**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.
2. Crie um programa que simule o uso de senha com tentativas infinitas até digitar a senha correta (use while True).
3. Monte um sistema que repita um menu até o usuário escolher sair. Use while e break.
4. Crie um programa que peça dois números inteiros e exiba todos os números entre eles que são primos. Use for.
5. O usuário tem 3 tentativas para acertar a senha. Se errar todas, o acesso é bloqueado. Use while.
6. Peça 10 números e separe em duas listas: pares e ímpares. Mostre as duas no final.
7. Peça uma frase e conte quantas vogais há nela. Mostre o total de cada uma (a, e, i, o, u).
8. Simule o lançamento de uma moeda até sair "cara" três vezes seguidas.  
   (Dica: usar random.choice(["cara", "coroa"]) e while).
9. Crie um programa que leia uma sequência de números e determine quantos números são menores que a média.
10. Crie um programa que leia uma sequência de números e determine o segundo maior número.
11. 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 | 4 |  |
| 3 | 3 | 6 | 8 |  |
| 4 | 7 | 6 | 14 |  |
| 5 | 7 | 30 | 37 |  |
| 5 | 7 | 30 | 37 | 8, 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 | 0 |  |
| 3 | 4 | 5 | 1 |  |
| 4 | 4 | 5 |  |  |
| 4 | 5 | 9 | 0 |  |
| 4 | 6 | 14 | 1 |  |
| 5 | 6 | 14 |  |  |
| 5 | 7 | 20 | 0 |  |
| 5 | 8 | 27 | 1 |  |
| 6 | 8 | 27 |  |  |
| 6 | 9 | 35 | 0 |  |
| 6 | 10 | 44 | 1 |  |
| 7 | 10 | 44 |  |  |
| 7 | 11 | 54 | 0 |  |
| 7 | 12 | 65 | 1 |  |
| 8 |  |  |  |  |
| 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.

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