos as informações possíveis sobre esse número.

```
msg = input('Digite uma palavra/valor: ')
```

```
1
2
3
4
5
6
7
8
9
```

```
A mensagem é maiúscula? msg.isupper()
A mensagem é maiúscula? msg.isupper()
A mensagem é minúscula? msg.islower()
A mensagem é Númerica? msg.isnumeric()
A mensagem é Alfabética? msg.isalpha()
A mensagem é alfanumérica? msg.isalnum()
```

Run: aula06b

```
C:\Users\lucas\PycharmProjects\pythonExercicios\venv\Scripts\p
Digite uma palavra/valor: LUCAS
A mensagem é maiúscula? True
A mensagem é maiúscula? True
A mensagem é minúscula? False
A mensagem é Númerica? False
A mensagem é Alfabética? True
A mensagem é alfanumérica? True
```

### Exercício 06 – Antecessor e Sucessor

Faça um programa que leia um número inteiro e mostre na tela o seu sucessor e o seu antecessor

```
num1 = int(input('Digite um número: '))
print('O valor do sucessor é: {} \n'
      'e o seu antecessor é: {}'.format(num1 + 1, num1 - 1))
```

```
Digite um número: 2
O valor do sucessor é: 3
e o seu antecessor é: 1
```

---

### Exercício 07 – Dobro, Triplo e Raíz Quadrada

Faça um algoritmo que leia um número e mostre o dobro, o triplo e a sua raiz quadrada.

```
num1 = int(input('Digite um número: '))
print('O seu dobro: {} \n'
      'O seu triplo: {} \n'
      'A raiz quadrada: {} \n'
      'A raiz quadrada no Pow: {}'
      .format(num1 * 2, num1 * 3, num1 ** (1/2), pow(num1, (1/2))))
```

```
Digite um número: 25
O seu dobro: 50
O seu triplo: 75
A raiz quadrada: 5.0
A raiz quadrada no Pow: 5.0
```

### Exercício 08 – Média dos alunos

Desenvolva um programa que leia as duas notas de um aluno, calcule e mostre sua média

```
nota1 = int(input('Digite a primeira nota: '))
nota2 = int(input('Digite a segunda nota: '))
media = (nota1 + nota2)/2
print('A média final do aluno é: {}'.format(media))
```

```
teste ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\S
Digite a primeira nota: 3
Digite a segunda nota: 2
A média final do aluno é: 2.5
```

---

### Exercício 09 – Conversos de medidas

Escreva um programa que leia um valor em metros e o exiba convertido em centímetros e milímetros

```
valor = float(input('Digite o valor em metros: '))
print('O valor em centímetros é de {:.2f} centímetros \n '
      'e o valor em milímetros é de {:.2f} milímetros'
      .format(valor*100, valor*1000))
```

```
teste ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe -u "C:/Users/lucas/PycharmProjects/pythonProject/venv/Scripts/teste.py"
Digite o valor em metros: 1
O valor em centímetros é de 100.00 centímetros
e o valor em milímetros é de 1000.00 milímetros
```

```
medida = float(input('Digite uma medida em metros: '))
cm = medida * 100
mm = medida * 1000
dam = medida / 10
hm = medida / 100
km = medida / 1000
polegada = medida * 39.37
jardas = medida * 1.094
print('O valor de {} m em:\n'
      'Centímetros: {} cm\n'
      'Milímetros: {} mm\n'
      'Decâmetro: {} \n'
      'Hectômetro: {} hm\n'
      'Kilometro: {} km\n'
      'Polegada: {:.4f} polegadas\n'
      'Jardas: {:.4f} jardas\n'
      .format(medida, cm, mm, dam, hm, km, polegada, jardas))
```

### Exercício 09b - Tabuada

Faça um programa que leia um número inteiro qualquer e mostre na tela a sua tabuada

```
tabuada = int(input('Digite o número para tabuada: '))
print('{} x 1 = {}'.format(tabuada, tabuada * 1))
print('{} x 2 = {}'.format(tabuada, tabuada * 2))
print('{} x 3 = {}'.format(tabuada, tabuada * 3))
print('{} x 4 = {}'.format(tabuada, tabuada * 4))
print('{} x 5 = {}'.format(tabuada, tabuada * 5))
print('{} x 6 = {}'.format(tabuada, tabuada * 6))
print('{} x 7 = {}'.format(tabuada, tabuada * 7))
print('{} x 8 = {}'.format(tabuada, tabuada * 8))
print('{} x 9 = {}'.format(tabuada, tabuada * 9))
```

teste ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts  
Digite o número para tabuada: 2  
2 x 1 = 2  
2 x 2 = 4  
2 x 3 = 6  
2 x 4 = 8  
2 x 5 = 10  
2 x 6 = 12  
2 x 7 = 14  
2 x 8 = 16  
2 x 9 = 18  
2 x 10 = 20

### Exercício 10 – Mostrar valor

Crie um programa que leia quanto dinheiro uma pessoa tem na carteira e mostre quantos dólares ela pode comprar. Considere USD 1,00 = R\$ 3,27

```
carteira = float(input('Quanto você tem na carteira: '))
dolar = 3.27
print('Você tem R$ {:.2f} na carteira e pode comprar '
      'USD {:.3f} em dólar.'
      .format(carteira, carteira/dolar))
```

teste ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python  
Quanto você tem na carteira: 3.27  
Você tem R\$ 3.27 na carteira e pode comprar USD 1.000 em dólar.

### Exercício 11 – Conversor de área (Tinta de parede)

Faça um programa que leia a largura de uma parede em metros, calcule sua área e a quantidade de tinta necessária para pintá-la. Sabendo que cada litro de tinta, pinta uma área de 2m<sup>2</sup>.

```
base = float(input('A base da parede: '))
largura = float(input('A largura da parede: '))
tinta = 2
area = base * largura
print('A parede tem {} metros \nVocê precisa de {} litros de tinta'
      .format(area, area/tinta))
```

```
teste ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
A base da parede: 4
A largura da parede: 4
A parede tem 16.0 metros
Você precisa de 8.0 litros de tinta
```

### Exercício 12 – Preço x Desconto

Faça um algoritmo que leia o preço de um produto e mostre seu novo preço com 5% de desconto.

```
produto = float(input('Digite o valor do produto: '))
desconto = float(input('Digite o valor do desconto em %: '))/100

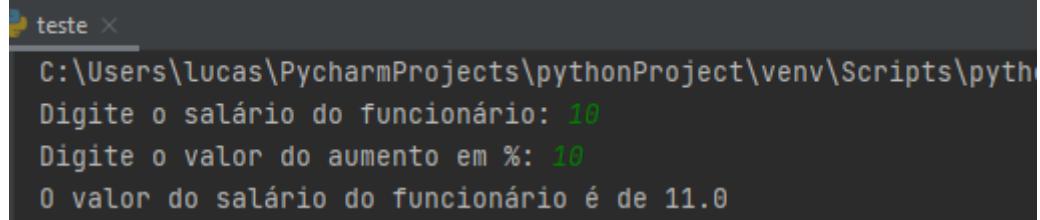
print('O valor do desconto será de {} reais e o valor do produto final é: {} reais'
      .format(desconto * produto, produto - (produto * desconto)))
```

```
teste ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Lucas,
Digite o valor do produto: 20
Digite o valor do desconto em %: 5
O valor do desconto será de 1.0 reais e o valor do produto final é: 19.0 reais
```

### Exercício 13 – Aumento do Salário do Funcionário

Faça um algoritmo que leia o salário de um funcionário e mostre seu novo salário com 15% de aumento.

```
salario = float(input('Digite o salário do funcionário: '))
aumento = float(input('Digite o valor do aumento em %: '))/100
print('O valor do salário do funcionário é de {}'
      .format(salario + (salario * aumento)))
```



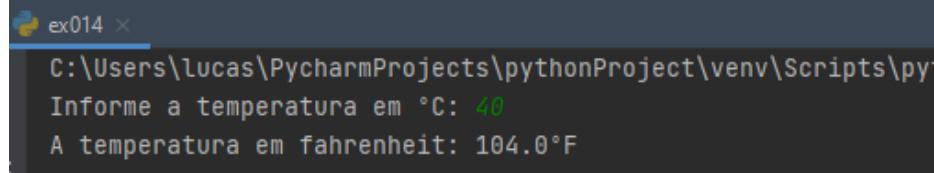
```
teste ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
Digite o salário do funcionário: 10
Digite o valor do aumento em %: 10
O valor do salário do funcionário é de 11.0
```

### Exercício 14 - Transformador de Temperatura

Crie um algoritmo que mostre a temperatura em °C e faça a transformação dessa unidade de medida para °F

```
temperatura = float(input('Informe a temperatura em °C: '))
fahrenheit = ((temperatura * 9 / 5) + 32)

print('A temperatura em fahrenheit: {}°F'.format(fahrenheit))
```

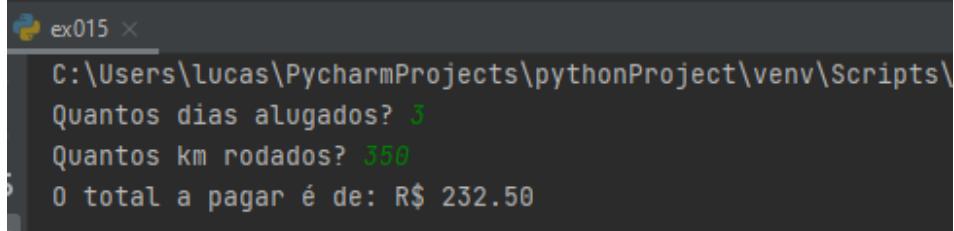


```
ex014 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
Informe a temperatura em °C: 40
A temperatura em fahrenheit: 104.0°F
```

### Exercício 15 – Cálculo do carro alugado

Escreva um programa que pergunte a quantidade de km rodado por um carro alugado e a quantidade de dias pelos quais ele foi alugado. Calcule o preço a pagar, sabendo que o carro custa R\$ 60,00 por dia e R\$ 0,15 por km rodado.

```
dias = int(input('Quantos dias alugados? '))
km = float(input('Quantos km rodados? '))
totalValor = (dias * 60) + (km * 0.15)
print('O total a pagar é de: R$ {:.2f}'.format(totalValor))
```



```
ex015 x
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\
Quantos dias alugados? 3
Quantos km rodados? 350
O total a pagar é de: R$ 232.50
```

### Exercícios de Importações de Módulos

#### Exercício 16 – Truncate – Corte de decimal

Crie um programa que leia um número Real qualquer pelo teclado e mostre na tela a sua porção inteira. Por exemplo: Um número 6,127 tem a sua parcela inteira o 6 (Utilizar o truncate).

```
import math

numreal = float(input('Digite o número: '))
print('O valor digitado foi {}. A porção inteira desse número é: {}'.format(numreal, math.trunc(numreal)))
```

```
from math import trunc

numreal = float(input('Digite o número: '))
print('O valor digitado foi {}. A porção inteira desse número é: {}'.format(numreal, trunc(numreal)))
```

### Exercício 17 – Cálculo do Comprimento do Triângulo

Faça um programa que leia o comprimento do cateto oposto e do cateto adjacente de um triângulo retângulo. Calcule e mostre o comprimento da hipotenusa.

```
import math

catoposto = float(input('Digite o valor do Cateto Oposto: '))
catadj = float(input('Digite o valor do Cateto Adjacente: '))
hipotenusa = math.hypot(catoposto, catadj)
print('O valor da Hipotenusa é {:.2f}'.format(hipotenusa))
```

```
ex016 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\pyi
Digite o valor do Cateto Oposto: 2
Digite o valor do Cateto Adjacente: 2.5
O valor da Hipotenusa é 3.20
```

---

### Exercício 18 – Cálculo do Ângulo do Triângulo

Faça um programa que leia um ângulo e mostre na tela o valor do seno, cosseno e tangente desse ângulo

É necessário transformar o “30” em GRAUS. Porque é um grau de ângulo e não centígrados. Por isso, precisa do **RADIANS**

```
from math import sin, cos, tan, radians

angulo = float(input('Digite o valor do Ângulo: '))
cosseno = cos(radians(angulo))
seno = sin(radians(angulo))
tangente = tan(radians(angulo))

print('O valor do Cosseno é {:.2f} \nSeno {:.2f} \nTangente é {:.2f}'.
      format(cosseno, seno, tangente))
```

```
ex018 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
Digite o valor do Ângulo: 30
O valor do Cosseno é 0.87
Seno 0.50
Tangente é 0.58
```

### Exercício 19 - Sorteio

Um professor quer sortear um dos seus quatro alunos para apagar o quadro. Faça um programa que ajude ele, lendo o nome deles e escrevendo o nome do escolhido.

The screenshot shows the PyCharm interface with two main sections: the code editor and the run terminal.

**Code Editor:**

```
1  from random import choice
2  n1 = str(input('Primeiro Aluno: '))
3  n2 = str(input('Segundo Aluno: '))
4  n3 = str(input('Terceiro Aluno: '))
5  n4 = str(input('Quarto Aluno: '))
6  lista = [n1, n2, n3, n4]
7  escolhido = choice(lista)
8
9  print('A lista de alunos é: {}'.format(lista))
10 print('O Aluno(a) escolhido foi: {}'.format(escolhido))
```

**Run Terminal:**

```
Run: ex019 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\
Primeiro Aluno: laura
Segundo Aluno: tony
Terceiro Aluno: nina
Quarto Aluno: lucas
A lista de alunos é: ['laura', 'tony', 'nina', 'lucas']
O Aluno(a) escolhido foi: tony
```

### Exercício 20 – Sorteio com Ordem

O mesmo professor do desafio anterior quer sortear a ordem das apresentações dos trabalhos dos alunos. Faça um programa que leia o nome dos quatro alunos e mostre a ordem sorteada.

```
from random import shuffle

n1= str(input('Primeiro aluno: '))
n2 = str(input('Segundo aluno: '))
n3 = str(input('Terceiro aluno: '))
n4 = str(input('Quarto aluno: '))
lista = [n1, n2, n3, n4]
shuffle(lista)

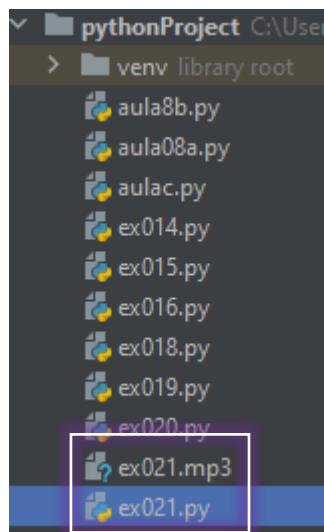
print('A ordem da apresentação será: {}'.format(lista))
```

```
ex020 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe ex020.py
Primeiro aluno: laura
Segundo aluno: nina
Terceiro aluno: lucas
Quarto aluno: tony
A ordem da apresentação será: ['lucas', 'laura', 'nina', 'tony']
```

### Exercício 21 – Tocador de Mp3

Faça um programa em Python que abra e reproduza um áudio de um arquivo MP3.

Primeiro é necessário ter uma música .mp3 separada. Após eu criar o meu projeto, eu preciso dar um ctrl + c na música, e dar um ctrl + v no PyCharm. Ele vai criar um arquivo novo:



É preciso colocar essas linhas de código:

The screenshot shows the PyCharm interface. The code editor window is open with the file 'ex021.py' containing the following code:

```
1 import pygame
2 pygame.mixer.init()
3 pygame.mixer.music.load('ex021.mp3')
4 pygame.mixer.music.play()
5 while(pygame.mixer.music.get_busy()): pass
```

Below the code editor is the run terminal window titled 'Run: ex021'. It shows the command line output:

```
C:\Users\lucas\PycharmProjects\pythonProject
pygame 2.0.0 (SDL 2.0.12, python 3.9.1)
Hello from the pygame community. https://www.pygame.org
Process finished with exit code -1
```

## Exercícios de Manipulações de Strings

### Exercício 022 – Manipulações de Strings + Funções

Faça um programa que leia o nome completo de uma pessoa e mostre:

- O nome com todas as letras maiúsculas
- O nome com todas as letras minúsculas
- Quantas letras ao todo (sem considerar os espaços)
- Quantas letras tem o primeiro nome

É necessário ao pedir a variável, precisa já iniciar com o strip para ele **eliminar os espaços antes e depois.**

Para encontrar a resposta da letra C, é preciso pedir o tamanho da variável onde se procure por espaços e substitua(replace) por um espaço nulo.

Para encontrar a resposta da Letra D, é preciso pedir o tamanho da variável sendo ela separada e aí, conforme o split tornaria ['Lucas', 'Pereira', 'Galves'] basta apenas para ele pegar a posição[0] que seria a primeira.

The screenshot shows the PyCharm IDE interface. The top window is titled "ex022.py" and contains the following Python code:

```
1 nome = str(input('Digite seu nome: ')).strip()
2 print('Seu nome maiúsculo é: {}'.format(nome.upper()))
3 print('Seu nome minúsculo é: {}'.format(nome.lower()))
4 print('Seu nome tem {} letras'.format(len(nome.replace(' ', ''))))
5 print('Seu primeiro nome tem {} letras'.format(len(nome.split()[0])))
```

The bottom window is titled "Run: ex022" and shows the terminal output for the code execution:

```
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
Digite seu nome: Lucas Pereira Galves
Seu nome maiúsculo é: LUCAS PEREIRA GALVES
Seu nome minúsculo é: lucas pereira galves
Seu nome tem 18 letras
Seu primeiro nome tem 5 letras
```

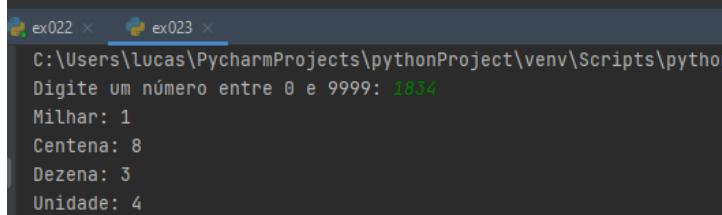
### Exercício 023 – [Erro]

Faça um programa que leia um número de 0 a 9999 e mostre na tela cada um dos dígitos separados. Ex: 1834 (unidade: 4 / dezena: 3 / centena: 8 / milhar: 1).

Muito cuidado:

Nesse primeiro print parece funcionar, mas dará erro

```
numero = str(input('Digite um número entre 0 e 9999: ')).strip()
numero = numero.replace(' ', '')
print('Milhar: {}'.format(numero.split()[0]))
print('Centena: {}'.format(numero.split()[1]))
print('Dezena: {}'.format(numero.split()[2]))
print('Unidade: {}'.format(numero.split()[3]))
```



Se eu digitar um número com menos de 4 dígitos terei problemas

```
Digite um número entre 0 e 9999: 25
Milhar: 2
Centena: 5
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\pythonProject\ex023.py", line 5, in <module>
    print('Dezena: {}'.format(numero.split()[2]))
IndexError: list index out of range
```

Essa é o jeito único e correto para resolver esse exercício

2345	100
45	23(div inteira)

23	10
3(Resto)	2(divinteira)

Explicando, eu primeiro pego o número 2345 e faço uma divisão inteira nele (//) e o resultado dessa divisão inteira que é 34 (na verdade da 23,45 mas como é inteira eu ignoro o ,45 e fico somente com o 23). Daí eu pego novamente esse 23 e faço a divisão (o módulo (%) ainda age como uma divisão) só que nessa divisão eu quero saber o resto dessa divisão, que é 3. Causando assim, sendo, o módulo de 2345 na casa da centena seja o 3.

```

numero = int(input('Digite um número entre 0 e 9999: '))
milhar = numero // 1000 % 10
centena = numero // 100 % 10
dezena = numero // 10 % 10
unidade = numero // 1 % 10
print('Analizando o numero {}'.format(numero))
print('O valor do milhar: {}'.format(milhar))
print('O valor da centena: {}'.format(centena))
print('O valor da dezena: {}'.format(dezena))
print('O valor da unidade: {}'.format(unidade))

```

```

ex022 × ex023 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Script
Digite um número entre 0 e 9999: 2345
Analizando o numero 2345
O valor do milhar: 2
O valor da centena: 3
O valor da dezena: 4
O valor da unidade: 5

```

Outra solução e mais correta, aliás, mas que continuará apresentando erro:

```

numero = int(input('Digite um número entre 0 e 9999: '))
num = str(numero)
print('Analizando o numero {}'.format(numero))
print('O valor do milhar {}'.format(num[0]))
print('O valor da centena {}'.format(num[1]))
print('O valor da dezena {}'.format(num[2]))
print('O valor da unidade {}'.format(num[3]))

```

```

ex022 × ex023 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Script
Digite um número entre 0 e 9999: 1834
Analizando o numero 1834
O valor do milhar 1
O valor da centena 8
O valor da dezena 3
O valor da unidade 4

```

```

Digite um número entre 0 e 9999: 26
Analizando o numero 26
O valor do milhar 2
O valor da centena 6
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\pythonProject\ex023.py", line 6, in <module>
    print('O valor da dezena {}'.format(num[2]))
IndexError: string index out of range

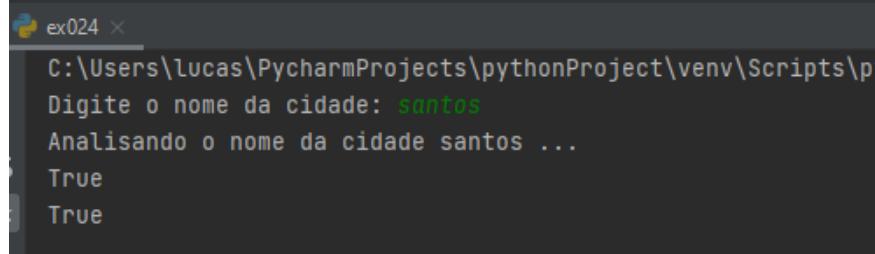
```

Portanto, somente a solução matemática para esse exercício ao invés da solução por String.

### Exercício 024 – Strip

Crie um programa que leia o nome de uma cidade e diga se ela **começa** ou **não** com o nome = “SANTO”.

```
cidade = str(input('Digite o nome da cidade: ')).strip()
print('Analizando o nome da cidade {} ...'.format(cidade))
print(cidade[: 5].upper() == 'SANTO')
#Com "in":|
analise = 'SANTO' in cidade[: 5].upper()
print(analise)
```



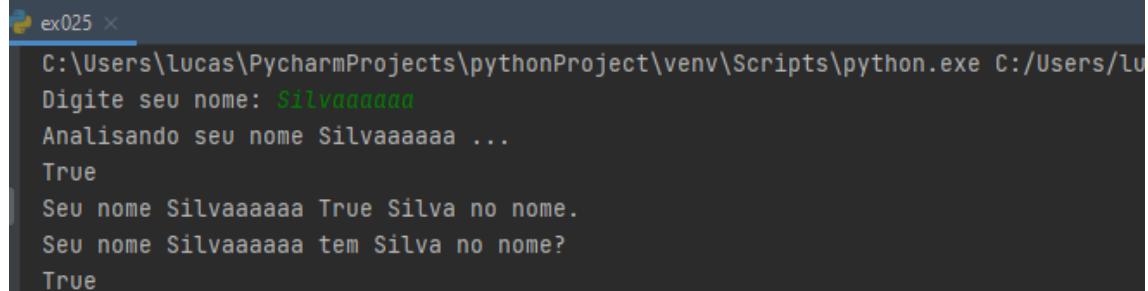
```
ex024 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/pythonProject/ex024.py
Digite o nome da cidade: santos
Analizando o nome da cidade santos ...
True
True
```

---

### Exercício 025 – Análise de Nome

Faça um programa que leia o nome de uma pessoa e diga se ela **tem** “SILVA” no nome. True or False.

```
nome = str(input('Digite seu nome: ')).strip()
print('Analizando seu nome {} ...'.format(nome))
analiseNome = 'SILVA' in nome.upper()
print(analiseNome)
#sem necessidade de variavel:
print('Seu nome {} {} Silva no nome.'.format(nome, 'SILVA' in nome.upper()))
print('Seu nome {} tem Silva no nome?\n{}'.format(nome, 'silva' in nome.lower()))
```



```
ex025 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/pythonProject/ex025.py
Digite seu nome: Silvaaaaaaaa
Analizando seu nome Silvaaaaaaaa ...
True
Seu nome Silvaaaaaaaa True Silva no nome.
Seu nome Silvaaaaaaaa tem Silva no nome?
True
```

## Exercício 026 – Manipulações de Nomes

Crie um programa que leia que leia uma frase qualquer e:

- A. Quantas vezes aparece a letra “A”
- B. Em que posição a letra “A” aparece pela primeira vez
- C. Em que posição a letra “A” aparece pela última vez

A letra C foi a mais complicada, pois é necessário fazer um ‘rfind’ que é para ele ir procurando sempre a direita e trazer o mais da direita possível.

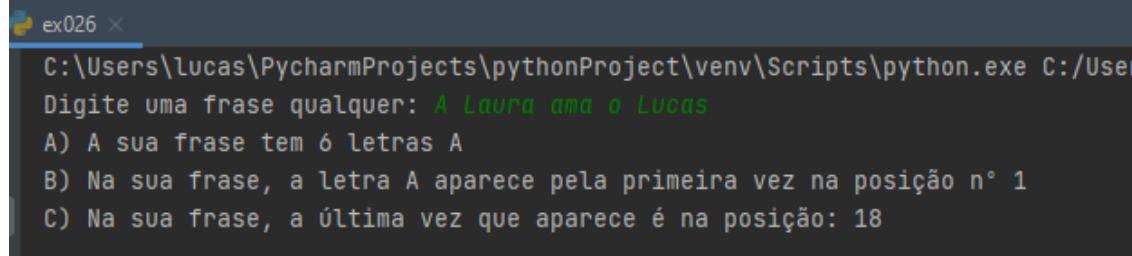
Vale lembrar que em Python, a primeira posição é a zero, mas, para humanos é muito estranho algo na posição zero, portanto, após ele encontrar, adicionamos + 1 para apresentar “mais legível” para o usuário.

```
frase = str(input('Digite uma frase qualquer: ')).strip()

print('A) A sua frase tem {} letras A '.format(frase.upper().count('A')))

print('B) Na sua frase, a letra A aparece pela primeira vez na posição nº {}'.format(frase.upper().find('A')+1))

print('C) Na sua frase, a última vez que aparece é na posição: {}'.format(frase.upper().rfind('A')+1))
```

The screenshot shows the PyCharm terminal window titled 'ex026'. The command line shows the path 'C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/User' followed by the user input 'Digite uma frase qualquer: A Laura ama o Lucas'. The program then prints three lines of output corresponding to the questions: 'A) A sua frase tem 6 letras A', 'B) Na sua frase, a letra A aparece pela primeira vez na posição nº 1', and 'C) Na sua frase, a última vez que aparece é na posição: 18'.

### Exercício 027 – Manipulações de Nomes, pt2.

Faça um programa que leia o nome inteiro de uma pessoa e mostre em seguida, o primeiro e o último nome da pessoa separadamente. Vale lembrar que se a pessoa tiver o nome com 5 sobrenomes, deve aparecer o primeiro e o último. Ex: Ana Maria de Souza (primeiro: Ana / último: Souza).

Pode utilizar a função Length dentro como se fosse um parâmetro/posição da lista para ele retornar a última posição do nome. Não sabia que era possível utilizar um método como uma posição de lista. Portanto, para achar o penúltimo nome e o nome do meio também segue essa lógica.

```
nome = str(input('Digite seu nome inteiro: ')).strip()
nomeSplitado = nome.split()
print(nomeSplitado)
print('O seu primeiro nome é: {}'.format(nomeSplitado[0]))
print('O seu último nome é: {}'.format(nomeSplitado[len(nomeSplitado)-1]))
```

```
ex027 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/pythonProject/ex027.py
Digite seu nome inteiro: laura viana da rocha
['laura', 'viana', 'da', 'rocha']
O seu primeiro nome é: laura
O seu último nome é: rocha
```

## Exercícios de Estrutura Condicional (If-Else)

### Exercício 028 – Número escolhido x Jogador

Escreva um programa que faça o computador “pensar” em um número inteiro entre 0 e 5 e peça para o usuário tentar descobrir qual foi o número escolhido pelo computador. E escreva na tela se o usuário venceu ou perdeu.

```
import random
from time import sleep

numint = random.randint(0, 5) #Computador emite um número aleatório
#print(numint)
print('---' * 20)
numUser = int(input('Digite um número entre 0 e 5: ')) #Usuário digita um número aleatório
print('---' * 20)
print('Processando os dados...')
sleep(1)
if numUser == numint: #Faz a Validação entre Computador x Usuário
    print('Seu número {} é igual ao {} escolhido pelo computador. \nParabéns!'.format(numUser, numint))
else:
    print('Seu número {} não é o mesmo número {} que o computador escolheu. \nTente novamente!'.format(numUser, numint))

ex028 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmPro
-----
Digite um número entre 0 e 5: 1
-----
Processando os dados...
Seu número 1 não é o mesmo número 3 que o computador escolheu.
Tente novamente!
```

### Exercício 029 – Radar de Velocidade

Escreva um programa que leia a velocidade de um carro na Estrada.

- A. Se ele ultrapassar a velocidade de 80km/h, mostre uma mensagem dizendo que ele foi multado.
- B. A multa vai custar 7 reais por cada km excedente, acima do limite.

Mostre na tela os valores.

```
from time import sleep

limite = 80
velocidade = float(input('Valor do carro no radar: '))
print('Analizando velocidade...')
sleep(0.5)
if velocidade > limite:
    print('Você foi multado! Velocidade: {} km/h'.format(velocidade))
    print('Valor da multa de R$ {:.2f}'.format((velocidade - limite) * 7))
else:
    print('Boa viagem.')

ex029 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Us
Valor do carro no radar: 150
Analizando velocidade...
Você foi multado! Velocidade: 150.0 km/h
Valor da multa de R$ 490.00
```

### Exercício 030 – Par ou Ímpar

Crie um programa que leia um número inteiro e mostre na tela se ele é PAR ou ÍMPAR:

```
numero = int(input('Digite um número qualquer: '))
if numero % 2 == 0:
    print('O seu número {} escolhido é PAR!'.format(numero))
else:
    print('O seu número {} escolhido é ÍMPAR!'.format(numero))

else

Python ex030 x
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/pythonProject/ex030.py
Digite um número qualquer: 11
O seu número 11 escolhido é ÍMPAR!
```

---

### Exercício 031 – Cálculo da Viagem

Crie um programa que pergunte a distância de uma viagem em KM. Calcule o preço da passagem, cobrando R\$ 0,50 por KM para viagens de até 200km. E R\$ 0,45 para viagens acima de 200 km. Mostre o valor.

```
viagem = float(input('Digite a distância da viagem: '))

if viagem <= 200:
    print('O valor da viagem será de: R$ {:.2f}'.format(viagem * 0.50))
else:
    print('O valor da viagem será de: R$ {:.2f}'.format(viagem * 0.45))

else

Python ex031 x
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/pythonProject/ex031.py
Digite a distância da viagem: 210
O valor da viagem será de: R$ 94.50
```

### Exercício 032 – Ano Bиссexto

Faça um programa que leia um ano qualquer e mostre se ele é BISSEXTO ou não.

```
from datetime import date

ano = int(input('Digite o ano que você quer analisar. Se ano atual, digite 0: '))
if ano == 0:
    ano = date.today().year
if ano % 4 == 0 and ano % 100 != 0 or ano % 400 == 0:
    print('O ano {} é BISSEXTO'.format(ano))
else:
    print('O ano {} NÃO é BISSEXTO'.format(ano))

if ano == 0
    ex032 ×

C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas
Digite o ano que você quer analisar. Se ano atual, digite 0: 0
0 ano 2020 é BISSEXTO
```

### Exercício 033 – Maior ou Menor

Faça um programa que leia 3 números e diga qual é o maior e qual o menor.

```
num1 = float(input('Digite o primeiro número: '))
num2 = float(input('Digite o segundo número: '))
num3 = float(input('Digite o terceiro número: '))

if num1 > num2 and num1 > num3:
    print('Número 1 é maior.')
elif num2 > num1 and num2 > num3:
    print('Número 2 é maior')
else:
    print('Número 3 é maior.')

if num1 < num2 and num1 < num3:
    print('Número 1 é menor.')
elif num2 < num1 and num2 < num3:
    print('Número 2 é menor')
else:
    if num1 > num2 and num1 > num3
        ex033 ×

C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas
Digite o primeiro número: 500
Digite o segundo número: 100
Digite o terceiro número: 2000
Número 3 é maior.
Número 2 é menor
```

### Exercício 034 – Aumento Salarial

Escreva um programa que pergunte o salário do funcionário e calcule o valor do seu aumento.

Para salários acima de R\$ 1.250,00, calcule um aumento de 10%

Para salários abaixo ou iguais, o aumento será de 15%

```
salario = float(input('Digite seu salário: '))

if salario >= 1250:
    print('Seu salário corrigido será de: R$ {}'.format((salario * 0.10) + salario))
else:
    print('Seu salário corrigido será de R$ {}'.format((salario * 0.15) + salario))

else
```

ex034 ×

C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas  
Digite seu salário: 1000  
Seu salário corrigido será de R\$ 1150.0

---

### Exercício 035 – Cálculo do Triângulo, pt3.

Desenvolva um programa que leia o comprimento de 3 retas e diga ao usuário se elas podem ou não formar um triângulo. Dica: Existe um princípio matemático que não permite que seja triângulo, se uma reta for muito maior que as outras 2, não será possível.

```
cateto1 = float(input('Digite a primeira reta: '))
cateto2 = float(input('Digite a segunda reta: '))
base = float(input('Digite a terceira reta: '))

if cateto1 + cateto2 > base:
    print('Pode formar um triângulo')
else:
    print('Não é possível formar um triângulo')

if cateto1 + cateto2 > base
```

ex035 ×

C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas  
Digite a primeira reta: 2  
Digite a segunda reta: 4  
Digite a terceira reta: 6  
Não é possível formar um triângulo

## Exercícios de Estruturas Aninhadas

### Exercício 036

Escreva um programa para aprovar o empréstimo bancário para a compra de uma casa. O programa vai perguntar o valor da casa, o salário do comprador e quantos anos ele vai pagar.

Calcule o valor da prestação mensal, sabendo que ele não pode exceder 30% do salário ou então o empréstimo será negado.

```
"""Programa para calcular empréstimo da casa"""

valor = float(input('Digite o valor da casa: R$ '))
salario = float(input('Digite o salário do comprador: R$ '))
anos = int(input('Digite em quantos anos: '))

parcelas = valor / (anos * 12)

print('Parcelas no valor de R$ {:.2f}'.format(parcelas))

if parcelas > (salario * 0.30):
    print('\nO seu salário em 30% é de R$ {:.2f}'.format(salario * 0.30))
    print('Não é possível realizar o empréstimo.')
    print('\nFaltam R$ {:.2f}'.format((salario * 0.30) - parcelas))
else:
    print('\nVocê pode realizar o empréstimo.')
    print('O seu salário em 30% é de R$ {:.2f}'
          '\nPortanto, te sobra R$ {:.2f}'
          .format(salario * 0.30, (salario * 0.30) - parcelas))

C:\Users\lucas\PycharmProjects\pythonProject1>
Digite o valor da casa: R$ 80000
Digite o salário do comprador: R$ 10000
Digite em quantos anos: 7
Parcelas no valor de R$ 952.38

Você pode realizar o empréstimo.
O seu salário em 30% é de R$ 3000.00
Portanto, te sobra R$ 2047.62
```

### Exercício 037

Escreva um programa que leia um número inteiro qualquer e peça para o usuário escolher qual será a base de conversão:

- 1 – para binário
- 2 – para octal
- 3 – para hexadecimal

```
"""Conversor de Inteiros"""
n1 = int(input('Digite o número inteiro a ser convertido: '))
print('''Escolha uma das opções para converter:
[1] Converte para Binário
[2] Converte para Octal
[3] Converte para Hexadecimal''')

opcao = int(input('Sua opção: '))

if opcao == 1:
    #Conversor de Inteiros em binário
    print('O número {} convertido em binário é: {}'.format(n1, bin(n1)[2:]))
elif opcao == 2:
    #Conversor de Inteiros em Octal
    print('O numero {} convertido em Octal é: {}'.format(n1, oct(n1)[2:]))
elif opcao == 3:
    #Conversos de Inteiros em Hexadecimal
    print('O número {} convertido em Hexadecimal é: {}'.format(n1, hex(n1)[2:].upper()))
else:
    print('Opção Inválida, rode o programa novamente.')
```

```
C:\Users\lucas\PycharmProjects\pythonProject\venv
Digite o número inteiro a ser convertido: 360
Escolha uma das opções para converter:
[1] Converte para Binário
[2] Converte para Octal
[3] Converte para Hexadecimal
Sua opção: 2
O numero 360 convertido em Octal é: 550
```

### Exercício 038

Escreva um programa que leia dois números inteiros e compare-os, mostrando na tela uma mensagem:

- O primeiro valor é maior;
- O segundo valor é maior;
- Não existe valor maior, os dois são iguais

```
"""Programa que compara dois números inteiros e retorna infos de igualdade"""

n1 = int(input('Digite o Primeiro Número Inteiro: '))
n2 = int(input('Digite o Segundo Número Inteiro: '))

if n1 > n2:
    print('O nº {} é maior que o nº {}'.format(n1, n2))
elif n1 < n2:
    print('O nº {} é maior que o nº {}'.format(n2, n1))
else:
    print('O nº {} é igual ao nº {}'.format(n1, n2))

else
```

ex038 ×

C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas

Digite o Primeiro Número Inteiro: 12

Digite o Segundo Número Inteiro: 12

O nº 12 é igual ao nº 12.

### Exercício 039 – Serviço Militar

Faça um programa que leia o ano de nascimento de um jovem e informe, de acordo com a sua idade:

- Se ele ainda vai se alistar ao serviço militar.
- Se está na hora de alistar.
- Se ele já passou da idade de alistar.

Seu programa também deverá mostrar quanto tempo falta ou quanto tempo passou do prazo.

```
"""Cálculo da Idade para Alistamento Militar"""
from datetime import date

anoAtual = date.today().year

sexo = int(input('Qual o seu sexo?\n    \'\\nEscolha as opções\'\n    \'\\n[1]Masculino \''
    '\n[2]Feminino\'\n    \'\\nSeleção: '))

if sexo == 1:
    nascimento = int(input('Ano de Nascimento: '))
    idade = anoAtual - nascimento
    alistamento = nascimento + 18
    if idade < 16:
        print('Você tem {} anos de idade.\n        \'\\nFique tranquilo, você ainda tem tempo até se alistar.\'
        \'\\nData prevista para {}''.format(idade, alistamento))
    elif 16 <= idade < 18:
        print('Você tem {} anos de idade.\n        \'\\nPrepare-se, você se alistará no exercíco em breve!\'
        \'\\nVocê vai se alistar no ano de {}''.format(idade, alistamento))
    elif idade == 18:
        print('Você tem {} anos de idade.\n        \'\\nAtenção! Está na hora de alistar.'.format(idade))

else:
    print('Você tem {} anos de idade.\n        \'\\nVocê já passou do tempo de se alistar.\'
        \'\\nVocê já se alistou/deveria ter se alistado no ano de {}''.
        format(idade, alistamento))
else:
    print('Seu alistamento não é necessário, obrigado.')
```

```
Qual o seu sexo?  
Escolha as opções  
[1]Masculino  
[2]Feminino  
Selecione: 1  
Ano de Nascimento: 1994  
Você tem 26 anos de idade.  
Você já passou do tempo de se alistar.  
Você já se alistou/deveria ter se alistado no ano de 2012
```

## Exercício 040

Crie um programa que leia duas notas de um aluno e calcule a sua média, mostrando uma mensagem no final, de acordo com a média atingida:

- Média abaixo de 5.0: “Reprovado”
- Média entre 5.0 e 6.9: “Recuperação”
- Média maior que 7.0: “Aprovado”

```
"""Cálculo da Média do Aluno na escola"""
from statistics import mean
from time import sleep

nota1 = float(input('Digite a primeira nota: '))
nota2 = float(input('Digite a segunda nota: '))

# media = (nota1 + nota2) / 2 Não use!
media = mean([nota1, nota2])

print('A média do aluno foi de: {:.1f}'.format(media))

if media >= 7.0:
    print('Aluno aprovado!')
elif 5.0 <= media < 7.0:
    print('Aluno em Recuperação!')
    print('Aluno está em recuperação')
    sleep(0.9)
    nota3 = int(input('Digite o valor da Nota da Prova de Recuperação: '))
    sleep(0.9)
    if nota3 > 7.0:
        print('Parabéns, você recuperou a nota!')
        print('\nAluno aprovado!')
    else:
        print('Não recuperou a nota, aluno reprovado.')
else:
    print('Não recuperou a nota, aluno reprovado.')
else:
    print('Sua média foi inferior a 5.0.')
    print('\nReprovado.')
```

```
Digite a primeira nota: 8
Digite a segunda nota: 4
A média do aluno foi de: 6.0
Aluno em Recuperação!
Aluno está em recuperação
Digite o valor da Nota da Prova de Recuperação: 5
Não recuperou a nota, aluno reprovado.
```

### Exercício 041

A confederação nacional de natação precisa de um programa que leia o ano de nascimento de um atleta e mostre sua categoria de acordo com a sua idade:

- Até 9 anos: 'Mirim'
- Até 14 anos: 'Infantil'
- Até 20 anos: 'Junior'
- Até 25 anos: 'Sênior'
- Acima: 'Master'

```
from datetime import date

nascimento = int(input('Qual ano vcê nasceu? '))
hoje = date.today().year
idade = hoje - nascimento

print('Você tem {} anos. \nSua Classificação: '.format(idade))

if idade <= 9:
    print('Você está selecionado na categoria: Mirim')
elif idade <= 14:
    print('Você está selecionado na categoria: Infantil')
elif idade <= 19:
    print('Você está selecionado na categoria: Júnior')
elif idade <= 25:
    print('Você está selecionado na categoria: Sênior')
else:
    print('Você está selecionado na categoria: Master')

elif idade <= 25
ex041 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
Qual ano vcê nasceu? 2015
Você tem 6 anos.
Sua Classificação:
Você está selecionado na categoria Mirim
```

## Exercício 042

Refaça o exercício 035 dos triângulos. Acrescentando o recurso de mostrar que tipo de triângulo será formado:

- Equilátero: Todos os lados são iguais
- Isósceles: Dois lados são iguais
- Escaleno: Todos os lados são diferentes

```
cateto1 = float(input('Digite a primeira reta: '))
cateto2 = float(input('Digite a segunda reta: '))
base = float(input('Digite a base: '))

if cateto1 + cateto2 > base:
    print('Pode formar um triângulo. ', end='')
    if cateto1 == cateto2 == base:
        print('Seu triângulo é EQUILÁTERO, todos os lados iguais.')
    elif cateto1 == cateto2 or cateto1 == base or cateto2 == base:
        print('Seu triângulo é ISÓSCELES, com dois lados iguais.')
    else:
        print('Seu triângulo é ESCALENO, nenhum lado é igual.')
else:
    print('Não é possível formar um triângulo')

if cateto1 + cateto2 > base
```

ex042 ×

C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/  
Digite a primeira reta: 5  
Digite a segunda reta: 5  
Digite a base: 5  
Pode formar um triângulo. Seu triângulo é EQUILÁTERO, todos os lados iguais.

### Exercício 043

Desenvolva um programa que leia o peso e a altura de uma pessoa, calcule seu Índice de Massa Corporal (IMC) e mostre seu status, de acordo com a tabela abaixo:

- Abaixo de 18.5: 'Abaixo do Peso'
- Entre 18.5 e 25: 'Peso ideal'
- Entre 25 a 30: 'Sobrepeso'
- Entre 30 a 40: 'Obesidade'
- Acima de 40: 'Obesidade Mórbida'

```
"""Cálculo do IMC - Índice de Massa Corpórea"""

pergunta1 = str(input('Você sabe o que é o IMC? '
                     '\n[1] Sim'
                     '\n[2] Não'
                     '\nSelecione: ')).strip()

if pergunta1 == '2':
    print('O IMC (Índice de Massa Corporal) é um cálculo '
          'que ajuda a avaliar se a pessoa está dentro do '
          'seu peso ideal, de acordo com a altura.')

pergunta2 = str(input('Vamos calcular ou seu IMC? '
                     '\n[1] Sim'
                     '\n[2] Não'
                     '\nSelecione: ')).strip()

if pergunta2 == '1':
    peso = float(input('Digite seu peso em kgs: '))
    altura = float(input('Digite sua altura em metros: '))
    IMC = (peso / (altura * altura))
    print('Você tem o IMC de {:.2f} kg/m². \n'.format(IMC), end='')
    if IMC < 18.5:
        print('Situação: Abaixo do Peso.')
    elif 18.5 <= IMC <= 25.0:
        print('Situação: Peso Ideal.')
    elif 25.0 <= IMC <= 30.0:
        print('Situação: Peso Ideal.')
    elif 25.0 <= IMC <= 30.0:
        print('Situação: Sobre peso.')
    elif 30.0 <= IMC <= 40.0:
        print('Situação: Obesidade.')
    else:
        print('Situação: Obesidade Mórbida.')
else:
    print('Você deveria calcular o seu IMC!')
```

```
Seleciona: 1
Vamos calcular ou seu IMC?
[1] Sim
[2] Não
Seleciona: 1
Digite seu peso em kgs: 97
Digite sua altura em metros: 1.96
Você tem o IMC de 25.25 kg/m².
Situação: Sobre peso.
```

## Exercício 044

Elabore um programa que calcule o valor a ser pago por um produto considerando o seu preço normal e condição de pagamento:

- À vista com dinheiro/ cheque: 10% de desconto
- À vista no cartão: 5% de desconto
- Em até 2x no cartão: Preço normal
- A partir de 3x ou mais no cartão: 20% de juros no valor total.

```
preco = float(input('Digite o preço do produto: R$ '))
pagamento = int(input('Forma de pagamento: '
                      '\n[1] À vista com dinheiro'
                      '\n[2] Cheque'
                      '\n[3] À vista no cartão'
                      '\n[4] Em até 2x no cartão'
                      '\n[5] A partir de 3x ou mais no cartão'
                      '\nSelecione: '))

if pagamento == 1 or pagamento == 2:
    if pagamento == 1:
        print('Você selecionou a opção Pagamento À vista com dinheiro'
              '\nO valor a pagar é de R$ {:.2f}'.format(preco - (preco * 0.10)))
    else:
        print('Você selecionou a opção Pagamento em Cheque'
              '\nO valor a pagar é de R$ {:.2f}'.format(preco - (preco * 0.05)))
elif pagamento == 3:
    print('Você selecionou a opção Pagamento À vista no Cartão'
          '\nO valor a pagar é de R$ {:.2f}'.format(preco - (preco * 0.05)))
elif pagamento == 4:
    print('Você selecionou a opção Pagamento em até 2x no Cartão'
          '\nO valor a pagar é de R$ {:.2f}'
          '\nCada parcela no valor de R$ {:.2f}'.format(preco, preco / 2))
else:
    print('Você selecionou a opção Pagamento em 3x ou mais no Cartão'
          '\nJuros de R$ {:.2f}'
          '\nO valor a pagar é de R$ {:.2f}'
          .format(preco * 0.20, preco + (preco * 0.20)))
    parcelas = int(input('Digite o valor de parcelas: '))
    if parcelas < 3:
        print('Erro! Não pode parcelar menor que 3x. Operação Cancelada.')
    else:
        print('O valor de cada parcela é de R$ {:.2f}'.format((preco + (preco * 0.20)) / parcelas))

print('Obrigado e volte sempre!')
```

Digite o preço do produto: R\$ 50  
Forma de pagamento:  
[1] À vista com dinheiro  
[2] Cheque  
[3] À vista no cartão  
[4] Em até 2x no cartão  
[5] A partir de 3x ou mais no cartão  
Selecione: 4  
Você selecionou a opção Pagamento em até 2x no Cartão  
O valor a pagar é de R\$ 50.00  
Cada parcela no valor de R\$ 25.00  
Obrigado e volte sempre!

## Exercício 045

Crie um programa que faça o computador jogar ‘Jokenpô’ com você.

Código disponível no Pycharm.

## Exercícios de Estrutura de Repetição For

### Exercício 046

Faça um programa que mostre na tela uma contagem regressiva para o estouro dos fogos de artifícios. Indo de 10 até 0, com uma pausa de 1 segundo entre eles.

```
from time import sleep

for regressiva in range(10, 0, -1):
    print('Contagem regressiva: {}'.format(regressiva))
    sleep(1)
print('FELIZ ANO NOVO!')  
  
for regressiva in range(10, 0, ...)  
ex047 ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts  
Contagem regressiva: 10  
Contagem regressiva: 9  
Contagem regressiva: 8  
Contagem regressiva: 7  
Contagem regressiva: 6  
Contagem regressiva: 5  
Contagem regressiva: 4  
Contagem regressiva: 3  
Contagem regressiva: 2  
Contagem regressiva: 1  
FELIZ ANO NOVO!
```

### Exercício 047

Crie um programa que mostre na tela todos os números **pares** que estão compreendidos entre 1 e 50.

```
for pares in range(2, 50 + 1, 2):
    print(pares)
print('FIM')
|
```

ex047 ×  
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe  
2  
4  
6  
8  
10  
12  
14  
16  
18  
20  
22  
24  
26

Nesse exercício, poderia ter um “if” fazendo a condição se o módulo de um número par fosse resto 0, mostraria somente os números pares. Mas, sabendo que número par, é de 2 em 2, eu posso adicionar esse intervalo no parâmetro para iteração e pular de dois em dois e assim evitar que meu programa consuma muito processamento.

### Exercício 048

Faça um programa que calcule a **soma** entre todos os números **ímpares** que são **múltiplos de três** e que se encontram no intervalo de 1 até 500.

```
soma = 0
cont = 0
for multiplos in range(1, 501, 2):
    if multiplos % 3 == 0:
        cont += 1
        soma += multiplos
print('Valor da soma dos {} números: {}'.format(cont, soma))
print('end')

for multiplos in range(1, 501, ... > if multiplos % 3 == 0
| ex048 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
Valor da soma dos 83 números: 20667
end
```

### Exercício 049

Refaça o exercício do Desafio 009b, mostrando a tabuada de um número que o usuário escolher, só que agora utilizando o laço For.

```
tabuada = int(input('Digite um número: '))

for numero in range(0, 11):
    print('{} x {} = {}'.format(tabuada, numero, tabuada * numero))
print('FIM')

for numero in range(0, 11)
    ex049 x
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
Digite um número: 9
9 x 0 = 0
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90
FIM
```

### Exercício 050

Desenvolva um programa que leia **seis** números inteiros e mostre a **soma** apenas daqueles que forem pares. Se o valor digitado for ímpar, desconsidere

```
soma = 0
cont = 0
for numero in range(0, 6):
    pares = int(input('Digite o {}º número: '.format(numero + 1)))
    if pares % 2 == 0:
        cont += 1
        soma += pares
print('O total de números pares é: {}'
      '\nA soma dos números pares é: {}'.format(cont, soma))
print('FIM')

for numero in range(0, 6)
    ex050 x
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
Digite o 1º número: 1
Digite o 2º número: 2
Digite o 3º número: 3
Digite o 4º número: 4
Digite o 5º número: 5
Digite o 6º número: 6
O total de números pares é: 3
A soma dos números pares é: 12
FIM
```

### Exercício 051

Desenvolva um programa que leia o primeiro termo e a razão de uma PA (Progressão Aritmética). No final, mostre os **10 primeiros** termos dessa progressão. ~Contagem de números pulando, usando intervalos~ razão é intervalo.

```
primeiro = int(input('Digite um número para Calcular a PA: '))
razao = int(input('Digite a razão: '))
decimo = primeiro + (10 - 1) * razao #Fórmula para calcular o enésimo termo da PA.
#Se for 20 primeiros, se troca o 10 por 20.
count = 0

for contador in range(primeiro, decimo + razao, razao):
    count += 1
    print('O {}º termo é {}'.format(count, contador))
print('FIM')

for contador in range(primeiro,...
```

ex051 ×

C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/P

Digite um número para Calcular a PA: 10

Digite a razão: 50

O 1º termo é 10

O 2º termo é 60

O 3º termo é 110

O 4º termo é 160

O 5º termo é 210

O 6º termo é 260

O 7º termo é 310

O 8º termo é 360

O 9º termo é 410

O 10º termo é 460

FIM

### Exercício 052

Faça um programa que leia um número inteiro e diga se ele é ou não um número primo.

The screenshot shows the PyDev IDE interface with the following details:

- Code Area:** Displays the Python code for Exercise 052. The code uses ANSI escape sequences to print colored output (green for prime numbers, red for non-prime numbers).
- Title Bar:** Shows the file name "ex052.py".
- Toolbars:** Includes standard PyDev toolbars for navigation and debugging.
- Variables View:** Shows the current variable state:
  - contador = {int} 1
  - num = {int} 3
  - tot = {int} 0
- Console View:** Displays the output of the program:

```
Connected to pydev debugger (build 203.5981.165)
Digite um número: >? 3
```

Exercício resolvido apenas com Debug, indicar a linha para [Toggle Breakpoint](#) (selecionar a linha com CTRL + F8 ) e Clicar para rodar o [Debug](#) (CTRL + F5) ou clicar no inseto do Debug.

### Exercício 053

Faça um programa que leia uma frase qualquer e diga se ela é um **palíndromo**, desconsiderando os espaços.

Ex: *Apos a Sopa* // *A Sacada da Casa* // *A torre da derrota* // *o lobo ama o bolo* // *anotaram a data da maratona*.

```
palindromo = str(input('Digite uma frase: ')).strip().upper() #espacos eliminados
palavras = palindromo.split() #gerou uma lista
frase = ''.join(palavras) #juntou a lista sem espaco
inverso = frase[::-1]

print('O inverso de {} é {}'.format(frase, inverso))
if inverso == frase:
    print('A frase é um palíndromo!')
else:
    print('A frase não é um palíndromo')
```

```
ex053 x
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/Pych
Digite uma frase: anotaram a data da maratona
O inverso de ANOTARAMADATADAMARATONA é ANOTARAMADATADAMARATONA
A frase é um palíndromo!
Digite uma frase: o lebo ama o bolo
O inverso de OLOBOAMAOBOL0 é OLOBOAMAOBOL0
A frase é um palíndromo!
```

### Exercício 054 – Maioridade 21

Crie um programa que leia o ano de nascimento de 7 pessoas. No final, mostre quantas pessoas ainda não atingiram a maioridade e quantos já são maiores. Maioridade 21 anos.

```
from datetime import date
maioridade = 21
hoje = date.today().year
countMaior = 0
countMenor = 0
for c in range(0, 7):
    nascimento = int(input('Digite o ano de nascimento do {}: '.format(c + 1)))
    if hoje - nascimento > maioridade:
        countMaior += 1
    else:
        countMenor += 1
print('0 total de maiores: {} '
      '\n0 total de menores: {}'.format(countMaior, countMenor))
```

```
ex054 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/luc
Digite o ano de nascimento do 1: 1994
Digite o ano de nascimento do 2: 2015
Digite o ano de nascimento do 3: 2013
Digite o ano de nascimento do 4: 2012
Digite o ano de nascimento do 5: 1990
Digite o ano de nascimento do 6: 2019
Digite o ano de nascimento do 7: 2020
0 total de maiores: 2
0 total de menores: 5
```

### Exercício 055 – Maior e Menor com For

Crie um programa que leia o peso de 5 pessoas. No final, mostre qual foi o maior e o menor peso lidos.

```
maior = 0
menor = float('inf')

for c in range(0, 5):
    peso = float(input('Digite peso da {}° pessoa: '.format(c + 1)))
    if peso > maior:
        maior = peso
    if peso < menor:
        menor = peso
print('O maior peso foi o: {} kg'.format(maior))
print('O menor peso foi o: {} kg'.format(menor))
```

```
ex055 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:
Digite peso da 1° pessoa: 80
Digite peso da 2° pessoa: 20
Digite peso da 3° pessoa: 50
Digite peso da 4° pessoa: 100
Digite peso da 5° pessoa: 25
O maior peso foi o: 100.0 kg
O menor peso foi o: 20.0 kg
```

### **Exercício 056 – Características com For**

Desenvolva um programa que leia o nome, idade e sexo de 4 pessoas. No final, do programa mostre:

- a) A média de idade do grupo.
- b) Qual é o nome do homem mais velho.
- c) Quantas mulheres tem menos de 20 anos de idade.

Fazer exercício posteriormente com OBJETO ou LISTA.

## Exercícios de Estrutura de Repetição While

### Exercício 057

Faça um programa que leia o sexo de uma pessoa, mas só aceite os valores ‘M’ ou ‘F’. Caso esteja errado, peça a digitação **novamente** até ter um valor **correto**.

```
sexo = str(input('Digite o sexo da pessoa: ')).strip().upper()[0]

while sexo not in 'MmFm':
    sexo = str(input('Inválido! Digite o sexo da pessoa: ')).strip().upper()[0]
print('O sexo digitado foi: {}'.format(sexo))

while sexo not in 'MmFm':
    ex057 x
    C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/luc
    Dige...  
Inválido! Digite o sexo da pessoa: p  
Inválido! Digite o sexo da pessoa: g  
Inválido! Digite o sexo da pessoa: u  
Inválido! Digite o sexo da pessoa: r  
Inválido! Digite o sexo da pessoa: m  
O sexo digitado foi: M.
```

### Exercício 058

Melhore o jogo do DESAFIO 028, onde o computador vai “pensar” em um número entre 0 e 10. Só que agora o jogador vai tentar adivinhar **até acertar**. Mostrando no final, **quantos palpites** foram necessários para **vencer** o computador.

```
from random import randint

computador = randint(1, 10)
usuario = int(input('Digite um valor entre 1 a 10: '))
tentativas = 1

while usuario != computador:
    if usuario > 10:
        print('\033[41mEI! PARE DE ROUBAR! O LIMITE É ATÉ 10!\033[m')
        '\nTente novamente, computador escolheu: {}'.format(computador))
    else:
        print('Quase! Você pensou no número diferente do computador: {}'.format(computador))
    if usuario != computador:
        tentativas += 1
    computador = randint(1, 10)
    usuario = int(input('Digite um valor entre 1 a 10: '))

print('\033[1;30;42mParabéns!\033[m Você chutou o nº {} e adivinhou o nº {} do computador!'
      '\nTotal de tentativas: {}'.format(usuario, computador, tentativas))

while usuario != computador > if usuario > 10
ex058 ×
```

Digite um valor entre 1 a 10: 6  
Quase! Você pensou no número diferente do computador: 2  
Digite um valor entre 1 a 10: 3  
Quase! Você pensou no número diferente do computador: 9  
Digite um valor entre 1 a 10: 12  
EI! PARE DE ROUBAR! O LIMITE É ATÉ 10!

## Exercício 059

Crie um programa que leia dois valores e mostre um menu na tela:

- [1] somar
- [2] multiplicar
- [3] maior
- [4] novos números
- [5] sair do programa

O programa deverá realizar a operação solicitada em cada caso.

```
numero1 = float(input('Digite um número: '))
numero2 = float(input('Digite outro número: '))
parar = 's'
somar = subtrair = multiplicar = dividir = maior = menor = 0
from time import sleep

while parar != 'N':
    menu = int(input('Selecione um item abaixo:
                      '\n[ 1 ] Somar
                      '\n[ 2 ] Subtrair
                      '\n[ 3 ] Multiplicar
                      '\n[ 4 ] Dividir
                      '\n[ 5 ] Maior e Menor
                      '\n[ 6 ] Digitar novos números
                      '\n[ 7 ] Sair do Programa
                      '\nSelecionar: '))

    if menu == 1:
        somar = numero1 + numero2
        print('Os dois números somados: {}'.format(somar))
    elif menu == 2:
        subtrair = numero1 - numero2
        print('A subtração dos dois números: {}'.format(subtrair))
    elif menu == 3:
        ...
    elif menu == 3:
        multiplicar = numero1 * numero2
        print('A multiplicação dos dois números: {}'.format(multiplicar))
    elif menu == 4:
        dividir = numero1 / numero2
        print('A divisão do {} pelo {} é: {}'.format(numero1, numero2, dividir))
    elif menu == 5:
        if numero1 > numero2:
            print('O número 1 ({}) é maior que o número 2 ({})'.format(numero1, numero2))
        else:
            print('O número 2 ({}) é maior que o número 1 ({})'.format(numero2, numero1))
    elif menu == 6:
        numero1 = float(input('Digite um número: '))
        numero2 = float(input('Digite outro número: '))
    elif menu == 7:
        parar = 'N'
    else:
        print('Opção Inválida.')
    sleep(1)
print('FIM')
```

## Exercício 060

Faça um programa que leia um número qualquer e mostre seu fatorial. Ex: 5! = 5x4x3x2x1 = 120. Ex2: 10! (lê-se: 10 factorial: 10x9x8x7x6x5x4x3x2x1 = valor

```
numero = int(input('Digite um número para calcularmos o fatorial: '))
atingiu = False
fatorial = 0
proximo = numero

while atingiu != True:
    fatorial = numero * (proximo - 1)
    numero = fatorial
    proximo -= 1
    if proximo == 1:
        atingiu = True
print('Fatorial: {}'.format(fatorial))
```

```
numero = int(input('Digite um número para calcularmos o fatorial: '))
c = numero
f = 1

print('Calculando o {}! = '.format(numero), end='')
while c > 0:
    print('{} '.format(c), end='')
    print('x ' if c > 1 else ' = ', end='')
    f *= c
    c -= 1
print('{}'.format(f))
print('\nFIM')
```

```
#Resolvendo o 060 com Módulo

#Módulo para calcular o fatorial:
from math import factorial

n = int(input('Digite um número para cálculo do fatorial: '))
fatorial = factorial(n)
print('O fatorial do número {}! é: {}'.format(n, fatorial))
```

### Exercício 061

Refaça o DESAFIO 051, lendo o primeiro termo e a razão de uma PA, mostrando os 10 primeiros termos da progressão aritmética usando o While.

```
numero = int(input('Digite um número: '))
razao = int(input('Digite a razão: '))
fim = False
c = 0
pa = 0
while c < 10:
    print('O {}º termo da PA é: {}'.format(c + 1, numero))
    pa = numero + razao
    numero = pa
    c += 1
print('\nfim')
```

```
ex060b × ex061 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\py
Digite um número: 5
Digite a razão: 2
O 1º termo da PA é: 5
O 2º termo da PA é: 7
O 3º termo da PA é: 9
O 4º termo da PA é: 11
O 5º termo da PA é: 13
O 6º termo da PA é: 15
O 7º termo da PA é: 17
O 8º termo da PA é: 19
O 9º termo da PA é: 21
O 10º termo da PA é: 23
```

## Exercício 062

Melhore o desafio 061, perguntando ao usuário se ele quer mostrar mais alguns termos. O programa encerra quando ele disser que quer mostrar 0 termos.

```
p_termo = int(input('Primeiro termo: '))
razao = int(input('Razão: '))
termos = int(input('Qtd Termos: '))
|
while termos > 0:
    print(p_termo, end=' ')
    p_termo += razao
    termos -= 1
    if termos == 0:
        termos = int(input('\nAumentar mais números na sequência: '))
```

```
ex062 x
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/pythonProject/ex062.py
Primeiro termo: 5
Razão: 5
Qtd Termos: 1
5
Aumentar mais números na sequência: 10
10 15 20 25 30 35 40 45 50 55
Aumentar mais números na sequência:
```

```
primeiro = int(input('Primeiro Termo: '))
razão = int(input('Razão: '))
termo = primeiro
cont = 1
total = 0
mais = 10

while mais != 0:
    total = total + mais
    while cont <= total:
        print('{} -> '.format(termo), end=' ')
        termo += razão
        cont += 1
    print('PAUSA')
    mais = int(input('Quantos termo você quer mostrar a mais? '))
print('Progressão finalizada com {} termos mostrados.'.format(total))
```

```
while mais != 0 > while cont <= total
ex062 x ex062b x
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/pythonProject/ex062b.py
Gerador de PA, por: Guanabara
=====
Primeiro Termo: 2
Razão: 2
2 ->4 ->6 ->8 ->10 ->12 ->14 ->16 ->18 ->20 ->PAUSA
Quantos termo você quer mostrar a mais? 3
22 ->24 ->26 ->PAUSA
Quantos termo você quer mostrar a mais? 4
```

### Exercício 063

Escreva um programa que leia um número n inteiro qualquer e mostre na tela os n primeiros elementos de uma Sequência Fibonacci. Ex: 0 -> 1 -> 1 -> 2 -> 3 -> 5 -> 8

```
print('-' * 30)
print('Sequência de Fibonacci')
print('-' * 30)

n = int(input('Termos: '))
t1 = 0
t2 = 1
print('~' * 30)
print('{} ~> {}'.format(t1, t2), end=' ')
count = 3
while count <= n:
    t3 = t1 + t2
    print(' ~> {}'.format(t3), end=' ')
    t1 = t2
    t2 = t3
    count += 1

while count <= n
    ex063 ×
    nsole | Variables | ⚡ | ⏪ ⏴ ⏵ ⏹ ⏹ | 📁
    Console
    -----
    -----
    Sequência de Fibonacci
    -----
    Termos: ~~~~~
    0 ~> 1 ~> 1 ~> 2 ~> 3
    Process finished with exit code -1
```

### Exercício 064

Crie um programa que leia vários números inteiros pelo teclado. O programa só vai parar quando o usuário digitar 999, que é a condição de parada. No final, mostre quantos números foram digitados e qual foi a soma entre eles. Desconsiderando o valor do Flag.

```
from time import sleep

numero = soma = count = 0
while numero != 999:
    numero = int(input('Digite um número: [999 Encerra o programa]: '))
    if numero != 999:
        soma += numero
        count += 1
    else:
        print('Encerrando o programa...')
        sleep(1)

print('A soma entre os números: {}'
      '\nA quantidade de números: {}'.format(soma, count))

while numero != 999
```

ex064 ×

```
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/
Digite um número: 10
Digite um número: 10
Digite um número: 10
Digite um número: 20
Digite um número: 999
Encerrando o programa...
A soma entre os números: 50
A quantidade de números: 4
```

### Exercício 066

Crie um programa que leia vários números inteiros pelo teclado. O programa só vai parar quando o usuário digitar o valor ‘999’, que é a condição de parada. No final, mostre quantos números foram digitados e qual foi a soma entre eles. Desconsidere o valor do flag.

```
num = soma = cont = 0

while True:
    num = int(input('Digite um número [999 Encerra o programa]: '))
    if num == 999:
        break
    soma += num
    cont += 1
print(f'A soma dos {cont} é: {soma}')
```

```
ex066 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe
Digite um número [999 Encerra o programa]: 15
Digite um número [999 Encerra o programa]: 15
Digite um número [999 Encerra o programa]: 15
Digite um número [999 Encerra o programa]: 999
A soma dos 3 é: 45
```

### Exercício 065

Crie um programa que leia vários números inteiros pelo teclado. No final da execução, mostre a média entre todos os valores e qual foi o maior e o menor valores lidos. O programa deve perguntar ao usuário se ele quer ou não continuar a digitar valores.

```
continuar = 's'
contar = 0
qtdNumeros = qtd = somar = media = maior = menor = 0

while continuar != 'N':
    contador = int(input('Quantos números deseja: '))
    while contador > contar:
        numero = int(input('Digite um número({}): '.format(contar + 1)))
        contar += 1
        somar += numero
        qtd += 1
        if qtd == 1:
            maior = menor = numero
        else:
            if numero > maior:
                maior = numero
            else:
                menor = numero
        media = somar
        qtdNumeros += contador
    print('A média entre os valores: {:.2f}'
          '\nO maior valor lido: {}'
          '\nO menor valor lido: {}'.format(media / qtdNumeros, maior, menor))
    continuar = str(input('Deseja continuar? [s/n]')).strip().upper()
    while continuar != 'N':

        continuar = str(input('Deseja continuar? [s/n]')).strip().upper()
        contar = 0
print('Fim')

while continuar != 'N' > while contador > contar
ex065 ×
Quantos números deseja: 2
Digite um número(1): 4
Digite um número(2): 6
A média entre os valores: 5.00
O maior valor lido: 6
```

## Exercício 067

Faça um programa que mostre a tabuada de vários números, **um de cada vez**, para cada valor digitado pelo usuário. O programa será interrompido quando o número solicitado for **negativo**.

```
print('-' * 30)
print('GERADOR DE TABUADA COM WHILE')
print('-' * 30, '\n')

cont = 0

while True:
    tabuada = int(input('Digite um número: '))
    if tabuada < 0:
        break
    while cont <= 10:
        print(f'{tabuada} x {cont} = {tabuada * cont}')
        cont += 1
    cont = 0
print('-' * 20)

ex067 x
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
-----
Digite um número: -2
Obrigado por utilizar o Gerador de Tabuadas.
Volte Sempre!
```

## Com WHILE + FOR

```
print('-' * 30)
print('GERADOR DE TABUADA COM FOR')
print('-' * 30, '\n')

while True:
    tabuada = int(input('Digite um número: '))
    if tabuada < 0:
        break
    for c in range(0, 10):
        print(f'{tabuada} * {c} = {tabuada * c}')
    print('-' * 20)

print('Obrigado por utilizar o Gerador de Tabuadas. '
      '\n\033[42mVolte Sempre!\033[m')

while True
ex067b x
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
-----
Digite um número: -1
Obrigado por utilizar o Gerador de Tabuadas.
Volte Sempre!
```

## Exercício 068

Faça um programa que jogue Par ou Ímpar com o computador. O jogo só será interrompido quando o jogador **perder**. Mostre na tela o total de vitórias consecutivas que ele conquistou no final do jogo.

```
from random import randint

num = cont = 0
perdeu = False
parOuImpar = ''
while True:
    numUsuario = int(input('Digite um número: '))
    numComputador = randint(0, 10)
    while parOuImpar not in 'PpIi':
        parOuImpar = str(input('Par ou ímpar [P/I]: ')).strip().upper()[0]
    soma = numUsuario + numComputador
    if parOuImpar == 'P' and soma % 2 == 0:
        print('Jogador Venceu!')
        escolha = 'par'
    elif parOuImpar == 'I' and soma % 2 == 1:
        print('Jogador Venceu!')
        escolha = 'ímpar'
    else:
        print('GAME OVER! HA-HA-HA - Computador Venceu!')
        perdeu = True
    print(f'Total de Vitórias Consecutivas do Jogador: {cont}')
    if soma % 2 == 0:
        escolha = 'par'

    print(f'Total de Vitórias Consecutivas do Jogador: {cont}')
    if soma % 2 == 0:
        escolha = 'par'
    else:
        escolha = 'ímpar'
    if perdeu == True:
        break
    cont += 1
    parOuImpar = ''
    print('-' * 40)

print(f'\nO computador escolheu {numComputador}.')
print(f'\nO jogador digitou {numUsuario}.')
print(f'\nE a soma {soma} é {escolha}')
print('-' * 40)

if cont >= 2:
    print(f'\033[32mIncrível! Você venceu {cont} vezes\033[m')
elif cont >= 1:
    print(f'\033[34mUau, você venceu {cont} vezes\033[m')
else:
    print('\033[31mVocê é uma vergonha.\033[m')
```

```
C:\Users\lucas\PycharmProjects\pythonProject\ver
Digite um número: 2
Par ou ímpar [P/I]: p
Jogador Venceu!
-----
Digite um número: 1
Par ou ímpar [P/I]: p
Jogador Venceu!
-----
Digite um número: 3
Par ou ímpar [P/I]: p
GAME OVER! HA-HA-HA - Computador Venceu!
Total de Vitórias Consecutivas do Jogador: 2

O computador escolheu 8.
O jogador digitou 3.
E a soma 11 é ímpar
-----
Incrivel! Você venceu 2 vezes
```

### Exercício 069

Crie um programa que leia a idade e o sexo de várias pessoas. A cada pessoa cadastrada, o programa deverá perguntar ao usuário se quer ou não continuar. No final, mostre:

- A. Quantas pessoas tem mais de 18 anos
- B. Quantos homens foram cadastrados
- C. Quantas mulher tem menos de 20 anos

```
cont = contIdade = contHomens = contMulher20 = 0

print('=-' * 15, '\nCadastrando de Pessoas')
print('=-' * 15)

while True:
    idade = int(input('\033[1mDigite a idade:\033[m '))
    if idade >= 18:
        contIdade += 1
    sexo = ''
    while sexo not in 'MmFf':
        sexo = str(input('\033[1mDigite o sexo [M/F]:\033[m')).strip().upper()[0]
    if sexo == 'M':
        contHomens += 1
    if sexo == 'F' and idade < 20:
        contMulher20 += 1
    cont += 1
    print('=-' * 15)
    continuar = ''
    while continuar not in 'SN':
        continuar = str(input('\033[7mDeseja continuar? [S/N]:\033[m')).strip().upper()[0]
    if continuar == 'N':
        break
    while continuar not in 'SN':
        continuar = str(input('\033[7mDeseja continuar? [S/N]:\033[m')).strip().upper()[0]
    if continuar == 'N':
        break

print(f'\nQuantidade de pessoas cadastradas: {cont}')
print(f'\nQuantidade de Pessoas maiores que 18: {contIdade}')
print(f'\nQuantidade de Homens cadastrados: {contHomens}')
print(f'\nQuantidade de Mulher menores de 20 anos: {contMulher20}'')
```

```
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/Py
=====
Cadastro de Pessoas
=====
Digite a idade: 26
Digite o sexo [M/F]: F
=====

Deseja continuar? [S/N]: S
Digite a idade: 24
Digite o sexo [M/F]: m
=====

Deseja continuar? [S/N]: a
Deseja continuar? [S/N]: a

Deseja continuar? [S/N]: s
Digite a idade: s
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\pythonProject\ex069.py", line 7, in <module>
    idade = int(input('Digite a idade:'))
ValueError: invalid literal for int() with base 10: 's'

Process finished with exit code 1
```

## Exercício 070

Crie um programa que leia o nome e o preço de vários produtos. O programa deverá perguntar se o usuário vai continuar. No final, mostre:

- A. Qual é o total gasto na compra
- B. Quantos produtos custam mais de 1000
- C. Qual é o nome do produto mais barato

```
print('~~~' * 10, '\nPet Shop Baracão', '~~~')
print('~~~' * 10)

cont = total = prod100 = prodCaro = prodBarato = maior = menor = 0

while True:
    nome = str(input('Nome do Produto: ')).strip()
    preco = float(input('Preço: R$ '))
    if cont == 0:
        maior = menor = preco
        prodCaro = prodBarato = nome
    elif preco > maior:
        maior = preco
        prodCaro = nome
    else:
        menor = preco
        prodBarato = nome
    cont += 1
    total += preco
    if preco > 100:
        prod100 += 1
    continuar = ' '
    while continuar not in 'SN':
        continuar = str(input('Deseja continuar? [S/N]: ')).upper().strip()[0]
    if continuar == 'N':
        break
    print()
print(f'~~~' * 15,
      f'\nO valor Total Gasto na compra: R$ {total:.2f}',
      f'\n{prod100} produto(s) custa(m) mais de R$ 100,00',
      f'\nO produto mais caro foi {prodCaro}: R$ {maior:.2f}',
      f'\nO produto mais barato foi {prodBarato}: R$ {menor:.2f}')
```

```
~~~~~  
Pet Shop Baracão ~_.~  
~~~~~  
Nome do Produto: Shampoo  
Preço: R$ 10  
Deseja continuar? [S/N]: s  
Nome do Produto: Vacina de Raiva  
Preço: R$ 200  
Deseja continuar? [S/N]: s  
Nome do Produto: Petiscos  
Preço: R$ 5  
Deseja continuar? [S/N]: n  
~~~~~  
O valor Total Gasto na compra: R$ 215.00  
1 produto(s) custa(m) mais de R$ 100,00  
O produto mais caro foi Vacina de Raiva: R$ 200.00  
O produto mais barato foi Petiscos: R$ 5.00  
  
Process finished with exit code 0
```

### **Exercício 071**

Crie um programa que simule o funcionamento de um caixa eletrônico. No início, pergunte ao usuário qual será o valor a ser sacado (número inteiro) e o programa vai informar quantas cédulas de cada valor serão entregues. Considere que o caixa possui cédulas de R\$ 50, R\$ 20, R\$ 10 e R\$ 1

## Exercícios de Estrutura de Tuplas

### Exercício 072

Crie um programa que tenha uma tupla 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.

```
tupla = ('zero', 'um', 'dois', 'tres', 'quatro', 'cinco', 'seis', 'sete', 'oito',
         'nove', 'dez', 'onze', 'doze', 'treze', 'quatorze', 'quinze', 'dezesseis', 'dezessete',
         'dezoito', 'dezenove', 'vinte')
sair = ''
while sair != 'S':
    while True:
        numero = int(input('Digite um número entre 0 e 20: '))
        if 0 <= numero <= 20:
            break
        print('Tente novamente.')
    print(f'Você digitou o número {tupla[numero]}')
    sair = ''
    while sair not in 'SsNn':
        sair = str(input('Deseja sair [S/N]: ')).upper().strip()[0]
    while sair != 'S'
ex072 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Lucas/Py
Digite um número entre 0 e 20: 15
Você digitou o número quinze
Deseja sair [S/N]: n
Digite um número entre 0 e 20: 88
Tente novamente.
Digite um número entre 0 e 20: 12
Você digitou o número doze
Deseja sair [S/N]: s
```

## Exercício 073

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

- A. Apenas os 5 primeiros colocados
  - B. Os últimos 4 colocados da tabela
  - C. Uma lista com os times em ordem alfabética
  - D. Em qual posição o time da Chapecoense está.

```
times = ('São Paulo', 'Internacional', 'Atletico-MG', 'Flamegno', 'Gremio', 'Palmeiras',
         'Fluminense', 'Santos', 'Corinthians', 'Athletico-PR', 'Ceara', 'Atletico-GO',
         'Bragantino', 'Sport', 'Fortaleza', 'Vasco', 'Bahia', 'Goias', 'Botafogo', 'Coritiba')

print(f'Os 5 primeiros times no momento: {times[:5]}')
print('-x' * 40)
print(f'Os últimos 4 colocados no momento: {times[-4:]}')
print('-x' * 40)
print(f'A lista em ordem alfabética: {sorted(times)}')
print('-x' * 40)
print(f'A posição que o São Paulo está: {times.index("São Paulo") + 1}º')
```

#### Exercício 074

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

```
from random import randint

tupla = (randint(1, 10), randint(1, 10), randint(1, 10),
         randint(1, 10), randint(1, 10))
#com for
print('Os valores da Lista com For rodando: ', end='')
for c in tupla:
    print(f'{c} ', end='')

#Sem for:
print(f'\nA lista vale: {tupla}')
print(f'O maior elemento da lista: {max(tupla)}')
print(f'O menor elemento da lista: {min(tupla)}')

ex072 × ex074 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Script
Os valores da Lista com For rodando: 1 8 7 10 7
A lista vale: (1, 8, 7, 10, 7)
O maior elemento da lista: 10
O menor elemento da lista: 1
```

## Exercício 075

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

- A. Quantas vezes o nº 9 apareceu
- B. Em que posição apareceu o primeiro valor 3
- C. Quais foram os números pares

```
tupla = (int(input('Digite o número 1: ')), int(input('Digite o número 2: ')),
         int(input('Digite o número 3: ')), int(input('Digite o número 4: ')),
         int(input('Digite o número 5: ')))

print(f'Tupla: {tupla}')
print(f'O número 9 apareceu: {tupla.count(9)} vezes')
if 3 in tupla:
    print(f'O número 3 apareceu pela primeira vez na posição {tupla.index(3) + 1}')
else:
    print('O número 3 não estava na lista!'''

par = impar = 0
for elemento in tupla:
    if elemento % 2 == 0:
        par += 1
    else:
        impar += 1
print(f'O total de pares: {par}')
print(f'O total de ímpares: {impar}')
```

```
C:\Users\lucas\PycharmProjects\py
Digite o número 1: 1
Digite o número 2: 4
Digite o número 3: 5
Digite o número 4: 4
Digite o número 5: 9
Tupla: (1, 4, 5, 4, 9)
O número 9 apareceu: 1 vezes
O total de pares: 2
O total de ímpares: 3
```

### Exercício 076

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

```
produtos = ('Lápis', 1.75, 'Borracha', 2.00, 'Caderno', 15.90, 'Estojo', 25.00, 'Transferidor', 4.20,
    'Compasso', 9.99, 'Mochila', 120.32, 'Canetas', 22.30, 'Livro', 34.90)

for prod in range(0, len(produtos)):
    if prod % 2 == 0:
        print(f'{produtos[prod]:<30}', end=' ')
    else:
        print(f'R$ {produtos[prod]:.2f}')

for prod in range(0, len(produtos)):
    if prod % 2 == 0:
        print(f'{produtos[prod]:<30}', end=' ')
    else:
        print(f'R$ {produtos[prod]:.2f}')


ex072 x ex076 x
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/py
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
```

## Exercício 077

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.

The screenshot shows the PyCharm IDE interface with the code editor, console, variables, and file browser tabs. The code editor contains the following Python script:

```
palavras = ('Laura', 'Lucas', 'Tony', 'Nina', 'Casa', 'Apartamento', 'Trabalho', 'Estagio', 'Faculdade', 'Europa')

for p in palavras:
    print(f'\nA palavra {p} tem vogais: ', end='')
    for vogal in p:
        if vogal.lower() in 'aeiou':
            print(f'{vogal}', end='')
```

The console output shows the execution of the script:

```
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition 2020.3\plugins\python-ce\helpers\pydev\pydevd.py" --multiproc --qt-support=auto --client 127.0.0.1 --port 50033 --file C:/Users/lucas/PycharmProjects/pythonProject/ex077.py
Connected to pydev debugger (build 203.5981.165)

A palavra Laura tem vogais:
>>>
```

The Variables panel on the right shows the current state of variables:

- p = 'Laura'
- palavras = ('Laura', 'Lucas', 'Tony', 'Nina', 'Casa', 'Apartamento', 'Trabalho', 'Estagio', 'Faculdade', 'Europa')
- vogal = 'L'
- Special Variables

O exercício foi resolvido, onde o primeiro laço for lê, para cada elemento ‘p’ na tupla ‘palavras’, ele irá printar o elemento, só que a tupla contém várias inserções e dados, portanto, ele vai ler a primeira ocorrência que é ‘Laura’, e no próximo For que receberá para cada elemento ‘vogal’ dentro da posição ‘p’, se lê: “Se Laura é o elemento zero da Tupla e eu estou pedindo dentro desse elemento zero, quero que ele me traga cada fatiamento desse elemento zero.

## Exercícios de Estrutura de Listas

### Exercício 078

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

```
lista = list()
cont = 0

for contador in range(0, 5):
    lista.append(int(input(f'Digite o número {contador + 1}: ')))

'''while cont < 5:
    lista.append((int(input(f'Digite o número {cont + 1}:'))))
    cont += 1'''

maior = max(lista)
menor = min(lista)

print(f'A lista: {lista}')

print(f'O maior valor da lista: {maior} nas posições: ', end='')
for posicao, valores in enumerate(lista):
    if valores == maior:
        print(f'{posicao}...', end='')

print(f'\nO menor valor da lista: {min(lista)}, nas posições: ', end='')
for posicao, valores in enumerate(lista):
    if valores == menor:
        print(f'{posicao}...', end='')
```

```
C:\Users\lucas\PycharmProjects\pythonProject\venv\Sc
Digite o número 1: 9
Digite o número 2: 9
Digite o número 3: 1
Digite o número 4: 1
Digite o número 5: 2
A lista: [9, 9, 1, 1, 2]
O maior valor da lista: 9 nas posições: 0...1...
O menor valor da lista: 1, nas posições: 2...3...
Process finished with exit code 0
|
```

## Exercício 079

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.

```
lista = list()
continuar = ''

while True:
    num = (int(input('Digite um número: ')))
    if num in lista:
        print(f'\033[31mNúmero {num} já está na lista, não vou adicionar...\033[m')
    else:
        lista.append(num)
        print(f'\033[32mValor {num} adicionado com sucesso!\033[m')
    continuar = str(input('\033[30;47mDeseja continuar? [S/N]:\033[m')).strip().upper()[0]
    if continuar == 'N':
        break

print('_' * 15)
print(f'\033[7mOs valores da lista: {sorted(lista)}\033[m')
```

## Exercício 080

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.

```
lista = []
#pegar o último valor da lista: lista[len(lista) - 1] -> Na lista, a última posição
# OU : lista[-1]

for contador in range(0, 5):
    numero = int(input(f'Digite um número {contador}: '))
    if contador == 0 or numero > lista[-1]: #Se o nº for a primeira vez ou se os próximos
        #forem maiores que o que já está na ultima posição, adiciona na última posição.
        lista.append(numero)
    else:
        pos = 0 #se o nº não é o primeiro e nem é maior que o último, assumo zero
        while pos < len(lista): #enquanto a posição for menor que o tamanho da lista:
            if numero <= lista[pos]: #se o número digitado for menor igual a posição da lista,
                lista.insert(pos, numero) #insere esse número nessa posição percorrida
                break #após achar a posição, break o While e volta ao Input na linha 8
            pos += 1 #após, a posição ganha contador.

print('=-' * 30)
print(f'A lista em ordem crescente sem Sort: {lista}')
```

```
Digite um número 0: 5
Digite um número 1: 56
Digite um número 2: 89
Digite um número 3: 90
Digite um número 4: 52
=====
A lista em ordem crescente sem Sort: [5, 52, 56, 89, 90]

Process finished with exit code 0
```

### **Exercício 081**

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

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

```

lista = []

while True:
    lista.append(int(input('\033[1mDigite um valor:\033[m ')))
    continuar = ' '
    while continuar not in 'SsNn':
        continuar = str(input('\033[30;47mDeseja continuar? [S/N]:\033[m')).strip().upper()
    if continuar == 'N':
        break

print('==' * 30)
print(f'A lista: {lista}')
print('==' * 30)
lista.sort(reverse=True)
print(f'\033[7mA lista Reversa: {lista}\033[m')
print('==' * 30)
lista.sort(reverse=False)
print(f'A lista em ordem crescente: {lista}')
print('==' * 30)
print(f'O tamanho da lista: {len(lista)}')
print('==' * 30)
if 5 in lista:
    print(f'\033[32mO valor 5 está na lista.\033[m')
else:
    print(f'\033[31mO valor 5 não está nessa lista.\033[m')

Digite um valor: 25
Deseja continuar? [S/N]: s
Digite um valor: 36
Deseja continuar? [S/N]: s
Digite um valor: 14
Deseja continuar? [S/N]: s
Digite um valor: 5
Deseja continuar? [S/N]: n
=====
A lista: [25, 36, 14, 5]
=====
A lista Reversa: [36, 25, 14, 5]
=====
A lista Retornou ao normal: [5, 14, 25, 36]
=====
O tamanho da lista: 4
=====
O valor 5 está na lista.

```

## Exercício 082

Crie um programa que vai ler vários números e colocar 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.

```
listaprin = []
listapar = []
listaimpar = []

while True:
    listaprin.append(int(input('Digite um número: ')))
    continuar = ' '
    while continuar not in 'SsNn':
        continuar = str(input('\033[7mDeseja continuar: [S/N]:\033[m ')).strip().upper()[0]
    if continuar == 'N':
        break
for valor in listaprin:
    if valor % 2 == 0:
        listapar.append(valor)
    else:
        listaimpar.append(valor)

listapar.sort()
listaimpar.sort()
print('x-' * 20)
print(f'\033[33mLista: {listaprin}\033[m')
print(f'\033[34mLista Par: {listapar}\033[m')
print(f'\033[35mLista Ímpar: {listaimpar}\033[m')
```

### Exercício 083

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 ou fechados na ordem correta.

```
palavras = []
countprimeiro = countsegundo = 0

palavras.append(str(input('Digite uma expressão: ')))
for p in palavras:
    for vogal in p:
        if vogal == '(':
            countprimeiro += 1
            #print(vogal, end='')
        elif vogal == ')':
            countsegundo += 1
            #print(vogal)
    if countprimeiro > countsegundo:
        print('\033[31mERRO! Tem um "(" a mais.\033[m')
    elif countsegundo > countprimeiro:
        print('\033[33mERRO! Tem um ")" a mais.\033[m')
    else:
        print('\033[32mSua expressão é válida!\033[m')
```

```
C:\Users\lucas\PycharmProjects\pytho
Digite uma expressão: ((a+b)*c
ERRO! Tem um "(" a mais.
```

```
C:\Users\lucas\PycharmProjects\pytho
Digite uma expressão: (a+b)-c
ERRO! Tem um ")" a mais.
```

```
C:\Users\lucas\PycharmProjects\pytho
Digite uma expressão: ((a+b)*c)
Sua expressão é válida!
```

## Exercícios de Estrutura de Listas – Parte 2

### Exercício 084

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

- A. Quantas pessoas foram cadastradas
- B. Uma listagem com as pessoas mais pesadas

- C. Uma lista com as pessoas mais leves

```
lista = list()
pessoas = []
maisPesado = maisLeve = 0

while True:
    lista.append(str(input('Nome: ')).strip())
    lista.append(int(input('Peso: ')))
    if len(pessoas) == 0:#realizar o maior e menor antes de limpar a lista.
        maisPesado = maisLeve = lista[1]
    else:
        if lista[1] > maisPesado:
            maisPesado = lista[1]
        elif lista[1] < maisLeve:
            maisLeve = lista[1]
    pessoas.append(lista[:])
    lista.clear()
    continuar = ''
    while continuar not in 'SsNn':
        continuar = str(input('Deseja continuar? [S/N]: ')).strip().upper()
    if continuar == 'N':
        break
```

```
print(pessoas)
print(f'A quantidade de cadastradas foi: {len(pessoas)}')
print(f'O maior peso: {maisPesado} kgs, as pessoas: ', end='')
for p in pessoas:
    if p[1] == maisPesado:
        print(f'{p[0]}', end=' ')
print(f'\nO menor peso: {maisLeve} kgs, as pessoas: ', end='')
for p in pessoas:
    if p[1] == maisLeve:
        print(f'{p[0]}', end='')
```

```
Nome: Lucas
Peso: 82
Deseja continuar? [S/N]: s
Nome: Laura
Peso: 45
Deseja continuar? [S/N]: s
Nome: Tony
Peso: 82
Deseja continuar? [S/N]: s
Nome: Nina
Peso: 40
Deseja continuar? [S/N]: n
[['Lucas', 82], ['Laura', 45], ['Tony', 82], ['Nina', 40]]
A quantidade de cadastradas foi: 4
O maior peso: 82 kgs, as pessoas: "Lucas" "Tony"
O menor peso: 40 kgs, as pessoas: "Nina"
Process finished with exit code 0
```

### Exercício 085

Crie um programa que o usuário possa digitar 7 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.

```
num = [[], []]

for c in range(0, 7):
    valor = int(input(f'Número {c + 1}: '))
    if valor % 2 == 0:
        num[0].append(valor)
    else:
        num[1].append(valor)

num[0].sort()
num[1].sort()
print(f'Os números pares: {num[0]}')
print(f'Os números ímpares: {num[1]}')
```

```
Número 1: 5
Número 2: 6
Número 3: 7
Número 4: 9
Número 5: 1
Número 6: 3
Número 7: 2
Os números pares: [2, 6]
Os números ímpares: [1, 3, 5, 7, 9]
```

### Exercício 086

Crie um programa que crie uma matriz de dimensão 3x3 e preencha com valores lidos pelo teclado. No final mostre a matriz na tela com a formatação correta

```
matrix = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
# FOR de alimentação da matriz
for linha in range(0, 3):#de 0 a 3 porque tem 0 1 e 2, o 3 ignora.
    for coluna in range(0, 3):#mesma lógica acima
        matrix[linha][coluna] = int(input(f'Número para [{linha}][{coluna}]:'))

#for de impressão da matriz:
for linha in range(0, 3):
    for coluna in range(0, 3):
        print(f'{matrix[linha][coluna]:^5}', end='')#5 casas decimais e centralizado (^)
    print()#quebra a cada coluna que ele lê

Número para [0][0]:255
Número para [0][1]:3
Número para [0][2]:1
Número para [1][0]:1405
Número para [1][1]:8888
Número para [1][2]:999
Número para [2][0]:4
Número para [2][1]:3
Número para [2][2]:1547
[ 255 ][ 3 ][ 1 ]
[1405 ][8888 ][ 999 ]
[ 4 ][ 3 ][1547 ]

Process finished with exit code 0
```

### Exercício 087

Aprimorando o exercício 086, mostrando no final:

- A. A **soma** de **todos** os valores **pares** digitados
- B. A **soma** dos valores da **terceira coluna**
- C. O **maior** valor digitado na **segunda linha**

```

lista = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

par = terceiraCol = segundaLinhaSoma = segundaLinhaMaior = 0

for posicao in range(0, 3):
    for elemento in range(0, 3):
        lista[posicao][elemento] = int(input(f'Número [{posicao}][{elemento}]: '))
        if lista[posicao][elemento] % 2 == 0:
            par += lista[posicao][elemento]

    for posicao in range(0, 3):
        terceiraCol += lista[posicao][2]

for elemento in range(0, 3):
    segundaLinhaSoma += lista[1][elemento]
    if elemento == 0:
        segundaLinhaMaior = lista[1][elemento]
    elif lista[1][elemento] > segundaLinhaMaior:
        segundaLinhaMaior = lista[1][elemento]

for posicao in range(0, 3):
    for elemento in range(0, 3):
        print(f'{lista[posicao][elemento]:^5}', end=' ')
    print()

print(f'A soma dos pares: {par}')
print(f'A soma dos números na 3º Coluna: {terceiraCol}')
print(f'A soma da Segunda Linha: {segundaLinhaSoma}')
print(f'O maior valor da Segunda Linha: {segundaLinhaMaior}')

```

```

Número [2][1]: 8
Número [2][2]: 9
[ 1 ][ 2 ][ 3 ]
[ 4 ][ 5 ][ 6 ]
[ 7 ][ 8 ][ 9 ]
A soma dos pares: 20
A soma dos números na 3º Coluna: 18
A soma da Segunda Linha: 15
O maior valor da Segunda Linha: 6

```

### Exercício 088

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 **aleatórios** entre 1 a 60 para cada jogo, cadastrando tudo em uma lista composta.

```
from random import randint
from time import sleep

lista = []

sorteios = int(input('Digite quantos jogos gerados: '))
for qtdJogos in range(0, sorteios):
    for jogo in range(0, 6):
        numero = randint(1, 60)
        if numero not in lista:
            lista.append(numero)
    lista.sort()
    sleep(1)
    print(f'Jogo nº {qtdJogos + 1}: [{lista}]')
    lista.clear()

print('=-' * 30)
print('<<<< BOA SORTE! >>>>')
```

```
Digite quantos jogos gerados: 3
Jogo nº 1: [[1, 30, 34, 37, 45, 46]
Jogo nº 2: [[3, 26, 27, 30, 53]
Jogo nº 3: [[5, 16, 18, 34, 52, 56]
=====
<<<< BOA SORTE! >>>>
```

### **Exercício 089**

Crie um programa que leia o 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. 3 níveis de listas.

## Exercícios de Estrutura de Dicionários

### Exercício 090

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

```
aluno = dict()

aluno['nome'] = str(input('Nome: '))
aluno['media'] = float(input('Média: '))
situacao = ' '
if aluno['media'] >= 7:
    aluno['situacao'] = 'Aprovado'
elif aluno['media'] >= 5:
    aluno['situacao'] = 'Recuperação'
else:
    aluno['situacao'] = 'Reprovado'

print(aluno)

print(f'O aluno {aluno["nome"]} teve a média {aluno["media"]} e está {aluno["situacao"]}')
print('=' * 15)

for k,v in aluno.items():
    print(f'{k} é igual a {v}')
```

```
Nome: Lucas
Média: 8
{'nome': 'Lucas', 'media': 8.0, 'situacao': 'Aprovado'}
O aluno Lucas teve a média 8.0 e está Aprovado
=====
nome é igual a Lucas
media é igual a 8.0
situacao é igual a Aprovado
```

### Exercício 091

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

```
from random import randint
from time import sleep
from operator import itemgetter

jogador = {'jogador 1': randint(1, 6),
            'jogador 2': randint(1, 6),
            'jogador 3': randint(1, 6),
            'jogador 4': randint(1, 6)
            }

#dicionário ranking para criar ordem em jogadores:
ranking = {} #poderia ser uma lista ranking = []

print('A lista: ')
print(jogador)#sómente mostra a lista
print('-=*' * 20)

#percorre/varre o dicionário e imprime:
print('Valores sorteados: ')
#para cada "chave" tirou o número "valor Da Chave"
for k, v in jogador.items():
    print(f'O {k} tirou o número: {v}')
    sleep(1)
```

```

print('=-'* 20)
#ordena o dicionário e ranking recebe o dicionário em forma de lista
ranking = sorted(jogador.items(), key=itemgetter(1), reverse=True)

print('=-'* 25)
#percorre a lista ranking, o enumerate serve para o índice
#o valor serve para encontrar os valores de cada posição na lista do ranking
for indice, valor in enumerate(ranking):
    print(f'0 {indice + 1}º lugar foi para: {valor[0]} que tirou {valor[1]} nos dados')

```

```

A lista:
{'jogador 1': 3, 'jogador 2': 2, 'jogador 3': 6, 'jogador 4': 1}
=====
Valores sorteados:
0 jogador 1 tirou o número: 3
0 jogador 2 tirou o número: 2
0 jogador 3 tirou o número: 6
0 jogador 4 tirou o número: 1
=====
=====
0 1º lugar foi para: jogador 3 que tirou 6 nos dados
0 2º lugar foi para: jogador 1 que tirou 3 nos dados
0 3º lugar foi para: jogador 2 que tirou 2 nos dados
0 4º lugar foi para: jogador 4 que tirou 1 nos dados

Process finished with exit code 0
|

```

## Exercício 092

Crie um programa que leia o nome, ano de nascimento, carteira de trabalho e cadastre-os(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 essa pessoa vai se aposentar.

```
from datetime import date
hoje = date.today().year

individuo = {}
tempo = 35

individuo['nome'] = str(input('Nome: '))
anoNascimento = int(input('Ano de Nascimento: '))
idade = hoje - anoNascimento
individuo['idade'] = idade
individuo['ctps'] = int(input('Número da Carteira de Trabalho(0 não possui): '))
if individuo['ctps'] != 0:
    individuo['contratacao'] = int(input('Digite ano da contratação: '))
    individuo['salario'] = float(input('Salário: R$ '))
    individuo['aposentadoria'] = (individuo['contratacao'] + 35) - hoje + idade

print('=-' * 25)

for k, v in individuo.items():
    print(f' - {k} tem o valor {v}')
```

```
Nome: Lucas
Ano de Nascimento: 1994
Número da Carteira de Trabalho(0 não possui): 140494
Digite ano da contratação: 2021
Salário: R$ 3500
=====
- nome tem o valor Lucas
- idade tem o valor 27
- ctps tem o valor 140494
- contratacao tem o valor 2021
- salario tem o valor 3500.0
- aposentadoria tem o valor 62
```

### Exercício 093

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.

```
jogador = dict()
listaDeGols = list()
somaGols = 0

jogador['nome'] = str(input('Nome do Jogador: '))
jogador['partidas'] = int(input(f'Quantidade de partidas do {jogador["nome"]}: '))

for cont in range(0, jogador['partidas']): #Laco para contar partidas
    golsPartida = int(input(f'Gol da {cont + 1}ª partida: ')) #Input de Gols por partida
    listaDeGols.append(golsPartida) #Lista que armazena cada gol x partida
    somaGols += golsPartida #total de gols

jogador['gols'] = listaDeGols[:] #dicionário que armazena a lista de gols.
jogador['totalGols'] = somaGols

print('=' * 10, 'Dicionário do Jogador ', '=' * 10)
print(jogador)

print('=' * 11, 'Campos do Jogador ', '=' * 11)
for k, v in jogador.items():
    print(f'O campo {k} tem valor {v}')

print('=' * 10, 'Estatísticas do Jogador ', '=' * 10)
print(f'O jogador {jogador["nome"]}, jogou {jogador["partidas"]} partidas.')
for indice, valor in enumerate(jogador["gols"]):
    print(f'  => Na partida {indice + 1}, o {jogador["nome"]} marcou {valor} gols.')

Nome do Jogador: Lucas
Quantidade de partidas do Lucas: 3
Gol da 1ª partida: 3
Gol da 2ª partida: 1
Gol da 3ª partida: 2
===== Dicionário do Jogador =====
{'nome': 'Lucas', 'partidas': 3, 'gols': [3, 1, 2], 'totalGols': 6}
===== Campos do Jogador =====
O campo nome tem valor Lucas
O campo partidas tem valor 3
O campo gols tem valor [3, 1, 2]
O campo totalGols tem valor 6
===== Estatísticas do Jogador =====
O jogador Lucas, jogou 3 partidas.
=> Na partida 1, o Lucas marcou 3 gols.
=> Na partida 2, o Lucas marcou 1 gols.
=> Na partida 3, o Lucas marcou 2 gols.
```

### Exercício 094 – Exercício mais Completo e Complexo de Dicionário e Lista

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:

Entrada dos dados, somente das leituras e das adições dos dicionários dentro da lista.

Vale lembrar que é **SEMPRE** para limpar o dicionário com o **CLEAR!**

```
from datetime import date

pessoas = {}
listaPessoas = []
hoje = date.today().year

while True:
    pessoas.clear() #pode usar clear em dicionário.
    pessoas['nome'] = str(input('Nome: '))
    while True:
        pessoas['sexo'] = str(input('Sexo: ')).upper()[0]
        if pessoas['sexo'] not in 'MmFf':
            print('ERRO! Digite apenas "M" ou "F".')
        else:
            break
    idade = int(input('Ano de Nascimento [aaaa]: '))
    pessoas['idade'] = hoje - idade
    listaPessoas.append(pessoas.copy())
    while True:
        continuar = str(input('Deseja continuar cadastrando pessoas? [S/N]: ')).strip().upper()
        if continuar in 'SN':
            break
        print('Erro! Digite apenas "S" ou "N".')
    if continuar == 'N':
        break

Nome: Lucas
Sexo: t
ERRO! Digite apenas "M" ou "F".
Sexo: M
Ano de Nascimento [aaaa]: 1994
Deseja continuar cadastrando pessoas? [S/N]: t
Erro! Digite apenas "S" ou "N".
Deseja continuar cadastrando pessoas? [S/N]: s
Nome: Laura
Sexo: f
Ano de Nascimento [aaaa]: 1996
Deseja continuar cadastrando pessoas? [S/N]: n
[{'nome': 'Lucas', 'sexo': 'M', 'idade': 27}, {'nome': 'Laura', 'sexo': 'F', 'idade': 25}]
```

Nesse exercício o controle de Autenticação dos Inputs foi Alterado! Verificar e tomar cuidado para não cometer erros de indentação e lógica!

### A. Quantas pessoas foram cadastradas

```
print(listaPessoas)
print(f'Quantidade de Pessoas Cadastradas: {len(listaPessoas)}')

ex094 x
Deseja continuar cadastrando pessoas? [S/N]: n
[{'nome': 'Lucas', 'sexo': 'M', 'idade': 27}, {'nome': 'Laura', 'sex
Quantidade de Pessoas Cadastradas: 2.
```

### B. A média de idade do grupo

Criação das variáveis para controle das idades:

```
somaIdade = mediaIdade = 0
```

Adição da variável somaidade logo após a leitura de cada Idade somando a idade do dicionário. Vale lembrar que essa idade no começo do laço infinito **SEMPRE SERÁ LIMPA. DAÍ A NECESSIDADE DO CLEAR** no dicionário.

```
idade = int(input('Ano de Nascimento [aaaa]: '))
pessoas['idade'] = hoje - idade
somaIdade += pessoas['idade']
listaPessoas.append(pessoas.copy())
while True:
```

Limpeza do dicionário após a saída do While infinito, apenas para bom código.

A Média das Idades **será lida pela LISTA** e **NÃO** pelo dicionário!

```
pessoas.clear()

print(listaPessoas)
print(f'Quantidade de Pessoas Cadastradas: {len(listaPessoas)}')

mediaIdade = somaIdade / len(listaPessoas) # calcula a media da idade

print(f'Média das Idades: {mediaIdade:.2f} anos.' )#mediaIdade: 5 casas ao todos sendo 2 decimais.
```

### C. Uma lista com todas as mulheres

Para “varrer” a lista de mulheres, esse é mais complexo, pois utilizamos o laço **FOR** para cada indivíduo/pessoa/ser humano/ cidadão/ **dentro** da lista Pessoas (que foi a lista que a lista que “absorveu” todos os dicionários criados).

Realizamos a condicional “Se” o indivíduo/pessoa/ser humano/ cidadão que foi declarado no laço FOR a sua CHAVE que é a [‘sexo’] receber o Ff , pode-se printar o indivíduo/pessoa/ser humano/ cidadão que a sua CHAVE seja [“nome”].

Esse é muito completo pois essa leitura não implica no DICIONÁRIO. Essa leitura do laço FOR e do IF está diretamente VINCULADA a LISTA PESSOAS que absorveu todos os dicionários, por isso é possível usar o para cada alguma coisa que tenha o valor de chave tal.

```
#Varrer a lista de pessoas procurando por mulheres:  
print('As mulheres cadastradas: ', end='')  
for individuo in listaPessoas:  
    if individuo['sexo'] in 'Ff':  
        print(f'{individuo["nome"]}', end=' ')  
print()  
  
Média das Idades: 26.00 anos.  
As mulheres cadastradas: Laura
```

Note que o dicionário está LIMPO, pois utilizamos o CLEAR(duas vezes, uma sempre que ele recebia a cópia pra dentro da lista e mais uma após o laço infinito apenas para limpeza. Mas, note que o dicionário está sempre VAZIO.

**Enquanto a LISTA DE PESSOAS está completa COM os Dicionários!**

```
01 continuar = {str} 'N'  
01 hoje = {int} 2021  
01 idade = {int} 1996  
> 2 listaPessoas = {list: 2} [{nome: 'Lucas', sexo: 'M', idade: 27}, {nome: 'Laura', sexo: ... Vie  
01 medialdade = {float} 26.0  
> 3 pessoas = {dict: 0}{ }  
01 somaldade = {int} 52  
> 4 Special Variables
```

Conforme vamos avançando no DEBUG podemos notar que o “indíviduo” que foi selecionado como elemento dentro do FOR

```
for individuo in listaPessoas:    individuo:  
    if individuo['sexo'] in 'Ff':  
        print(f'{individuo["nome"]}', end=' ')  
print()
```

É atribuído a posição ZERO da LISTA DE PESSOAS, veja que recebeu o valor “Individuo” que é para cada pessoa, indivíduo na lista! Exatamente, a posição zero da lista. Mas, lembrando que nos próximos passos, deverá receber A CHAVE que está contida no dicionário que está dentro dessa lista!

```
01 continuar = {str} 'N'  
01 hoje = {int} 2021  
01 idade = {int} 1996  
> 01 individuo = {dict: 3} {'nome': 'Lucas', 'sexo': 'M', 'idade': 27}  
> 02 listaPessoas = {list: 2} [{'nome': 'Lucas', 'sexo': 'M', 'idade': 27}, {'nome':  
01 medaldade = {float} 26.0  
> 01 pessoas = {dict: 0} {}  
01 somaldade = {int} 52  
> 01 Special Variables
```

A única pessoa/indivíduo cadastrado(a) nessa lista que tem a sua chave[‘sexo’] como feminino é a Laura. Repare bem que a validação não é em cima do [‘nome’] ou do [‘idade’] e sim em cima do [‘sexo’], por isso cada indivíduo que receba a sua chave [‘sexo’] é autenticado.

#### D. Uma lista com todas as pessoas com idade acima da média

Podemos associar novamente para cada indivíduo na lista de pessoas, que ele assuma que dentro do seu dicionário, a chave [‘idade’] esteja acima da média de idade que já foi calculada na Letra B, então printe o indivíduo que tenha a chave nome o seu valor nome e na chave idade o seu valor de idade.

```
print('=-' * 20)  
#Varrer a lista de pessoas com idade acima da média:  
print('As pessoas com Idade acima da média: ')  
for individuo in listaPessoas:  
    if individuo['idade'] >= mediaIdade:  
        print(f'O(a) {individuo["nome"]} tem idade acima da média '  
              f'com idade de: {individuo["idade"]} anos.')  
print()
```

## Exercício 095

Aprimore o desafio 093 para que ele funcione com vários jogadores, incluindo um sistema de visualização de detalhes do aproveitamento de cada jogador.

```
jogador = {}
jogadores = []
listaGols = []

#INÍCIO DA LEITURA DOS DADOS DOS JOGADORES
while True:
    jogador.clear()
    jogador['nome'] = str(input('Nome do jogador: '))
    jogador['partidas'] = int(input(f'Partidas do {jogador["nome"]}: '))
    for cont in range(0, jogador['partidas']):
        gols = int(input(f'Gols na {cont + 1}ª partida: '))
        listaGols.append(gols)
        jogador['gols'] = listaGols[:]
        jogador['saldoGol'] = sum(jogador['gols'])
    jogadores.append(jogador.copy())
    listaGols.clear()
    while True:
        continuar = str(input('Deseja continuar? [S/N]: ')).upper()[0]
        if continuar in 'SN':
            break
        print('Erro! Digite apenas "Sim" ou "Não".')
    if continuar in 'N':
        break
#FIM DA LEITURA DOS DADOS

#INÍCIO DAS IMPRESSÕES DA LISTA DOS JOGADORES
print('--' * 15, 'Estatísticas do Jogador ', '--' * 15)
print('cod', end='')
for i in jogador.keys():
    print(f'{i:<15}', end=' ')
print()
print('-' * 60)
for chave, valor in enumerate(jogadores):
    print(f'{chave:>4} ', end=' ')
    for dados in valor.values():
        print(f'{str(dados):<15}', end=' ')
    print()
print('-' * 60)
#FIM DA IMPRESSÃO DA LISTA DOS JOGADORES
```

```

#INÍCIO DA BUSCA DOS JOGADORES:
while True:
    busca = int(input('Digite o jogador que deseja obter dados [999 Finaliza o programa]: '))
    if busca == 999:
        break
    if busca >= len(jogadores):
        print(f'Erro! Não existe jogador com código {busca}.')
    else:
        print(f' -- Estatísticas do jogador {jogadores[busca]["nome"]} -- ')
        for i, g in enumerate(jogadores[busca]['gols']):
            print(f'      No jogo {i + 1} fez {g} gols.')
print('-' * 60)

```

```

Nome do jogador: Lucas
Partidas do Lucas: 2
Gols na 1º partida: 1
Gols na 2º partida: 0
Deseja continuar? [S/N]: s
Nome do jogador: Laura
Partidas do Laura: 2
Gols na 1º partida: 1
Gols na 2º partida: 1
Deseja continuar? [S/N]: s
Nome do jogador: Tony
Partidas do Tony: 2
Gols na 1º partida: 3
Gols na 2º partida: 1
Deseja continuar? [S/N]: s
Nome do jogador: Nina
Partidas do Nina: 2
Gols na 1º partida: 3
Gols na 2º partida: 2
Deseja continuar? [S/N]: n
===== Estatísticas do Jogador =====
codnome      partidas      gols      saldoGol
-----
 0 Lucas      2          [1, 0]      1
 1 Laura       2          [1, 1]      2
 2 Tony        2          [3, 1]      4
 3 Nina        2          [3, 2]      5
-----
```

```
----- Estatisticas do Jogador -----
codnome      partidas      gols      saldoGol
-----
  0 Lucas        2          [1, 0]       1
  1 Laura        2          [1, 1]       2
  2 Tony         2          [3, 1]       4
  3 Nina         2          [3, 2]       5
-----
Digite o jogador que deseja obter dados [999 Finaliza o programa]: 0
-- Estatisticas do jogador Lucas --
  No jogo 1 fez 1 gols.
  No jogo 2 fez 0 gols.
-----
Digite o jogador que deseja obter dados [999 Finaliza o programa]: 1
-- Estatisticas do jogador Laura --
  No jogo 1 fez 1 gols.
  No jogo 2 fez 1 gols.
-----
Digite o jogador que deseja obter dados [999 Finaliza o programa]: 6
Erro! Não existe jogador com código 6.
-----
Digite o jogador que deseja obter dados [999 Finaliza o programa]: 999

Process finished with exit code 0
```

Note que ao usar o for chave, valor in enumerate(jogadores) onde jogadores é uma lista composta com vários dicionários e lá também é composta em alguns dos campos com listas.

Que a chave vale 0 (zero) no enumerate e imprime uma “espécie” de índice, de chave valendo zero, enquanto o valor assume a posição 0 (zero) da lista.

```
for chave, valor in enumerate(jogadores):    chave: 0    valor:  
    print(f'{chave}>{valor}', end='')  
    for dados in valor.values():  
        print(f'{str(dados):<15}', end='')  
    print()  
  
for chave, valor in enumerate(j...  
ex095 x  
onsole | Variables          
Console  
Partidas do Laura: >? 2  
Gols na 1º partida: >? 3  
Gols na 2º partida: >? 3  
Deseja continuar? [S/N]: >? n  
[{'nome': 'Lucas', 'partidas': 2, 'gols': [1, 1], 'saldoGol': 2}, {'nome': 'Laura',  
'partidas': 2, 'gols': [3, 3], 'saldoGol': 6}]  
===== Estatisticas do Jogador =====  
0  
Variables  
+ chave = (int) 0  
+ cont = (int) 1  
+ continuar = (str)'N'  
+ gols = (int) 3  
> jogador = (dict: 4) {'nome': 'Laura', 'partidas': 2, 'gols': [3, 3], 'saldoGol': 6}  
> jogadores = (list: 2) [{"nome": "Lucas", "partidas": 2, "gols": [1, 1], "saldoGol": 2}, {"nome": "Laura", "partidas": 2, "gols": [3, 3], "saldoGol": 6}]  
> listaGols = (list: 0)  
> valor = (dict: 4) {'nome': 'Lucas', 'partidas': 2, 'gols': [1, 1], 'saldoGol': 2}  
> Special Variables
```

Enquanto entra no próximo for, dados vai assumir o primeiro ou zero elemento dentro desse dicionário e imprimir somente os valores, não vai imprimir uma key por exemplo.

The screenshot shows a Python script named 'ex095' running in a terminal window. The code uses a nested loop to print values from a dictionary. A yellow box highlights the inner loop iteration where 'dados' is set to 'Lucas'. The output in the console shows the name 'Lucas' being printed twice. The variables pane on the right shows the state of variables at this point.

```
for chave, valor in enumerate(jogadores):    chave: 0    valor:
    print(f'{chave:>3}', end=' ')
    for dados in valor.values():    dados: 'Lucas'
        print(f'{str(dados):<15}', end=' ')
    print()
```

Console output:

```
Partidas do Laura: >? 2
Gols na 1° partida: >? 3
Gols na 2° partida: >? 3
Deseja continuar? [S/N]: >? n
[{'nome': 'Lucas', 'partidas': 2, 'gols': [1, 1], 'saldoGol': 2}, {'nome': 'Laura',
 'partidas': 2, 'gols': [3, 3], 'saldoGol': 6}]
===== Estatísticas do Jogador =====
 0Lucas      2          [1, 1]
```

Variables pane:

- chave = (int) 0
- cont = (int) 1
- continuar = (str) 'N'
- dados = (str) 'Lucas'
- gols = (int) 3
- jogador = (dict: 4) {'nome': 'Laura', 'partidas': 2, 'gols': [3, 3], 'saldoGol': 6}
- jogadores = ([list: 2] [{"nome": "Lucas", "partidas": 2, "gols": [1, 1], "saldoGol": 2}, {"nome": "Laura", "partidas": 2, "gols": [3, 3], "saldoGol": 6}])
- listaGols = ([list: 0])
- valor = (dict: 4) {'nome': 'Lucas', 'partidas': 2, 'gols': [1, 1], 'saldoGol': 2}

Repare, apenas que utilizando dessa forma, o laço vai varrer elemento por elemento e imprimir elemento por elemento também.

The screenshot shows the same Python script 'ex095' running again. This time, the inner loop iteration where 'dados' is set to 'Lucas' is highlighted with a yellow box. The output in the console shows the name 'Lucas' followed by its value '[1, 1]'. The variables pane shows the state of variables at this point.

```
for chave, valor in enumerate(jogadores):    chave: 0    valor:
    print(f'{chave:>3}', end=' ')
    for dados in valor.values():    dados: 'Lucas'
        print(f'{str(dados):<15}', end=' ')
    print()
```

Console output:

```
Partidas do Laura: >? 2
Gols na 1° partida: >? 3
Gols na 2° partida: >? 3
Deseja continuar? [S/N]: >? n
[{'nome': 'Lucas', 'partidas': 2, 'gols': [1, 1], 'saldoGol': 2}, {'nome': 'Laura',
 'partidas': 2, 'gols': [3, 3], 'saldoGol': 6}]
===== Estatísticas do Jogador =====
 0Lucas      2          [1, 1]
```

Variables pane:

- chave = (int) 0
- cont = (int) 1
- continuar = (str) 'N'
- dados = (list: 2) [1, 1]
- gols = (int) 3
- jogador = (dict: 4) {'nome': 'Laura', 'partidas': 2, 'gols': [3, 3], 'saldoGol': 6}
- jogadores = ([list: 2] [{"nome": "Lucas", "partidas": 2, "gols": [1, 1], "saldoGol": 2}, {"nome": "Laura", "partidas": 2, "gols": [3, 3], "saldoGol": 6}])
- listaGols = ([list: 0])
- valor = (dict: 4) {'nome': 'Lucas', 'partidas': 2, 'gols': [1, 1], 'saldoGol': 2}

Em um determinado momento, a lista na posição zero irá sumir, dando lugar a posição um:

The screenshot shows a Python code editor and a variables panel. The code in the editor is:

```
for chave, valor in enumerate(jogadores):
    chave: i    valor:
        print(f'{chave}: {valor}', end='')
    for dados in valor.values():
        dados: 'Laura'
            print(f'{str(dados):<15}', end=' ')
    print()
```

The line `dados: 'Laura'` is highlighted with a yellow box. In the variables panel, the variable `dados` is also highlighted with a yellow box.

The console output shows:

```
Gols na 1º partida: >? 3
Gols na 2º partida: >? 3
Deseja continuar? [S/N]: >? n
[{'nome': 'Lucas', 'partidas': 2, 'gols': [1, 1], 'saldoGol': 2}, {'nome': 'Laura', 'partidas': 2, 'gols': [3, 3], 'saldoGol': 6}]
----- Estatísticas do Jogador -----
Lucas      2           [1, 1]      2
Laura
```

The variable `dados` in the variables panel is set to the string 'Laura'.

## Exercícios de Funções – Parte 1

### Exercício 096

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.

Exemplo:

```
Controle de Terrenos
-----
LARGURA (m): 4.5
COMPRIMENTO (m): 8
A área de um terreno 4.5x8.0 é de 36.0m².

Process finished with exit code 0

def area(largura, comprimento):
    totalArea = largura * comprimento
    print(f'O valor da área do terreno {largura:.2f}m² e '
          f'{comprimento:.2f}m² é de {totalArea:.2f}m²')

largura = float(input('Largura: '))
comprimento = float(input('Comprimento: '))
area(largura, comprimento)

C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts
Largura: 4.5
Comprimento: 8
O valor da área do terreno 4.50m² e 8.00m² é de 36.00m²

Process finished with exit code 0
```

### Exercício 097

Faça um programa que tenha uma função chamada “escreva()”, que recebe um texto qualquer como parâmetro e mostre uma mensagem com tamanho variável.

Exemplo:

The screenshot shows the PyCharm IDE interface. On the left is the code editor with the following Python script:

```
def escreva(palavra):
    print('~' * n)
    print(f' {palavra} ')
    print('~' * n)

palavra = str(input('Digite algo: '))
n = len(palavra) + 4

escreva(palavra)

escreva()
```

On the right is the terminal window showing the execution of the script and its output:

```
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts> ex097
Digite algo: Oi, meu nome é Bruce
~~~~~
Oi, meu nome é Bruce
~~~~~
```

### Exercício 098

Faça um programa que tenha uma função chamada “contador()” que receba **três parâmetros**: Início, Fim e Passo. Realize a contagem. Seu programa tem que realizar três contagens através da função criada: (Sem desempacotamento) – (Tem que funcionar com passo negativo também, ex: - 2) – (Se passo = 0, assumir passo = 1 ou -1).

- A. De 1 até 10, de 1 em 1
- B. De 10 até 0, de 2 em 2
- C. Uma contagem personalizada

```
from time import sleep
def mostraLinha():
    print('-' * 30)

def contador(inicio, fim, passo):
    print(f'Contagem de {inicio} a {fim} de {passo} em {passo}')
    if passo < 0:
        passo *= -1 #converte de negativo pra positivo
    if passo == 0:
        passo = 1
    if fim > inicio:
        cont = inicio
        while fim >= cont:
            print(f'{cont}', end=' ')
            #sleep(0.4)
            cont += passo
        print('Fim')
    else:
        if inicio > fim:
            cont = inicio
            while cont >= fim:
                print(f'{cont}', end=' ')
                #sleep(0.4)
                cont -= passo
            print('Fim')

    inicio = 1
    fim = 10
    passo = 1
    contador(inicio, fim, passo)
```

```
mostraLinha()
inicio = 10
fim = 0
passo = 2
contador(inicio, fim, passo)

mostraLinha()
# Programa Principal:
print('Contagem Personalizada')
mostraLinha()
inicio = int(input('Inicio: '))
fim = int(input('Fim: '))
passo = int(input('Passo: '))
contador(inicio, fim, passo)
```

```
Contagem de 1 a 10 de 1 em 1
1 2 3 4 5 6 7 8 9 10 Fim
-----
Contagem de 10 a 0 de 2 em 2
10 8 6 4 2 0 Fim
-----
Contagem Personalizada
-----
Início: 200
Fim: 50
Passo: 10
Contagem de 200 a 50 de 10 em 10
200 190 180 170 160 150 140 130 120 110 100 90 80 70 60 50 Fim

Process finished with exit code 0
```

### Exercício 099

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

```
from time import sleep
lista = []

def maior (* numeros):
    for i in numeros:
        lista.append(i)
    print('Analizando os dados...')
    sleep(1)
    print(f'Foram informados os valores: {lista}.')
    sleep(1)
    print(f'O tamanho da lista é de {len(lista)} números.')
    sleep(1)
    if len(lista) >= 1:
        print(f'O maior número na lista é o {max(lista)}.')
    else:
        print(f'Não há valores para comparação.')
    print('-' * 30)
    lista.clear()
```

```
maior(2, 9, 4, 5, 7, 1)
maior(4, 7, 0)
maior(1, 2)
maior(6)
maior()
```

```
Analisando os dados...
Foram informados os valores: [2, 9, 4, 5, 7, 1].
O tamanho da lista é de 6 números.
O maior número na lista é o 9.
-----
Analisando os dados...
Foram informados os valores: [4, 7, 0].
O tamanho da lista é de 3 números.
O maior número na lista é o 7.
-----
Analisando os dados...
Foram informados os valores: [1, 2].
O tamanho da lista é de 2 números.
O maior número na lista é o 2.
-----
Analisando os dados...
Foram informados os valores: [6].
O tamanho da lista é de 1 números.
O maior número na lista é o 6.
-----
Analisando os dados...
Foram informados os valores: [].
O tamanho da lista é de 0 números.
Não há valores para comparação.
```

### **Exercício 0100**

Faça um programa que tenha uma lista chamada de 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 PARES sorteados pela função anterior.

Sem parâmetros

```
from random import randint

numeros = []

def sorteia():
    count = 0
    while count < 5:
        sorteio = randint(1, 10)
        if sorteio not in numeros:
            numeros.append(sorteio)
            count += 1
    numeros.sort()
    print(numeros)
```

```
def somarPar():
    somapar = 0
    for valor in numeros:
        if valor % 2 == 0:
            somapar += valor
    print(f'O valor dos pares: {somapar}')
    numeros.clear()
```

```
sorteia()
somaPar()
somaParParametro(numeros)
mostraLinha()
sorteia()
somaPar()
somaParParametro()

ex096 × ex100 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python
[3, 5, 7, 8, 10]
O valor dos pares: 18
A soma dos Pares com Função Recebendo Parâmetro é: 0
=====
[2, 3, 4, 5, 9]
O valor dos pares: 6
A soma dos Pares com Função Recebendo Parâmetro é: 0
=====
```

*Passando parâmetros:*

```
from random import randint

numeros = []

def sorteia():
    count = 0
    while count < 5:
        sorteio = randint(1, 10)
        if sorteio not in numeros:
            numeros.append(sorteio)
            count += 1
    numeros.sort()
    print(numeros)

def somarParParametro(numeros):
    somaParParametro = 0
    for valor in numeros:
        if valor % 2 == 0:
            somaParParametro += valor
    print(f'A soma dos Pares com Função Recebendo Parâmetro é: {somaParParametro}')
    numeros.clear()
```

```
sorteia()
#somarPar()
somarParParametro(numeros)
mostraLinha()
sorteia()
#somarPar()
somarParParametro(numeros)
```

```
=====
[3, 6, 7, 9, 10]
A soma dos Pares com Função Recebendo Parâmetro é: 16
=====
[1, 3, 4, 7, 9]
A soma dos Pares com Função Recebendo Parâmetro é: 4
```

## Exercícios de Funções – Parte 2

### Exercício 0101

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 a pessoa tem voto **NEGADO**, **OPCIONAL** ou **OBRIGATÓRIO** nas eleições. O retorno literal são as palavras grifadas e em negrito.

```
def voto(nascimento):
    """
    Informa com base na idade, se o voto da pessoa é:
    Obrigatório, não obrigatório ou Opcional.
    :param idade: Pergunta a idade do usuário.
    """

    from datetime import date
    hoje = date.today().year

    idade = hoje - nascimento
    if idade < 16:
        print(f'Sua idade de {idade}, seu voto é Negado.')
    elif idade >= 18 and idade <= 65:
        print(f'Sua idade de {idade}, seu voto é obrigatório.')
    else:
        print(f'Sua idade de {idade}, seu voto é opcional')

#help(voto)
nascimento = int(input('Digite o ano de nascimento: '))
voto(nascimento)

voto()
ex101 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Script
Digite o ano de nascimento: 1978
Sua idade de 43, seu voto é obrigatório.
```

### Exercício 0102

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

The screenshot shows a PyCharm code editor with a script named 'ex102.py'. The code defines a function 'fatorial' that calculates the factorial of a number and has an optional parameter 'show' to print the multiplication steps. The code is as follows:

```
def fatorial(num=1, show=False):
    num = int(input('Digite um número: '))
    i = 1
    for c in range(num, 0, -1):
        i *= c
    if show == True:
        for c in range(num, 0, -1):
            print(f'{c} {"x " if c > 1 else " = "}', end=' ')
    return i

print(fatorial(5, show=True))
```

Below the code, the terminal window shows the execution of the script 'ex102.py' and its output:

```
fatorial() > for c in range(num, 0, -1)
ex102 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\pyth
Digite um número: 6
6 x 5 x 4 x 3 x 2 x 1 = 720
```

Veja, que solicitarmos apenas a função factorial(5), sem pedir o print, como só temos o return, nenhum valor será formatado na tela.

```
def factorial(num=1, show=False):
    """
    Calcula o fatorial de um número.
    Se 'show' True, então mostra o cálculo.
    :param num: O número a calcular
    :param show: (Opcional) Mostra ou não a conta
    :return: O valor do fatorial de um número "num".
    """

    num = int(input('Digite um número: '))
    i = 1
    for c in range(num, 0, -1):
        i *= c
    if show == True:
        for c in range(num, 0, -1):
            print(f'{c}{" x " if c > 1 else " = "}', end=' ')
    return i

print(factorial(5, show=True))
factorial(5)

fatorial0
```

The screenshot shows a PyCharm interface with a code editor and a terminal window. The code editor contains a Python script named `ex102.py`. The script defines a function `factorial` that calculates the factorial of a number. If `show` is set to `True`, it prints the calculation step-by-step. The script then calls `print(factorial(5, show=True))` and `factorial(5)`. The terminal window below shows the output of the script. It prompts the user to 'Digite um número:' and receives '5'. It then prints the calculation '5 x 4 x 3 x 2 x 1 = 120' followed by another prompt for input.

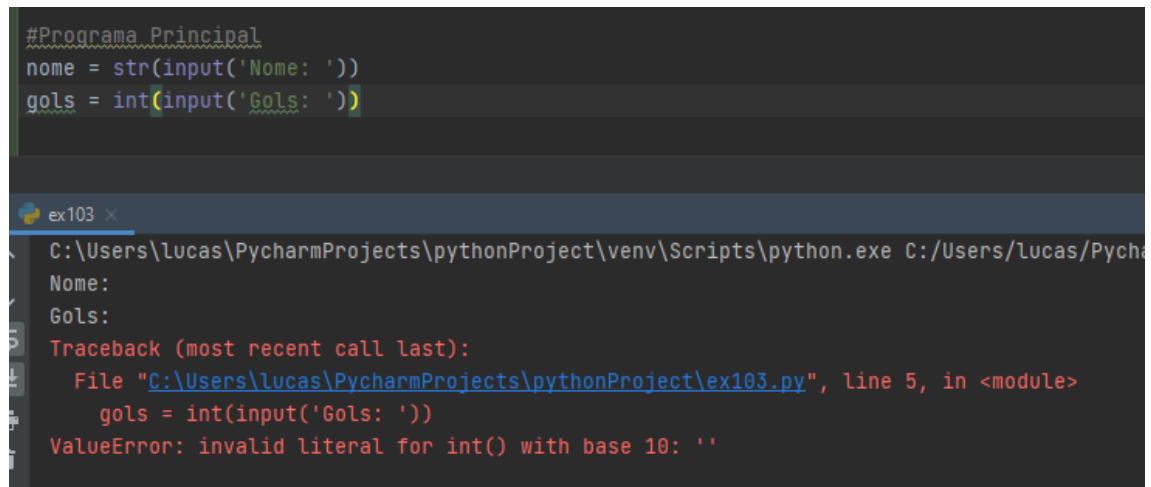
```
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\pyt
Digite um número: 5
5 x 4 x 3 x 2 x 1 = 120
Digite um número: 5
```

### Exercício 0103 – Erro de Traceback

Crie 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 totalmente informado corretamente.

Da forma abaixo, caso deixarmos o nome em branco, não teria problema, mas o valor (int) de gols, ocorreria um erro de **Traceback**.

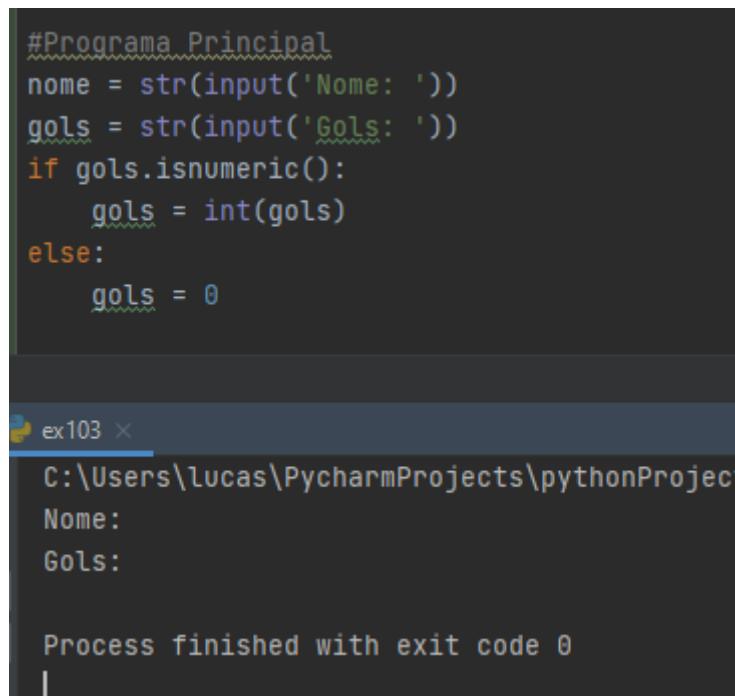


```
#Programa Principal
nome = str(input('Nome: '))
gols = int(input('Gols: '))
```

ex103 ×

```
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Lucas/Pycha
Nome:
Gols:
Traceback (most recent call last):
  File "C:/Users/lucas/PycharmProjects/pythonProject/ex103.py", line 5, in <module>
    gols = int(input('Gols: '))
ValueError: invalid literal for int() with base 10: ''
```

Para isso, no lugar do int de gols, colocamos como string, pois dessa forma, o programa não ocasiona o erro do **Traceback**



```
#Programa Principal
nome = str(input('Nome: '))
gols = str(input('Gols: '))
if gols.isnumeric():
    gols = int(gols)
else:
    gols = 0
```

ex103 ×

```
C:\Users\lucas\PycharmProjects\pythonProject
Nome:
Gols:

Process finished with exit code 0
```

```
def ficha(nome=<desconhecido>, gols=0):
    print(f'O {nome} fez {gols} gol(s) no campeonato.')

#Programa Principal
nome = str(input('Nome: '))
gols = str(input('Gols: '))

if gols.isnumeric():
    gols = int(gols)
else:
    gols = 0

if nome.strip() == '':
    ficha(gols=gols)
else:
    ficha(nome, gols)

ex103 ×
C:\Users\lucas\PycharmProjects\pythonProject\venv\Scripts\|
Nome:
Gols:
0 <desconhecido> fez 0 gol(s) no campeonato.
```

#### Exercício 0104

Crie um programa que tenha uma função chamada “leiaint()”, que vai funcionar de forma semelhante à função “Input()” do Python, só que fazendo a validação para aceitar apenas um valor numérico.

```
def leiaint():
    numero = input('Digite um número: ')
    if numero.isnumeric():
        numero = int(numero)
        return numero
    else:
        while numero.isnumeric() == False:
            print(f'\033[031mErro, número {numero} não é inteiro.\033[m')
            numero = input('Digite um número: ')
            if numero.isnumeric():
                break
    return numero

n = leiaint()
print(f'O valor digitado foi: {n}')
```

```
C:\Users\lucas\PycharmProjects\python
Digite um número:
Erro, número não é inteiro.
Digite um número: 1,2
Erro, número 1,2 não é inteiro.
Digite um número: 2
O valor digitado foi: 2

Process finished with exit code 0
```

### Exercício 0105

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

- A. A quantidade de notas
- B. A maior nota
- C. A menor nota
- D. A média da turma
- E. A situação (opcional)
- F. Adicione também um *docstring* na função.

Exemplo:

Código Guanabara:

```
def notas(*n, sit=False):  
    r = dict()  
    r['total'] = len(n)  
    r['maior'] = max(n)  
    r['menor'] = min(n)  
    r['média'] = sum(n)/len(n)  
    if sit:  
        if r['média'] >= 7:  
            r['situação'] = 'BOA'  
        elif r['média'] >= 5:  
            r['situação'] = 'RAZOÁVEL'  
        else:  
            r['situação'] = 'RUIM'  
    return r  
  
# Programa Principal  
resp = notas(9, 10, 5.5, 2.5, 8.5, sit=True)  
print(resp)  
  
x105 ✘  
/Users/guanabara/PycharmProjects/Python/venv/bin/python /Users/guanabara/PycharmP  
{'total': 5, 'maior': 10, 'menor': 2.5, 'média': 7.1, 'situação': 'BOA'}  
Process finished with exit code 0
```

Meu código:

```
def notas():
    dictNotas = {}
    listaNotas = []
    from statistics import mean
    qtdNotas = int(input('Quantas notas: '))
    for c in range(qtdNotas, 0, -1):
        notas = float(input('Digite uma nota: '))
        listaNotas.append(notas)
    dictNotas['notas'] = listaNotas[:]
    dictNotas['total'] = len(listaNotas)
    dictNotas['maior'] = max(listaNotas)
    dictNotas['menor'] = min(listaNotas)
    dictNotas['media'] = mean(listaNotas)
    if dictNotas['media'] < 6:
        situacao = 'Má situação'
    elif dictNotas['media'] <= 8:
        situacao = 'Situação Ok'
    else:
        situacao = 'Aluno está muito bem!'
    dictNotas['situacao'] = situacao
    return dictNotas

lucas = notas()
print(f'Os valores de Lucas: \n{lucas}')


Quantas notas: 2
Digite uma nota: 8
Digite uma nota: 9
Os valores de Lucas:
{'notas': [8.0, 9.0], 'total': 2, 'maior': 9.0, 'menor': 8.0, 'media': 8.5, 'situacao': 'Aluno está muito bem!'} 
```

```
Help on function notas in module __main__:

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



### Exercício 0106

Faça um *mini-sistema* que utilize o **interactive help** do Python. O usuário vai digitar o comando e o manual vai aparecer. Quando o usuário digitar “**FIM**”, o programa encerará. Obs: **Utilize cores**.

Exemplo:

```
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 > print
Acessando o manual do comando 'print'
Help on built-in function print in module builtins:

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

## Exercícios de Módulos e Pacotes

### Exercício 0107

Crie um módulo chamado moeda.py que tenha as funções incorporadas “aumentar(), diminuir(), dobrar(), metade()”. Faça também um programa que importe esse módulo e use alguma dessas funções

Exemplo:

```
teste.py
1 from ex107 import moeda
2
3 p = float(input('Digite o preço: R$'))
4 print(f'A metade de {p} é {moeda.metade(p)}')
5 print(f'O dobro de {p} é {moeda.dobrar(p)}')
6 print(f'Aumentando 10%, temos {moeda.aumentar(p, 10)}')
7 print(f'Diminuindo 13%, temos {moeda.diminuir(p, 13)}')

Run: teste (2) ×
/usr/local/bin/python3.7 /Users/guanabara/Documents/Curs
Digite o preço: R$500
A metade de 500.0 é 250.0
O dobro de 500.0 é 1000.0
Aumentando 10%, temos 550.0
Diminuindo 13%, temos 435.0
```

Main Program:

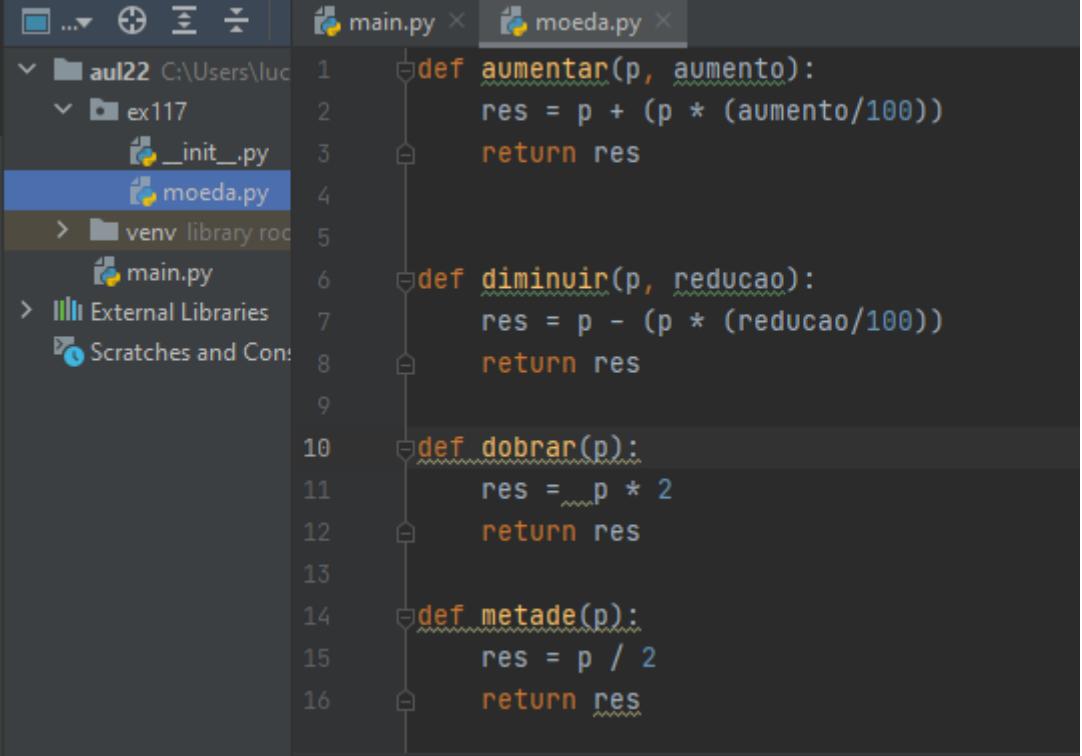
```
aul22 C:\Users\luc
  ex117
    __init__.py
    moeda.py
  venv library root
    main.py
External Libraries
Scratches and Con

main.py
from ex117 import moeda

p = float(input('Digite o preço: R$ '))
print(f'A metade de {p} é R$ {moeda.metade(p)}')
print(f'O dobro de {p} é R$ {moeda.dobrar(p)}')
print(f'Aumentando 10%, temos R$ {moeda.aumentar(p, 10)}')
print(f'Diminuindo 13%, temos R$ {moeda.diminuir(p, 13)}')

Run: main ×
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/Py
Digite o preço: R$ 500
A metade de 500.0 é R$ 250.0
O dobro de 500.0 é R$ 1000.0
Aumentando 10%, temos R$ 550.0
Diminuindo 13%, temos R$ 435.0
```

Pacote ex117 > Módulo moeda.py



The screenshot shows the PyCharm interface with the 'moeda.py' file open. The code defines four functions: 'aumentar', 'diminuir', 'dobrar', and 'metade'. Each function takes a parameter 'p' and returns a result 'res' calculated using a formula involving division by 100.

```
def aumentar(p, aumento):
    res = p + (p * (aumento/100))
    return res

def diminuir(p, reducao):
    res = p - (p * (reducao/100))
    return res

def dobrar(p):
    res = p * 2
    return res

def metade(p):
    res = p / 2
    return res
```

Seguindo a melhor proposta, poderíamos aplicar o return diretamente, porém, a longo prazo, é melhor ter cada uma das funções gerando variáveis que armazenam as operações, por isso os “res” como variáveis globais em cada uma de suas respectivas funções e o return delas propriamente.

## Exercício 0108

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

Exemplo:

```
from ex108 import moeda

p = float(input('Digite o preço: R$'))
print(f'A metade de {moeda.moeda(p)} é {moeda.moeda(moeda.metade(p))}')
print(f'O dobro de {moeda.moeda(p)} é {moeda.moeda(moeda.dobro(p))}')
print(f'Aumentando 10%, temos {moeda.moeda(moeda.aumentar(p, 10))}')

teste (1) ×
/usr/local/bin/python3.7 /Users/guanabara/Documents/CursoemVideo/Python/Python/ex108.py
Digite o preço: R$500
A metade de R$500,00 é R$250,00
O dobro de R$500,00 é R$1000,00
Aumentando 10%, temos R$550,00

Process finished with exit code 0
```

Esse exercício é um pouco mais complexo. Note:

Fizemos uma cópia da aula anterior, somente para não mexermos no projeto do 107. Copiamos todas as funções, mas, adicionamos a monetFormat().

Essa função é responsável por receber dois parâmetros, sendo um o próprio preço e o outro uma string, e como ela não vai ser informado no programa principal, pode receber o “R\$” do opcional.

No retorno da função, na variável, vamos fazer um formato que mostra o “R\$” e logo em seguida o preço que vai receber o parâmetro e com duas casas decimais. E também, sempre que receber um “Ponto” vai ser substituído pela “Vírgula”.

The screenshot shows the PyCharm IDE interface. On the left, the Project tool window displays a file structure for 'aul22' containing 'ex107', 'ex108', and 'main.py'. The 'moeda.py' file is selected and shown in the main editor area. The code in 'moeda.py' is as follows:

```
def dobrar(preco=0):
    res = preco * 2
    return res

def metade(preco=0):
    res = preco / 2
    return res

def monetFormat(preco=0, moeda='R$'):
    res = (f'{moeda}{preco:.2f}').replace('.', ',')
    return res
```

Enquanto isso, no Main, continuamos com a importação do 108, como se fosse um novo projeto. Pedimos o preço no programa principal. E agora vem:

Logo após a linha 4, onde “A metade de...” existe o {moeda.monetFormat(preco)}, vai puxar a função que foi criada que printa o “R\$ 500,00” (vai printar com vírgula por causa do replace, se não, seria um ponto).

E após o “vale:” foi preciso chamar a função que mostra o valor “bonitinho” de “R\$ ” e dentro dessa função, aí sim, chamaremos a função que mostra a metade do preço.”.)

```
moeda.py × main.py ×
1 from ex108 import moeda
2
3 preco = float(input('Digite um preço: R$ '))
4 print(f'A metade de {moeda.monetFormat(preco)} vale: {moeda.monetFormat(moeda.metade(preco))}')
5 print(f'O dobro de {moeda.monetFormat(preco)} vale: {moeda.monetFormat(moeda.dobrar(preco))}')
6 print(f'Aumentando 10% de {moeda.monetFormat(preco)} vale: {moeda.monetFormat(moeda.aumentar(preco, 10))}')
7 print(f'Diminuindo em 13% de {moeda.monetFormat(preco)} vale: {moeda.monetFormat(moeda.diminuir(preco, 13))}')


C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/aul22/main.py
Digite um preço: R$ 500
A metade de R$500,00 vale: R$250,00
O dobro de R$500,00 vale: R$1000,00
Aumentando 10% de R$500,00 vale: R$550,00
Diminuindo em 13% de R$500,00 vale: R$435,00
```

### Exercício 0109

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.

Exemplo:

```
from ex109 import moeda

p = float(input('Digite o preço: R$'))
print(f'A metade de {moeda.moeda(p)} é {moeda.metade(p, False)}')
print(f'O dobro de {moeda.moeda(p)} é {moeda.dobro(p, True)}')
print(f'Aumentando 10%, temos {moeda.aumentar(p, 10, True)}')
print(f'Reduzindo 13%, temos {moeda.diminuir(p, 13, True)}')

teste (5) <
/usr/local/bin/python3.7 /Users/guanabara/Documents/CursoemVideo/F
Digite o preço: R$500
A metade de R$500,00 é 250.0
O dobro de R$500,00 é R$1000,00
Aumentando 10%, temos R$550,00
Reduzindo 13%, temos R$435,00
```

No pacote do ex 109

```
def diminuir(preco=0, porc=0, formato=False):
    res = preco - (preco * porc/100)
    return res if formato is False else monetFormat(res)

def metade(preco=0, formato=False):
    res = preco / 2
    return res if formato is False else monetFormat(res)

dobro()

main <
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Us
Digite um preço: R$ 500
A metade de R$500,00 vale 250.0
O dobro de R$500,00 vale 1000.0
Aumentando 10% de R$500,00 vale R$550,00
Diminuindo 13% de R$500,00 vale 435.0
```

Criamos um parâmetro dentro de cada função que vai receber um parâmetro False(lá no programa principal vamos passar ele sendo True or False também). No retorno, se lê:

Retorne o res(o valor da fórmula) se o parâmetro Formato for Falso (que por padrão no parâmetro opcional é falso), se não, retorne a função monetFormat( que vai formatar a moeda) e aplique isso no resultado (res) da função.

E no programa principal, precisamos passar esse parâmetro True or False para ele ser lido nas funções:

```
from ex109 import moeda

preco = float(input('Digite um preço: R$ '))

print(f'A metade de {moeda.monetFormat(preco)} vale {moeda.metade(preco)}')
print(f'O dobro de {moeda.monetFormat(preco)} vale {(moeda.dobro(preco))}')
print(f'Aumentando 10% de {moeda.monetFormat(preco)} vale {moeda.aumentar(preco, 10, True)}')
print(f'Diminuindo 13% de {moeda.monetFormat(preco)} vale {moeda.diminuir(preco, 13)}').
```

```
main ×
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/a
Digite um preço: R$ 500
A metade de R$500,00 vale 250.0
O dobro de R$500,00 vale 1000.0
Aumentando 10% de R$500,00 vale R$550,00
Diminuindo 13% de R$500,00 vale 435.0
```

Veja que aí podemos tirar toda aquela função gigantesca que tínhamos antes. E se não passarmos dessa forma, vamos perder a formatação que deixava com R\$ 0,00 (com o R\$ e a vírgula).

```
from ex109 import moeda

preco = float(input('Digite um preço: R$ '))

print(f'A metade de {moeda.monetFormat(preco)} vale {moeda.metade(preco, True)}')
print(f'O dobro de {moeda.monetFormat(preco)} vale {(moeda.dobro(preco, True))}')
print(f'Aumentando 10% de {moeda.monetFormat(preco)} vale {moeda.aumentar(preco, 10, True)}')
print(f'Diminuindo 13% de {moeda.monetFormat(preco)} vale {moeda.diminuir(preco, 13, True)}').
```

```
main ×
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas/PycharmProjects/a
Digite um preço: R$ 500
A metade de R$500,00 vale R$250,00
O dobro de R$500,00 vale R$1000,00
Aumentando 10% de R$500,00 vale R$550,00
Diminuindo 13% de R$500,00 vale R$435,00
```

### Exercício 0110

Adicione ao 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.

Exemplo:

```
from ex110 import moeda

p = float(input('Digite o preço: R$'))
moeda.resumo(p, 80, 35)

-----
```

teste (3) /usr/local/bin/python3.7 /Users/guanabara/

Digite o preço: R\$500

-----

RESUMO DO VALOR

-----

Preço analisado: R\$500,00  
Dobro do preço: R\$1000,00  
Metade do preço: R\$250,00  
80% de aumento: R\$900,00  
35% de redução: R\$325,00

-----

```
def resumo(preco=0, aumento=0, desconto=0):
    mostrarLinha()
    print(f'RESUMO DO VALOR'.center(30))
    mostrarLinha()
    print(f'Preço Analisado: \t{monetFormat(preco)}')
    print(f'Dobro do Preço: \t{dobro(preco, True)}')
    print(f'Metade do Preço: \t{metade(preco, True)}')
    print(f'{aumento}% de aumento: \t{aumentar(preco, aumento, True)}')
    print(f'{desconto}% de desconto: \t{diminuir(preco, desconto, True)}')
    mostrarLinha()

resumo()
```

main ×

Digite um preço: R\$ 500

-----

RESUMO DO VALOR

-----

Preço Analisado: R\$500,00  
Dobro do Preço: R\$1000,00  
Metade do Preço: R\$250,00  
80% de aumento: R\$900,00  
35% de desconto: R\$325,00

-----

### Exercício 0111

Crie um pacote chamado utilidadesCEV() que tenha dois módulos internos chamados moeda e dado.

Transfira todas as funções utilizadas nos desafios 107 ao 110 para o primeiro pacote e mantenha tudo funcionando.

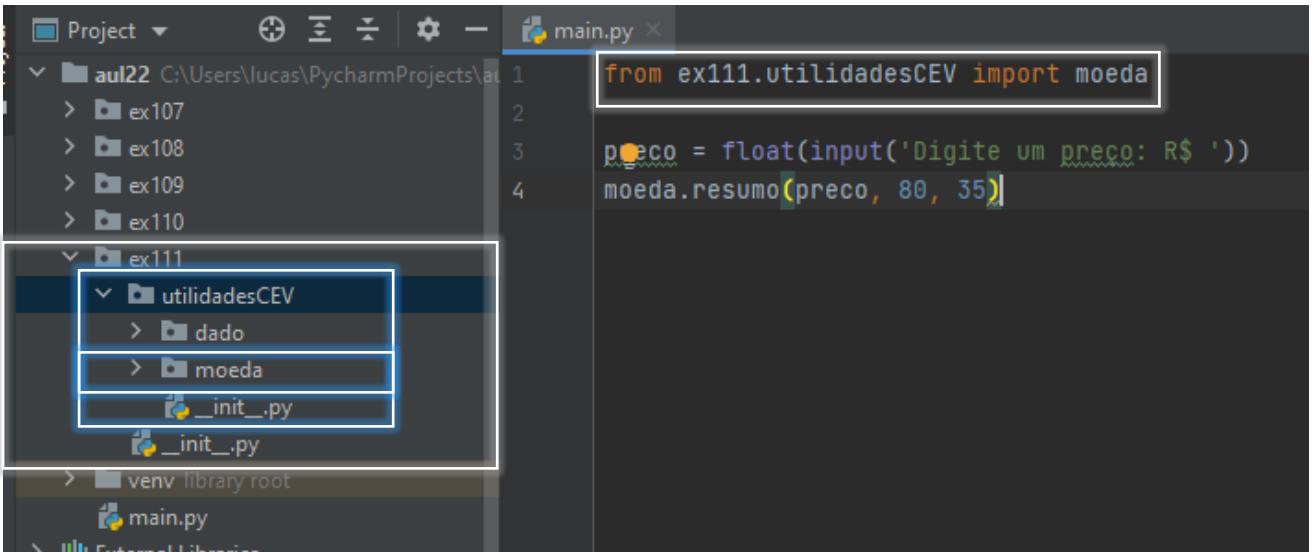
Estávamos utilizando módulos, dessa vez vamos diretamente aos pacotes

Criamos o pacote ex111 (que vai ser o Projeto) e dentro desse projeto, criamos o **Pacote utilidadesCEV**. Por quê?

Porque se criarmos mais e mais pacotes, pode ser que em algum outro pacote, tenha um pacote/módulo com o mesmo nome, assim vamos diferenciar.

Dentro do **Pacote UtilidadesCEV > Pacote Moeda** > O módulo antigo `moeda.py` vamos transferir todas as funções para o módulo `__init__.py` e aí faremos a importação do **Pacote UtilidadesCEV** e importando o pacote moeda dentro do programa principal. E vamos excluir o antigo módulo `moeda.py` também.

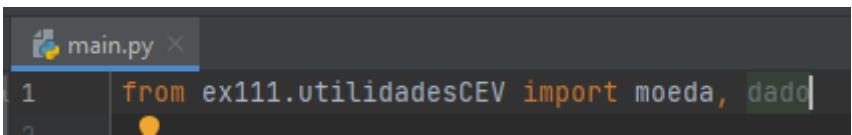
Ao fazer isso, é extremamente importante fazer a importação correta:



The screenshot shows the PyCharm interface. On the left, the Project tool window displays a file tree. Under the 'aul22' project, there are several subfolders: 'ex107', 'ex108', 'ex109', 'ex110', and 'ex111'. The 'ex111' folder is expanded, revealing its contents: 'utilidadesCEV', 'dado', 'moeda', and two files named 'init\_.py'. The 'main.py' file is open in the code editor on the right. The code in 'main.py' is as follows:

```
from ex111.utilidadesCEV import moeda
preco = float(input('Digite um preço: R$ '))
moeda.resumo(preco, 80, 35)
```

Pode realizar a importação assim também, uma vez que o `.utilidadesCEV` já foi importando, podemos importar cada pacote que desejarmos.



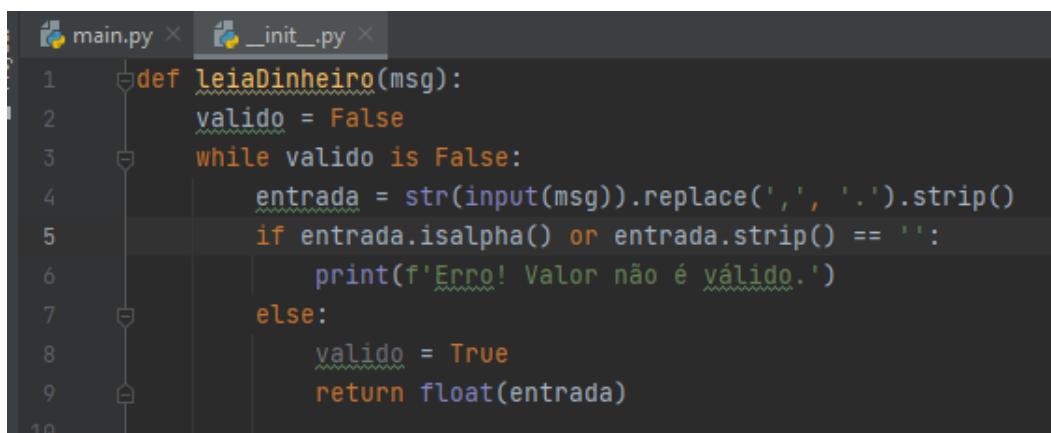
The screenshot shows the PyCharm interface with the 'main.py' file open. The code in 'main.py' includes two import statements:

```
from ex111.utilidadesCEV import moeda, dado
```

## Exercício 0112

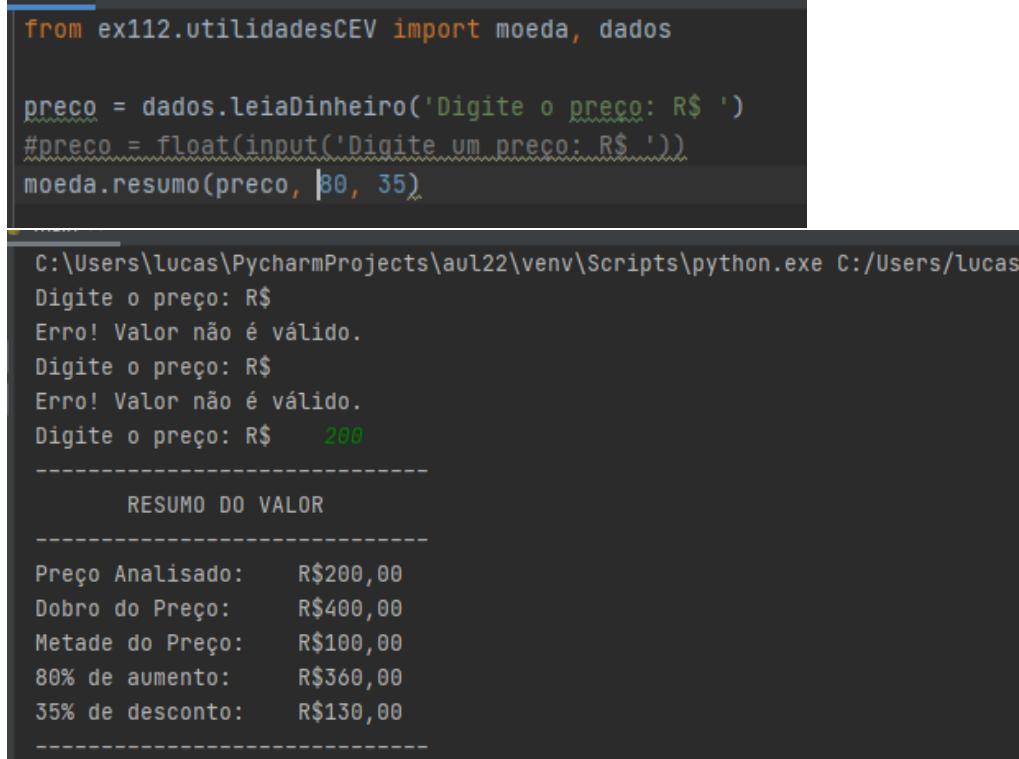
Dentro do pacote utilidadesCEV que criamos no desafio 111, temos um módulo chamado dado. Crie uma função chamada “leiaDinheiro()” que seja capaz de funcionar com a função input(), mas com uma validação de dados para aceitar apenas valores que sejam monetários.

No pacote de Dados, criamos a função leiaDinheiro(que vai receber um parâmetro que será uma mensagem vinda lá do Programa Principal). E terá uma validação de dados: Se o usuário digitar R\$ 500,60, daria erro porque o Python não reconhece a vírgula como o separador numérico. Correção na 5; Se o usuário digitar tudo certo, vai retornar um valor float de uma string que foi criada para poder validar o número.



```
main.py x __init__.py x
1 def leiaDinheiro(msg):
2     valido = False
3     while valido is False:
4         entrada = str(input(msg)).replace(',', '.').strip()
5         if entrada.isalpha() or entrada.strip() == '':
6             print(f'Erro! Valor não é válido.')
7         else:
8             valido = True
9     return float(entrada)
10
```

No programa principal, ainda assim, precisaremos criar uma variável preço que fará a chamada da função leiaDinheiro e vai passar o parâmetro (que vai ser o valor digitado pelo usuário) e vai passar tudo que a função precisa. E aí o resumo se encaminha de realizar os prints. Dados serviu como validação e o resumo pra fazer as contas.



```
from ex112.utilidadesCEV import moeda, dados

preco = dados.leiaDinheiro('Digite o preço: R$ ')
#preco = float(input('Digite um preço: R$ '))
moeda.resumo(preco, 80, 35)

C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/lucas
Digite o preço: R$
Erro! Valor não é válido.
Digite o preço: R$
Erro! Valor não é válido.
Digite o preço: R$    200
-----
      RESUMO DO VALOR
-----
Preço Analisado:    R$200,00
Dobro do Preço:    R$400,00
Metade do Preço:   R$100,00
80% de aumento:   R$360,00
35% de desconto:  R$130,00
-----
```

Mas e se o usuário digitar um:

200f

Não é string pra poder converter pra número, não é número com ponto pra converter pra numero com vírgula, não é espaço vazio que se resolva com strip, é um erro do usuário de digitar número com string e isso só pode ser resolvido com Tratamento de Erros.

```
C:\Users\Lucas\PycharmProjects\aul22\venv\Scripts\python.exe C:/Users/Lucas/PycharmProjects/aul22/main.py
Digite o preço: R$ 200f
Traceback (most recent call last):
  File "C:\Users\lucas\PycharmProjects\aul22\main.py", line 3, in <module>
    preco = dados.leiaDinheiro('Digite o preço: R$ ')
  File "C:\Users\lucas\PycharmProjects\aul22\ex112\utilidadesCEV\dados\__init__.py", line 9, in leiaDinheiro
    return float(entrada)
ValueError: could not convert string to float: '200f'
```

## Exercícios de Tratamento de Erros e Exceções

### Exercício 0113

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

```
main.py x __init__.py x
1  def leiaint():
2      while True:
3          try:
4              numero = int(input('Digite um número Inteiro: '))
5          except (ValueError):
6              print(f'\033[31mErro! Valor digitado não é um número inteiro.\033[m')
7              continue
8          except (KeyboardInterrupt):
9              print('Usuário não digitou informações.')
10             return 0
11         except Exception as error:
12             print(f'Infelizmente aconteceu um erro de {error.__class__}')
13         else:
14             #print('Muito obrigado, volte sempre!')
15             return numero
16             break
17
18
19 def leiaFloat(msg):
20     while True:
21         try:
22             numero = float(input(msg).replace(',', '.', 1))
23         except (ValueError, TypeError):
24             print('\033[1;31mERRO: por favor, digite um número real válido.\033[m')
25             continue
26         except (KeyboardInterrupt):
27             print('\n\033[1;31mO usuário preferiu não digitar esse número.\033[m')
28             return 0
29         else:
30             return numero
31
32
33 from ex113 import leiaint, leiaFloat
34
35 numeroInteiro = leiaint()
36 numeroReal = leiaFloat('Digite um valor Real: ')
37 print(f'O número inteiro foi {numeroInteiro} '
38       f'\nO número real digitado foi {numeroReal}')
```

### Exercício 0114

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

```
main114.py × __init__.py ×
1 import urllib
2 import urllib.request
3
4 def conectar():
5     try:
6         site = urllib.request.urlopen('http://www.pudim.com.br')
7     except:
8         print(f'Deu erro!')
9     else:
10        print(f'Conexão estabelecida com sucesso!')
```

```
main114.py × __init__.py ×
1 from ex114 import conectar
2
3 conexao = conectar()
4
C:\Users\lucas\PycharmProjects\aul22\ver
Conexão estabelecida com sucesso!
Process finished with exit code 0
```

```
try:
    site = urllib.request.urlopen('http://www.pudim')
except:
    print(f'Deu erro!')
```

Fazendo o pedido diretamente ao usuário, na página abaixo.

```
#conexao = conectar()
conexao = conectarInput()

def conectarInput():
    try:
        site = str(input('Digite um site inteiro: '))
        host = urllib.request.urlopen(site)
    except urllib.error.URLError:
        print(f'O site não está acessível no momento.')
    except Exception as error:
        print(f'Ocorreu um erro! Error: {error.__class__}')
    else:
        print(f'Conexão estabelecida com sucesso!')

C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\p
Digite um site inteiro: https://www.google.com
Conexão estabelecida com sucesso!

Process finished with exit code 0
```

Nesse caso ele cairá na Exceção que foi digitada para tratar o erro do UrlLib.

```
C:\Users\lucas\PycharmProjects\aul22\venv\
Digite um site inteiro: https://wwwwww
O site não está acessível no momento.

Process finished with exit code 0
```

Para esse caso, não foi definida uma exceção, somente encontrou um erro de “valor”.

```
C:\Users\lucas\PycharmProjects\aul22\venv\Scripts\p
Digite um site inteiro: www.google
Ocorreu um erro! Error: <class 'ValueError'>

Process finished with exit code 0
```

### Exercício 0115

Crie um pequeno sistema modularizado que permita cadastrar pessoas pelo seu nome e idade em um arquivo de texto simples. O sistema só vai ter duas opções: **Cadastrar** uma nova pessoa e **Listar** todas as pessoas cadastradas.

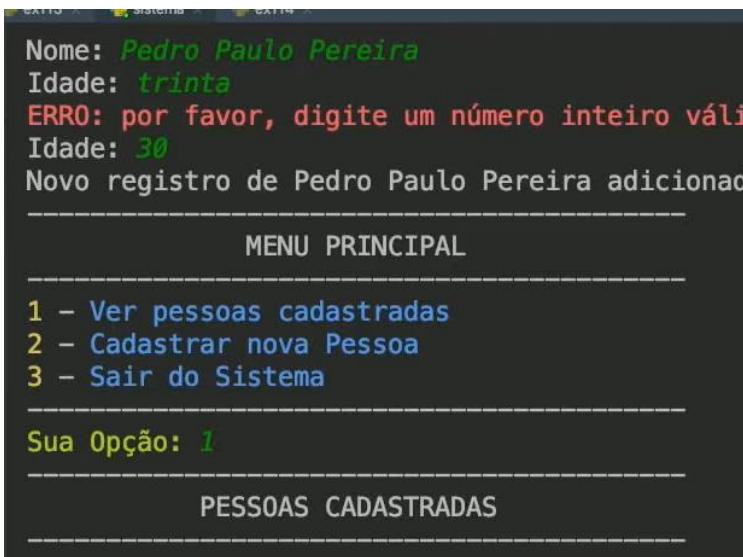
```
MENU PRINCIPAL
1 - Ver pessoas cadastradas
2 - Cadastrar nova Pessoa
3 - Sair do Sistema
Sua Opção: 1
PESSOAS CADASTRADAS
Ana Paula Vieira      32 anos
Cláudio Mendonça     18 anos
Gustavo Guanabara    41 anos
Maria Clara Peixoto   65 anos
Maurício Souza       19 anos
Nilce Pedrosa         43 anos
Pedro Gonçalves      18 anos
Rafael Albuquerque    38 anos
Renata Soares        13 anos
MENU PRINCIPAL
1 - Ver pessoas cadastradas
2 - Cadastrar nova Pessoa
3 - Sair do Sistema
Sua Opção:
Sua Opção: 2
NOVO CADASTRO
Nome: Zuleide Lima
Idade: 55
Novo registro de Zuleide Lima adicionado.
MENU PRINCIPAL
1 - Ver pessoas cadastradas
2 - Cadastrar nova Pessoa
3 - Sair do Sistema
Sua Opção: 1
PESSOAS CADASTRADAS
Ana Paula Vieira      32 anos
Cláudio Mendonça     18 anos
Gustavo Guanabara    41 anos
Maria Clara Peixoto   65 anos
Maurício Souza       19 anos
Nilce Pedrosa         43 anos
Pedro Gonçalves      18 anos
Rafael Albuquerque    38 anos
Renata Soares        13 anos
Zuleide Lima          55 anos
```

```

MENU PRINCIPAL
-----
1 - Ver pessoas cadastradas
2 - Cadastrar nova Pessoa
3 - Sair do Sistema
-----
Sua Opção: 1
-----
PESSOAS CADASTRADAS
-----
Ana Paula Vieira      32 anos
Cláudio Mendonça     18 anos
Gustavo Guanabara    41 anos
Maria Clara Peixoto   65 anos
Maurício Souza       19 anos
Nilce Pedrosa         43 anos
Pedro Gonçalves      18 anos
Rafael Albuquerque    38 anos
Renata Soares        13 anos
Zuleide Lima          55 anos
-----
MENU PRINCIPAL
-----
1 - Ver pessoas cadastradas
2 - Cadastrar nova Pessoa
3 - Sair do Sistema
-----
Sua Opção: 3
-----
Saindo do sistema... Até logo!
-----
Process finished with exit code 0

```

O melhor é, mesmo quando apertarmos o PAUSE e rodarmos o programa de novo, a ZULEIDE que foi criada, ainda deve estar lá! Não tem que sumir!!!



```

Nome: Pedro Paulo Pereira
Idade: trinta
ERRO: por favor, digite um número inteiro válido
Idade: 30
Novo registro de Pedro Paulo Pereira adicionado

MENU PRINCIPAL
-----
1 - Ver pessoas cadastradas
2 - Cadastrar nova Pessoa
3 - Sair do Sistema
-----
Sua Opção: 1
-----
PESSOAS CADASTRADAS
-----
```

## Principal

```
main115.py ×  cadastro\__init__.py ×  arquivo\__init__.py ×  cadastroTexto.txt ×
1 | from ex115.cadastro import menu
2 |
3 | menu()
```

Módulo Cadastro > Menu()

```
def linha(tamanho=42):
    print('-' * tamanho)

def cabecalho(txt):
    linha()
    print(txt.center(42))
    linha()

def menu():
    from datetime import date
    from time import sleep
    from ex115.arquivo import arquivoExiste, criarArquivo, lerArquivo, cadastrar
    hoje = date.today().year
    dicioPessoas = {}
    cor = [...] # 3 - Texto Vermelho
    arquivo = 'cadastroTexto.txt'
    if not arquivoExiste(arquivo):
        criarArquivo(arquivo)

    while True:
        dicioPessoas.clear()
        cabecalho('MENU PRINCIPAL')
        print(f'{cor[1]}1 - {cor[2]}Ver pessoas cadastradas {cor[0]}')
        print(f'{cor[1]}2 - {cor[2]}Cadastrar nova pessoa{cor[0]}')
        print(f'{cor[1]}3 - {cor[2]}Sair do Sistema {cor[0]}')
        linha()
        #sleep(0.5)
        while True:
            try:
                opcao = int(input(f'{cor[1]}Opcão{cor[0]}: '))
                if opcao > 3:
                    print(f'Opcão inválida!')
                else:
                    break
            except ValueError:
                print(f'Erro! Opcão Inválida')
```

```

# Opção que Lista as Pessoas Cadastradas:
if opcao == 1:
    '''if len(listaPessoas) <= 0:
        sleep(1)
        print(f'{cor[3]}Não existem pessoas cadastradas!{cor[0]}')
    else:'''
    cabecalho('PESSOAS CADASTRADAS')
    lerArquivo(arquivo)
#sleep(1)

#Opção que Cadastra as Pessoas no Sistema
if opcao == 2:
    cabecalho('CADASTRAR NOVO USUÁRIO')
    while True:
        nome = str(input('Digite seu nome: '))
        if nome.isalpha():
            #print(f'Nome aceito!')
            dicioPessoas['nome'] = nome
            break
        else:
            print('Nome não válido.')
    while True:
        try:
            nascimento = int(input('Nascimento: '))
            dicioPessoas['idade'] = hoje - nascimento
        except (ValueError):
            print(f'Erro com o formato do dado. Por favor repetir.')
        except Exception as error:
            print(f'Deu erro fora da validação {error.__class__}')
        else:
            cadastrar(arquivo, dicioPessoas['nome'], dicioPessoas['idade'])
            break
    #break

#sleep(1)
if opcao == 3:
    print(f'Obrigado por utilizar o sistema, até logo!')
    break

```

Módulo do Arquivo que tem as funções do Cadastro

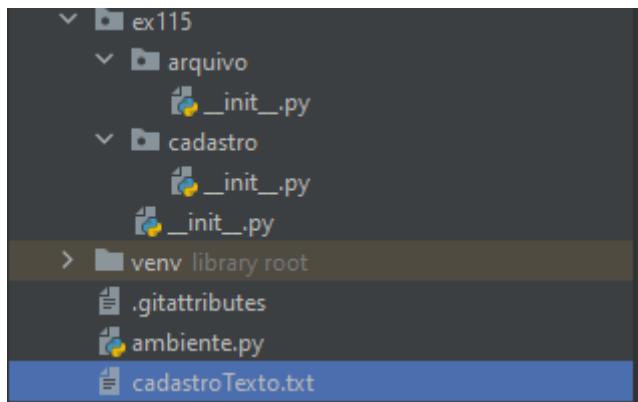
```
from ex115.cadastro import cabecalho

def arquivoExiste(nome):
    try:
        #open abre arquivos.
        a = open(nome, 'rt') #rt: read text
        a.close()
    except FileNotFoundError:
        return False
    else:
        return True

def criarArquivo(nome):
    try:
        a = open(nome, 'wt+') #wt+: Write Text o + se não existir criar arquivo(+)
        a.close()
    except Exception as error:
        print(f'ERRO {error.__class__}. Erro na criação do arquivo.')
    else:
        print(f'Arquivo criado com sucesso')

def lerArquivo(nome):
    try:
        a = open(nome, 'rt')
    except:
        print(f'Erro ao ler o arquivo.')
    else:
        for linha in a:
            dado = linha.split(';')
            dado[1] = dado[1].replace('\n', '') #Replace o espaço no cadastro por nada
            print(f'{dado[0]}:{<30}{dado[1]:>3} anos')
    finally:
        a.close()

def cadastrar(arquivo, nome='desconhecido', idade='0'):
    try:
        a = open(arquivo, 'at') #append text - grava no texto
    except:
        print(f'Houve um erro na abertura do arquivo!')
    else:
        try:
            a.write(f'{nome};{idade}\n')
        except:
            print(f'Houve um erro na hora de escrever os dados.')
        else:
            print(f'Novo registro, {nome} adicionado(a).')
```



Resultado:

```
-----  
          MENU PRINCIPAL  
-----  
1 - Ver pessoas cadastradas  
2 - Cadastrar nova pessoa  
3 - Sair do Sistema  
-----  
Opção:  
-----  
Opção:  
Erro! Opção Inválida  
Opção: 1  
-----  
          PESSOAS CADASTRADAS  
-----  
Lucas           27 anos  
LAURA          25 anos  
Tony            3 anos  
-----  
          MENU PRINCIPAL  
-----  
1 - Ver pessoas cadastradas
```

```
5 Sair do sistema
-----
Opção: 2
-----
        CADASTRAR NOVO USUÁRIO
-----
Digite seu nome: Lucas
Nascimento: 1994
Novo registro, Lucas adicionado(a).
-----
        MENU PRINCIPAL
-----
Opção: 3
Obrigado por utilizar o sistema, até logo!
```





## **Exercício de Cores no Terminal**

Esse desafio consiste em voltar nos exercícios anteriores e inserir cores para deixar o programa mais apresentável.