**REGRAS**:

* Todos os exercícios devem funcionar com qualquer valor de entrada. Faça testes.
* Todas as variáveis, exceção das constantes e quando o valor é informado, devem ter seu valor solicitado ao usuário (input).

**EXERCÍCIOS**

1. Peça 10 números e conte quantos são múltiplos de 3. Use for.

i = 0  
lista\_jafoi = []  
lista\_resultado = []  
  
for i in range(10):  
 numero = int(input("Digite um número: "))  
 if numero % 3 == 0 and numero not in lista\_jafoi:  
 lista\_resultado.append(numero)  
 lista\_jafoi.append(numero)  
  
quantidade = len(lista\_resultado)  
  
print(f"A quantidade de números digitados divisíveis por 3 é: {quantidade}")

1. Crie um programa que simule o uso de senha com tentativas infinitas até digitar a senha correta (use while True).

senha = "senha"  
  
while True:  
 senha\_tentativa = input("Digite sua tentativa da senha: ")  
 if senha\_tentativa == senha:  
 print("Você acertou a senha!")  
 break  
 else:  
 print("Senha incorreta! Tente novamente!")

1. Monte um sistema que repita um menu até o usuário escolher sair. Use while e break.

while True:  
 print("Bem vindo ao meu menu: \n"  
 "Menu \n"  
 "Menu também \n"  
 "Adivinha? Menu TAMBÉM \n"  
 "Sair")  
 opcao = input("Digite a opção que você quer: ")  
 if opcao.lower() == "sair":  
 print("Vcoê saiu do menu!")  
 break  
 else:  
 print("Você continua no menu!")

1. Crie um programa que peça dois números inteiros e exiba todos os números entre eles que são primos. Use for.

numero1 = int(input("Digite o número inicial: (ele dever ser positivo): "))  
numero2 = int(input("Digite o número final: (ele dever ser positivo): "))  
primo = True  
  
if numero1 < numero2:  
 for n in range(numero1+1, numero2):  
 i = 2 #divisor  
 while i < n:  
 if n % i == 0 or n == 1:  
 primo = False  
 break  
 else:  
 primo = True  
 i+=1  
  
 if primo:  
 print(n)  
  
else:  
 print("Números inválidos")

1. O usuário tem 3 tentativas para acertar a senha. Se errar todas, o acesso é bloqueado. Use while.

senha = "senha"  
tentativa = 0  
  
while tentativa != 3:  
 tentativa += 1  
 senha\_tentativa = input("Digite sua tentativa da senha: ")  
 if senha\_tentativa == senha:  
 print("Você acertou a senha!")  
 break  
 if tentativa == 3:  
 print("Você esgotou suas tentativas!")  
 else:  
 print("Senha incorreta! Tente novamente!")

1. Peça 10 números e separe em duas listas: pares e ímpares. Mostre as duas no final.

i = 0  
lista\_jafoi = []  
lista\_resultado\_impar = []  
lista\_resultado\_par = []  
  
for i in range(10):  
 numero = int(input("Digite um número: "))  
 if numero % 2 == 0 and numero not in lista\_jafoi:  
 lista\_resultado\_par.append(numero)  
 elif numero % 2 != 0 and numero not in lista\_jafoi:  
 lista\_resultado\_impar.append(numero)  
 lista\_jafoi.append(numero)  
  
print(f"Os número ímpares são: {lista\_resultado\_impar}\n"  
 f"Os números pares são: {lista\_resultado\_par}")

1. Peça uma frase e conte quantas vogais há nela. Mostre o total de cada uma (a, e, i, o, u).

frase = input("Digite uma frase: ")  
frase\_formatada = frase.lower()  
  
  
contador\_a = 0  
contador\_e = 0  
contador\_i = 0  
contador\_o = 0  
contador\_u = 0  
  
tamanho = len(frase)  
  
for i in range (0,tamanho):  
 if frase\_formatada[i] == "a":  
 contador\_a += 1  
 if frase\_formatada[i] == "e":  
 contador\_e += 1  
 if frase\_formatada[i] == "i":  
 contador\_i += 1  
 if frase\_formatada[i] == "o":  
 contador\_o += 1  
 if frase\_formatada[i] == "u":  
 contador\_u += 1  
  
print(f"O total de 'a' na frase é {contador\_a}\n"  
 f"O total de 'e' na frase é {contador\_e}\n"  
 f"O total de 'i' na frase é {contador\_i}\n"  
 f"O total de 'o' na frase é {contador\_o}\n"  
 f"O total de 'u' na frase é {contador\_u}\n")

1. Simule o lançamento de uma moeda até sair "cara" três vezes seguidas.  
   (Dica: usar random.choice(["cara", "coroa"]) e while).

import random  
  
i = 0  
tentativas = 0  
  
while i != 3:  
 opcoes = random.choice(["cara", "coroa"])  
 print(opcoes)  
 if opcoes == "coroa":  
 i = 0  
 elif opcoes == "cara":  
 i+=1  
 tentativas += 1  
  
print(f"A quantidade de vezes que a moeda foi lançada para sair 'cara' três vezes seguidas foi de: {tentativas}")

1. Crie um programa que leia uma sequência de números e determine quantos números são menores que a média.

lista\_tamanho = int(input("Digite o tamanho da lista: "))  
lista = []  
lista\_menores = []  
  
for i in range(lista\_tamanho):  
 i += 1  
 numero = float(input(f"Digite o número {i}: "))  
 lista.insert(1, numero)  
soma = sum(lista)  
media = soma / lista\_tamanho  
  
for n in range(lista\_tamanho):  
 if lista[n] < media:  
 lista\_menores.append(lista[n])  
  
print(f"Os números abaixo da média são: {lista\_menores}")

1. Crie um programa que leia uma sequência de números e determine o segundo maior número

lista\_tamanho = int(input("Digite o tamanho da lista: "))  
lista = []  
maior = ""  
  
for i in range(lista\_tamanho):  
 i += 1  
 numero = float(input(f"Digite o número {i}: "))  
 lista.insert(1, numero)  
  
lista.remove(max(lista))   
  
maior = max(lista)  
  
print(f"O maior é {maior}")

.

1. Mostre a tabela de teste de mesa preenchida referente ao programa abaixo:

a = 1

b = 2

c = 0

for i in range(1, 6):

if i % 2 == 0:

a += i

else:

b \*= i

c = a + b

print(a, b, c)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **i** | **a** | **b** | **c** | **print(a, b, c)** |
|  | 1 | 2 | 0 |  |
| 1 | 1 | 2 | 3 |  |
| 2 | 3 | 2 | 5 |  |
| 3 | 3 | 6 | 9 |  |
| 4 | 7 | 6 | 13 |  |
| 5 | 7 | 30 | 37 |  |
| 5 | 7 | 30 | 37 | 7 30 37 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. Mostre a tabela de teste de mesa preenchida referente ao programa abaixo:

x = 3

y = 2

z = 0

while x < 8:

for j in range(2):

z += y

y += 1

x += 1

print(x, y, z)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **y** | **z** | **j** | **print(x, y, z)** |
| 3 | 2 | 0 |  |  |
| 3 | 3 | 2 | 1 |  |
| 3 | 4 | 5 | 2 |  |
| 4 | 5 | 9 | 1 |  |
| 4 | 6 | 14 | 2 |  |
| 5 | 7 | 20 | 1 |  |
| 5 | 8 | 27 | 2 |  |
| 6 | 9 | 35 | 1 |  |
| 6 | 10 | 44 | 2 |  |
| 7 | 11 | 54 | 1 |  |
| 7 | 12 | 65 | 2 |  |
| 8 | 12 | 65 |  | 8 12 65 |
|  |  |  |  |  |

**DESAFIO - obrigatório**

1. Simulação de Populações de Coelhos

Crie um programa que simule o crescimento de uma população de coelhos ao longo de várias gerações. Os coelhos se reproduzem a uma taxa fixa a cada geração, e uma porcentagem deles morre a cada geração. O programa deve solicitar ao usuário a taxa de reprodução, a taxa de mortalidade e o número inicial de coelhos. Use um loop for ou while para simular várias gerações e exiba a população de coelhos após um número de gerações especificado pelo usuário.

numero\_inicial = int(input("Digite a quantidade inicial de coelhos: "))  
geracoes = int(input("Digite a quantidade de gerações: "))  
taxa\_reproducao = float(input("Digite a taxa de reprodução em %: "))  
taxa\_mortalidade = float(input("Digite a taxa de mortalidade em %: "))  
  
populacao = numero\_inicial  
  
for i in range(geracoes):  
 populacao = populacao + (populacao \* (taxa\_reproducao / 100))  
 populacao = populacao - (populacao \* (taxa\_mortalidade / 100))  
  
print(f"População final é de {int(populacao)} coelhos")

1. Jogo da Forca. Crie um jogo da forca, onde:

* **Palavra oculta**: A palavra é escolhida aleatoriamente de uma lista de palavras pré-definidas. A palavra deve ser exibida com espaços (\_) representando cada letra. O jogador deve tentar adivinhar as letras da palavra.
* **Feedback dinâmico**:
  + O jogo deve mostrar a palavra com as letras corretas já adivinhadas a cada tentativa.
  + O jogo também deve mostrar as **letras erradas** que o jogador já tentou, para evitar que ele repita a mesma letra.
  + Caso o jogador tente uma letra que já tenha sido usada (correta ou incorreta), o jogo deve **informar que ele já tentou essa letra**, pedindo que ele tente outra.
* **Número de tentativas**: O jogador tem um total de **6 tentativas** para errar antes de perder o jogo. A cada erro, o número de tentativas diminui.
* **Mensagens de vitória ou derrota**:
  + O jogo deve informar ao jogador quando ele **ganhar**, revelando a palavra completa.
  + Caso o jogador **perca**, o jogo deve revelar a palavra e informar que ele perdeu

import random

#Lista de palavras

lista\_palavras = ["Abacaxi","Banana","Uva","Jabuticaba","Maracujá","Morango","Limão","Laranja"]

#Pegar palavra aleatória da lista e transforma-la em lista

palavra\_aleatoria = random.choice(lista\_palavras).lower()

lista\_palavra\_aleatoria = list(palavra\_aleatoria)

print(lista\_palavra\_aleatoria)

#Encriptografar a palavra como \_\_\_

lista\_palavra\_crip = ['\_' for \_ in palavra\_aleatoria]

#Definir a lista com as letras de acertos e erros

lista\_chutes = []

lista\_erros = []

tentativas = 0

while True:

chute = input("Digite seu chute: ").lower()

if chute in lista\_chutes:

print(f"Você já seleciou essa letra antes!\n"

f"Esses são os seus chutes até agora: {lista\_chutes}")

else:

if chute in palavra\_aleatoria:

lista\_chutes.append(chute)

#percorrer a palavra e descobrir se tem algo de acordo com chute

i = 0

for letra in lista\_palavra\_aleatoria:

if letra == chute: #se a letra naquele i for == chute

lista\_palavra\_crip[i] = chute #revelar o chute na posição do i

i += 1

print(f"Você acertou")

else:

lista\_chutes.append(chute)

lista\_erros.append(chute)

tentativas += 1

palavra\_atual = ''.join(lista\_palavra\_crip) #lista em string

print(f"Essa é a palavra atualmente: {palavra\_atual}.\n"

f"Essas são as letras erradas até agora: {lista\_erros}")

if '\_' not in lista\_palavra\_crip: #se não tem mais \_ na lista da palavra encriptada

print(f"Você ganhou!\n"

f"A palavra é {palavra\_atual}")

break

if tentativas >= 6:

print(f"Você perdeu!\n"

f"A palavra correta era: {palavra\_aleatoria}")

break