PROGRAMAÇÃO
EM LINGUAGEM PYTHON

Autoria de Carolina Soares

A LINGUAGEM PYTHON

- Uma das linguagens mais populares
- De alto nível
- Orientada a objetos
- Interpretada de script
- Imperativa
- Funcional
- Tipagem dinâmica e forte
- Surgiu em 1991
- Lançada por Guido van Rossum
- Desenvolvida em comunidade aberta
- Mantida pela organização Python Software Foundation

TODAS AS AULAS E EXERCÍCIOS

https://github.com/MariaCarolinass/aulas-python

A COMUNIDADE PYTHON

grupy.rn

PYTHON BRASIL 2023

django girls
Pernaíba 2023

PythonSul
Jaraguá do Sul 2022

PYTHON NORTE 2023

AFROPYTHON

APyB
Associação Python Brasil

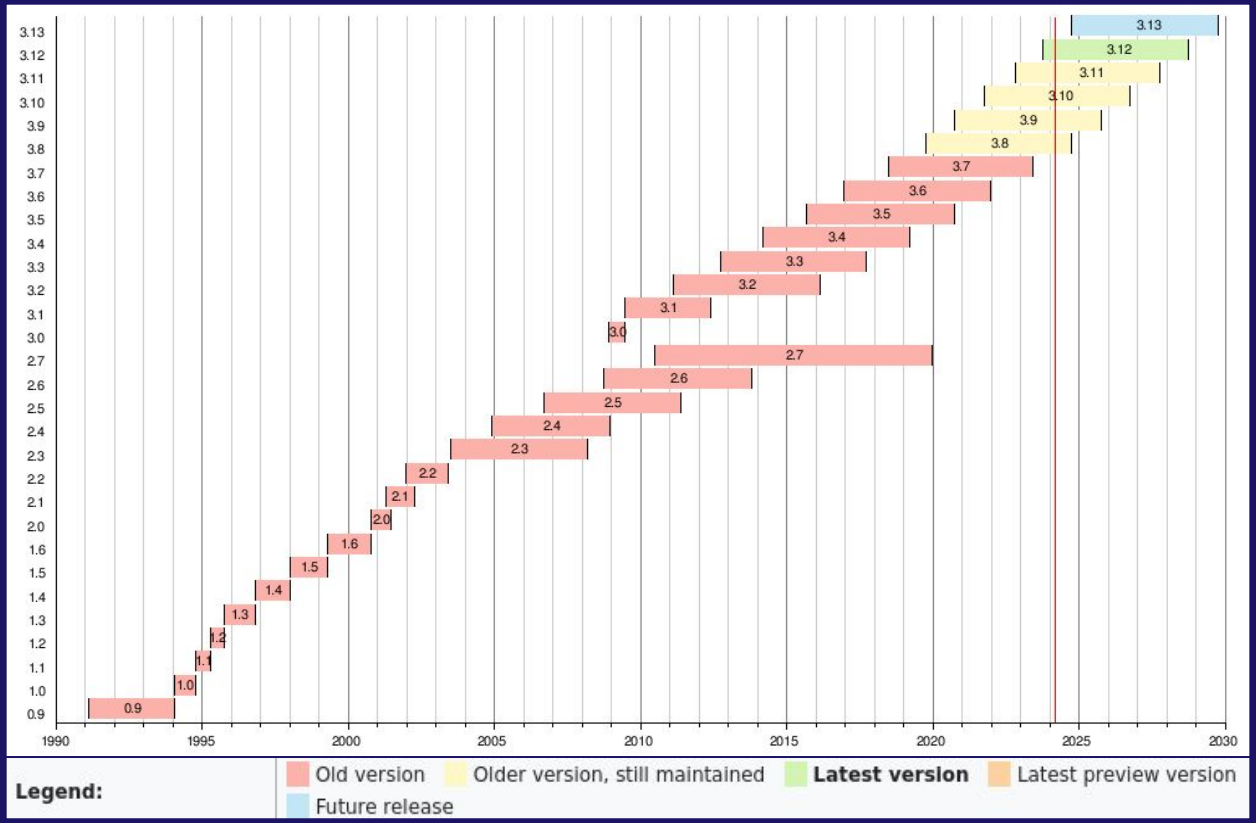
ONDE UTILIZAR PYTHON?

- Construção de sistemas Web
- Análise de dados
- Inteligência Artificial
- Machine Learning
- Construção de aplicativos
- Construção de sistemas desktop

TUDO SOBRE PYTHON

https://wiki.python.org.br

ATUALIZAÇÕES DO PYTHON



https://en.wikipedia.org/wiki/History_of_Python

COMO COMEÇAR?

- Ter acesso ao um Computador / Notebook / Smartphone com Internet
- Ter um editor (texto ou código) / IDE
- Fazer o Download do Python

EXIBINDO UMA MENSAGEM

```
print("Hello, World!")
```

```
Hello, World!
```

EXECUTANDO UM PROGRAMA

```
$ python nome_arquivo.py
```

Execução: terminal / cmd

Editores de código: Notepad++, Sublime Text, Atom

Editores de código e execução: Visual Studio Code, PyCharm, Google Colab

VARIÁVEIS

```
num1 = 10
num2 = 5.2
msg = "Olá, eu estou estudando Python"
b = True
nome = input("Digite o seu nome: ")
```

TIPOS PRIMITIVOS

Tipo	Descrição	Exemplos
`int`	Número de precisão fixa	`42, 105, -12, 7`
`float`	Ponto flutuante	`3.7436, 2.1, 0.72632`
`bool`	Booleano	`True` ou `False`
`str`	Cadeia de caracteres	`'Olá, mundo'`

CONVERTENDO TIPO DA VARIÁVEL

```
num = float(“25.30”)
print(type(num))
```

Saída do código:

```
<class 'float'>
```

SOLICITANDO UMA ENTRADA

```
nome = input("Digite o seu nome:")
print(nome)

num = input("Digite um número inteiro: ")
print(num)

num = input("Digite um número flutuante: ")
print(num)
```

Saída do código:

```
Maria
12
8.2
```

DECLARANDO TIPO DA ENTRADA

```
num = int(input("Digite um número inteiro: "))
num = float(input("Digite um número com ponto flutuante: "))
nome = str(input("Digite seu nome: "))
nome = bool(input("Digite True ou False: "))
```

FORMATANDO MENSAGENS COM FORMAT

```
print("Olá, {}.".format("Maria")
print("Meu nome é {nome}.".format(nome = "Maria"))
print("Meu nome é {} eu tenho {} anos".format(nome = "Maria",
idade = 18))
```

Saída do código:

```
Meu nome é Maria eu tenho 18 anos
```

FORMATANDO MENSAGENS COM F-STRINGS

```
nome = "João"
print(f"Nome: {nome}")
```

Saída do código:

```
Nome: João
```

FORMATANDO CASAS DECIMAIS

```
pi = 3.14159265359
print(f"Pi é igual a {pi:.2f}")
```

Saída do código:

```
Pi é igual a 3.14
```

FORMATANDO CASAS DECIMAIS COM ROUND

```
pi = round(3.14159265359, 2)
print(f"Pi é igual a {pi}")
```

Saída do código:

```
Pi é igual a 3.14
```

EXERCÍCIOS

1. Imprima o seu nome completo.
2. Declare 3 variáveis para guardar o nome, idade e altura de uma pessoa, atribua os valores nas variáveis e imprima cada uma.
3. Peça para o usuário digitar uma frase e imprima a frase.
4. Peça para o usuário informar o seu nome e idade. Imprima em uma mensagem o nome e a idade usando a formatação f-strings. Não esqueça de adicionar os tipos de entrada para nome e idade.
5. Faça um programa que pergunte qual a cor favorita do usuário e guarde em uma variável, logo em seguida imprima a mensagem “a cor favorita do usuário é” com a cor impressa usando a formatação format.

OPERADORES E EXPRESSÕES

Autoria de Carolina Soares

OPERADORES ARITMÉTICOS

Operador	Nome	Exemplo	Saída
+	Adição	5 + 2	7
-	Subtração	5 - 2	3
*	Multiplicação	5 * 2	10
/	Divisão	5 / 2	2.5
//	Divisão Inteira	5 // 2	2
%	Resto da divisão	5 % 2	1
**	Exponenciação	5 ** 2	25

OPERADORES ARITMÉTICOS

```
soma = 5 + 2

subt = 5 - 2

multi = 5 * 2

divi = 5 / 2

print(f"Soma: {soma} \n
Subtração: {subt} \n
Multiplicação: {multi} \n
Divisão: {divi}")
```

Saída do código:

```
Soma: 7

Subtração: 3

Multiplicação: 10

Divisão: 2.5
```

OPERADORES RELACIONAIS

Operador	Descrição	Exemplo	Saída
==	Igual	5 == 2	False
!=	Diferente	5 != 2	True
>	Maior	5 > 2	True
<	Menor	5 < 2	False
>=	Maior igual	5 >= 2	True
<=	Menor igual	5 <= 2	False

OPERADORES RELACIONAIS

```
a, b = 5, 3
a > b
a == b
a != b
a < b
```

Saída do código:

```
True
False
True
False
```

OPERADORES LÓGICOS

Operador	Descrição	Exemplo	Saída
and	e	True and True True and False False and True False and False	True False False False
or	ou	True or True True or False False or True False or False	True True True False
not	não	not True not False	False True

OPERADORES LÓGICOS

```
a, b, c = 10, 4, 2
a > b and b < c
a > b or b < c
not a > b or b < c
```

Saída do código:

```
False
True
False
```

OPERADORES DE ATRIBUIÇÃO

Operador	Exemplo	Equivalente a
=	x = valor	x = valor
+=	x += valor	x = x + valor
-=	x -= valor	x = x - valor
/=	x /= valor	x = x / valor
//=	x //= valor	x = x // valor
%=	x %= valor	x = x % valor
*=	x *= valor	x = x * valor

OPERADORES DE ATRIBUIÇÃO

```
a = 10
a %= 2
print(a)
```

Saída do código:

```
0
```

PRECEDÊNCIA DE OPERADORES

Operador
() **
* / % //
+ -
<= < > >=
== !=
= %= /= //= -= += = *=
not or and

PRECEDÊNCIA DE OPERADORES

```
print(2 ** 5 + 10 - (2 + 3))
```

Saída do código:

37

EXERCÍCIOS

1. Faça um programa que pergunte a altura e peso do usuário e depois calcule o IMC usando a seguinte fórmula: IMC = peso / (altura²). Imprima o resultado.

2. Receba dois números inteiros, em seguida exiba a soma, subtração, multiplicação, divisão, divisão inteira e o resto entre os dois números.

3. Faça um programa que receba três notas de um aluno, calcule a média entre as notas e mostre o resultado da média. Crie uma variável para guardar um valor do tipo booleano referente a aprovação do aluno, se o aluno tiver obtido uma média maior ou igual que 6 imprima uma mensagem de "aprovado" com o resultado da aprovação igual a True, senão imprima "reprovado" e mostre False como resultado do aluno.

4. Receba três variáveis a, b e c e aplique na fórmula para calcular o valor de delta: b² - 4 x a x c. Com o resultado calcule as raízes declarando duas variáveis x1 e x2 e aplique a fórmula: x1 = - b + √delta / 2 x a; x2 = - b - √delta / 2 x a. Imprima o resultado de delta, X1 e X2. Confira sempre a precedência das operações na fórmulas e use delta na potência de 1/2 para calcular as raízes quadráticas.

5. Pergunte um número ao usuário e atribua com o operador de atribuição, o valor do dobro do número e exiba o resultado.

ESTRUTURAS CONDICIONAIS

Autoria de Carolina Soares

ESTRUTURA CONDICIONAL IF

O condicional if verifica se uma condição é verdadeira e executa o bloco de código dentro dele.

```
if <condicao>:
    # executar caso a condição seja verdade
```

ESTRUTURA CONDICIONAL IF

```
if 5 > 3:
    print("5 é maior que 3")

if 3 == 5:
    print("5 é igual 3")

print("testando a condição if")
```

Saída do código:

```
5 é maior que 3
testando a condição if
```

ESTRUTURA CONDICIONAL IF

```
num = int(input("Digite um número: ")
if num != 0:
    print("Número é diferente de zero")
if num == 0:
    print("Número é igual a 0")
if num >= 1 and num <= 10:
    print("Número está entre 1 e 10")
```

ESTRUTURA CONDICIONAL ELSE

O condicional else executa um bloco de código dentro dele caso todas as outras condições que estão em cima tenham retornado falso.

```
if <condicao>:
    # se a condição for verdadeira execute
else:
    # execute caso a primeira condição não seja verdade
```

ESTRUTURA CONDICIONAL ELSE

```
num = int(input("Digite um número: "))
if num == 0:
    print("Número é igual a 0")
else:
    print("Número é diferente de zero")
```

ESTRUTURA CONDICIONAL ELSE

```python
linguagem = "Python"
if linguagem == "Python":
    print("A linguagem é Python.")
else:
    print("Linguagem não identificada!")
```

ESTRUTURA CONDICIONAL ELIF

O condicional elif é executado quando o primeiro if da estrutura retorna falso. O elif pode ser utilizado várias vezes após o if.

```
if <condicao>:
    # execute se a condição for verdadeira
elif <condicao>:
    # execute se a primeira condição não for satisfeita
elif <condicao>:
    # execute se a segunda condição não for satisfeita
else:
    # execute se nenhuma das condições acima forem atendidas
```

ESTRUTURA CONDICIONAL ELIF

```python
linguagem = "Python"
if linguagem == "Python":
    print("A linguagem é Python.")
elif linguagem == "C#":
    print("A linguagem é C#.")
elif linguagem == "Java":
    print("A linguagem é Java.")
else:
    print("Linguagem não identificada!")
```

EXERCÍCIOS

1. Escreva um programa que receba três notas de um aluno, calcule a sua média e compare, se a média for menor que 4 exiba uma mensagem “Reprovado por média”, se a média for maior ou igual que 4 e menor que 6 exiba uma mensagem “Em recuperação” e se a média for maior ou igual que 6 exiba uma mensagem “Aprovado por média”. Considere a menor nota como 0 e a maior nota igual a 10.

2. Faça um programa que receba um número e verifica se ele é par ou ímpar e imprima a mensagem “É par” para números pares e “É ímpar” para números ímpares.

3. Solicite a idade do usuário e se já possui Carteira de habilitação com a resposta Sim ou Não. Verifique se o usuário é maior ou menor de idade, se a idade for maior ou igual que 18 anos e a resposta para habilitação for Sim, exiba uma mensagem “Hábito para dirigir”, mas se a resposta para habilitação for Não, independente da idade exiba uma mensagem “NÃO hábito para dirigir”.

4. Escreva um programa que receba dois números e pergunte qual operação matemática simples o usuário deseja realizar dentre as opções de soma, subtração, multiplicação e divisão. Verifique qual opção foi escolhida e mostre o resultado da conta entre os dois números.

EXERCÍCIOS

5. Faça um programa que pergunte qual conversão de graus o usuário deseja realizar de Celsius para Fahrenheit ou de Fahrenheit para Celsius, depois solicite a temperatura (int) e exiba o resultado da conversão. De Celsius para Fahrenheit utilize: 9 / 5 * Celsius + 32; E de Fahrenheit para Celsius utilize: 5 / 9 * (Fahrenheit - 32).

LAÇOS DE REPETIÇÕES

Autoria de Carolina Soares

COMO EXIBIR CADA LETRA DA PALAVRA “PYTHON”?

EXIBINDO AS LETRAS DE UMA PALAVRA

```
print("P")
print("y")
print("t")
print("h")
print("o")
print("n")
```

Saída do código:

```
P
y
t
h
o
n
```

COMO SIMPLIFICAR O CÓDIGO USANDO UM LOOP?

ESTRUTURA DE REPETIÇÃO FOR

O for executa cada item dado por uma sequência de dados como listas, tuplas, conjuntos ou strings, percorrendo pelos elementos até o fim da sequência.

```
for elemento in sequencia:
    print(elemento)
```

PERCORRENDO UMA STRING

```
for letra in "Python":
    print(letra)
```

Saída do código:

```
P
y
t
h
o
n
```

SEM E COM ESTRUTURA DE REPETIÇÃO

```
print("P")
print("y")
print("t")
print("h")
print("o")
print("n")
```

```
for letra in "Python":
    print(letra)
```

Saída do código:

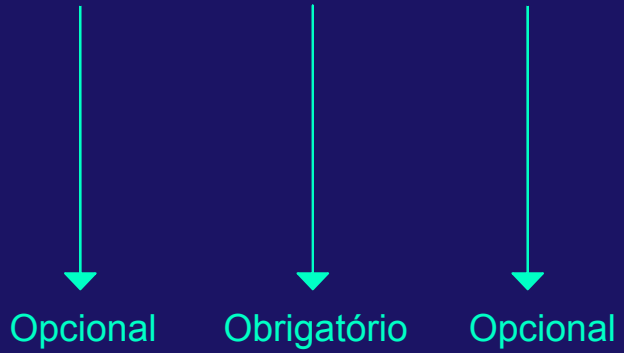
```
P
y
t
h
o
n
```

COMO EXIBIR NÚMEROS DE 1 – 10 COM ESTRUTURA DE REPETIÇÃO?

PERCORRENDO UMA SEQUÊNCIA NUMÉRICA

A função range combinada ao for itera a cada loop a um intervalo específico de repetições, que por padrão inicia em 0.

```
for elemento in range(iniciar, parar, passo):
    print(elemento)
```



USANDO A FUNÇÃO RANGE

```
for numero in range(10):
    print(numero)
```

1 2 3 4 5 6 7 8 9 10

Saída: 0 1 2 3 4 5 6 7 8 9

USANDO A FUNÇÃO RANGE

```
for numero in range(11):
    print(numero)
```

1 2 3 4 5 6 7 8 9 10 11

Saída: 0 1 2 3 4 5 6 7 8 9 10

USANDO A FUNÇÃO RANGE

```
for numero in range(1, 10+1):
    print(numero)
```

1 2 3 4 5 6 7 8 9 10

Saída: 1 2 3 4 5 6 7 8 9 10

USANDO A FUNÇÃO RANGE

```
for numero in range(10, 0, -1):
    print(numero)
```

1 2 3 4 5 6 7 8 9 10

Saída: 10 9 8 7 6 5 4 3 2 1

USANDO A FUNÇÃO RANGE

```
for numero in range(1, 11, 2):
    print(numero)
```

1 2 3 4 5

Saída: 1 3 5 7 9

ESTRUTURA DE REPETIÇÃO WHILE

O while executa um bloco de código ENQUANTO a condição aplicada a ele for satisfeita.

```
while <condicao>:
    # Bloco a ser executado
```

ESTRUTURA DE REPETIÇÃO WHILE

```
contador = 0
while contador < 10:
    print(contador)
    contador += 1
```

Saída do código:

```
0
1
2
3
4
5
6
7
8
9
```

MAIS SOBRE LAÇOS DE REPETIÇÕES

UTILIZANDO O ELSE NO FOR

```
for numero in range(10):
    print(numero)
else:
    print("Fim do loop")
```

Saída do código:

```
0
1
2
3
4
5
6
7
8
9
Fim do loop
```

UTILIZANDO O ELSE NO WHILE

```
contador = 0
while contador < 10:
    print(contador)
    contador += 1
else:
    print("Fim do loop")
```

Saída do código:

```
0
1
2
3
4
5
6
7
8
9
Fim do loop
```

AUXILIARES USADOS NO FOR E WHILE

```
for numero in range(10):
    if numero == 5:
        break # interromper um loop
    else:
        print(numero)
```

Saída do código:

```
0
1
2
3
4
5
```

AUXILIARES USADOS NO FOR E WHILE

```
contador = 0
while contador < 10:
    contador += 1
    if contador == 5:
        continue # pula um código abaixo dele
    print(contador)
```

Saída do código:

```
1
2
3
4
6
7
8
9
10
```

AUXILIARES USADOS NO FOR E WHILE

```
for numero in range(5000):
    pass # um código que não irá ser realizado

while False:
    pass # um código que não irá ser realizado
```

NA PRÓXIMA AULA VEREMOS LAÇOS DE REPETIÇÃO COM LISTAS E MUITO MAIS

EXERCÍCIOS

1. Faça um programa que mostre a contagem do 0 até 100 pulando a cada dezena, o último número a ser exibido deve ser o 100.
2. Faça um programa que mostre a contagem regressiva de 10 a 1.
3. Escreva um programa que pergunte um número inteiro ao usuário e enquanto o número digitado pelo usuário for positivo pergunte outro número. Ao final do programa exiba a soma dos números recebidos. O programa só para de pedir novos números se o número for menor que 0.
4. Solicite um número de 1 a 10 ao usuário e com o número escolhido exiba a sua tabuada inteira, usando laços de repetição. Por exemplo: 2 x 1 = 2, 2 x 2 = 4, ..., 2 x 10 = 20.
5. Pergunte 10 números para o usuário e a cada número exiba se ele é par ou ímpar.
6. Solicite um número ao usuário e verifique se número é primo ou não.

ESTRUTURAS DE DADOS

Autoria de Carolina Soares

ESTRUTURAS DE DADOS

O que veremos:

- Listas
- Tuplas
- Sets
- Dicionários

CARACTERÍSTICAS DAS ESTRUTURAS

Estrutura	Ordenado	Mutável	Construtor	Exemplo
`list`	sim	sim	`[]` ou `list()`	`[1, 2.5, 'Pedro', True]`
`tuple`	sim	não	`()` ou `tuple()`	`(1, 2.5, 'Pedro', True)`
`set`	não	sim	`{}` ou `set()`	`{1, 2.5, 'Pedro', True}`
`dict`	sim	sim	`{}` ou `dict()`	`{6: 'jun', 7: 'jul'}`

https://phylos.net/2021-03-08/python-sequencias-e-colecoes

ARMAZENANDO ITENS EM LISTAS

Uma lista é uma coleção ordenada de valores, separados por vírgula e dentro de colchetes [], elas são utilizadas para armazenar diversos itens em uma única variável.

```
list = []
print(type(list))
```

Saída do código:

```
<class 'list'>
```

CRIANDO LISTAS COM O RANGE

```
list(range(10))
```

Saída: `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`

CRIANDO LISTAS DE STRINGS

```
lista = ['uva', 'maçã', 'pêra', 'morango']
print(lista)
```

Saída: `['uva', 'maçã', 'pêra', 'morango']`

CRIANDO LISTAS MISTAS

```
lista = ['abc', 0.5, True, 90]
print(lista)
```

Saída: `['abc', 0.5, True, 90]`

VERIFICANDO TAMANHO DA LISTA

```
lista = ['abc', 0.5, True, 90]

print(len(lista))
```

Saída: 4

ACESSANDO UM ITEM NA LISTA

```
lista = ['uva', 'maçã', 'pêra', 'morango']
print(lista[0])
```

Saída: uva

```
print(lista[2])
```

Saída: pêra

```
print(lista[-1])
```

Saída: morango

LISTA DENTRO DA LISTA

```
usuarios = [
    ["Maria", 18],
    ["Roberto", 20],
    ["Laura", 25],
    ["Carlos", 19]
] # lista de usuários que guarda
uma lista de nomes e idades.
```

ACESSANDO UM ITEM DENTRO DE OUTRA LISTA

```
lista = [[2, 3, 5], [1, 2], [9, 6, 9]]

print(lista[1][0])
```

Saída: 1

FATIAMENTO DE LISTA

```
lista = [2, 4, 12, 20, 34, 1, 9]
lista[ inicio : fim : passo ]
```

FATIANDO UMA LISTA

```
lista = [2, 4, 12, 20, 34, 1, 9]
```

```
print(lista[2:5])
```

Saída: [12, 20, 34]

```
print(lista[:5])
```

Saída: [2, 4, 12, 20, 34]

```
print(lista[5:])
```

Saída: [1, 9]

```
print(lista[-3:6])
```

Saída: [34, 1]

```
print(lista[:5:2])
```

Saída: [2, 12, 34]

PERCORRENDO UMA LISTA

```
lista = ["uva", "maçã", "pêra", "morango"]

for l in lista:
    print(l)
```

Saída do código:

```
uva
maçã
pêra
morango
```

VERIFICANDO SE UM ITEM EXISTE

```
lista = ["uva", "maçã", "pêra", "morango"]

"pêra" in lista
```

Saída: True

```
"pêra" not in lista
```

Saída: False

VERIFICANDO SE UM ITEM EXISTE

```
lista = ["uva", "maçã", "pêra", "morango"]

if "pêra" in lista:
    print("Pêra existe na lista!")
else:
    print("Pêra NÃO existe na lista!")
```

Saída: Pêra existe na lista!

MÉTODOS DAS LISTAS

- `lista.append("valor1") # adiciona um valor`
- `lista.clear() # limpar lista`
- `lista.copy()  # copiar lista`
- `lista.count("valor1") # contar quantos elementos tem na lista com esse valor`
- `lista.extend(lista2) # adiciona no final da lista os elementos de outra lista`
- `lista.index("valor1") # retorna o índice onde o valor especificado estar`
- `lista.insert(2, "valor2") # insere elemento em posição especificada`
- `lista.pop() # remove e retorna elemento em posição especificada (se nenhum elemento é passado é removido o último elemento)`
- `lista.remove("valor2") # remove elemento especificado da lista`
- `lista.reverse() # inverte a ordem da lista`
- `lista.sort() # ordena a lista`

ARMAZENANDO ITENS EM TUPLAS

As tuplas funcionam da mesma maneira que as listas, mas uma vez criadas elas não podem ser mais mudadas.

```
tupla = ()
print(type(tupla))
```

Saída do código:

```
<class 'tuple'>
```

CRIANDO TUPLAS

```
pontoXY = (-2, 5)

pontoXYZ = (-1, 0.5, 3)

usuarioId = ("143535", "Maria")

produto = ("Camisa", "P", 56.00, True)
```

ACESSANDO UMA TUPLA

```
tupla = (2, 5, 3)

print(tupla[1])
```

Saída: 5

MÉTODOS DAS TUPLAS

- `tupla.count("valor1") # contar quantos elementos tem na lista com esse valor`
- `tupla.index("valor1") # retorna o índice onde o valor especificado estar`

ARMAZENANDO ITENS EM SETS

Os sets são conjuntos criados para armazenar vários itens, mas diferente das listas e tuplas, os sets são desordenados, parcialmente imutáveis, não possuem itens duplicados e não podem ser indexados.

```
conjunto = set()
print(type(conjunto))
```

Saída do código:

```
<class 'set'>
```

CRIANDO E OPERANDO COM SETS

```
x = {'a', 'f', 'c', 'j'}
y = {'c', 'a', 'f', 'd'}
uniao = x | y # ou x.union(y)
print(uniao)
Saída: {'c', 'd', 'f', 'a', 'j'}
diferenca = x - y  # ou x.symmetric_difference_update(y)
print(diferenca)
Saída: {'j'}
intercessao = x & y # ou x.intersection_update(y)
print(intercessao)
Saída: {'a', 'c', 'f'}
```

MÉTODOS DOS CONJUNTOS

- `conjunto.add("valor1") # insere elemento no set`
- `conjunto.clear() # remove todos os elementos do set`
- `conjunto.copy() # retorna cópia do set`
- `conjunto.difference(conjunto1) # retorna um set com a diferença entre 2 ou mais sets`
- `conjunto.difference_update(conjunto1) # remove elementos incluídos no segundo set`
- `conjunto.discard("valor2") # remove item especificado`
- `conjunto.intersection(conjunto1) # retorna o set interseção de 2 sets`
- `conjunto.intersection_update(conjunto1) # remove itens do set não presentes no segundo set especificado`
- `conjunto.isdisjoint(conjunto1) # retorna True se os 2 sets são disjuntos`
- `conjunto.issubset(conjunto1) # retorna True se o set é subconjunto do segundo set`
- `conjunto.issuperset(conjunto1) # retorna True se o set contém o segundo set`
- `conjunto.pop() # remove (e retorna) um elemento arbitrário do set`
- `conjunto.remove("valor3") # remove o elemento especificado`
- `conjunto.symmetric_difference(conjunto1) # retorna o set com a diferença simétrica de dois sets`
- `conjunto.symmetric_difference_update(conjunto1) # insere a diferença simétrica desse set em outro`
- `conjunto.union(conjunto1) # retorna um set com a união dos sets`
- `conjunto.update("valor1") # atualiza o primeiro set com sua união com um ou mais sets`

ARMAZENANDO ITENS EM DICIONÁRIOS

Os dicionários são usados para armazenar elementos com uma chave única, seguida de um valor, que pode ser do tipo lista, dicionário, inteiro, string e etc.

```
dicionario = {"chave": "valor"}
```

CRIANDO UM DICIONÁRIO

```
dicionario = {
    "nome": "maria",
    "idade": 22,
    "sexo": "feminino"
} print(dicionario)
```

Saída: `{'nome': 'maria', 'idade': 22, 'sexo': 'feminino'}`

ACESSANDO UM ITEM NO DICIONÁRIO

```
dicionario = {"nome": "maria", "idade": 22, "sexo": "feminino"}

print(dicionario["nome"])
```

Saída: `'maria'`

VERIFICANDO TAMANHO DO DICIONÁRIO

```
dicionario = {"nome": "maria", "idade": 22, "sexo": "feminino"}

print(len(dicionario))
```

Saída: 3

PERCORRENDO UM DICIONÁRIO

```
livros = {1: ["Dom Casmurro", "Machado de Assis"], 2: ["1984", "George Orwell"], 3: ["O Pequeno Príncipe", "Antoine de Saint-Exupéry"]}


for id in livros:
    print(f"Id: {id} | Livro: {livros[id]}")
```

Saída: Id: 1 | Livro: ['Dom Casmurro', 'Machado de Assis']

Saída: Id: 2 | Livro: ['1984', 'George Orwell']

Saída: Id: 3 | Livro: ['O Pequeno Príncipe', 'Antoine de Saint-Exupéry']

MÉTODOS DOS DICIONÁRIOS

- `dicionario.clear() # remove todos os elementos do dicionário`
- `dicionario.copy() # retorna uma cópia do dicionário`
- `dicionario.fromkeys("chave") # retorna dicionário com chaves e valores especificados`
- `dicionario.get("chave") # retorna o valor relativo à chave dada, ou valor default dado`
- `dicionario.items() # retorna uma lista contendo uma tupla para cada par chave:valor`
- `dicionario.keys() # retorna objeto iterável com as chaves do dicionário`
- `dicionario.pop() # remove o elemento relativo à chave especificada`
- `dicionario.popitem() # remove o último par chave:valor inserido`
- `dicionario.setdefault("chave") # retorna o valor relativo à chave dada. Se a chave não existe insere chave:valor`
- `dicionario.update({"chave": "valor"}) # Atualiza o dicionário com pares chave:valor dados`
- `dicionario.values() # retorna objeto iterável os valores do dicionário`

LISTAS E TUPLAS EM UM DICIONÁRIO

DICIONÁRIO COM TUPLA NA CHAVE

```
livros = {
    ("Dom Casmurro", "Machado de Assis"): 1,
    ("1984", "George Orwell"): 2,
    ("O Pequeno Príncipe", "Antoine de Saint-Exupéry"): 3
} print(livros)
```

Saída: `{('Dom Casmurro', 'Machado de Assis'): 1, ('1984', 'George Orwell'): 2, ('O Pequeno Príncipe', 'Antoine de Saint-Exupéry'): 3}`

DICIONÁRIO COM LISTA NO VALOR

```
livros = {
    1: ["Dom Casmurro", "Machado de Assis"],
    2: ["1984", "George Orwell"],
    3: ["O Pequeno Príncipe", "Antoine de Saint-Exupéry"]
} print(livros)
```

Saída: {1: ['Dom Casmurro', 'Machado de Assis'], 2: ['1984', 'George Orwell'], 3: ['O Pequeno Príncipe'], 4: ['Antoine de Saint-Exupéry']}

EXERCÍCIOS

1. Solicite 3 notas de um aluno e guarde em uma lista. Em seguida, mostre qual foi a maior nota da lista acessando ela e depois exiba a média entre as três notas.

2. Crie um programa que calcula a distância entre dois pontos, para isso crie duas tuplas para o ponto a e b. O ponto a irá receber valores para xa e ya e o ponto b receberá valores para xb e yb, cada valor será solicitado ao usuário. Em seguida, crie uma variável para calcular a distância entre os pontos das tuplas, use a fórmula: $\sqrt{((Xb - Xa)^2 + (Yb - Ya)^2)}$ e ao final mostre o resultado do cálculo. Lembre-se que para obter a raiz quadrática de um número eleve a potência de $^1/_2$.

3. Crie dois conjuntos e insira no primeiro conjunto números de 50 a 70 e no segundo conjunto insira números de 65 a 80, utilize o método range para simplificar a inserção dos números nos conjuntos. Logo, crie três variáveis, uma que irá mostrar a união entre esses conjuntos, outra que mostra a diferença entre os conjuntos e a última que exibe a interseção dos conjuntos.

4. Faça um programa que cria um dicionário de avaliação de linguagens de programação, onde conterá os elementos Python, JavaScript, C++, Java e PHP como chaves do dicionário. E o valor de cada elemento será solicitado para o usuário digitar uma avaliação de 0 a 5 para as linguagens. Em seguida, mostre qual foi a linguagem com melhor e pior avaliação do dicionário.

EXERCÍCIOS

5. Crie uma lista de pessoas com sub-listas que contenham nome, idade e estado civil de 5 pessoas. Utilize um loop para perguntar essas informações 5 vezes. Depois, pegue cada idade e mostre quem é o mais velho da lista e mostre a média entre todas as idades. Também mostre a quantidade de pessoas com estado civil, solteiro e casado.

FUNÇÕES

Autoria de Carolina Soares

O QUE É UMA FUNÇÃO

É um bloco de código que realiza certa funcionalidade, a fim de simplificar o código. Uma função executa tudo o que está dentro dela.

```
def funcao():
    # Bloco de código
```

CRIANDO UMA FUNÇÃO

```
def imprimirMsg():
    print("Criando uma função com Python!")
```

CHAMANDO UMA FUNÇÃO

```
def funcao():
    # Bloco de código

funcao()
```

Para que o bloco de código da função seja executado é preciso chamá-lo.

CRIANDO E CHAMANDO UMA FUNÇÃO

```
def imprimirMsg():
    print("Uma função Python!")
imprimirMsg()
```

Saída do código:

```
Uma função Python!
```

PALAVRA RESERVADA NA FUNÇÃO

```
def funcao():
    pass # Nada vai acontecer
funcao()
```

PARÂMETROS NA FUNÇÃO

```
def funcao(valor1, valor2):
    # Bloco de código

funcao(2, 3)
```

As funções podem ter parâmetros de diversos tipos: float, string, boolean, list, map...

PASSANDO PARÂMETROS NA FUNÇÃO

```python
# Recebendo dados
def somarNumeros(num1, num2):
    soma = num1 + num2
    print(soma)
somarNumeros(5, 2)
```

Saída do código:

```
7
```

PASSANDO PARÂMETROS NA FUNÇÃO

```
pessoas = ["Maria", "Pedro", "Eduarda", "José"]

def addValorNaLista(lista, valor):
    lista.append(valor)
def removerValorNaLista():
    lista.remove(valor)
def contarTamanhoDaLista(lista):
    print(len(lista))
```

```
addValorNaLista(pessoas, "Letícia")
removerValorNaLista(pessoas, "Pedro")
contarTamanhoDaLista(pessoas)
verificarSeExisteNaLista(pessoas,
"Ricardo")
```

PARÂMETROS PADRÕES NA FUNÇÃO

```
def somarNumeros(num1=0, num2=0):
    soma = num1 + num2
    print(soma)

somarNumeros()
```

Como não passei valores para função, ela considera seus valores padrão.

Saída do código:

```
0
```

PARÂMETROS ARGS E KWARGS

```python
# Número variável de parâmetros
def mostrarNumeros(*args):
    print(args)
    print(type(args))
mostrarNumeros(10, 20, 30, 40)
```

Saída:

```
(10, 20, 30, 40)
<class 'tuple'>
```

```python
# Número variável de parâmetros nomeados
def guardarPessoas(**kwargs):
    print(kwargs)
    print(type(kwargs))
guardarPessoas(nome='João', idade=35)
```

Saída:

```
{'nome': 'João', 'idade': 35}
<class 'dict'>
```

RETORNANDO VALORES NA FUNÇÃO

```
def funcao(valor1, valor2):
    return valor1 + valor2
somaValores = funcao(5, 2)
```

RETORNANDO UM VALOR NA FUNÇÃO

```
def somarNumeros(num1, num2):
    return num1 + num2
soma = somarNumeros(5, 2)
print(soma)
```

Saída do código:

```
7
```

RETORNANDO UM VALOR NA FUNÇÃO

```
def contarTamanhoDaLista(lista):
    return len(lista)
pessoas = ["Maria", "Pedro", "José", "Eduarda"]
tamanhoLista = contarTamanhoDaLista(pessoas)
print(f"A lista tem tamanho: {tamanhoLista}")
```

Saída do código:

```
4
```

RETORNANDO MÚLTIPLOS VALORES

```python
def somarNumerosEFazerMedia(num1, num2):
    soma = num1 + num2
    media = soma / 2
    return soma, media
print(somarNumerosEFazerMedia(10, 2))
```

Saída do código:

```
(12, 6.0)
```

CRIANDO BOAS FUNÇÕES

MELHORANDO FUNÇÕES

Função 1:

```
def somarNumeros():
    print(5 + 2)
```

Função 2:

```
def somarNumeros(num1, num2):
    print(num1 + num2)
somarNumeros(5, 2)
```

Função 3:

```
def somarNumeros(num1, num2):
    return num1 + num2
print(somarNumeros(5, 2))
```

MELHORANDO FUNÇÕES

Exemplo 1:

```
def somarNumerosEFazerMedia(num1, num2):
    soma = num1 + num2
    media = soma / 2
    return soma, media
print(somarNumerosEFazerMedia(10, 2))
```

Exemplo 2:

```
def somarNumeros(num1, num2):
    return num1 + num2
def fazerMedia(soma_valores):
    return soma_valores / 2
soma = somarNumeros(10, 2)
media = fazerMedia(soma)
print(soma, media)
```

MAIS FUNÇÕES

FUNÇÕES LAMBDA

Uma função lambda também chamada de função anônima, é uma função simples que não precisa ser nomeada e são úteis para resolver problemas simples, onde não precisa ser reutilizada no código.

```
f = lambda {argumentos}: {expressão}
```

FUNÇÕES LAMBDA

```
doblo = lambda x: x * 2
print(doblo(3))
```

Saída do código:

```
6
```

FUNÇÕES LAMBDA

Função tradicional:

```
def dobrarNumero(numero):
    return numero * 2
dobrarNumero(3)
```

Função anônima (lambda):

```
doblo = lambda x: x * 2
doblo(3)
```

FUNÇÕES LAMBDA

```
soma = lambda x, y: x + y
print(soma(2, 3))
```

Saída do código:

```
5
```

FUNÇÕES EMBUTIDAS

Funções comuns:

- ❑ `print()`
- ❑ `len()`
- ❑ `type()`
- ❑ `range()`
- ❑ `input()`

Funções para tipo:

- ❑ `int()`
- ❑ `float()`
- ❑ `str()`
- ❑ `list()`
- ❑ `tuple()`
- ❑ `dict()`

Funções de iteração:

- ❑ `map()`
- ❑ `max()`
- ❑ `min()`
- ❑ `sum()`

Funções matemáticas:

- ❑ `abs()`
- ❑ `pow()`

FUNÇÕES EMBUTIDAS

Funções embutidas

A
abs()
aiter()
all()
any()
anext()
ascii()

B
bin()
bool()
breakpoint()
bytearray()
bytes()

C
callable()
chr()
classmethod()
compile()
complex()

D
delattr()
dict()
dir()
divmod()

E
enumerate()
eval()
exec()

F
filter()
float()
format()
frozenset()

G
getattr()
globals()

H
hasattr()
hash()
help()
hex()

I
id()
input()
int()
isinstance()
issubclass()
iter()

L
len()
list()
locals()

M
map()
max()
memoryview()
min()

N
next()

O
object()
oct()
open()
ord()

P
pow()
print()
property()

R
range()
repr()
reversed()
round()

S
set()
setattr()
slice()
sorted()
staticmethod()
str()
sum()
super()

T
tuple()
type()

V
vars()

Z
zip()

_
__import__()

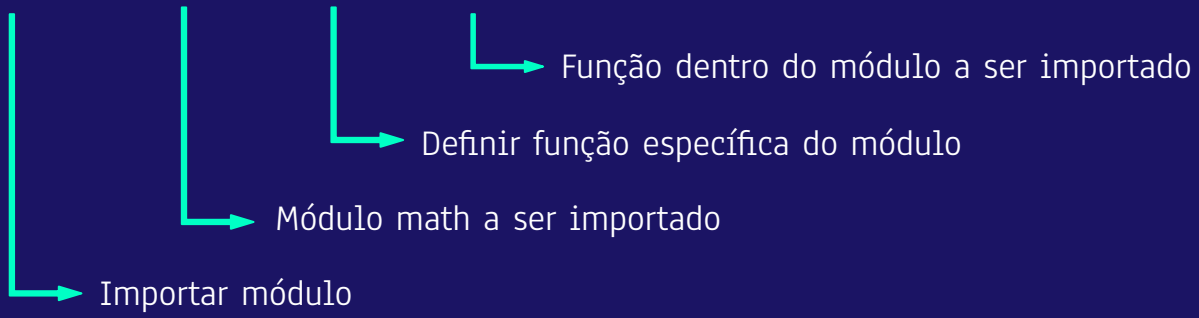
https://docs.python.org/pt-br/3/library/functions.html

MÓDULOS E FUNÇÕES

MÓDULOS

Um módulo é um arquivo contendo definições e instruções Python, onde pode conter classes, funções, variáveis ou qualquer outra coisa que você possa escrever em um script Python normal.

```
import math from sqrt
```



IMPORTANDO MÓDULOS

`import os` → Funções de sistemas operacional

`import math` → Funções matemáticas básicas

`import random` → Gerar números aleatórios

`import datetime` → Datas e horas

`import time` → Medições de tempo e espera

`import csv` → Arquivos CSV

`import re` → Expressões regulares (Regex)

`import json` → Dados em formato JSON

`import requests` → Requisições HTTP

IMPORTANDO FUNÇÕES DO MÓDULO MATH

```
import math
raiz = math.sqrt(25) # Saída: 5.0
sen = math.sin(45) # Saída: 0.8509035245341184
cos = math.cos(35) # Saída: -0.9036922050915067
pi = math.pi # 3.141592653589793
```

IMPORTANDO FUNÇÕES ESPECÍFICAS

```
from math import sqrt
raiz = sqrt(25) # Saída: 5.0
```

DEFININDO APELIDO DO MÓDULO

```
import math as m

raiz = m.sqrt(25) # Saída: 5.0
```

IMPORTANDO FUNÇÕES DO MÓDULO RANDOM

```
import random

# Gerar um número aleatório entre 1 e 100
random_number = random.randint(1, 100)

# Gerar um número aleatório entre 0 e 1
random_float = random.random()

# Gerar uma sequência de 5 números aleatórios entre 1 e 100
random_sequence = random.sample(range(1, 101), 5)
```

FUNÇÕES DO MÓDULO RANDOM PARA LISTAS

```
import random

frutas = ['maçã', 'banana', 'cereja', 'laranja']

# Escolher um item aleatório da lista

print(random.choice(frutas)) # Saída: banana

# Embaralhar a lista

random.shuffle(frutas)

print(frutas) # Saída: ['maçã', 'banana', 'cereja', 'laranja']
```

BIBLIOTECAS

Uma biblioteca é um conjunto de módulos com funcionalidades específicas para serem reutilizadas no código. Algumas bibliotecas são padrões e já vem na instaladas como o `math`, `random`, `json` e outras.

Bibliotecas externas (requerem instalação):

- ❏ NumPy
- ❏ Pandas
- ❏ Matplotlib
- ❏ Unittest
- ❏ Tkinter
- ❏ Django

EXERCÍCIOS

1. Crie uma função que receba três números inteiros e retorne o dobro da soma dos três números.
2. Faça uma função que retorne se o número é par ou ímpar.
3. Crie uma lista de produtos qualquer de uma loja de roupas e desenvolva 4 funções que irão operar a lista, são elas: adicionar, remover, editar e exibir um produto. Pense em como irá criar cada função e faça verificações para saber por exemplo se um produto primeiro existe na lista para poder ser removido. Ao final, mostre a lista após as alterações realizadas nas funções.
4. Crie 4 funções lambdas que irão realizar operações matemáticas simples de soma, subtração, multiplicação e divisão entre dois números fornecidos pelo usuário.
5. Faça uma função que sorteie um número de 1 a 6 de um dado 10 vezes e a cada chamada na função, mostre o valor sorteado. Também, contabilize o número de vezes que cada número foi sorteado e exiba as quantidades ao final do programa.
6. Crie três funções para calcular o seno, cosseno e tangente de um número, converta os resultados para graus, utilizando a função degrees do math e retorne os resultados arredondados em duas casas decimais. Ao final utilize as funções max e min para saber qual é o valor máximo e mínimo dos três resultados. Exiba todos os resultados das funções.

EXERCÍCIOS

7. A sequência de Fibonacci é uma sequência infinita de números que começa com 0 e 1 e cada número subsequente é a soma dos dois números anteriores. Um exemplo de sequência: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34... Note que o resultado da soma dos primeiros números da sequência 0 + 1 é igual a 1 que significa o próximo número da sequência, a mesma lógica se aplica para os próximos números como 1 + 1 é igual a 2. Crie uma função que tenha como parâmetro o número máximo da sequência de Fibonacci e retorne uma lista com todos os números que foram somados na essa sequência até o número máximo.

STRINGS

Autoria de Carolina Soares

STRINGS

Uma string é um tipo de dado que armazena uma sequência de caracteres.

```
texto = 'Olá mundo'
print(texto)
print(type(texto))
```

Saída do código:

```
Olá mundo
<class 'str'>
```

STRINGS DE DOCUMENTAÇÃO

```
texto = """

Está é uma String em Python com multiplas linhas. O Python me permite guardar uma String com várias linhas de caracteres. As Strings são textos que podem conter letras e números. Além disso, as Strings podem está dentro de aspas simples ou duplas."""
```

ÍNDICE DA STRING

```
nome = "Maria"
print(nome[0])
print(nome[1])
print(nome[2])
print(nome[3])
print(nome[4])
```

Saída do código:

```
M
a
r
i
a
```

ÍNDICE DA STRING

```
nome = "Maria"
print(nome[-1])
print(nome[-2])
print(nome[-3])
print(nome[-4])
print(nome[-5])
```

Saída do código:

```
a
i
r
a
M
```

INTERVALO DA STRING

índice	0	1	2	3	4	5	6	7	8	9
nome	J	o	ã	o		S	i	l	v	a

```
nome[ início : fim : pulo]
```

INTERVALO DA STRING

índice	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
nome	J	o	ã	o		S	i	l	v	a

```
nome[ início : fim : pulo]
```

INTERVALO DA STRING

```
nome = "João Silva"
print(nome[5:])
print(nome[-3:])
print(nome[3:8])
print(nome[-10:4])
print(nome[:7:2])
```

Saída do código:

```
Silva
lva
o Sil
João
Jã i
```

CONCATENANDO STRINGS

```
texto1 = 'Aprendendo'
texto2 = 'Python'
print(texto1 + ' ' + texto2)
```

Saída do código:

```
Aprendendo Python
```

MULTIPLICANDO A STRING

```
linha = '-'
print(linha * 20)
print('Bloco de código')
print(linha * 20)
```

Saída do código:

```
--------------------
Bloco de código
--------------------
```

OBTENDO O TAMANHO DA STRING

```
texto = "abc"
print(len(texto))
```

Saída do código:

```
3
```

STRINGS EM ESTRUTURAS DE REPETIÇÃO

```python
txt = 'Python'
for item in txt:
    print(item)
```

```python
txt = 'Python'
indice = 0
while indice < len(txt):
    print(txt[indice])
    indice += 1
```

Saída do código:

```
P
y
t
h
o
n
```

VERIFICAR SE STRING EXISTE EM OUTRA

```
texto = "Tenha um ótimo dia"
if 'dia' in texto:
    print("A sequência de caractere existe na string!")
```

Saída: A sequência de caractere existe na string!

VERIFICAR SE STRING EXISTE EM OUTRA

```
texto = "Tenha um ótimo dia"
if 'ótimo' not in texto:
    print("A palavra 'ótimo' não existe no texto")
else:
    print("Sim, 'ótimo' existe no texto!")
```

Saída: Sim, 'ótimo' existe no texto!

FUNÇÕES DA STRING

```
palavra = "Céu Limpo"

palavra.upper() # CÉU LIMPO

palavra.lower() # céu limpo

palavra.replace("Limpo", "Nublado") # Céu Nublado

palavra.split(" ") # ['Céu', 'Limpo']

palavra.startswith("Céu") # True

palavra.endswith(".") # False
```

CARACTERES DE ESCAPE NA STRING

```
print("Programação em \"Python\"")
```

Saída: Programação em "Python"

```
print("Programação \n em Python")
```

Saída: Programação

em Python

```
print("Programação \t em Python")
```

Saída: Programação em Python

```
print("Programação e\bm\b Python")
```

Saída: Programação Python

CARACTERES DE ESCAPE

Sequência de escape	Significado
`\<newline>`	A barra invertida e a nova linha foram ignoradas
`\\`	Contrabarra (`\`)
`\'`	Aspas simples (`'`)
`\"`	Aspas duplas (`"`)
`\a`	ASCII Bell (BEL) - um sinal audível é emitido
`\b`	ASCII Backspace (BS) - apaga caractere à esquerda
`\f`	ASCII Formfeed (FF) - quebra de página
`\n`	ASCII Linefeed (LF) - quebra de linha
`\r`	ASCII Carriage Return (CR) - retorno de carro
`\t`	ASCII Horizontal Tab (TAB) - tabulação horizontal
`\v`	ASCII Vertical Tab (VT) - tabulação vertical
`\`*ooo*	Caractere com valor octal *ooo*
`\x`*hh*	Caractere com valor hexadecimal *hh*

https://docs.python.org/pt-br/3/reference/lexical_analysis.html#strings

EXERCÍCIOS

1. Utilizando estrutura de repetição imprima cada letra da palavra Python de trás para frente.

2. Um palíndromo é uma palavra ou frase que quando lida de trás para frente permanece igual. Com as palavras a seguir adicione em uma lista e percorra cada elemento, mostrando se a palavra é um palíndromo ou não: "ana", "pelo", "ovo", "reviver", "a grama é amarga", "a mala nada na lama", "telhado", "abacate", "radar", "osso", "viver para viver", "reler", "ame o poema", "sol" , "rever", "sala". Também exiba a quantidade total de palavras que são palíndromos.

3. Considere a seguinte string com nomes de frutas: "banana uva maçã melão abacaxi". Coloque todas as palavras da string em maiúsculo, depois separe as frutas da string adicionando em uma lista utilizando o split. Verifique se uva está na lista e troque pela fruta morango. Por último imprima a lista.

4. Receba uma entrada que pede o nome completo do usuário, verifique se o nome completo do usuário possui "de" se sim utilize o replace para remover a palavra da string, conte o tamanho total do nome completo e por último pegue o primeiro e último nome do usuário para concatenar em outra string. Exiba o nome completo, tamanho e a string com o primeiro e último nome do usuário.

5. Receba uma data do usuário no formato dd/mm/aaaa e imprima a data com o mês escrito por extenso.