Algoritmos Computacionais

Conteúdo: Estruturas de decisão

Prof. Dsc. Giomar Sequeiros
giomar@eng.uerj.br

Documentação de código

Documentação

- Documentar código é importante para torná-lo mais legível e fácil de entender.
- Existem várias maneiras de documentar o código Python, a mais comum é o uso de **comentários** e **docstrings**.

  - **Comentários**: Os comentários são linhas de **texto** que começam com o símbolo **#** e são usados para **explicar** o que o **código** está fazendo. Os comentários devem ser **claros** e **concisos** e não devem ser usados para explicar o óbvio.

  - **Docstrings**: As docstrings são **strings** que aparecem no **início** de uma **função**, **método**, classe ou módulo e são usados para **descrever** o que a **função**, método, classe ou módulo faz. As **docstrings** são cercadas por **aspas triplas** e podem ser acessadas usando o atributo **__doc__.**

Exemplo documentação usando comentário

```
# Define uma função que calcula a area de um círculo
def calculate_area(raio):
    # A área é pi vezes o raio ao quadrado
    area = 3.14 * raio ** 2
    return area
```

Exemplo documentação usando Docstrings

```
def calcular_area(raio):
    """
    Calcula a área de um círculo.

    Argumentos:
        raio (float): O raio do círculo.

    Retorna:
        float: A área do círculo.
    """
    area = 3.14 * raio ** 2
    return area
```

Documentação recomendada na disciplina!

Python Type Hints

- É uma ferramenta introduzida no Python 3.5 para ajudar a indicar o tipo de dados esperado para argumentos de função e valores de retorno.
- O objetivo é **fornecer informações** adicionais para **facilitar** a **leitura** e manutenção do **código**, bem como melhorar a detecção de erros em tempo de execução.

```
def soma(a: float, b: float) -> float:
    """
    Soma dois números de ponto flutuante.

    :param a: O primeiro número a ser somado.
    :type a: float
    :param b: O segundo número a ser somado.
    :type b: float
    :return: A soma dos dois números.
    :rtype: float
    """
    return a + b
```

Estruturas de decisão

Estruturas de decisão

- As estruturas de **decisão** são usadas para **permitir** que um programa faça **escolhas** com base em uma **condição**.
- Existem duas estruturas de decisão principais em Python:
  - if-else
  - if-elif-else.

if-else

- A estrutura if-else **verifica** se uma **condição** é **verdadeira** ou falsa.
- Se a condição for **verdadeira**, o código dentro do **bloco** if será **executado**. **Caso contrário**, o código dentro do bloco **else** será **executado**.

- A sintaxe é a seguinte:

```
if condição:
    # código a ser executado se a condição for verdadeira
else:
    # código a ser executado se a condição for falsa
```

if-else: exemplo

- O seguinte trecho de código recebe um número e verifica se pertence a uma pessoa maior ou menor de idade

```
idade = 18

if idade >= 18:
    print("Você é maior de idade!")
else:
    print("Você é menor de idade.")
```

if-else: exemplo 2

- Função que determina se um número é par ou ímpar

```
def par_impar(num):
    """
    Verifica se um número é par ou ímpar.

    Argumentos:
    num -- o número a ser verificado (int)

    Retorna:
    Uma string indicando se o número é par ou ímpar
    """
    if num % 2 == 0:
        return "O número é par."
    else:
        return "O número é ímpar."
```

if-else: exemplo 3

- Função que retorna o maior número entre dois

```
def maior_numero(num1, num2):
    """
    Retorna o maior número entre dois.

    Argumentos:
    num1 -- o primeiro número (int ou float)
    num2 -- o segundo número (int ou float)

    Retorna:
    O número maior entre num1 e num2 (int ou float)
    """
    if num1 > num2:
        return num1
    else:
        return num2
```

if-elif-else

- A estrutura **if-elif-else** é usada quando há **várias condições** a serem verificadas.
- Se a primeira **condição** for **falsa**, a **próxima** condição será **verificada** e assim por **diante**. Se **nenhuma** das condições for **verdadeira**, o bloco **else** será **executado**
- A sintaxe é a seguinte:

```
if condição1:
    # código a ser executado se a condição1 for verdadeira
elif condição2:
    # código a ser executado se a condição2 for verdadeira
elif condição3:
    # código a ser executado se a condição3 for verdadeira
else:
    # código a ser executado se nenhuma das condições for verdadeira
```

if-elif-else: Exemplo 1

- Trecho de código que verifica uma nota

```
nota = 7

if nota >= 9:
    print("Você tirou uma nota excelente!")
elif nota >= 7:
    print("Você tirou uma nota boa!")
elif nota >= 5:
    print("Você tirou uma nota razoável.")
else:
    print("Você precisa estudar mais!")
```

if-elif-else: Exemplo 2

- Função que classifica o nível de poluição do ar

```
def classificar_poluição(poluicao):
    """
    Classifica o nível de poluição do ar de acordo com a quantidade de partículas PM2.5.

    Argumentos:
    poluicao -- a quantidade de partículas PM2.5 em microgramas por metro cúbico (float)

    Retorna:
    Uma string indicando o nível de poluição do ar (str)
    """
    if poluicao < 12.1:
        return "Bom"
    elif poluicao < 35.5:
        return "Moderado"
    elif poluicao < 55.5:
        return "Não saudável para grupos sensíveis"
    elif poluicao < 150.5:
        return "Não saudável"
    elif poluicao < 250.5:
        return "Muito não saudável"
    else:
        return "Perigoso"
```

Operadores lógicos

Operadores lógicos

- Os operadores lógicos são usados para combinar ou comparar expressões **booleanas**.
- Em Python, existem três operadores lógicos: **and**, **or** e **not**.
  - **and**: Retorna **True** se **ambas** as expressões booleanas são **True**.
  - **or**: Retorna **True** se pelo **menos uma** das expressões booleanas é **True**.
  - **not**: **Inverte** o valor de uma expressão booleana. Se a expressão for True, retorna False. Se a expressão for False, retorna True.
- Aqui estão alguns exemplos de como esses operadores podem ser usados em Python:

Operadores lógicos : tabelas verdade

- Operador **and**

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

- Operador **or**

A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False

- Operador **not**

A	not A
True	False
False	True

Operadores lógicos: Exemplo 1

- Execute o seguinte trecho de código

```
# Exemplo com operador and
x = 5
y = 10
if x > 0 and y > 0:
    print("Ambas as variáveis são positivas.")

# Exemplo com operador or
x = 5
y = -2
if x > 0 or y > 0:
    print("Pelo menos uma das variáveis é positiva.")

# Exemplo com operador not
x = True
if not x:
    print("O valor de x é False.")
else:
    print("O valor de x é True.")
```

Operadores lógicos: Exemplo 2

- Função que classifica o aluno com base na sua nota

```
def verifica_aprovacao(nota1, nota2, presenca):
    """
    Verifica se um aluno foi aprovado ou reprovado com base em suas notas e presença.

    Parâmetros:
    nota1 (float): primeira nota do aluno.
    nota2 (float): segunda nota do aluno.
    presenca (float): porcentagem de presença do aluno.

    Retorna:
    str: "Aprovado" se o aluno tiver média de notas maior ou igual a 6 e presença igual
         ou superior a 75%.
        "Reprovado" caso contrário.
    """
    media = (nota1 + nota2) / 2
    if media >= 6 and presenca >= 75:
        return "Aprovado"
    else:
        return "Reprovado"
```

Exercício 1

- Escreva uma função que receba um número inteiro e retorne uma string indicando se ele é positivo, negativo ou zero.

Exercício 2

- Escreva uma função que calcule o IMC (Índice de Massa Corporal) de uma pessoa com base em sua altura (em metros) e massa (em quilogramas). A função deve retornar uma string indicando se a pessoa está abaixo do peso, no peso normal, com sobrepeso ou obesa.

$$IMC = \frac{\text{massa (kg)}}{[\text{altura (m)}]^2}$$

IMC	Classificação
<18,5	Abaixo do peso ideal
18,5 - 25	Peso normal
25 - 30	Excesso de peso
30 - 35	Obesidade (grau I)
35 - 40	Obesidade (grau II)
> 40	Obesidade (grau III)

Exercício 3

- O perfil de uma pessoa pode ser determinado a partir da sua data de nascimento, conforme exemplificado a seguir. Crie uma função que dada uma data de nascimento, retorne o perfil correspondente

Exemplo: 13/06/1970

① 1306 + 1970 = 3276
② 32 + 76 = 108
③ 108 | 5
105 21
3

consulte a tabela ao lado para saber o perfil correspondente ao número 3!

R	Perfil
0	*Tímido*
1	*Sonhador*
2	*Paquerador*
3	*Atraente*
4	*Irresistível*

Exercício 4

- Leia a distância em Km e a quantidade de litros de gasolina consumidos por um carro em um percurso, calcule o consumo em Km/l e escreva uma mensagem de acordo com a tabela abaixo:

CONSUMO (Km/l)	MENSAGEM
menor que 8	"Venda o carro!"
entre 8 e 14	"Econômico"
maior que 14	"Super econômico"

Exercício 5

- Crie uma função que receba os 3 lados A, B e C de um triângulo, onde A é o maior lado, e retorne em qual caso este triângulo se encaixa.

  - Se A >= B + C, então nenhum triângulo é formado
  - Se $A^2 = B^2 + C^2$, então temos um triângulo retângulo
  - Se $A^2 > B^2 + C^2$, então temos um triângulo obtusângulo
  - Se $A^2 < B^2 + C^2$, então temos um triângulo acutângulo

Desafio

- O importo de renta IR depende de uma tabela dividida em quatro faixas de renda, com uma alíquota progressiva que vai de 7,5% a 27,5%. A faixa máxima atinge os salários acima de R$ 4.664,68.
- Faixas e as respectivas alíquotas
  - **Faixa 1**: Até R$ 1.903,98: **isento**
  - **Faixa 2**: De R$ 1.903,99 até R$ 2.826,65: **7,5%**
  - **Faixa 3**: De R$ 2.826,66 até R$ 3.751,05: **15%**
  - **Faixa 4**: De R$ 3.751,06 até R$ 4.664,68: **22,5%**
  - **Faixa 5**: Acima de R$ 4.664,68: **27,5%**

Desafio

- Faixas e as respectivas alíquotas
  - **Faixa 1**: Até R$ 1.903,98: **isento**
  - **Faixa 2**: De R$ 1.903,99 até R$ 2.826,65: **7,5%**
  - **Faixa 3**: De R$ 2.826,66 até R$ 3.751,05: **15%**
  - **Faixa 4**: De R$ 3.751,06 até R$ 4.664,68: **22,5%**
  - **Faixa 5**: Acima de R$ 4.664,68: **27,5%**
- Por exemplo, um contribuinte com renda mensal tributável no valor **de R$5000,00** pagará **R$505,64** de imposto de renda, calculado da forma descrita na Tabela:

Faixa da Base de Cálculo		**Alíquota**	**Valor do Imposto (R$)**
1^a Faixa	1.903,98	Isento	0,00
2^a Faixa	922,67	7,5%	69,20
3^a Faixa	924,40	15,0%	138,66
4^a Faixa	913,63	22,5%	205,57
5^a Faixa	335,32	27,5%	92,21
Total	**5000,00**	**10,11%**	**505,64**

- Nesse caso, a alíquota efetiva (percentual final do imposto incidente sobre o rendimento do contribuinte) foi de 10,11% (= 505,64/5000,00)

Desafio

- Escreva uma função em Python que receba como entrada a **renda mensal tributável** (em reais) de um contribuinte. A função devera calcular o **valor** do **imposto** de renda devido e a alíquota efetiva e retornar uma string com esses valores
  - Considere o seguinte conjunto de casos de teste para avaliar seu programa:

Teste 1

Entrada	Saída
R$1524.99	imposto: R$0.00 alicota: 0%

Teste 2

Entrada	Saída
R$2000.00	imposto: R$7.20 alicota: 0.36%

Teste 3

Entrada	Saída
R$2900.00	imposto: R$80.20 alicota: 2.76%

Teste 4

Entrada	Saída
R$4500.50	imposto: R$376.48 alicota: 8.36%

Teste 5

Entrada	Saída
R$15000.00	imposto:R$3255.64.00 alicota:21.7%