Introdução a programação com Python - AT

Nome: Coloque seu nome no arquivo main

exercicio01.py

```
# Exercício 01
"""
Calcule a soma de todos os números pares de 1 a N, onde N é um
número informado pelo usuário.

Exemplo de execução:
Digite um número: 10
A soma dos números pares de 1 a 10 é: 30
"""
# Solicita o numero indice de repetições no programa
n = int(input(" Digite um numero limite, para calcular a soma dos
pares: \n"))
i = 0
soma = 0
# Repetir o laço ate que chegue no valor informado em n
while i < n:
# Incrementa o indice
  i = i+1
# Verifica o resto para ver se eh par
  if i % 2 == 0:
# Incrementa a soma
    soma += i
print (f"Num total de {i} numeros, a soma dos numeros pares eh:
{soma}\n")
```

Saída

```
Digite um numero limite, para calcular a soma dos pares:
Num total de 5 numeros, a soma dos numeros pares eh: 6
```

exercicio02.py

```
# Exercício 02
"""
Crie uma lista com números que o usuário informar até que ele
digite 'q'.
Exiba a soma dos números informados, a média e o desvio-padrão.
Não utilize bibliotecas ou funções para o cálculo da soma, média e
desvio padrão.

Exemplo de execução:
Digite um número (q para sair): 5
Digite um número (q para sair): 10
Digite um número (q para sair): -3
Digite um número (q para sair): q
A soma dos números informados é: 12
A média dos números informados é: 4
O desvio-padrão dos números informados é: 6.5574

No moodle, é possível ver a fórmula para cálculo do desvio padrão.
Veja um exemplo em: https://pt.khanacademy.org/math/statistics-
probability/summarizing-quantitative-data/variance-standard-
deviation-sample/a/population-and-sample-standard-deviation-review


"""
from math import sqrt

# Pede o primeiro numero para o usuario
num = input("Digite o primeiro numero. Para sair, digite q: \n")
# Cria lista para fazer o somatorio e declara as variaveis
numericas dando valor 0
lista = []
soma = 0
somatorio = 0
n = 0
media = 0

# Repete enquanto num for diferente de q
while num.lower() != "q":
# Verifica se é digito positivo ou negativo
  if num.isdigit() or num[1:].isdigit():
# Incrementa a o contador
    n+=1
# Transforma em float
    num = float(num)
```

```
# Soma os valores
    soma += num
# Faz a media dos valores
    media = soma / n
# Adiciona em uma lista o numero
    lista.append(num)

    num = input("por gentileza, digite um novo numero ou q para
sair: \n")
  else:
    num = input("por gentileza, digite um numero ou q pra sair:
\n")

for i in lista:
  somatorio += (i - media)**2

desvio_padrao = round(sqrt(somatorio/(n -1)),4)

print(f"Soma dos numeros: {soma}\nMedia: {media}\nDesvio padrao:
{desvio_padrao}\n Fim!")
```

Saída

```
Digite o primeiro numero. Para sair, digite q:
por gentileza, digite um novo numero ou q para sair:
por gentileza, digite um novo numero ou q para sair:
por gentileza, digite um novo numero ou q para sair:
Soma dos numeros: 12.0
Media: 4.0
Desvio padrao: 6.5574
 Fim!
```

exercicio03.py

```
# Exercício 03
"""
Escreva um programa que recebe uma lista de números inteiros do
usuário.
O programa deve criar uma nova lista contendo apenas os números
pares da lista original e imprimi-la.

Exemplo de execução:
Entre com os números separados por espaço: 1 2 3 4 5 6
Números pares: [2, 4, 6]
"""
num = input("Digite uma lista de numeros separado por espaços:
\n").replace(" ","")
tam = len(num)
list = []
soma_par = 0
for i in range(tam):
  if int(num[i]) % 2 == 0:
    list.append(int(num[i]))

print(f'Os numeros pares são: {list}\n')
```

Saída

```
Digite uma lista de numeros separado por espaços:
Os numeros pares são: [2, 4]
```

exercicio04.py

```
# Exercício 04
"""
Escreva um código que peça um número para o usuário e responda se
ele é primo ou não.
Lembrando que um número primo é um número maior que 1 que só pode
ser dividido por 1 e por ele mesmo.
Seu código deve conter comentários.
"""
# Solicita numero
num = int(input("Digite um numero para saber se ele eh primo ou
nao:\n"))
primo = 0
# Faz a repeticao ate o numero indicado
for i in range(num):
#verifica se o numero eh divisivel, o indice maior que 0 e
diferente dele mesmo
  if num % (i+1) == 0 and i > 0 and num != i +1:
    primo += 1
# Mostra se nao teve repeticao, eh promo
if primo == 0:
    print("O numero eh primo!")
else:
    print("O numero nao eh primo!")
```

Saída

```
Digite um numero para saber se ele eh primo ou nao:
O numero eh primo!
```

exercicio05.py

```
# Exercício 05
"""
Escreva um programa que dado o valor de N da variável abaixo
calcula o valor da soma:
1 + 1/2 + 1/3 + 1/4 + ... + 1/N

Exemplo de execução:
Se N = 4,
o programa deverá calcular:
1 + 1/2 + 1/3 + 1/4 = 2.083333333333333

Saída esperada:
2.083333333333333
"""
n = 4
soma = 0

for i in range(n):
  soma += 1/(i+1)

print (round(soma,15))
```

Saída

```
2.083333333333333
```

exercicio06.py

```
# Exercício 06
"""
No TP-2, Alice e Bob inventaram o seguinte método para codificar
mensagens:
1) Dividem a mensagem ao meio
2) A primeira metade da mensagem vai ocupar as posições pares do
texto. Já a segunda metade vai ocupar as posições ímpares.
Considerando que a primeira posição é a 0.

Por exemplo, a mensagem "Olá Alice." seria transformada em:

        posicao |0 1 2 3 4 5 6 7 8 9
           pares|O # l # á #   # A #
        ímpares |# l # i # c # e # .
         ___________________________

mensagem_encriptada = Olliác eA.
(Veja no moddle uma figura com esse exemplo)

A sua tarefa agora é escrever o código para ENCRIPTAR a mensagem.
A mensagem a ser encriptada está definida na variável
mensagem_original. Seu código deve conter comentários.
"""

mensagem_original = "Olá Alice! Resolvi mudar nossa criptografia,
acredito que nossa última mensagem foi interceptada. Mas agora com
esse novo método ninguém vai conseguir ler nossas mensagens."

tam = len(mensagem_original)

men_p1 = mensagem_original[0:tam-1]

print(len(men_p1))
```

Saída

```
171
```

exercicio07.py

```
# Exercício 07
"""
Um investidor tem um capital de R$ 1000,00 para investir em renda
fixa.
Atualmente, ele tem duas opções de investimento:
- Poupança: rende 0.5% ao mês
- LCA: rende 0.7% ao mês

Ao final de 10 anos de investimento, quanto o investidor terá
ganho a mais se escolher o LCA ao invés da poupança?
"""
```

Saída

---

exercicio08.py

```
# Exercício 08
"""
Calcule o fatorial de um número informado pelo usuário.
O fatorial de um número n é o produto de todos os inteiros de 1
até n.

Exemplo de execução:
Digite um número: 5
O fatorial de 5 é 120
"""
```

Saída

---

exercicio09.py

```
# Exercício 09
"""
Adrian, Bruno e Goran queriam se juntar ao clube dos amantes de
pássaros. No entanto, eles não sabiam que todos os candidatos
devem passar por um exame de admissão. O exame consiste em
perguntas, cada uma com três possíveis respostas: A, B e C.

Infelizmente, eles não conseguiam distinguir um pássaro de uma
baleia, então estão tentando adivinhar as respostas corretas. Cada
um dos três garotos tem uma teoria sobre qual conjunto de
respostas funcionará melhor:

Adrian afirma que a melhor sequência é: A B C A B C A B C A B
C ...

Bruno está convencido de que esta é melhor: B A B C B A B C B A B
C B ...

Goran ri deles e usará esta sequência: C C A A B B C C A A B B C
C ...

O gabarito do exercício é dado na variável gabarito.

Escreva um código que diga quanto cada questão cada um dos garotos
acertou. Seu código deve conter comentários.
"""

gabarito =
"BBCBCBBCAACABBACCCBCBCABBACBCACBCCBACACCCCABBCCCCABACAABBCAACACBA
BBACACBBBCABBABCAABCCCBABAAAAABBCBBCABABAAABCCCCACBBBCAABCBCBABCBC
BABAACBCCCACAAABCCCCCABBAABACAACCABCBABACBBACCCCCAACBBBCBAACACCACA
AAC"
```

Saída

---

exercicio10.py

```
# Exercício 10
"""
Peça ao usuário para informar um número e exiba a tabuada de
multiplicação desse número de 1 a 10.

Exemplo de execução:
Informe um número: 5
A tabuada de 5 é:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
"""
```

Saída

---

exercicio11.py

```
# Exercício 11
"""
Dado as duas listas de números inteiros abaixo, crie uma nova
lista de mesmo tamanho onde cada elemento é a soma dos elementos
correspondentes das listas originais.
Imprima o valor das listas originais e da lista resultante.

Exemplo de execução:
lista0: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
lista1: [11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
lista_resultante: [12, 14, 16, 18, 20, 22, 24, 26, 28, 30]
"""

import random

lista0 = random.sample(range(1, 100), 10)
lista1 = random.sample(range(1, 100), 10)
```

Saída

---

exercicio12.py

```
# Exercício 12
"""
Quando eu como meus Skittles, sempre como os vermelhos por último. Eu os separo em grupos de cores e começo comendo os laranjas, depois os azuis, verdes, amarelos, rosas, violetas, marrons e, finalmente, os vermelhos. Os vermelhos são os melhores, então eu os como devagar, um de cada vez. As outras cores eu como rapidamente aos punhados (minha mão pode segurar no máximo 7 Skittles). Eu sempre termino todos os Skittles de uma cor antes de passar para a próxima, então às vezes o último punhado de uma cor não está completamente cheio. Mas espere, tem mais! Eu transformei meu hábito de comer Skittles em uma ciência. Eu sei que sempre levo exatamente 13 segundos para comer um punhado de Skittles que não são vermelhos e ajusto minha taxa de mastigação para que eu sempre leve 13 segundos, mesmo que minha mão não esteja completamente cheia. Quando como os Skittles vermelhos, gosto de levar meu tempo, então levo exatamente 16 segundos para comer cada um. Eu tenho uma grande caixa de Skittles. Depois de terminar de separar as cores, quanto tempo levará para comê-los?

 Abaixo está uma lista de cores de Skittles em minha caixa. Cada cor é representada por uma string.
"""

saco_de_skittles = [
    "vermelho",
    "marrom",
    "marrom",
    "violeta",
    "azul",
    "rosa",
    "azul",
    "azul",
    "rosa",
    "marrom",
    "amarelo",
    "marrom",
    "rosa",
    "violeta",
    "verde",
    "amarelo",
    "vermelho",
    "laranja",
    "laranja",
```

```
    "azul",
    "marrom",
    "rosa",
    "vermelho",
    "vermelho",
    "vermelho",
    "marrom",
    "laranja",
    "laranja",
    "verde",
    "vermelho",
    "laranja",
    "violeta",
    "azul",
    "rosa",
    "amarelo",
    "rosa",
    "marrom",
    "laranja",
    "verde",
    "vermelho",
    "azul",
    "amarelo",
    "verde",
    "laranja",
    "marrom",
    "laranja",
    "rosa",
    "violeta",
    "marrom",
    "vermelho",
]
```

Saída

exercicio13.py

```
# Exercício 13
"""
Escreva um programa que dado as variaveis lista0 e lista1, crie
uma nova lista na variável lista_intersecao contendo apenas os
elementos que estão presentes em ambas as listas.
"""

lista0 = [7, 15, "a", "r", "o", "c", 17, 7, "w", "b"]
lista1 = ["b", "o", 7, 3, 2, "x", "d", 6, "w", "n"]


lista_intersecao = None
```

Saída

---

exercicio14.py

```
# Exercício 14
"""
    Crie um jogo onde o computador escolhe um número aleatório
entre 1 e 100 e o usuário tenta adivinhá-lo.
    O programa deve informar se o palpite do usuário é maior ou
menor que o número escolhido.

    Exemplo de execução (supondo que o computador escolheu o
número 42):

    Computador escolhe um número secreto
    Usuário faz o palpite: 50
    Saída: "Seu palpite é maior que o número escolhido."

    Usuário faz o palpite: 30
    Saída: "Seu palpite é menor que o número escolhido."

    Usuário faz o palpite: 42
    Saída: "Parabéns! Você adivinhou o número!"
"""
```

Saída

---

exercicio15.py

```
# Exercício 15
"""
Faça um código para validar se um determinado cpf é válido.
O cpf deve ser informado no formato ddd.ddd.ddd-dd (onde d é um
dígito de 0 a 9).
Para validar o cpf, você deve seguir os seguintes passos:

A validação do CPF (Cadastro de Pessoas Físicas) é um processo que
verifica se um número de CPF é válido ou não. O CPF é composto por
11 dígitos e possui um algoritmo específico para validação.

O algoritmo de validação do CPF é baseado em cálculos matemáticos
que envolvem os dígitos do número. Aqui está uma descrição passo a
passo de como é feita a validação do CPF:

1. Primeiro, é importante remover os caracteres especiais (como
pontos e traços) do número do CPF, mantendo apenas os dígitos.
2. Em seguida, é necessário separar os 9 primeiros dígitos do CPF,
que são os dígitos base para o cálculo dos dígitos verificadores.
3. Para calcular o primeiro dígito verificador, multiplicamos cada
um dos 9 dígitos pela sequência decrescente de números de 10 a 2.
Somamos os resultados dessas multiplicações.
4. O resultado da soma é dividido por 11. O dígito verificador é o
resto da divisão. Se o resto for igual a 0 ou 1, o dígito
verificador é 0. Caso contrário, subtraímos o resto de 11 para
obter o dígito verificador.
5. Repetimos o mesmo processo para calcular o segundo dígito
verificador, mas agora utilizamos os 9 dígitos do CPF original
mais o primeiro dígito verificador calculado no passo anterior
multiplicando agora esses dígitos pela sequência decrescente de 11
a 2.
6. Após calcular o segundo dígito verificador, temos o CPF
completo com os 11 dígitos.
7. Para validar o CPF, comparamos os dígitos verificadores
calculados com os dois últimos dígitos do CPF original. Se forem
iguais, o CPF é considerado válido. Caso contrário, o CPF é
considerado inválido.

Veja um exemplo em: https://dicasdeprogramacao.com.br/algoritmo-
para-validar-cpf/
"""
```

Saída

exercicio16.py

```
# Exercício 16
"""
Parabéns você está chegou a última questão do curso de Introdução
ao Python!
Para comemorar escreva um código que imprime a mensagem "Parabéns!
Você concluiu o curso de Introdução ao Python!" 100 vezes na tela!
"""
```

Saída
