

Python

Clube de IA
IFSUL - Câmpus Gravataí
Marcelo Dias

AMBIENTE DE DESENVOLVIMENTO

Instalação Python

- Não é necessário fazer esta instalação para o Clube de IA
- Iremos utilizar a instalação via Anaconda
- URL para instalação do interpretador Python padrão

<https://www.python.org/downloads/>

Instalação Anaconda

- URL para download do Anaconda
- Software para Ciência de Dados
- Inclui: Python, pacotes para Ciência de Dados, Jupyter Notebooks, VS Code, R Studio,...

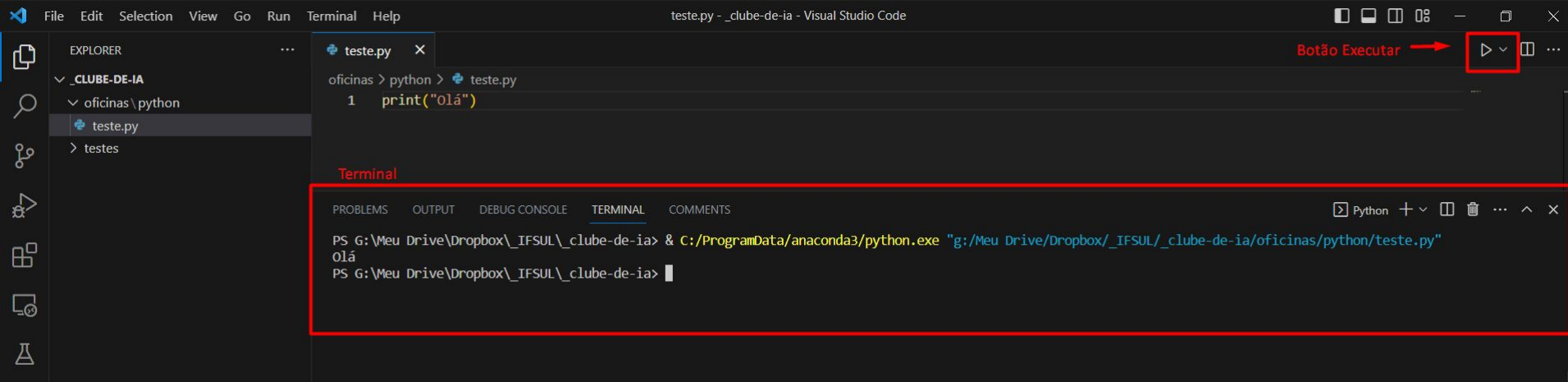
<https://www.anaconda.com/download>

Teste seu ambiente de desenvolvimento

1. Crie uma pasta para seus programas
2. Abra o Anaconda
3. Abra o VS Code a partir do Anaconda
4. Adicione a pasta criada ao ambiente do VS Code
5. Adicione a extensão “Python Extension Pack”
6. Adicione um arquivo teste.py na pasta
7. Insira o código abaixo no programa, salve e execute

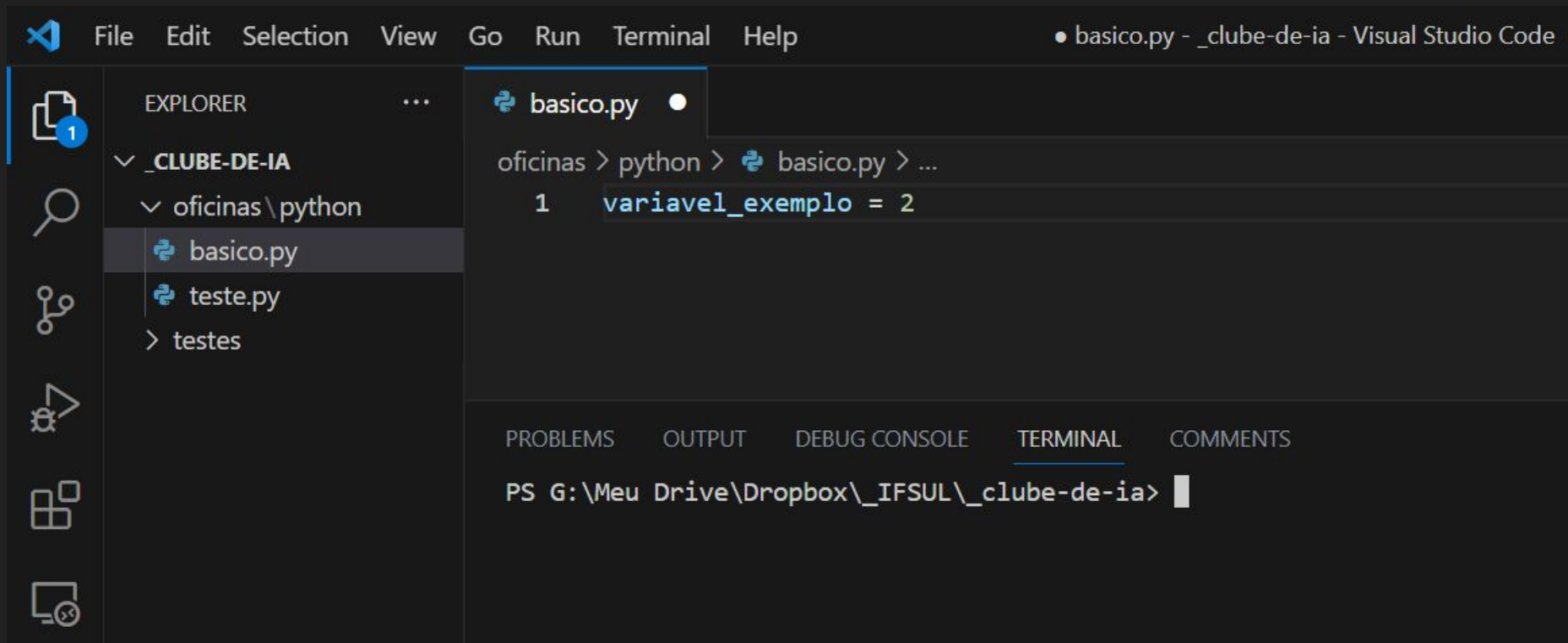
```
print('Olá')
```

Teste seu ambiente de desenvolvimento

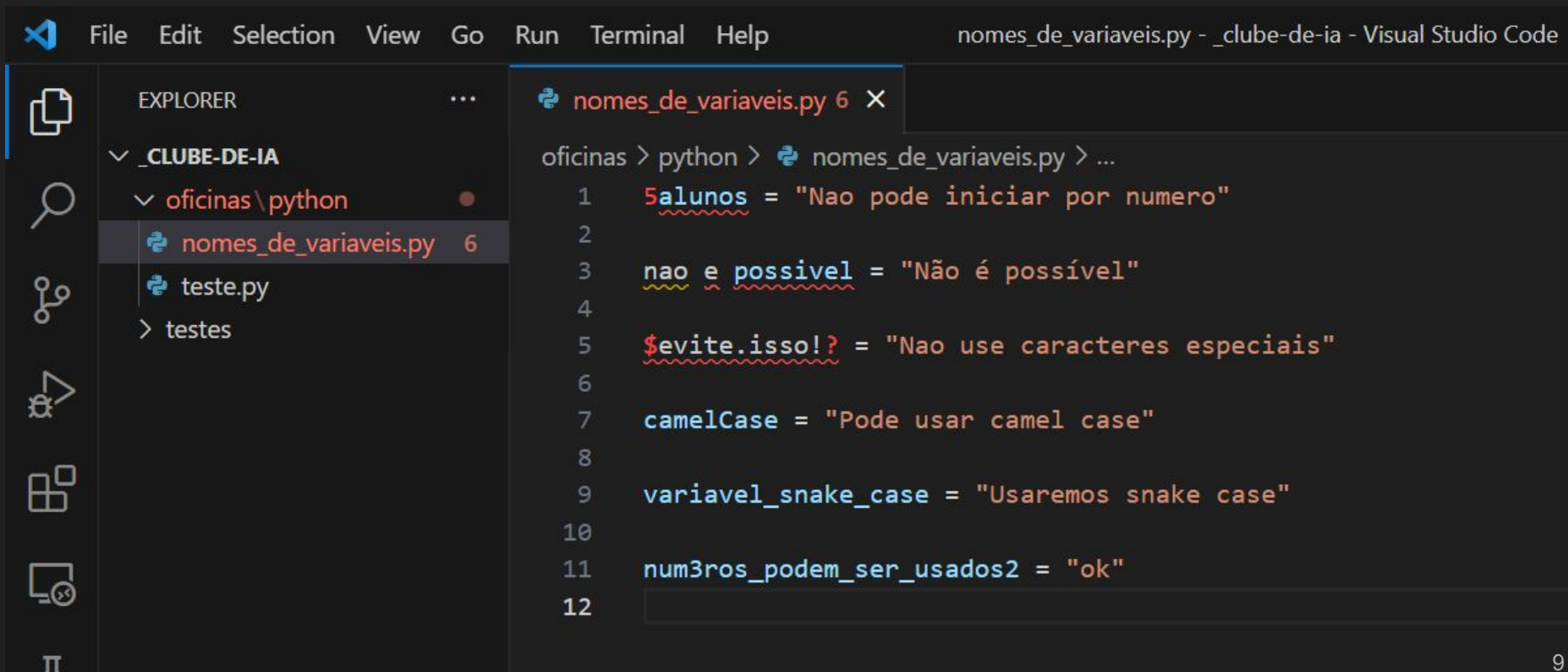


PYTHON BÁSICO

Variáveis e Atribuições



Nomes de Variáveis



Visual Studio Code interface showing the Explorer sidebar and the editor window.

Explorer Sidebar:

- EXPLORER
- CLUBE-DE-IA
 - oficinas\python
 - nomes_de_variaveis.py 6
 - teste.py
 - testes

Editor Window: nomes_de_variaveis.py - _clube-de-ia - Visual Studio Code

```
oficinas > python > nomes_de_variaveis.py > ...  
1  5alunos = "Nao pode iniciar por numero"  
2  
3  nao e possivel = "Não é possível"  
4  
5  $evite.isso!? = "Nao use caracteres especiais"  
6  
7  camelCase = "Pode usar camel case"  
8  
9  variavel_snake_case = "Usaremos snake case"  
10  
11 num3ros_podem_ser_usados2 = "ok"  
12
```

DICA PARA VIDA DE PROGRAMADOR

- NÃO USE ACENTOS OU CEDILHA PARA NOMES DE VARIÁVEIS
- NÃO USE ACENTOS, CEDILHA OU ESPAÇOS PARA NOMES DE ARQUIVOS DE PROGRAMAS.
- UTILIZE LETRAS MINÚSCULAS E _ PARA ARQUIVOS DE PROGRAMAS
- Apesar de ser “permitido”, vai evitar muitos problemas

Tipos de Datos Básicos

- float
- int
- boolean

```
1 float_1 = 1.2345
2 float_2 = -1.23
3 float_3 = 0.0
4 float_4 = 123.0
5 float_5 = -0.1
6
7 int_1 = 6
8 int_2 = -41
9 int_3 = 0
10 int_4 = 700
11 int_5 = -732
12
13 boolean_1 = True
14 boolean_2 = False
15
```

Comentários

- É ignorado durante a execução do programa
- Deve conter um espaço após o “#”

```
1  # Isto e um comentario
2
3  variavel_1 = 6 * 5 # Isto tambem e um comentario
4  |
```

Operadores (matemática)

```
1  adicao = 3 + 4      # 7
2  multiplicacao = 5 * 7  # 35
3  subtracao = 6 - 3     # 3
4  divisao = 30 / 5      # 6
5
6  exponenciacao = 2 ** 4  # 16
7  modulo = 17 % 3        # 2 (ou resto da divisao)
8  divisao_int = 17 // 3   # 5
```

Operadores de atribuição

```
1  variavel = 2
2  variavel = variavel + 3 # 5
3  variavel += 3    # (8) mesma operacao acima com menos codigo
4
5  variavel -= 3    # (5) subtracao
6  variavel *= 5    # (25) multiplicacao
7  variavel /= 5    # (5) divisao
8  variavel **= 3   # (125) exponenciacao
9  variavel //= 2   # (62) divisao inteira
10 variavel %= 10   # (2) resto da divisao
11
```

Precedência de operadores

1. `()`
2. `**`
3. `% // / *`
4. `+ -`

print

- Usado para imprimir uma saída

```
1  variavel_1 = 1.2345
2  variavel_2 = 54391
3  variavel_3 = True
4  variavel_4 = (7 + 3) ** 2
5
6  print(variavel_1)
7  print(variavel_2)
8  print(variavel_3)
9  print(variavel_4)
10
11 print(1234)
12 print((5 - 2) * 4)
13
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-  
/python/print.py"
```

```
1.2345
```

```
54391
```

```
True
```

```
100
```

```
1234
```

```
12
```


Erros de aproximação em floats

- Python usa a linguagem c que apresenta o mesmo comportamento
- Workarounds
 - Usar int
 - Função round

```
1  var_1 = 2.33 + 6.70    # 9.0300000000000001
2  print(var_1)
3
4  # workarounds
5  var_2 = (233 + 670) / 100 # 9.03
6  print(var_2)
7
8  var_3 = round(2.33 + 6.70, 2)    # 9.03
9  print(var_3)
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia> & C:/Pr
/python/float_erro_de_aproximacao.py"
9.0300000000000001
9.03
9.03
```

STRINGS

O que é uma String

- Sequência de caracteres (letras, dígitos, símbolos)

```
1  variavel_1 = 'Isto é uma string'
2  variavel_2 = "Isto também é uma string"
3
4  variavel_3 = "" # string vazia, porem diferente de " "
5  variavel_4 = "1762"
6  variavel_5 = "123!"
7  variavel_6 = "&%(!@hua&.,;:^~"
8  variavel_7 = "Que a ForcA esteJA COM vocÊ"
9
```

Índices em Strings

- Permitem acessar um ou mais caracteres da String
- Usado entre colchetes: [índice]

```
1  # Índice para acessar cada caractere
2  # Continua 10, 11, 12,...
3  variavel_1 = "0123456789"
4
5  variavel_2 = "Força"
6  print(variavel_2[3])
7  print("Força"[3])
8
```

Terminal (Ctrl+')

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia> /python/string_indices.py"
```

ç

ç

Substrings

- Acessar partes de uma string
- 3 formas de “fatiar”
- Índice negativo

```
1  variavel_1 = "jabuticaba"
2
3  print(variavel_1[:4])
4  print(variavel_1[2:6])
5  print(variavel_1[6:])
6
7  print(variavel_1[-2:])
8
```

PROBLEMS OUTPUT DEBUG CONSOLE 1

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clu
/python/substrings.py"
jabu
buti
caba
ba
```

Concatenação

- Une strings formando uma string maior

```
1  variavel_1 = "sorvete" + " " + "chocolate"
2  print(variavel_1)
3
4  print("(" + "7 - 5" + ")" + " ** " + "4")
5
```

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	COMMENTS
			PS G:\Meu Drive\Dropbox_IFSUL_clube-de-ia> & C:/Pr /python/strings_concatenacao.py sorvete chocolate (7 - 5) ** 4	

Função type()

- Retorna o tipo de uma variável ou literal

```
1  variavel_1 = 1.2345
2  variavel_2 = 54391
3  variavel_3 = True
4  variavel_4 = (7 + 3) ** 2
5
6  print(type(variavel_1))
7  print(type(variavel_2))
8  print(type(variavel_3))
9  print(type(variavel_4))
10 print("")
11 print(type(1234))
12 print(type((5 - 2) * 4))
13
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TER

PS G:\Meu Drive\Dropbox_IFSUL_clubes
/python/tipos_de_variaveis.py"

<class 'float'>

<class 'int'>

<class 'bool'>

<class 'int'>

<class 'int'>

<class 'int'>

Função str

- Converte uma variável ou valor para string

```
1  variavel_1 = 1.2345
2  variavel_2 = 54391
3  variavel_3 = True
4  variavel_4 = (7 + 3) ** 2
5
6  print(type(str(variavel_1)))
7  print(type(str(variavel_2)))
8  print(type(str(variavel_3)))
9  print(type(str(variavel_4)))
10 print("")
11 print(type(str(1234)))
12 print(type(str((5 - 2) * 4)))
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-
/python/funcao_str.py"
```

```
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
```

```
<class 'str'>
<class 'str'>
```


Sequências de Escape

- Permite incluir caracteres especiais

```
1  print("Que\ta\tforça\testeja\com\tvocê")    # \t
2  print("")
3  print("Linha 1\nLinha 2")                  # \n
4  print("")
5  print('"duplas dentro de simples" - OK')
6  print("'simples dentro de duplas' - OK")
7  print("")
8  print('\simples dentro de simples\' - OK') # \'
9  print("\duplas dentro de duplas\" - OK")  # \"
10 print("")
11 print("E para usar a barra: \\")          # \\
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia> & C:/ProgramData/Python/Python38-32/Scripts/python C:/ProgramData/Python/Python38-32/Scripts/python/python/sequencias_de_escape.py"
```

```
Que      a      força  esteja  com      você
```

```
Linha 1
```

```
Linha 2
```

```
"duplas dentro de simples" - OK
'simples dentro de duplas' - OK
```

```
'simples dentro de simples' - OK
"duplas dentro de duplas" - OK
```

```
E para usar a barra: \
```

Função input()

- Usada para entrada de informações
- Lê sempre uma string

```
1  variavel_1 = input()
2
3  print("Variavel_1 é igual a " + variavel_1 + ".")
4  print(type(variavel_1))
5
6  variavel_2 = input("Por favor, informe o valor: ")
7
8  print("Variavel_2 é igual a " + variavel_2 + ".")
9  print(type(variavel_2))
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

COMMENTS

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia> & C:/ProgramData
ola
Variavel_1 é igual a ola.
<class 'str'>
Por favor, informe o valor: 1.2345
Variavel_2 é igual a 1.2345.
<class 'str'>
```

Função int

- Converte uma string contendo um inteiro em int
- Converte float em int
- Erro caso não seja int

```
1  variavel_int = int(input("Informe o seu número de inscrição: "))
2  print(variavel_int)
3  print(type(variavel_int))
4  print("")
5  print(int(1.2345)) # mais próximo inteiro menor que valor
6  print(30 / 7)
7  print(int(30 / 7)) # mais próximo inteiro menor que valor
8  print("")
9  print(int("1.2345"))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia> & C:/ProgramData/anaconda3/python.exe "g:/Meu Drive/Dro
Informe o seu número de inscrição: 12345
12345
<class 'int'>

1
4.285714285714286
4

Traceback (most recent call last):
  File "g:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia\oficinas\python\funcao_int.py", line 9, in <module>
    print(int("1.2345"))
ValueError: invalid literal for int() with base 10: '1.2345'
```

Função float()

- Converte string contendo um número em um float
- Não pode conter ,

```
1  variavel_float = float(input("Informe o valor pago: "))
2  print(variavel_float)
3  print(type(variavel_float))
4  print("")
5  print(float(1.2345))
6  print(30 / 7)
7  print(float(30 / 7))
8  print("")
9  print(float("1,234"))
10
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

COMMENTS

PS G:\Meu Drive\Dropbox_IFSUL_clube-de-ia> & C:/ProgramData/anaconda3/python.exe "g:/Meu Drive/Dropbox"

Informe o valor pago: 34.99

34.99

<class 'float'>

1.2345

4.285714285714286

4.285714285714286

Traceback (most recent call last):

File "g:\Meu Drive\Dropbox_IFSUL_clube-de-ia\oficinas\python\funcao_float.py", line 9, in <module>
 print(float("1,234"))

ValueError: could not convert string to float: '1,234'

FUNÇÕES

Funções

```
1 print("Isto")
2 print("é")
3 print("um")
4 print("Exemplo")
5
6 print("Isto")
7 print("é")
8 print("um")
9 print("Exemplo")
10
11 print("Isto")
12 print("é")
13 print("um")
14 print("Exemplo")
15
```

PROBLEMS OUTPUT DEBUG CONSOLE

```
PS G:\Meu Drive\Dropbox\_IFSUL
Isto
é
um
Exemplo
Isto
é
um
Exemplo
Isto
é
um
Exemplo
```



```
1 def print_exemplo():
2     print("Isto")
3     print("é")
4     print("um")
5     print("Exemplo")
6
7 print_exemplo()
8 print_exemplo()
9 print_exemplo()
10
```

PROBLEMS OUTPUT DEBUG CONSOLE

```
PS G:\Meu Drive\Dropbox\_IFSUL\_c
Isto
é
um
Exemplo
Isto
é
um
Exemplo
Isto
é
um
Exemplo
```

Funções

- Nomes de função seguem as mesmas regras de nomes de variáveis
- Permitem reaproveitar código
- Escondem a complexidade de quem usa a função
- Diminuem o tamanho de programas
- Palavra reservada def
- Identação
- Pode ter parâmetros
- Pode retornar valor ou não
- Boa prática: use 2 quebras de linha após a função

Funções com parâmetros

```
1 def imprime_resultado(nome_aluno, matricula, disciplina, nota):
2     print("O aluno " + nome_aluno + "(" + str(matricula) + ") obteve a nota " + str(nota) + " em " + disciplina )
3
4
5 aluno = "João Silva"
6 matricula = 12345
7 nota = 7.56
8
9 imprime_resultado(aluno, matricula, "matemática", nota)
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia> & C:/ProgramData/anaconda3/python.exe "g:/Meu Drive/Dropbox/_IFSUL/_clube-de-ia
O aluno João Silva(12345) obteve a nota 7.56 em matemática
```


Funções com parâmetros com valor default(padrão)

- Parâmetros com valor default devem estar no final da lista
- Opcionalmente, o valor pode ser informado na chamada

```
1 def imprime_resultado(nome_aluno, matricula, nota, disciplina="língua portuguesa"):
2     print("O aluno " + nome_aluno + "(" + str(matricula) + ") obteve a nota " + str(nota) + " em " + disciplina )
3
4
5 aluno = "João Silva"
6 matricula = 12345
7 nota_lingua_portuguesa = 8.32
8 nota_matematica = 7.56
9
10 imprime_resultado(aluno, matricula, nota_lingua_portuguesa)
11 imprime_resultado(aluno, matricula, nota_matematica, "matemática")
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia> & C:/ProgramData/anaconda3/python.exe "g:/Meu Drive/Dropbox/_IFSUL/_clube-de-ia.
"
O aluno João Silva(12345) obteve a nota 8.32 em língua portuguesa
O aluno João Silva(12345) obteve a nota 7.56 em matemática
```

Funções com retorno

```
1  def imprime_media_final(nota_1=0, nota_2=0, nota_3=0):
2      |   print(round((nota_1 + nota_2 + nota_3) / 3,2))
3
4
5  imprime_media_final(6.75, 7.94, 8.34)
6
7  def calcula_media_final(nota_1=0, nota_2=0, nota_3=0):
8      |   return round((nota_1 + nota_2 + nota_3) / 3, 2)
9
10
11 print( "A média final do aluno Joao Silva é " + str(calcula_media_final(6.75, 7.94, 8.34)))
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia> & C:/ProgramData/anaconda3/python.exe "g:/Meu Drive/Dropbox/7.68
```

```
A média final do aluno Joao Silva é 7.68
```

Imports e módulos

- importar módulo
- importar uma função de um módulo
- universal import

```
1  # import genérico
2  import random
3
4  print(random.randint(1, 10))
5
6  # importar uma função de um módulo
7
8  from random import randint
9
10 print(randint(1, 10))
11
12 # universal import|
13 from random import *
14
15 print(random())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-i
8
10
0.9173009374708329
```

Escopo de variáveis

- Pode ter uma variável dentro de função com mesmo nome de uma global (não são a mesma variável)
- Pode usar o mesmo nome em escopos diferentes
- Local não é usada no escopo global
- Globais podem ser usadas no escopo local
- Local de uma função não vale em outra função

```
1  exemplo = "Este é uma global"
2
3
4  def exemplo_variavel_local():
5      exemplo = "Esta é uma local"
6      return exemplo
7
8
9  print(exemplo_variavel_local())
10 print(exemplo)
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-  
.py"  
Esta é uma local  
Este é uma global
```

Local não é usada no escopo global

```
1 def exemplo_fruta():
2     fruta = "jabuticaba"
3     return fruta
4
5 variavel = exemplo_fruta()
6 print(fruta)
7
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia> & C:/ProgramData/anaconda3/python.exe "g:/Meu Drive/Dropbox/_IFSUL/_clube-de-ia/py"
Traceback (most recent call last):
  File "g:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia\oficinas\python\variavel_local_no_escopo_global.py", line 6, in <module>
    print(fruta)
NameError: name 'fruta' is not defined
```

Globais podem ser usadas no escopo local

```
1 def exemplo_fruta():  
2     print(fruta)  
3  
4 fruta = "jabuticaba"  
5 variavel = exemplo_fruta()  
6
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS G:\Meu Drive\Dropbox_IFSUL_clupy"
jabuticaba

Alterando variáveis globais em escopo local

- Só é possível alterar uma variável global dentro de uma função usando a palavra reservada `global`

`global <nome-da-variavel>`

```
1  def exemplo_fruta():
2      global fruta
3      fruta = "laranja"
4      print(fruta)
5
6  fruta = "jabuticaba"
7  variavel = exemplo_fruta()
8  print(fruta)
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERM

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube
alteracao.py"
laranja
laranja
```

INSTRUÇÕES IF/ELSE/ELIF

Operadores relacionais

- == !=
- > < >= <=
- Podem comparar tipos básicos e strings

Operadores lógicos

- and
- or
- not

Instrução if

```
1  fruta = input("Informe o nome de uma fruta: ")
2
3  if fruta == "jabuticaba":
4      print("É uma jabuticaba.")
5
6  print("O nome da fruta é: " + fruta)
7  print("")
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia> & C:/Program Files/Python/Python37-32/python.exe C:/Program Files/Python/Python37-32/python.exe
Informe o nome de uma fruta: jabuticaba
É uma jabuticaba.
O nome da fruta é: jabuticaba

PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia> & C:/Program Files/Python/Python37-32/python.exe C:/Program Files/Python/Python37-32/python.exe
Informe o nome de uma fruta: laranja
O nome da fruta é: laranja
```

Instrução else

```
1  fruta = input("Informe o nome de uma fruta: ")
2
3  if fruta == "jabuticaba":
4      print("É uma jabuticaba.")
5  else:
6      print("Não é uma jabuticaba.")
7
8  print("O nome da fruta é: " + fruta)
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
PS G:\Meu Drive\Dropbox\_IFSUL\_clube-de-ia> & C:/Program
Informe o nome de uma fruta: laranja
Não é uma jabuticaba.
O nome da fruta é: laranja
```

if e else aninhados

```
1  fruta = input("Informe o nome de uma fruta: ")
2  verde_ou_madura = input("Informe se a fruta está verde ou madura: ")
3
4  if fruta == "jabuticaba":
5      if verde_ou_madura == "verde":
6          print("É uma jabuticaba verde.")
7      else:
8          print("É uma jabuticaba madura.")
9  else:
10     print("Não é uma jabuticaba.")
11     print("O nome da fruta é: " + fruta)
12     if verde_ou_madura == "verde":
13         print ("A fruta está verde.")
14     else:
15         print ("A fruta está madura")
16
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

COMMENTS

PS G:\Meu Drive\Dropbox_IFSUL_clube-de-ia> & C:/ProgramData/anaconda3/python

```
Informe o nome de uma fruta: jabuticaba
Informe se a fruta está verde ou madura: madura
É uma jabuticaba madura.
```

PS G:\Meu Drive\Dropbox_IFSUL_clube-de-ia> & C:/ProgramData/anaconda3/python

```
Informe o nome de uma fruta: laranja
Informe se a fruta está verde ou madura: verde
Não é uma jabuticaba.
O nome da fruta é: laranja
A fruta está verde.
```

Instrução elif

```
1  fruta = input("Informe o nome de uma fruta: ")
2
3  if fruta == "jabuticaba":
4      print("É uma jabuticaba.")
5  elif fruta == "laranja":
6      print("É uma laranja.")
7  elif fruta == "bergamota":
8      print("É uma bergamota.")
9  else:
10     print("O nome da fruta é: " + fruta)
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
PS G:\Meu Drive\Dropbox\_IFSUL\_club-de-ia> & C:/Progr
Informe o nome de uma fruta: bergamota
É uma bergamota.
```

LOOPS

Instrução while

- Condição para repetição
- Atenção com loops infinitos (a condição para término do loop nunca acontece)

Loop Infinito

```
oficinas > python > while.py > ...
```

```
1  cont = 1
2
3  while cont < 5:
4      print("Contador: ", cont)
5
```

```
oficinas > python > while.py > ...
```

```
1  cont = 1
2
3  while cont < 5:
4      print("Contador: ", cont)
5      cont += 1
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
• /bin/python3 "/mnt/g/Meu Drive/_IFSUL/_clul
Contador: 1
Contador: 2
Contador: 3
Contador: 4
○ root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/
```


Instrução for

- Percorre a string caractere por caractere
-

oficinas > python > for_in_string.py > ...

```
1 palavra = "IFSUL"
2
3 for letra in palavra:
4     print(letra)
5
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL,
I
F
S
U
L
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL,
```

Instrução for in range

- A função range gera uma sequência de valores
- For percorre a sequência atribuindo cada valor (
- Outras opções de range:

oficinas > python > for_in_range.py > ...

```
1 teste_range = range(5)
2
3 print(teste_range)
4 print(list(teste_range))
5
6 for i in teste_range:
7     print(i)
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
● root@note-marcelo:/mnt/g/Meu Drive/_IFSUL
range(0, 5)
[0, 1, 2, 3, 4]
0
1
2
3
4
○ root@note-marcelo:/mnt/g/Meu Drive/_IFSUL
```

oficinas > python > for_in_range_outros.py > ...

```
1 teste_range = range(5, 10)
2 print(teste_range)
3 print(list(teste_range))
4
5 teste_range = range(10, 21, 2)
6 print(teste_range)
7 print(list(teste_range))
8
9 teste_range = range(5, -1, -1)
10 print(teste_range)
11 print(list(teste_range))
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CC

```
● root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-
range(5, 10)
[5, 6, 7, 8, 9]
range(10, 21, 2)
[10, 12, 14, 16, 18, 20]
range(5, -1, -1)
[5, 4, 3, 2, 1, 0]
50
○ root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-
```

Instrução for in range

- A função range gera uma sequência de valores
- For percorre a sequência atribuindo cada valor (
- Outras opções de range:

oficinas > python > for_in_range.py > ...

```
1 teste_range = range(5)
2
3 print(teste_range)
4 print(list(teste_range))
5
6 for i in teste_range:
7     print(i)
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
● root@note-marcelo:/mnt/g/Meu Drive/_IFSUL
range(0, 5)
[0, 1, 2, 3, 4]
0
1
2
3
4
○ root@note-marcelo:/mnt/g/Meu Drive/_IFSUL
```

oficinas > python > for_in_range_outros.py > ...

```
1 teste_range = range(5, 10)
2 print(teste_range)
3 print(list(teste_range))
4
5 teste_range = range(10, 21, 2)
6 print(teste_range)
7 print(list(teste_range))
8
9 teste_range = range(5, -1, -1)
10 print(teste_range)
11 print(list(teste_range))
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CC

```
● root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-
range(5, 10)
[5, 6, 7, 8, 9]
range(10, 21, 2)
[10, 12, 14, 16, 18, 20]
range(5, -1, -1)
[5, 4, 3, 2, 1, 0]
51
○ root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-
```

STRINGS 2

.upper() e .lower()

- .isalpha() : somente letras
- .isalnum() : somente números e letras
- .isdecimal() : somente números
- .isspace() : somente espaços
- .istitle() : Estilo título
- .title() : Cada primeira letra de palavra maiuscula

oficinas > python > upoer_lower.py > ...

```
1 minusculas = "não há letras maiusculas!"
2 print(minusculas.upper())
3 print(minusculas)
4 maiusculas = "NAO HÁ LETRAS MINUSCULAS!"
5 print(maiusculas.lower())
6 print(maiusculas)
7
8 print(maiusculas.islower())
9 print(maiusculas.isupper())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEVIEW

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-
NÃO HÁ LETRAS MAIUSCULAS!
não há letras maiusculas!
nao há letras minusculas!
NAO HÁ LETRAS MINUSCULAS!
False
True
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-
```

.startswith() e .endswith()

```
oficinas > python > startswith_endswith.py
1 print("Hello World".startswith("Hello"))
2 print("Hello World".startswith("World"))
3 print()
4 print("Hello World".endswith("Hello"))
5 print("Hello World".endswith("World"))
6
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEW

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-i
True
False

False
True
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-i
```

.join()

- Une as palavras separadas por um separador (a string definida no início)

```
oficinas > python > string_join.py
1  print("---".join(["um", "dois", "tres"]))
2
3  print("@".join(["usuario", "dominio.com"]))
4

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  CODEWHISPERER

● root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# ,
  um---dois---tres
  usuario@dominio.com
○ root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# |
```

.split()

- Gera uma lista a partir de uma string com separadores
- O separador padrão é um espaço “ “

```
oficinas > python > string_split.py
```

```
1 print("nome data_nascimento email".split())
2
3 print("nome,data_nascimento,email".split(","))
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER

```
● root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bi
['nome', 'data_nascimento', 'email']
['nome', 'data_nascimento', 'email']
○ root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```


.split()

- Gera uma lista a partir de uma string com separadores
- O separador padrão é um espaço “ “

```
oficinas > python > string_split.py
```

```
1 print("nome data_nascimento email".split())
2
3 print("nome,data_nascimento,email".split(","))
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER

```
● root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bi
['nome', 'data_nascimento', 'email']
['nome', 'data_nascimento', 'email']
○ root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

.rjust() e ljust()

- Garante um tamanho mínimo para uma string alinhando à direita ou à esquerda e preenchendo os caracteres faltantes
- A string padrão é um espaço “ ”
- Também existe o .center()

```
oficinas > python > string_ljust_rjust.py
```

```
1 print("Hello World".rjust(20))
2 print("Hello World".rjust(20, "-"))
3 print("Hello World".ljust(20, "-"))
4 print("123".rjust(10, "0"))
5
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

● root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_club
    Hello World
    -----Hello World
    Hello World-----
    0000000123
○ root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_club
```

.strip(), rstrip() e .lstrip()

- Elimina caracteres/strings de uma string
- rstrip elimina ao final (direita)
- lstrip elimina no início (esquerda)

```
oficinas > python > string_strip.py
1 print("Hello World!aaaaaa".strip("a"))
2 print("Hello World!aaaaaa".lstrip("a"))
3 print("Hello World!aaaaaa".rstrip("a"))
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEV

```
● root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-
Hello World!
Hello World!aaaaaa
Hello World!
○ root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-
```

.replace()

- Substitui uma string por outra

```
oficinas > python > string_replace.py
```

```
1 print("Boa tarde.".replace("tarde", "noite"))  
2
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

CODEWHISPERER

```
● root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bi  
Boa noite.  
○ root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

.len()

- Retorna o tamanho de uma string

```
oficinas > python > string_len.py
1 print(len("Casa"))
2 print(len("Casa "))
```

PROBLEMS OUTPUT DEBUG CONSOLE

```
● root@note-marcelo:/mnt/g/Meu Drive: # python string_len.py
4
6
○ root@note-marcelo:/mnt/g/Meu Drive: #
```

.format()

- Formata uma string utilizando variáveis

```
oficinas > python > string_format.py > ...
```

```
1 nome = input("Qual o seu nome? ")
2 idade = input("Qual a sua idade? ")
3 peso = input("Qual o seu peso? ")
4
5 print(nome, "tem", idade, "anos e pesa", peso, "kg")
6 print("{nome} tem {idade} anos e pesa {peso} kg".format(nome=nome, idade=idade, peso=peso))
7 print(f"{nome} tem {idade} anos e pesa {peso} kg")
8 print("{} tem {} anos e pesa {} kg".format(nome, idade, peso))
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE LOG

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bin/python3 "/mnt/g/Meu Drive/_IFSUL/_clube-de-ia/of
Qual o seu nome? joao
Qual a sua idade? 15
Qual o seu peso? 61
joao tem 15 anos e pesa 61 kg
joao tem 15 anos e pesa 61 kg
joao tem 15 anos e pesa 61 kg
joao tem 15 anos e pesa 61 kg
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

LISTAS

Exemplos de Listas

- Contém elementos separados por vírgulas
- Cada elemento pode ser de qualquer tipo
- Uma lista pode ser um elemento em outra

oficinas > python >  listas_exemplos.py > ...

```
1  lista_1 = [3, 2, 1]
2  lista_2 = [3.14, 2.793]
3  lista_3 = ["ola", "mundo", "!", "Que", "belo", "dia!"]
4  lista_4 = [True, False, True, True, False]
5  lista_5 = [[1,2], [3,4,5], [6,7,8,9]]
6  lista_6 = [7, 3.14, "mundo", True, [1,2]]
7  lista_7 = []
```


list()

- Usada para converter o conteúdo de uma variável para uma lista

```
oficinas > python > listas_list.py
1  print(list("Olá mundo"))
2

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  CODEWHISPERER
```

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /b.
['O', 'l', 'á', ' ', 'm', 'u', 'n', 'd', 'o']
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

in e not in

- Verifica se um elemento existe em alguma posição da lista

```
oficinas > python > listas_in_not_in.py > ...
```

```
1  lista = [1,2,3,4,5,6,7,8,9,10]
2  print(4 in lista)
3  print(4 not in lista)
4  print(12 in lista)
5  print(12 not in lista)
6
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /b:
True
False
False
True
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

índices e slicing

oficinas > python >  listas_indices_e_slicing.py > ...

```
1  lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2  print("1)", lista[2])           # elemento de índice 2
3  print("2)", lista[5:])          # elementos de índice 5 até o final
4  print("3)", lista[2:5])         # elementos de índice 2 até o 5
5  print("4)", lista[2:5:2])       # elementos de índice 2 até o 5 pulando de dois em dois
6  print("5)", lista[::2])         # elementos de índice 0 até o final pulando de dois em dois
7  print("6)", lista[::-1])        # elementos de índice 0 até o final de trás pra frente
8  print("7)", lista[2:4] + lista[4:2:-1]) # elementos de índice 2 até o 4 e depois de 4 até o 2 de trás pra frente
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE LOG

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bin/python3 "/mnt/g/Meu Drive/_IFSUL/_clube-de-ia/oficinas/python/listas_i
1) 3
2) [6, 7, 8, 9, 10]
3) [3, 4, 5]
4) [3, 5]
5) [1, 3, 5, 7, 9]
6) [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
7) [3, 4, 5, 4]
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

Lista dentro de lista

- O primeiro índice indexa a lista mais externa
- O segundo índice indexa a lista retornada por lista[1] ([4, 5, 6])

```
oficinas > python > listas_elemento_lista.py > ...
```

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

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

CODEWHISPERER

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bi  
6  
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

Índice negativo

- Conta a partir de -1 que é o último elemento da lista

```
oficinas > python > listas_indice_negativo.py > ...
```

```
1  lista = [1,2,3,4,5,6,7,8,9,10]
2  print(lista[-1])
3  print(lista[-2])
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bi
10
9
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

Alterando um elemento

- É possível alterar um elemento específico de uma lista

oficinas > python > listas_alterando_elemento.py > lista

```
1 lista = [1,2,3,4,5,6,7,8,9,10]
2 print(lista)
3 lista[5] = 999
4 print(lista)
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

CODEWHISPERER

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bi
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 999, 7, 8, 9, 10]
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

del e .remove()

- del remove pelo índice
- remove() remove pelo valor do item

oficinas > python > listas_del.py > ...

```
1  lista = ["maçã", "banana", "laranja", "pera", "uva"]
2  del lista[0]
3  print(lista)
4  lista.remove("banana")
5  print(lista)
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

CODEWHISPERER REFERENCE

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bin/pytho
['banana', 'laranja', 'pera', 'uva']
['laranja', 'pera', 'uva']
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

.append() e .insert()

- append() adiciona um elemento no final da lista
- insert() insere um elemento em uma posição na lista

oficinas > python > listas_append.py > ...

```
1  lista = ["banana", "maçã", "laranja", "pera", "uva"]
2  print(lista)
3  lista.append("morango")
4  print(lista)
5  lista.insert(2, "melancia")
6  print(lista)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bin/python3
['banana', 'maçã', 'laranja', 'pera', 'uva']
['banana', 'maçã', 'laranja', 'pera', 'uva', 'morango']
['banana', 'maçã', 'melancia', 'laranja', 'pera', 'uva', 'morango']
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```


.sort()

- Ordena uma lista
- Ordenação reversa (reverse=True)
- Tipos mistos (False -> 0 e True -> 1)
- Ordenação ASCII
- Ordenação não case sensitive (key=str.lower)

```
oficinas > python > listas_sort.py > lista
1 lista = [3.14, 5, -16, 0.0001, 123000, 0]
2 print('Original:', lista)
3 lista.sort()
4 print(lista)
5 lista = ["banana", "maçã", "abacate", "melão"]
6 print('Original:', lista)
7 lista.sort()
8 print(lista)
9 lista.sort(reverse=True)
10 print(lista)
11 lista = [False, 3.14, -2, -5.93, True, -0.0001]
12 print('Original:', lista)
13 lista.sort()
14 print(lista)
15 lista = ["Banana", "maçã", "abacate", "Melão"]
16 print('Original:', lista)
17 lista.sort()
18 print(lista)
19 lista.sort(key=str.lower)
20 print(lista)
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER RE
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bin/
Original: [3.14, 5, -16, 0.0001, 123000, 0]
[-16, 0, 0.0001, 3.14, 5, 123000]
Original: ['banana', 'maçã', 'abacate', 'melão']
['abacate', 'banana', 'maçã', 'melão']
['melão', 'maçã', 'banana', 'abacate']
Original: [False, 3.14, -2, -5.93, True, -0.0001]
[-5.93, -2, -0.0001, False, True, 3.14]
Original: ['Banana', 'maçã', 'abacate', 'Melão']
['Banana', 'Melão', 'abacate', 'maçã']
['abacate', 'Banana', 'maçã', 'Melão']
```

.index()

- retorna a posição de um elemento em uma lista

oficinas > python >  listas_index.py > ...

```
1  lista = [1,2,3,4,5,6,7,8,9,10]
2  print(lista.index(7))
3  lista = ["banana","maçã","laranja","pera","uva","laranja"]
4  print(lista.index("laranja"))
5  |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE LOG

```
▶ root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bin/python3 "/n
6
2
▶ root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# []
```

.pop()

- Remove um elemento de uma lista retornando o valor
- Padrão é o último elemento

oficinas > python > listas_pop.py > ...

```
1  lista = ["banana", "maçã", "pera", "uva"]
2  print(lista)
3  print(lista.pop())
4  print(lista)
5  lista = ["banana", "maçã", "pera", "uva"]
6  print(lista)
7  print(lista.pop(1))
8  print(lista)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bi
['banana', 'maçã', 'pera', 'uva']
uva
['banana', 'maçã', 'pera']
['banana', 'maçã', 'pera', 'uva']
maçã
['banana', 'pera', 'uva']
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

listas e strings

- Listas são mutáveis
- Strings são imutáveis (não suportam atribuições para um caractere)

listas atribuídas por referência

- Existe apenas uma lista sendo manipulada
- Use `.copy()` para criar uma cópia independente


```
oficinas > python > listas_referencia.py > ...
1  lista = [1,2,3,4,5,6,7,8,9,10]
2  lista_2 = lista
3  lista_2[5] = 999
4  print("lista ", lista)
5  print("lista_2", lista_2)
6  lista_2 = lista.copy()
7  lista_2[5] = 777
8  print("lista ", lista)
9  print("lista_2", lista_2)
10
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  CODEWHISPERER

root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bi
lista  [1, 2, 3, 4, 5, 999, 7, 8, 9, 10]
lista_2 [1, 2, 3, 4, 5, 999, 7, 8, 9, 10]
lista  [1, 2, 3, 4, 5, 999, 7, 8, 9, 10]
lista_2 [1, 2, 3, 4, 5, 777, 7, 8, 9, 10]
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

\ para continuação

- Use \ para continuar uma expressão ou código na próxima linha

oficinas > python >  barra_para_continuacao.py > ...

```
1 soma = 4 + 3 + \  
2     2 + 1  
3 print(soma)  
4 string_longa = "Olá, mundo! \  
5 Que dia lindo!"  
6 print(string_longa)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bi  
10  
Olá, mundo! Que dia lindo!  
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

DICIONÁRIOS

Dicionários

- Estrutura de pares chave e valor
- cada item pode ser referenciado como uma variável qualquer
- Chave e valor podem ser de qualquer tipo
- Use a função `.get()` quando a chave pode não existir no dicionário

oficinas > python >  dicionarios.py > ...

```
1 professores = {"matematica": "Maria", "fisica": "Joao", "quimica": "Jose"}
2 print(professores["matematica"])
3 print(professores.get("fisica"))
4 print("O professor de fisica é " + professores["fisica"])
5 print(f'O professor de química é {professores["quimica"]}')
6
```

PROBLEMS

OUTPUT

DEBUG CONSOLE


TERMINAL

CODEWHISPERER REFERENCE LOG

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bin/python3 "/mnt/g/Meu Drive/_IFSUL/_clube-de-ia/dicionarios.py"
Maria
Joao
O professor de fisica é Joao
O professor de química é Jose
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```


in e not in

- Verifica se um valor é igual a alguma chave
- Para verificar pelos valores user .values()

oficinas > python >  dicionarios_in_not_in.py > ...

```
1 professores = {"matematica": "Maria", "fisica": "Joao", "quimica": "Jose"}
2 print("1)", "fisica" in professores)
3 print("2)", "biologia" in professores)
4 print("3)", "matematica" not in professores)
5 print("4)", "Jose" in professores)
6 print("5)", "Jose" in professores.values())
7
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE LOG

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bin/python3 "/mnt/g/Meu Drive/_IFSUL/_clube-de-ia/dicionarios_in_not_in.py"
1) True
2) False
3) False
4) False
5) True
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

.keys(), .values() e .items

- .keys() retorna chaves
- .values() retorna valores
- .items() retorna os pares chave e valor

oficinas > python > dicionarios_metodos.py > professores

```
1 professores = {"matematica": "Maria", "fisica": "Joao", "quimica": "Jose"}
2 print("KEYS:")
3 for chave in professores.keys():
4     print(chave)
5 print("VALORES:")
6 for valor in professores.values():
7     print(valor)
8 print("ITENS:")
9 for chave, valor in professores.items():
10     print(chave, valor)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE LOG

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bin/python3 "/mnt/g/Meu Drive/_IFSUL/_clube-de-ia/dicionarios_metodos.py"
KEYS:
matematica
fisica
quimica
VALORES:
Maria
Joao
Jose
ITENS:
matematica Maria
fisica Joao
quimica Jose
```

.pop()

- remove um par chave e valor pela chave retornando o valor

```
oficinas > python > dicionarios_pop_popitem.py > ...
```

```
1 professores = {"matematica": "Maria", "fisica": "Joao", "quimica": "Jose"}
2 print(professores.pop("fisica"))
3 print(professores)
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL


CODEWHISPERER REFERENCE LOG

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bin/python3 "/mnt/g/Meu Drive/_I
Joao
{'matematica': 'Maria', 'quimica': 'Jose'}
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

TUPLAS

Tuplas

- Podem ser criadas com parênteses: (<valores>)
- Podem ser criadas usando a função tuple
- Podem ser acessadas usando índices e slicing da mesma forma que listas
- São imutáveis
- Métodos .count(), .index()

```
oficinas > python >  tuplas.py > ...
```

```
1  tupla_1 = (1, 2, 3, 3, 2)
2  tupla_2 = (3.14, "João", False, [1, 2, 3])
3  print(tupla_1)
4  print(tupla_2)
5  tupla_3 = tuple([321, 3.14, 87])
6  tupla_4 = tuple("Python")
7  print(tupla_3[1])
8  print(tupla_4[1:3])
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

CODEWHISP

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
(1, 2, 3, 3, 2)
(3.14, 'João', False, [1, 2, 3])
3.14
('y', 't')
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

CONJUNTOS

Conjuntos

- Similar a lista, porém sem elementos repetidos e sem ordem

oficinas > python > conjuntos.py > ...

```
1 set_1 = {1, 2, 3, 4, 4, 3, 5}
2 set_2 = set(["a", "b", "c", "d", "d"])
3 print("c" in set_2)
4 print(set_1)
5 print(set_2)
6 for item in set_1:
7     print(item)
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISP

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
True
{1, 2, 3, 4, 5}
{'d', 'c', 'a', 'b'}
1
2
3
4
5
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia#
```

Métodos para conjuntos

- add()
- remove() e discard()
- copy() e is
- union ou |
- intersection ou &
- difference ou -

oficinas > python > conjuntos_metodos.py > ...

```
1 set_1 = {"banana", "maçã", "pera", "uva"}
2 set_1.add("laranja")
3 print(set_1)
4 set_1.remove("banana") # Gera erro se o elemento não existir
5 set_1.discard("banana") # Não gera erro se o elemento não existir
6 print(set_1)
7 set_2 = set_1.copy() # Cria uma cópia do conjunto
8 print(set_2 is set_1) # Verifica se referencia a mesma lista
9 set_2.discard("pera")
10 print(set_1)
11 print(set_2)
12 print( set_1.union({3, 4, 5}) ) # Faz a união entre dois conjunto
13 set_1.clear()
14 print(set_1)
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

CODEWHISPERER REFERENCE LOG

```
root@note-marcelo:/mnt/g/Meu Drive/_IFSUL/_clube-de-ia# /bin/python3 "/mnt/g/Meu
{'banana', 'laranja', 'uva', 'pera', 'maçã'}
{'laranja', 'uva', 'pera', 'maçã'}
False
{'laranja', 'uva', 'pera', 'maçã'}
{'maçã', 'uva', 'laranja'}
{'uva', 3, 4, 'pera', 5, 'maçã', 'laranja'}
set()
```


Referências

- <https://www.python.org/>
- <https://www.udemy.com/course/python-for-absolute-beginners-u/>