Fundamentos de Programação

Funções

Thiago Alves

Introdução

- Sequência de comandos que possui um nome e realiza uma funcionalidade específica
- Python fornece várias funções embutidas

```
print int('32')

print int(-3.999)

x = 32

print float(x)

print float('3.999')

print str(345.76)
```

Introdução

```
n1, n2 = raw_input().split()
n1 = int(n1)
n2 = int(n2)
print n1 + n2
```

Introdução

```
distancia = input()
tempo = input()
velMedia = float(distancia)/tempo
print 'Velocidade media:', velMedia
```

Introdução

- Python possui um módulo que fornece a maioria das funções matemáticas mais comuns
- Precisamos importar um módulo antes de utilizá-lo:
- `import math`
- `print math.log10(100)`
- `print math.sqrt(25)`
- `print math.sqrt(math.log10(1000))`

Introdução

- Podemos importar uma função específica:
- `from math import sqrt`
- `print sqrt(25)`

- Ou todas as funções de um módulo:
- `from math import *`
- `print sqrt(log10(1000))`

Exercício

- Um terreno tem forma de quadrado e o dono quer saber o tamanho de cada lado do terreno. Construa um programa em que o usuário digita a área do terreno, em metros quadrados. O programa deve mostrar o tamanho dos lados do terreno, em metros.

Exercício

- Um terreno tem forma de quadrado e o dono quer saber o tamanho de cada lado do terreno. Construa um programa em que o usuário digita a área do terreno, em metros quadrados. O programa deve mostrar o tamanho dos lados do terreno, em metros.

```
from math import sqrt

area = input()

print sqrt(area), 'metros'
```

Definição de Funções

- Podemos definir nossas próprias funções:

```
def soma(x1, x2):#x1 e x2 são parâmetros
  return x1+x2   #retorno da função

print soma(5,3) #chamada da função
#5 e 3 são argumentos e são passados
#para os parâmetros
```

Definição de Funções

- Podemos definir nossas próprias funções:

```
def soma(x1, x2):
  return x1+x2


x = 7
#variáveis como argumentos
print soma(x, 2)
```

Definição de Funções

- Podemos definir nossas próprias funções:

```
def soma(x1, x2):
  return x1+x2

x = 7
#podemos compor funcoes e operacoes
print soma(soma(x,2)*2,5)
```

Definição de Funções

- Crie uma função `aumento5(salario)` que recebe o salário e retorna 5% do salário
- Defina uma função `atualizaSal5(sal)` que recebe o salário e retorna o salário atualizado com um aumento de 5%

Definição de Funções

```
def aumento5(salario):
  acrescimo = salario*5/100.0
  return acrescimo
def atualizaSal5(sal):
  novoSal = sal + sal*5/100.0
  return novoSal
```

Definição de Funções

- Faça um programa que recebe os salários de dois funcionários de uma empresa e aumenta os salários em 5%. Seu programa deve mostrar o aumento de cada funcionário e o novo salário de cada funcionário.

Definição de Funções

```
salario1 = input()
print aumento5(salario1)
print atualizaSal5(salario1)
salario2 = input()
print aumento5(salario2)
print atualizaSal5(salario2)
```

Variáveis Locais

- Variáveis criadas dentro de funções são locais
- Só existem dentro da função

```
def aumento5(salario):
    acrescimo = salario*5/100.0
    return acrescimo

print aumento5(1734.43)
print acrescimo #erro! local na funcao!
```

Variáveis Locais

- Parâmetros de funções também são locais

```
def aumento5(salario):
  acrescimo = salario*5/100.0
  return acrescimo

print aumento5(1734.43)
print salario #erro!
```

Definição de Funções

- Crie uma função `acrescimo(sal,porc)` que recebe o salário e uma porcentagem e retorna a porcentagem do salário. Por exemplo,
```
>> print acrescimo(1500.0,10.0)
150
```
- Defina uma função `novoSal(sal, porc)` que recebe o salário e retorna o salário atualizado de acordo com a porcentagem dada. Utilize a função `acrescimo` para facilitar. A porcentagem deve ser fornecida da mesma forma que na função `acrescimo`.

Fluxo de Execução

```
def acrescimo(sal,porc):
  return sal*porc/100.0

def novoSal(sal,porc):
  return acrescimo(sal,porc) + sal

salario = 3632.78
print novoSal(salario, 7)
```

Vantagens

- Forma de agrupar uma sequência de instruções, de modo que possam ser executadas mais de uma vez em um programa
- Reutilização de código
- Maior legibilidade
- Mais fácil de mudar o código
- Permitem especificar parâmetros para diferir cada vez que a função é executada
- Decomposição das funcionalidades do programa

Exercícios

- Defina uma função `dist(x1,y1,x2,y2)` que recebe dois pontos (x1,y1) e (x2,y2) e retorna a distância entre os pontos.

Exercícios

- Defina uma função `areaCir(xc,yc,xp,yp)` que recebe o ponto do centro do círculo (xc,yc) e um ponto do perímetro do círculo (xp,yp). A função deve retornar a área do círculo. Assuma π = 3,14.

Definição de Funções

- Podemos definir funções sem retorno

```
def mensagem(tipo):
  print 'Digite um numero', tipo

mensagem('inteiro')
valor = input()
print valor
```

Definição de Funções

- Podemos definir funções sem retorno

```
def mensagem(tipo):
  print 'Digite um numero', tipo

mensagem('real')
valor = input()
print valor
```

Definição de Funções

- Podemos definir funções sem parâmetros

```
def mostrarMensagem():
  print 'Digite um numero inteiro'

mostrarMensagem()
valor = input()
print valor
```