DTECH

Módulo 6

Funções, Módulos, Debugging e Tratamento de Exceções

Algoritmos e Estruturas de Dados I - Python

(Rone Ilídio)

Conteúdo

- Funções
  - Comando return
  - Parâmetro opcional
  - Escopo de variável
- Definição de módulos
  - Import
- Debbuging
- Tratamento de Exceção

Funções

- Funções são trechos de código, identificados por um nome, que são criados uma única vez e chamados quantas vezes forem necessárias

- Reduzem as linhas de código e organizam o programa

- Facilita a reutilização do código → criação de módulos

Funções

- A declaração de uma função segue a seguinte sintaxe:

```
def nome_função(<parametro1>,<parametro2>,...):
        <comando1>
        <comando2>
         ...
```

- Para chamar uma função, basta escrever seu nome e passar os parâmetros necessários
- Quando não há parâmetros, colocar ()

Funções

- Exemplo 1 (função que calcula a área de um círculo)

```
def areacirculo(raio):
  return 3.14 * raio * raio

r = float(input("Digite o raio de um círculo:"))
a = areacirculo(r)
print(a)
```

Definição com apenas um parâmetro

Espaço opcional

Início da execução

Chamada com apenas um parâmetro

Funções

- Exemplo 2 (recebe dois parâmetros e calcula a área do retângulo)

```
def arearetangulo(base, altura):
  return base * altura

b = float(input("Digite a base:"))
a = float(input("Digite a altura:"))
area = arearetangulo(b,a)
print(area)
```

Definição com dois parâmetros

Início da execução

Chamada com dois parâmetros

Comando return

Comando return

- Duas funcionalidades:
  - Para a execução da função
  - Envia o resultado para a chamada

Comando return

Exemplo 2 – Função que verifica o maior entre três números

```
def maior(n1,n2,n3):
  if n1>=n2 and n1>=n3:
    return n1
  if n2>=n1 and n2>=n3:
    return n2
  if n3>=n1 and n3>=n2:
    return n3
a = int(input("Digite um número:"))
b = int(input("Digite um número:"))
c = int(input("Digite um número:"))
print(maior(a,b,c))
```

Comando return

Exemplo 2 – Função que verifica se um número é primo

```
def primo(n):
    for i in range(2,n):
        if n % i == 0:
            return False
    return True

num = int(input('Informe um número:'))
resp = primo(num)
if resp:
    print(num,'é primo.')
else:
    print(num,'não é primo.')
```

Função Sem Retorno

Função Sem Retorno

- Função que não possui um valor para ser retornado
- Não possui o comando return ou após o return não possui nenhum valor

Função Sem Retorno

- Exemplo 3 - cria uma função que exibe um título na tela

```
def tit(a):
  print("-"*30)
  print(a)
  print("-"*30)
tit("Título Inicial")
print("Agora vou colocar um blablablá")
print("Vou inventar qualquer coisa para escrever\n\n")
tit("Título Final")
```

Parâmetro Opcional

Parâmetro Opcional

- Basta colocar um valor no parâmetro ao definir a função
- Exemplo

```
def somar(a,b,c=0):
   return a+b+c
s = somar(1,2)
print(s)
```

A variável c é opcional, se não for informada recebe o valor 0

Chame a função da seguinte forma:
s = somar(1,2,3)

Escopo de Variável

Escopo de Variável

- Espaço no programa onde uma variável é válida
  - Variável global: válida em qualquer parte do programa
  - Variável local: válida somente dentro de uma função
- Quando existem duas variáveis com mesmo nome, uma local e outra global, a local é acessada

Escopo de Variável

Definição de funções

```
def minhafuncao():
  x=10
  print("x dentro da função:",x)
  print("g dentro da função:",g)
```

x é uma variável local, válida só na função

Programa principal

```
x=1
g=20
minhafuncao()
print("x fora da função:",x)
print("g fora da função:",g)
```

g é uma variável global, válida em qualquer lugar

Comando global

- Define que a variável a ser utilizada será a global
- Não cria outra variável local (com mesmo nome) dentro da função

Comando global

```
def minhafuncao():
  global x
  x=10
  print("x dentro da função:",x)
x=1
print("x antes da chamada da função:",x)
minhafuncao()
print("x após a chamada da função:",x)
```

Não cria outra variável x, utiliza a variável x global

Resultado:
x antes da chamada da função: 1
x dentro da função: 10
x após a chamada da função: 10

Import

Import

- O comando import adiciona um módulo, ou bibliotecas, a seu programa
- Um módulo é um conjunto de funções
- Exemplo - módulo para calculos matemáticos math

```
import math
num = float(input("Digite um número:"))
x = math.sqrt(num)
print("A raiz quadrada de",num, "é",x)
```

Import

- A biblioteca random gera números aleatórios

Exemplo – gera um número float aleatório entre 0 e 0.999999

```
import random
alea = random.random()
print(alea)
```

Exemplo – gera 100 números inteiros aleatórios entre 0 e 100

```
import random
for i in range(0,100):
  alea = random.randint(1,10)
  print(alea, end=" ")
```

Import

- Módulo winsound: toca sons no formato wav
- Exemplo (código fonte e arquivo .wav na mesma pasta):

```
import winsound
winsound.PlaySound('cartoon001.wav',winsound.
  SND_ASYNC)
n = input('Clique [Enter] para finalizar!')
```

Definição de Módulos

Definição de Módulos

- Um módulo é um arquivo com várias funções que podem ser importadas em outros programas
- Exemplo - crie dois aquivos com os códigos

Arquvo main.py

```
import uteis
num = 5
r = uteis.fatorial(num)
print(r)
```

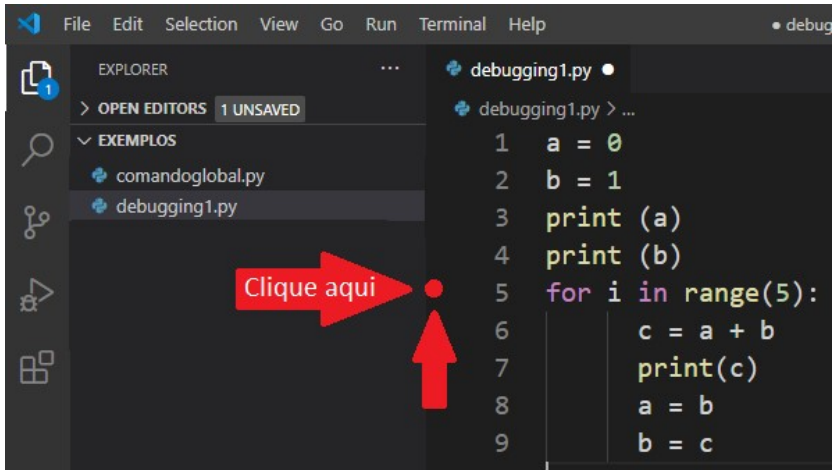
Arquivo uteis.py

```
def fatorial(n):
  f = 1
  for i in range(1,n+1):
    f = f * i
  return f
```

Debugging

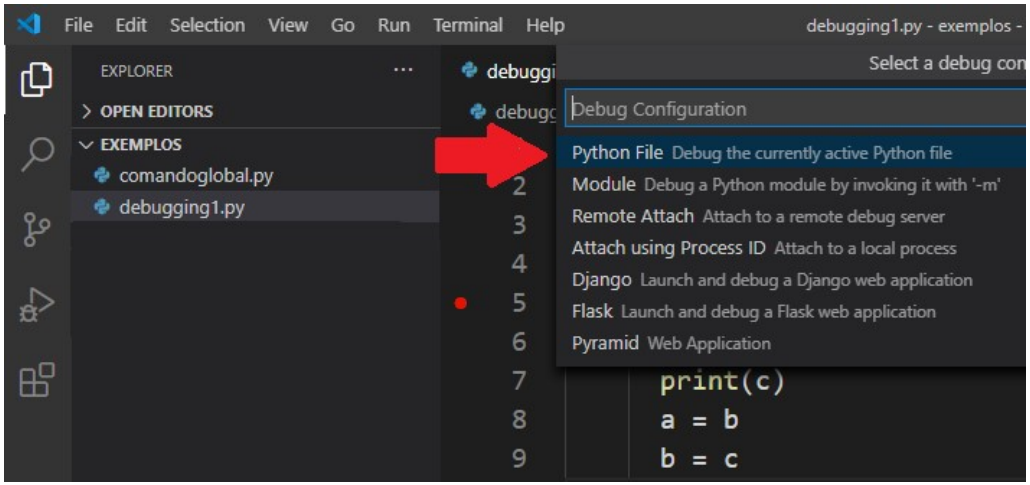
Debugging

- ‘Debugar’ ou fazer um ‘Debugging’ é realiza a interpretação do código linha a lina
- Primeiro passo um break point (linha onde a interpretação para)



Debugging

- Pressione F5 (executar e debugar)
- Configure de debug para Python



Debugging

- Controle de debug:
  - F10 – pula a função
  - F11 – entra na função
  - Shift + F11 – sai do bloco e vai para o bloco anterior
  - F5 – inicia ou continua a interpretação com debugging

Debugging

- Faça o debuggin do código a seguir

```
a = 0
b = 1
print (a)
print (b)
for i in range(5):
    c = a + b
    print(c)
    a = b
    b = c
```

Break point

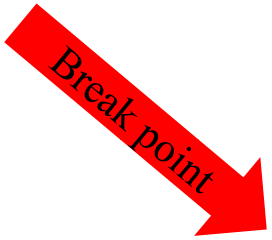
Variáveis

a	b	c	i
0	1	1	0
1	1	2	1
1	2	3	2
2	3	5	3
3	5	8	4
5	8		

Tela

0
1
1
2
3
5
8

Debugging




```
def primo(n):
    for i in range(2,n//2):
        if n % i == 0:
            return True
    return False


num = int (input('Informe um número:'))
resp = primo(num)
if resp:
    print(num,'é primo')
else:
    print(num,'não é primo‘)
```

Tratamento de Exceção

Tratamento de Exceção

- Exceção é um erro que ocorre em tempo de execução
- Exemplos de exceção:
  - divisão por zero
  - receber letra quando um número é esperado
  - acesso a posição inexistente de uma lista, tupla ou dicionário

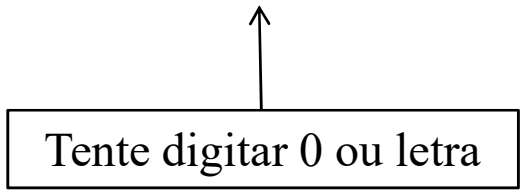
Tratamento de Exceção

- Bloco básico

```
try:
    <comandos com possível exceção>
except:
    <comandos executados em caso de exceção>
```

Tratamento de Exceção

```
try:
  a = float(input("Digite um número:"))
  b = float(input("Digite outro número:"))
  c = a/b
  print(c)
except:
  print("Existem problemas com os dados
  fornecidos.")
```



Tratamento de Exceção

- Blocos opcionais

```
try:
    <comandos com possível exceção>
except:
    <comandos executados em caso de exceção>
else:
    <comandos executados sem exceção>
finaly:
    <comandos sempre executados>
```

Exercícios

Exercícios

1. Crie uma função que recebe uma temperatura em Fahrenheit e converte para Celsius. Seu programa principal deve receber uma temperatura, chamar a função e exibir o resultado (C=(F-32)/1.8).
2. Crie um módulo com duas funções, uma que retorna o n-esimo termo (a_n) e outra que retorna a soma dos n primeiros termos (S_n) de uma PG. Crie um programa que pergunte para o usuário se ele deseja calcular a_n ou S_n, receba os parâmetros, chame a função correspondente e exiba o resultado

$$a_n = a_1 * q^{n-1} \text{ e } S_n = a_1(q^n-1)/(q-1)$$

1. Crie um módulo com funções para calcular delta, x1 e x2 de uma equação do 2º grau. Crie um programa que receba a, b e c e utilize tais funções para exibir x1 e x2. Atenção para equações com delta negativo.

Exercício 1

```
def celsius(f):
    c = (f-32)/1.8
    return c

fahr = float(input('Digite a temp em Fahrenheit:'))
resposta = celsius(fahr)
print(f'A temperatura em Celsius é {resposta:.2f}')
```

Exercício 2

- Formado por dois arquivos: pg.py e exercicio2.py
- pg.py

```
def an(a1,q,n):
    return a1 * q ** (n-1)
def sn(a1,q,n):
    return a1*(q**n-1)/(q-1)
```

```
import pg
op = input('Digite 1 para calcular an e 2 para ca
  lcular sn:')
a1 = float(input('Informe a1:'))
q = float(input('Informe q:'))
n = float(input('Informe n:'))
if op == '1':
    print('an=',pg.an(a1,q,n))
elif op == '2':
    print('sn=',pg.sn(a1,q,n))
else:
    print('Opção inválida!')
```

exercicio2.py

Exercício 3

- Formado por dois arquivos: equacao2ograu.py e exercicio3.py

equacao2ograu.py

```
import math
def delta(a,b,c):
    return b**2 - 4 * a * c
def x1(a,b,c):
    return (-b + math.sqrt(delta(a,b,c)))/(2*a)
def x2(a,b,c):
    return (-b - math.sqrt(delta(a,b,c)))/(2*a)
```

exercicio3.py

```
import equacao2ograu
a = float(input('Informe a:'))
b = float(input('Informe b:'))
c = float(input('Informe c:'))
d = equacao2ograu.delta(a,b,c)
if d<0:
    print('Equação sem raízes reais.')
else:
    print('x1:',equacao2ograu.x1(a,b,c))
    print('x2:',equacao2ograu.x2(a,b,c))
```