



Algoritmos Computacionais

Conteúdo: Estruturas de repetição

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



Aula 04 - Estruturas de decisão.pdf

Funções com argumentos opcionais

Funções com argumentos opcionais

- É possível definir funções com argumentos opcionais usando o sinal de igual (=) para atribuir um valor padrão ao argumento.
 - Se nenhum valor for fornecido para o argumento durante a chamada da função, o valor padrão será usado.
- A sintaxe básica para definir uma função com argumentos opcionais é a seguinte:

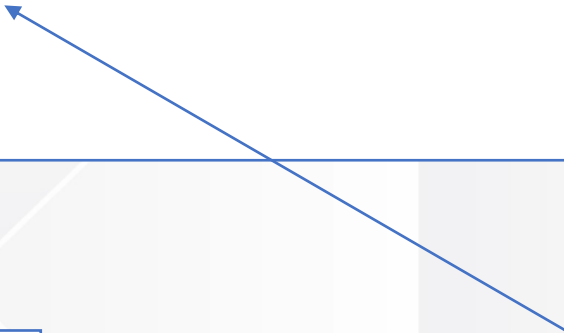
```
def nome_da_funcao(arg1, arg2=valor_padrao_arg2, arg3=valor_padrao_arg3):  
    # corpo da função
```

- Onde:
- **arg1** é um argumento **obrigatório**, enquanto **arg2** e **arg3** são **opcionais** e têm valores **padrão** de valor_padrao_arg2 e valor_padrao_arg3, respectivamente.

Funções com argumentos opcionais: Exemplo 1

- Crie a função soma

```
def soma(a, b=0):  
    """  
    Retorna a soma de dois números.  
    Argumentos:  
        a (float): O primeiro número a ser somado.  
        b (float, optional): O segundo número a ser somado. Se não for fornecido, o  
        valor padrão é zero.  
    Retorna:  
        float: A soma dos dois números.  
    """  
    return a + b
```



- Uso:

```
resultado = soma(3, 4) # retorna 7  
resultado = soma(3)   # retorna 3
```

Exemplo de documentação



Comando while

Comando while

- O comando "while" é usado para criar loops que executam repetidamente enquanto uma condição especificada for verdadeira.
- A sintaxe básica:

```
while condição:  
    #código a ser executado enquanto a condição for verdadeira
```

- A condição é avaliada no início de cada iteração do loop, e se for verdadeira, o código dentro do loop é executado. Após cada iteração, a condição é avaliada novamente e o loop continua enquanto a condição for verdadeira.

Comando while: Exemplo 1

- Por exemplo, o seguinte código usa um loop "while" para imprimir os números de 0 a 4:

```
i = 0
while i < 5:
    print(i)
    i += 1
```

Comando while: Exemplo 2

- Uma função para calcular a soma dos números de 1 a n, usando um loop "while" para iterar sobre os números.

```
def soma(n):  
    """  
    Esta função calcula a soma dos números de 1 a n.  
    """  
    resultado = 0  
    i = 1  
    while i <= n:  
        resultado += i  
        i += 1  
    return resultado
```


Comando while: Exemplo 3

- Função para determinar o fatorial de um número

```
def fatorial(numero):  
    """  
    Esta função calcula o fatorial de um número.  
    """  
    resultado = 1  
    while numero > 1:  
        resultado *= numero  
        numero -= 1  
    return resultado
```

Comando while: Exemplo 4

- Função que um imprime a tabuada de uma número

```
def tabuada(numero):  
    """  
    Esta função imprime a tabuada de um número.  
    """  
    mul = 1  
    while mul <= 10:  
        resultado = numero * mul  
        print(f"{numero} x {mul} = {resultado}")  
        mul += 1
```

>>tabuada(5)

```
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
```

Comando while: Exemplo 5

- Função para determinar o fatorial de um número

```
def fatorial(numero):  
    """  
    Esta função calcula o fatorial de um número.  
    """  
    resultado = 1  
    while numero > 1:  
        resultado *= numero  
        numero -= 1  
    return resultado
```

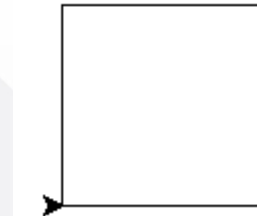
Comando while: Exemplo 6

- Função para desenha um quadrado usando o módulo turtle

```
import turtle as t
def quadrado(lado):
    """
    Desenha uma quadrado usando while
    """
    contador = 1

    while contador <= 4:
        t.forward(lado)
        t.left(90)
        contador = contador + 1
```

```
>>> quadrado(100)
```

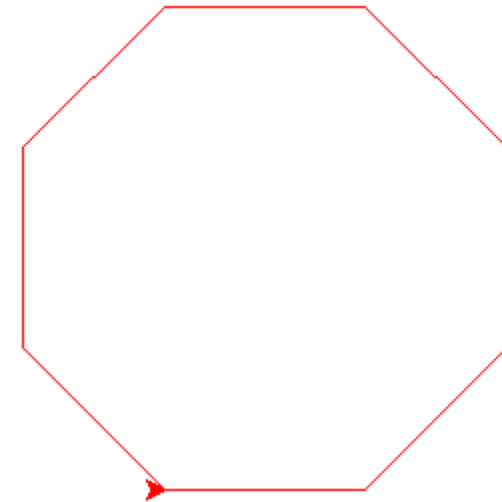


Comando while: Exemplo 7

- No mesmo arquivo do exemplo anterior acrescente a seguinte função

```
def desenha_poligono(lado, n, cor='red'):  
    """  
    Desenha uma figura usando while  
    Argumentos:  
    lado: int, número de pixels do lado  
    n: número de lados  
    cor: cor da figura  
    """  
    t.color(cor)  
    contador = 1  
    while contador <= n:  
        t.forward(lado)  
        t.left(360/n)  
        contador = contador + 1
```

```
>>> desenha_poligono(100,8)
```

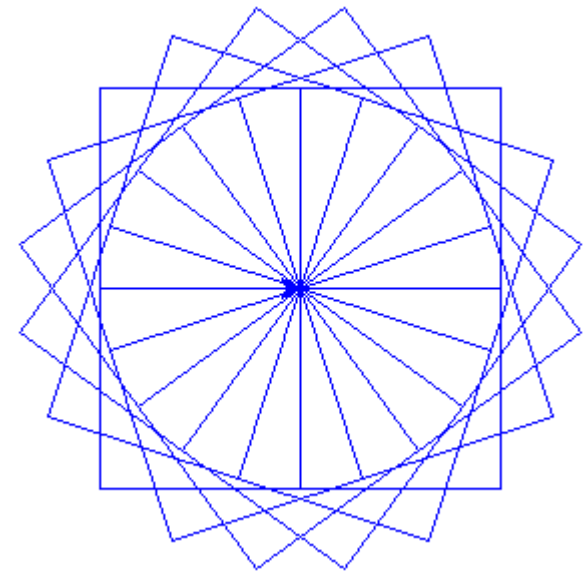


Comando while: Exemplo 8

- No mesmo arquivo do exemplo anterior acrescente a seguinte função

```
def desenha_figura(n,x, cor='blue'):  
    """  
    Desenha figura com n quadrados de lado  
x  
    """  
    t.speed('fastest') #velocidade  
    t.color(cor) # cor da caneta  
    contador = 1  
    while contador <= n:  
        quadrado(x)  
        t.left(360/n)  
        contador = contador + 1
```

```
>>> desenha_figura(20,100)
```



Exercício triângulo

- Crie uma função que imprime um triângulo usando *
- Exemplo, para $n=5$ a saída deve ser:

```
*  
**  
***  
****  
*****
```

Exercício triângulo: solução

- Uma possível solução

```
def imprimir_triângulo(n):  
    """  
    Esta função imprime um triângulo usando asteriscos.  
    """  
    i = 1  
    while i <= n:  
        print('*' * i)  
        i += 1
```


Exercício triângulo

- Crie um programa que imprima um triângulo invertido.

Exemplo:

```
>>> imprimir_triangulo_inv(5)
*****
****
***
**
*
```

Exercício Fibonacci

Crie uma função que solicite ao usuário um número **inteiro positivo** e, em seguida, imprima os primeiros "n" números da **série de Fibonacci**.

A série de **Fibonacci** é uma **sequência** de números em que cada **número subsequente** é a **soma** dos dois **números anteriores**.

- Os primeiros dois números da série são 0 e 1.
- Se por **exemplo** o usuário digitar **6**, o programa deve imprimir os primeiros 6 números da série:
0, 1, 1, 2, 3, 5.
- Se o usuário digitar 10, o programa deve imprimir os primeiros **10** números da série:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34.

Exercício Fibonacci : solução

```
def fibonacci(n):  
    """  
    Esta função solicita um número inteiro positivo ao usuário e imprime  
    os primeiros n números da série de Fibonacci.  
    """  
    # Inicialização dos primeiros dois números da série  
    num1, num2 = 0, 1  
  
    # Imprime os primeiros n números da série  
    contador = 0  
    while contador < n:  
        print(num1)  
        num1, num2 = num2, num1 + num2  
        contador += 1
```

Exercício: Jogo da adivinhação

- O jogo da adivinhação é um jogo em que o programa escolhe um **número aleatório** e o jogador deve tentar **adivinhar** esse número. O jogador deve **digitar** um **palpite** e o programa deve dizer se o palpite é **maior** ou **menor** do que o número secreto. O jogo deve continuar até que o jogador adivinhe corretamente o número.
- Dica use a função **randint** do módulo **random**

Exercício: Jogo da adivinhação - solução

- Uma possível solução

```
import random
def jogo_adivinhacao(max_tentativas = 5, limite = 100):
    numero_secreto = random.randint(1, limite) # Gera um número aleatório
    tentativas = 0
    print(f"Tente adivinhar o número entre 1 e {limite}.")

    while tentativas < max_tentativas:
        chute = int(input("Digite um número: "))
        if chute == numero_secreto:
            print("Parabéns! Você acertou!")
            break #sai do while
        elif chute < numero_secreto:
            print("Tente um número maior.")
        else:
            print("Tente um número menor.")
        tentativas += 1

    if tentativas == max_tentativas:
        print("Suas tentativas acabaram. O número secreto era:", numero_secreto)
    print("O jogo acabou!")
```

Exercício: Jogo da adivinhação - solução

- Exemplo de execução

```
>>> jogo_adivinacao()  
Tente adivinhar o número entre 1 e 100.  
Digite um número: 54  
Tente um número menor.  
Digite um número: 23  
Tente um número menor.  
Digite um número: 16  
Tente um número menor.  
Digite um número: 4  
Tente um número menor.  
Digite um número: 3  
Tente um número menor.  
Suas tentativas acabaram. O número secreto  
era: 2  
O jogo acabou!
```

Desafio

- Escreva uma função que calcule a soma de todos os números primos menores que um determinado número n .
- Exemplos de entradas e saídas:

```
>>> soma_primos(10)  
17
```

Os números primos menores que 10 são 2, 3, 5 e 7. A soma desses números é $2 + 3 + 5 + 7 = 17$.

```
>>> soma_primos(20)  
77
```

Os números primos menores que 20 são 2, 3, 5, 7, 11, 13 e 17. A soma desses números é $2 + 3 + 5 + 7 + 11 + 13 + 17 = 77$.