# Programação em Python


CEFET-MG

CEFET-MG

- **Instrução ‘match case’**
  - Equivalente ao ‘switch case’ do C

```
>>> var = valor
>>> match var:
>>>         case 1:
>>>                 instrução
>>>         case 2:
>>>                 instrução
>>>         case _:
>>>                 instrução
```

CEFET-MG

- **Instrução while**
  - Indicado para laço de repetição condicional
    - Caso contrário usar o for
  - Uso combinado:
    - break
    - continue

CEFET-MG

- **Instrução while**

```
>>> while True
>>>         age = imput('Qual a sua idade')
>>>         if age < 18:
>>>                 print('Você está mentindo')
>>>                 break
>>>         if age >18 and age < 40
>>>                 print('Você é muito jovem')
>>>                 continue
>>>         if age >= 40
>>>                 print('Você é muito jovem')
>>>                 continue
```

# Mãos à obra

- **Exercícios (Usar como desafio)**

26) Faça um programa gerenciar uma agenda de contatos. Para cada contato armazene o nome, a data de nascimento (dia, mês e ano), endereço e o telefone. O programa deve permitir:

- inserir contato
- remover contato
- pesquisar um contato pelo nome
- listar todos os contatos
- listar os contatos cujo nome inicia se com uma dada letra
- imprimir os aniversariantes do mês.

```
contatos = []  # Lista para armazenar os contatos

while True:
    print("\n=== MENU AGENDA ===")
    print("1. Inserir contato")
    print("2. Remover contato")
    print("3. Pesquisar contato por nome")
    print("4. Listar todos os contatos")
    print("5. Listar contatos por letra inicial")
    print("6. Imprimir aniversariantes do mês")
    print("0. Sair")

    opcao = input("Escolha uma opção: ")
    match opcao:
        case '0':
            print("Saindo da agenda...")
            break
```

CEFET-MG

```
case '1':
    dados = []
    nome = input('Digite o nome: ')
    dados.append(nome)
    dia_nascimento = input('Dia de nascimento: ')
    dados.append(dia_nascimento)
    mes_nascimento = input('Mês de nascimento: ')
    dados.append(mes_nascimento)
    ano_nascimento = input('Ano de nascimento: ')
    dados.append(ano_nascimento)
    endereco = input('Endereço: ')
    dados.append(endereco)
    tel = input("Telefone: ")
    dados.append(tel)
    contatos.append(dados)
    print("Contato adicionado com sucesso!")
```

```
case '2':
        rm_nome = input('Digite o nome para remover o contato: ')
        for contato in contatos:
            if contato[0].lower() == rm_nome.lower():
                contatos.remove(contato)
                print("Contato removido com sucesso!")
                break
        else :
            print("Contato não encontrado.")

case '3':
        pesq_nome = input('Digite o nome para pesquisar: ')
        encontrado = False
        for contato in contatos:
            if contato[0].lower() == pesq_nome.lower():
                print("Contato encontrado:", contato)
                encontrado = True
        if not encontrado:
            print("Contato não encontrado.")
```

```
case '4':
        if contatos:
            print("\n--- Lista de Contatos ---")
            for contato in contatos:
                print(contato)
        else:
            print("Nenhum contato cadastrado.")

case '5':
        letra = input('Digite a letra inicial: ').lower()
        encontrados = [c for c in contatos if c[0].lower().startswith(letra)]
        if encontrados:
            print("\n--- Contatos encontrados ---")
            for contato in encontrados:
                print(contato)
        else:
           print("Nenhum contato encontrado com essa letra.")
```

```
case '6':
        mes = input("Digite o mês (número): ")
        aniversariantes = [c for c in contatos if c[2] == mes]
        if aniversariantes:
            print("\n--- Aniversariantes do mês ---")
            for contato in aniversariantes:
                print(contato)
        else:
            print("Nenhum aniversariante neste mês.")

case _:
        print("Opção inválida.")
```

CEFET-MG

- **Exercícios**

Observe o código abaixo:

```
>>> a = [1, 2, 7, 4, 2, 6, 7]
>>> b = a
>>> b[0] = -1
```

a) Qual será a saída se mandarmos imprimir as listas a e b? Por que isto acontece?

b) Como criar uma cópia da lista a?

```
a = [1, 2, 7, 4, 2, 6, 7]
b = a
b[0] = -1
print(a)  # [-1, 2, 7, 4, 2, 6, 7]
print(b)  # [-1, 2, 7, 4, 2, 6, 7]
```

Para criar uma cópia:

```
c = a[:]
c = list(a)
c = a.copy()
```

- Quando fazemos b = a, **não criamos uma nova lista**.
- Apenas dizemos que b aponta para o **mesmo objeto na memória** que a.
- Logo, qualquer modificação feita em b também aparece em a.

- **Exercícios**

Observe o código abaixo:

```
>>> pessoas = list()
>>> dados = list()
>>> for c in range(3):
>>>     dado.append(str(input('Nome: ')))
>>>     dado.append(str(input('Idade: ')))
>>>     pessoas.append(dado[:])
>>>     dado.clear()
```

a) Por que utilizar [:] ? Para que serve o comando clear()?

b) Complete o código, para sejam recebidos dados de 5 pessoas, diga se a pessoa é maior ou menor de idade, e, no final, diga quantas pessoas são maiores e menores de idade?

Por que usar [:] ?

- [:] cria uma cópia da lista. Se usássemos apenas pessoas.append(dado), todas as posições de pessoas apontariam para a mesma lista dado. Assim, cada modificação sobrescreveria os valores anteriores. Usando dado[:], garantimos que uma cópia independente seja armazenada.

📌 Para que serve clear() ?

- clear() esvazia a lista, removendo todos os elementos. Isso é necessário porque, após copiar os dados de uma pessoa para pessoas, precisamos limpar dado para reutilizá-lo na próxima pessoa.

# Programação em Python

CEFET-MG

```
pessoas = list()
dado = list()
maiores = 0
menores = 0

for c in range(5):
    dado.append(str(input("Nome: ")))
    dado.append(int(input("Idade: ")))
    pessoas.append(dado[:])  # copia os dados para a lista principal
    dado.clear()  # limpa a lista para receber os dados da próxima pessoa

print("\n--- Resultado ---")
for p in pessoas:
    print(f"{p[0]} tem {p[1]} anos.", end=" ")
    if p[1] >= 18:
        print("É maior de idade.")
        maiores += 1
    else:
        print("É menor de idade.")
        menores += 1
print(f"\nTotal de maiores de idade: {maiores}")
print(f"Total de menores de idade: {menores}")
```

- **Exercícios**

Escreva um programa que leia n números reais e calcule o desvio padrão destes valores. O desvio padrão é dado pela seguinte equação:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Na qual, *n* é a quantidade de números, $x_i$ é o *i*-ésimo valor e $\bar{x}$ é a média dos valores.

# Programação em Python

CEFET-MG

```
import math

# Ler quantidade de números
n = int(input("Quantos números deseja informar? "))

valores = []

# Entrada dos números reais
for i in range(n):
    numero = float(input(f"Digite o {i+1}º número: "))
    valores.append(numero)

# Calcular média
media = sum(valores) / n

# Calcular soma dos quadrados das diferenças
soma = 0
for x in valores:
    soma += (x - media) ** 2

# Calcular desvio padrão amostral
desvio_padrao = math.sqrt(soma / (n - 1))

# Mostrar resultado
print(f"\nMédia: {media:.4f}")
print(f"Desvio padrão: {desvio_padrao:.4f}")
```

- Estrutura **set()**
  - Tipo nativo de dado
  - Não aceita elementos repetidos
    - numeros = set([1, 2, 2, 3, 3, 3])
    - print(numeros)  # {1, 2, 3}
  - **Não garante ordem** (os elementos não ficam necessariamente na ordem que foram inseridos).
    - frutas = {"maçã", "banana", "uva"}
    - print(frutas)  # a ordem pode variar

- Estrutura **set()**

  - Permite operações matemáticas de conjuntos:
    - União (|) → elementos que estão em um ou outro conjunto
    - Interseção (&) → elementos em comum
    - Diferença (-) → elementos que estão em um mas não no outro
    - Diferença simétrica (^) → elementos que estão em apenas um dos conjuntos

- Estrutura **set()**

```
A = {1, 2, 3, 4}
B = {3, 4, 5, 6}

print(A | B)      # União -> {1, 2, 3, 4, 5, 6}
print(A & B)      # Interseção -> {3, 4}
print(A - B)      # Diferença -> {1, 2}
print(A ^ B)      # Diferença simétrica -> {1, 2, 5, 6}
```

- **Exercícios:**

- Faça um programa que leia dois conjuntos de números inteiros distintos e imprima a interseção destes dois conjuntos (os números presentes em ambos os conjuntos).

```
# Ler o primeiro conjunto
n1 = int(input("Quantos números terá o primeiro conjunto? "))
conjunto1 = set()
for i in range(n1):
    num = int(input(f"Digite o {i+1}º número do primeiro conjunto: "))
    conjunto1.add(num)

# Ler o segundo conjunto
n2 = int(input("\nQuantos números terá o segundo conjunto? "))
conjunto2 = set()
for i in range(n2):
    num = int(input(f"Digite o {i+1}º número do segundo conjunto: "))
    conjunto2.add(num)

# Calcular interseção
intersecao = conjunto1 & conjunto2

# Mostrar resultado
print("\nInterseção dos conjuntos:", intersecao if intersecao else "Não há elementos em comum.")
```

- **Exercícios:**
  - Crie um programa que:
    - Permita ao usuário digitar números inteiros até que ele digite 0.
    - Armazene todos os números em um conjunto (set) para garantir que não haja repetição.
    - Quando o usuário digitar 0, o programa deve:
    - Mostrar todos os números únicos digitados.
    - Informar quantos números distintos foram digitados.
    - Informar o maior e o menor número digitado.

CEFET-MG

```
# Conjunto para armazenar os números únicos
numeros = set()

while True:
    n = int(input("Digite um número inteiro (0 para sair): "))
    if n == 0:
        break
    numeros.add(n)  # adiciona ao conjunto (não repete)

if numeros:
    print("\nNúmeros únicos digitados:", numeros)
    print("Quantidade de números distintos:", len(numeros))
    print("Maior número:", max(numeros))
    print("Menor número:", min(numeros))
else:
    print("\nNenhum número foi digitado.")
```

CEFET-MG

- **Exercícios:**

  Faça um programa que leia duas sequências de números inteiros ordenados e imprima uma sequência com os números ordenados das duas sequências anteriores.

  Dica: use a função map(tipo, *dados*)

  - O map(int, ...) é uma forma rápida de converter todos os elementos de uma lista de strings para inteiros

```
# Ler a primeira sequência
seq1 = list(map(int, input("Digite os números da 1° sequência, separados por espaço: ").split()))

# Ler a segunda sequência
seq2 = list(map(int, input("Digite os números da 2° sequência, separados por espaço: ").split()))

# Juntar as duas sequências
seq_completa = seq1 + seq2

# Ordenar
seq_completa.sort()

# Mostrar resultado
print("Sequência ordenada das duas sequências:", seq_completa)
```

- **Exercícios:**

  Faça um algoritmo que, dada a idade de um nadador, classifique-o em uma das seguintes categorias e retorne a categoria do nadador.

  - Infantil A: 5 a 7 anos;
  - Infantil B: 8 a 10 anos;
  - Juvenil A: 11 a 13 anos;
  - Juvenil B: 14 a 17 anos;
  - Sênior: maiores de 18 anos.

CEFET-MG

```
# Ler idade do nadador
idade = int(input("Digite a idade do nadador: "))

# Classificação usando match
match idade:
    case x if 5 <= x <= 7:
        categoria = "Infantil A"
    case x if 8 <= x <= 10:
        categoria = "Infantil B"
    case x if 11 <= x <= 13:
        categoria = "Juvenil A"
    case x if 14 <= x <= 17:
        categoria = "Juvenil B"
    case x if x >= 18:
        categoria = "Sênior"
    case _:
        categoria = "Idade fora da faixa para competição"

# Mostrar resultado
print(f"O nadador de {idade} anos pertence à categoria: {categoria}")
```

- **Exercícios**

36) Faça uma função que permita que o usuário escolha que tipo de média deseja calcular a partir de três notas. A função deve receber as 3 notas e a opção de média a ser calculada de acordo com as fórmulas abaixo e deve retornar o valor da média:

- Aritmética (A): $\frac{N1+N2+N3}{3}$
- Ponderada (P): $\frac{(N1*3)+(N2*3)+(N3*4)}{(3+3+4)}$
- Harmônica (H): $\frac{3}{\frac{1}{n1}+\frac{1}{n2}+\frac{1}{n3}}$

# Programação em Python

CEFET-MG

```
# Ler três notas do usuário
n1 = float(input("Digite a primeira nota: "))
n2 = float(input("Digite a segunda nota: "))
n3 = float(input("Digite a terceira nota: "))

# Escolher o tipo de média
print("\nEscolha o tipo de média:")
print("A - Aritmética")
print("P - Ponderada")
print("H - Harmônica")
opcao = input("Digite a opção desejada (A/P/H): ").upper()
```

```
# Calcular a média usando match
match opcao:
    case "A":
        media = (n1 + n2 + n3) / 3
        tipo = "Aritmética"
    case "P":
        media = (n1*3 + n2*3 + n3*4) / (3+3+4)
        tipo = "Ponderada"
    case "H":
        media = 3 / (1/n1 + 1/n2 + 1/n3)
        tipo = "Harmônica"
    case _:
        media = None
        tipo = None
        print("Opção inválida!")

# Mostrar resultado
if media is not None:
    print(f"\nA média {tipo} das notas é: {media:.2f}")
```

CEFET-MG

- **Exercícios**

Faça um algoritmo que lê o número de um vendedor de uma empresa, seu salário fixo e o total de vendas por ele efetuadas. Cada vendedor recebe um salário fixo, mais um a comissão proporcional às vendas por ele efetuadas. A comissão é de 3% sobre o total de vendas até 10.000 e 5% sobre o que ultrapassa este valor. Escrever o número do vendedor, o total de suas vendas, seu salário fixo e seu salário total.

# Programação em Python

CEFET-MG

```
while True:
    # Ler dados do vendedor
    numero_vendedor = input("Digite o número do vendedor
(ou '0' para sair): ")
    if numero_vendedor == "0":
        print("\nEncerrando o cadastro de vendedores.")
        break

    salario_fixo = float(input("Digite o salário fixo do vendedor:
"))
    total_vendas = float(input("Digite o total de vendas
efetuadas: "))

    # Calcular comissão
    if total_vendas <= 10000:
        comissao = total_vendas * 0.03
    else:
        comissao = 10000 * 0.03 + (total_vendas - 10000) * 0.05

    # Calcular salário total
    salario_total = salario_fixo + comissão

    # Mostrar resultado
    print("\n=== Resumo do Vendedor ===")
    print(f"Número do vendedor:
{numero_vendedor}")
    print(f"Total de vendas: R$ {total_vendas:.2f}")
    print(f"Salário fixo: R$ {salario_fixo:.2f}")
    print(f"Salário total (com comissão): R$
{salario_total:.2f}\n")
```

CEFET-MG

- **Exercícios**

  - Calculadora lógica: Implemente uma calculadora que recebe duas variáveis booleanas e um operador lógico (and, or, xor, not) e devolve o resultado.

# Programação em Python

CEFET-MG

```
# Ler duas variáveis booleanas
a = input("Digite o valor de A (True/False): ").strip().capitalize() == "True"
b = input("Digite o valor de B (True/False): ").strip().capitalize() == "True"

# Ler operador lógico
print("\nEscolha o operador lógico: and, or, xor, not")
operador = input("Digite o operador: ").strip().lower()

# Calcular resultado usando match
match operador:
    case "and":
        resultado = a and b
    case "or":
        resultado = a or b
    case "xor":
        resultado = a != b  # XOR lógico
    case "not":
        resultado = not a   # NOT unário, só sobre A
    case _:
        resultado = None
        print("Operador inválido!")

# Mostrar resultado
if resultado is not None:
    print(f"\nResultado da operação {operador.upper()}: {resultado}")
```

- **Exercícios**

  - Crie um algoritmo que leia dois inteiros a e b e exiba: soma, diferença, produto, divisão inteira, resto e exponenciação

```
while True:
  # Ler dois números inteiros
  a = int(input("Digite o valor de A: "))
  b = int(input("Digite o valor de B: "))

   # Menu de operações
   print("\nEscolha a operação:")
   print("1 - Soma")
   print("2 - Diferença (A - B)")
   print("3 - Produto")
   print("4 - Divisão inteira (A // B)")
   print("5 - Resto da divisão (A % B)")
   print("6 - Exponenciação (A ** B)")
   print("0 - Sair")

   opcao = input("Digite o número da operação: ").strip()
```

```
if opcao == "0":
    print("Encerrando a calculadora.")
    break

# Realizar operação usando match
match opcao:
    case "1":
        resultado = a + b
        operacao = "Soma"
    case "2":
        resultado = a - b
        operacao = "Diferença"
    case "3":
        resultado = a * b
        operacao = "Produto"
```

```
case "4":
    if b != 0:
        resultado = a // b
    else:
        resultado = None
    operacao = "Divisão inteira"
 case "5":
    if b != 0:
        resultado = a % b
    else:
        resultado = None
    operacao = "Resto da divisão"
```

```
case "6":
        resultado = a ** b
        operacao = "Exponenciação"
    case _:
        resultado = None
        operacao = None
        print("Opção inválida!")

# Mostrar resultado
if resultado is not None:
    print(f"{operacao} de {a} e {b} = {resultado}")
else:
    if operacao is not None:
        print(f"Não é possível realizar a operação {operacao} com B = 0.")
```

- **Exercícios**

  - Crie um programa que apresenta um menu com opções (1-Somar, 2-Subtrair, 3-Multiplicar, 4-Dividir, 5-Sair) e use match para executar a operação correspondente até que o usuário decida encerrar.

# Programação em Python

CEFET-MG

```
while True:
    # Menu
    print("\n=== MENU CALCULADORA ===")
    print("1 - Somar")
    print("2 - Subtrair")
    print("3 - Multiplicar")
    print("4 - Dividir")
    print("5 - Sair")

    opcao = input("Escolha a opção: ").strip()

    if opcao == "5":
        print("Encerrando a calculadora.")
        break

    # Ler os números apenas se a opção for válida
    if opcao in ["1", "2", "3", "4"]:
        a = float(input("Digite o primeiro número: "))
        b = float(input("Digite o segundo número: "))
```

```
# Executar operação usando match
    match opcao:
        case "1":
            resultado = a + b
            operacao = "Soma"
        case "2":
            resultado = a - b
            operacao = "Subtração"
        case "3":
            resultado = a * b
            operacao = "Multiplicação"
        case "4":
            if b != 0:
                resultado = a / b
            else:
                resultado = None
            operacao = "Divisão"
        case _:
            resultado = None
            operacao = None
            print("Opção inválida!")
```

```
# Mostrar resultado
if resultado is not None:
    print(f"{operacao} de {a} e {b} = {resultado}")
elif operacao == "Divisão":
    print("Não é possível dividir por zero.")
```

- **Exercícios**

  - Receba uma nota numérica (0 a 10) e classifique em conceitos A, B, C, D, F usando match.
    - 0 a 5,9: F
    - 6 a 6,9: D
    - 7 a 7,9: C
    - 8 a 8,9: B
    - 9 a 10: A

CEFET-MG

```
# Ler nota
nota = float(input("Digite a nota (0 a 10): "))

# Classificação usando match com guard clauses
match nota:
    case x if 0 <= x <= 5.9:
        conceito = "F"
    case x if 6 <= x <= 6.9:
        conceito = "D"
    case x if 7 <= x <= 7.9:
        conceito = "C"
    case x if 8 <= x <= 8.9:
        conceito = "B"
    case x if 9 <= x <= 10:
        conceito = "A"
    case _:
        conceito = None
        print("Nota inválida!")

# Mostrar resultado
if conceito is not None:
    print(f"A nota {nota} corresponde ao conceito: {conceito}")
```

- **Exercícios**

Leia um valor e use match para identificar se é int, float, str ou bool, imprimindo uma mensagem personalizada.

# Programação em Python

CEFET-MG

```
# Ler valor do usuário
valor = input("Digite um valor: ")

# Tentar converter para tipos conhecidos
if valor.lower() == "true":
    valor_convertido = True
elif valor.lower() == "false":
    valor_convertido = False
else:
    try:
        if "." in valor:
            valor_convertido = float(valor)
        else:
            valor_convertido = int(valor)
    except ValueError:
        valor_convertido = valor  # permanece como string
```

```
# Identificar tipo usando match
match type(valor_convertido):
    case int:
        print(f"O valor {valor_convertido} é um
número inteiro (int).")
    case float:
        print(f"O valor {valor_convertido} é um
número real (float).")
    case str:
        print(f"O valor '{valor_convertido}' é uma
string (str).")
    case bool:
        print(f"O valor {valor_convertido} é booleano
(bool).")
    case _:
        print(f"O valor {valor_convertido} é de tipo
desconhecido.")
```

- **Exercícios**
- A **Sequência de Collatz**, é uma sequência definida por uma regra muito simples, aplicada a qualquer número inteiro positivo:
  - Se o número for **par**, o próximo número da sequência é a sua **metade** (n/2).
  - Se o número for **ímpar**, o próximo número da sequência é o **triplo mais um** (3n+1).
  - Ex.: A sequência para o número 6 é: 6, 3, 10, 5, 16, 8, 4, 2, 1.
- Dado um número inteiro positivo, gere a sequência de Collatz usando while até chegar em 1.

CEFET-MG

```
# Ler número inteiro positivo
n = int(input("Digite um número inteiro positivo: "))

# Verificar se é positivo
if n <= 0:
    print("O número deve ser positivo.")
else:
    print("\nSequência de Collatz:")
    while n != 1:
        print(n, end=", ")
        if n % 2 == 0:
            n = n // 2  # número par: divide por 2
        else:
            n = 3 * n + 1  # número ímpar: 3n + 1
    print(1)  # último número da sequência
```

- **Exercícios**

  - Leia um valor de saque e, usando while, calcule a quantidade mínima de notas de 100, 50, 20, 10, 5 e 2 necessárias.

# Programação em Python

CEFET-MG

```
# Ler valor do saque
valor = int(input("Digite o valor do saque: "))

# Verificar valor válido
if valor < 2:
    print("O valor mínimo de saque é 2.")
else:
    notas = [100, 50, 20, 10, 5, 2]  # notas disponíveis
    contagem = {}  # dicionário para armazenar quantidade de
cada nota

    restante = valor
    i = 0
    while restante > 0 and i < len(notas):
        nota = notas[i]
        quantidade = restante // nota  # número de notas dessa
cédula
        if quantidade > 0:
            contagem[nota] = quantidade
            restante -= quantidade * nota
        i += 1

# Mostrar resultado
    print(f"\nSaque de R$ {valor} será entregue em:")
    for nota, qtd in contagem.items():
        print(f"{qtd} nota(s) de R$ {nota}")

    if restante > 0:
        print(f"Restante não atendido: R$ {restante}")
```

CEFET-MG

- **Exercícios**

O programa sorteia um número entre 1 e 100. O usuário tenta adivinhar com while, e o programa dá dicas: "maior" ou "menor". Após 10 tentativas, termina.

# Programação em Python

CEFET-MG

```
import random

# Sorteia um número entre 1 e 100
numero_sorteado = random.randint(1, 100)

tentativas = 0
max_tentativas = 10
acertou = False

print("Tente adivinhar o número entre 1 e 100. Você tem 10 tentativas.")

while tentativas < max_tentativas:
    chute = int(input(f"Tentativa {tentativas + 1}: Digite seu palpite: "))
    tentativas += 1
    if chute == numero_sorteado:
        print(f"Parabéns! Você acertou o número {numero_sorteado} em {tentativas} tentativas.")
        acertou = True
        break
    elif chute < numero_sorteado:
        print("O número sorteado é MAIOR.")
    else:
        print("O número sorteado é MENOR.")

if not acertou:
    print(f"\nFim das tentativas! O número sorteado era {numero_sorteado}.")
```

- **Desafio**

- Imagine um tabuleiro quadriculado com *m*×*n* casas dispostas em *m* linhas e *n* colunas. Algumas casas estão livres e outras estão bloqueadas. As casas livres são marcadas com - e as bloqueadas com #. Há um robô na casa (1,1), que é livre. O robô só pode andar de uma casa livre para outra. Em cada passo, só pode andar para a casa que está ao norte, a leste, ao sul ou a oeste. Ajude o robô a encontrar a saída, que está na posição (m,n).

- **Sugestão**: Faça uma moldura de casas bloqueadas.