LABORATÓRIOS DE PROGRAMAÇÃO

Funções

Aula 3

Profª. Kamilla Dória

TÓPICOS

- Introdução
- Criação de Funções

INTRODUÇÃO

FUNÇÕES

- Importante forma de abstração em programação (reuso de código);
- São inspiradas e parecidas com as funções matemáticas;
- Cada função possui um nome e um conjunto de argumentos (parâmetros).

FUNÇÕES

- Representam um cálculo que é feito sobre os valores passados como argumento;

- Sequência de comandos que calcula um valor a partir de argumentos.

FUNÇÕES

- Em programação:
  - Função para cálculo de média ponderada;
  - Função para calcular fatorial;
  - Função para calcular o troco de uma compra;
  - Função para realizar alguma ação no programa.

FUNÇÕES PREDEFINIDAS

- Função *divmod*:
  - Calcula a divisão inteira e o resto da divisão entre dois números.

```
x = int(input('x: '))
y = int(input('y: '))
print(divmod(x,y))
```

```
"FuncoesPreDefinidas.py"
x: 10
y: 3
(3, 1)

Process finished with exit code 0
```

FUNÇÕES PREDEFINIDAS

- Função *len*:
  - Retorna o comprimento de uma string (espaços em branco contam).

```
texto = input('Digite um texto: ')
print(f'Esse texto possui {len(texto)} caracteres')
```

```
"FuncoesPreDefinidas.py"
Digite um texto: Kamilla Dória
Esse texto possui 13 caracteres

Process finished with exit code 0
```

FUNÇÕES PREDEFINIDAS

- Função *help*:
  - Permite que interaja com a console para buscar documentações sobre a linguagem.

```
help()
```

```
Welcome to Python 3.8's help utility!

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help>
```

BIBLIOTECA DE FUNÇÕES

- Conjunto de funções disponíveis em um ou mais pacotes;

- Para utilizar é necessário indicar o pacote.

```
import math
print(math.factorial(10))
print(math.gcd(10,5))
print(math.pi)
```

```
"Biblioteca.py"
3628800
5
3.141592653589793

Process finished with exit code 0
```

BIBLIOTECA DE FUNÇÕES

- Mais Exemplos:

```
import random
for i in range(3):
    print(random.random())
for i in range(3):
    print(random.randrange(1,11))
for i in range(3):
    print(random.randint(1, 10))
```

```
"Biblioteca.py"
0.6267903214390255
0.0804704529432639 4
0.6517196338715099
9
5
3
9
5
3

Process finished with exit code 0
```

CRIAÇÃO DE FUNÇÕES

CRIAÇÃO DE FUNÇÕES

- Identificação do trecho de código que deve estar em uma função (abstração e reuso).

- Ex: cálculo de fatorial:

```
n = int(input('Número: '))
fat = 1
for i in range(2, n+1):
    fat *= i
print(n, '! = ', fat)
```

CRIAÇÃO DE FUNÇÕES

- Criação da função *fatorial*:
  - A função deve receber como argumento o número que deseja calcular o fatorial. O comando *def* é utilizado para definir a função:
    - def fatorial(n)
    - def fatorial(n-1)
  - Ao invés de solicitar o input do usuário, o número a ser calculado o fatorial será o que está passando por argumento.

CRIAÇÃO DE FUNÇÕES

- Criação da função *fatorial*:
  - A função irá retornar o valor calculado para o trecho do programa que invocou a função:
    - return fat
  - Ao invés de imprimir o dado, o número calculado será retornado.

CRIAÇÃO DE FUNÇÕES

- Criação da função *fatorial*:

```
def fatorial(n):
    fat = 1
    for i in range(2, n + 1):
        fat *= i
    return fat
```

CRIAÇÃO DE FUNÇÕES

- Parâmetro:
  - Variável utilizada entre parênteses na definição da função;
- Argumento:
  - Valor passado para a função quando ela é invocada.

CRIAÇÃO DE FUNÇÕES

- Exemplo:

```
def fatorial(n):
    fat = 1
    for i in range(2, n + 1):
        fat *= i
    return fat

num = int(input('Informe um número: '))
print(fatorial(num))
```

```
"NovaFuncao.py"
Informe um número: 10
3628800

Process finished with exit code 0
```

PARÂMETROS DEFAULT

- São parâmetros opcionais na função.
- Exemplo:
  - *range(10)*
  - *range(1, 10)*
  - *range(1, 10, 2)*

PARÂMETROS DEFAULT

- São parâmetros opcionais na função.
  - Exemplos:
    - *range(10)* - assume 0, 10, 1 como default
    - *range(1, 10)* – assume 1, 10, 1 como default
    - *range(1, 10, 2)*

    - *round(x)* – assume x, 0 como default
    - *round(x, 2)*

FUNÇÕES COM PARÂMETROS DEFAULT

- Basta colocar o nome do parâmetro informando o valor default:

```
def fatorial(n=1):
    fat = 1
    for i in range(2, n + 1):
        fat *= i
    return fat

num = int(input('Informe um número: '))
print(fatorial())
```

```
“NovaFuncao.py"
Informe um número: 10
1

Process finished with exit code 0
```

FUNÇÕES COM PARÂMETROS DEFAULT

- Exemplo:

```
def calcMedia(nota1, nota2, recup = 0):
    media = 0
    if recup == 0:
        media = (nota1+nota2)/2
    else:
        media = (nota1+nota2+recup)/3
    return media

n1 = int(input("Nota 1: "))
n2 = int(input("Nota 2: "))
resp = input("O aluno ficou de recuperação? [S/N]  ")
media = 0
if resp == 'S':
    recup = int(input('Nota de recuperação: '))
    media = calcMedia(n1, n2, recup)
else:
    media = calcMedia(n1, n2)
print(f'Média: {round(media, 2)}')
```

```
"Parametros.py"
Nota 1: 3
Nota 2: 6
O aluno ficou de recuperação? [S/N]  S
Nota de recuperação: 8
Média: 5.67

Process finished with exit code 0
```

FUNÇÕES SEM RETORNO

- Funções que não retornam valor para o programa que a invocou (não tem *return*).

```
def impressaoQuebraLinha(texto):
    print(f'{texto}\n')

entrada = input('Texto: ')
impressaoQuebraLinha(entrada)
entrada = input('Texto2: ')
impressaoQuebraLinha(entrada)
```

```
"FuncaoSemRetorno.py"
Texto: Kamilla
Kamilla

Texto2: Dória
Dória


Process finished with exit code 0
```

EXERCÍCIO 1

Utilizando as bibliotecas de funções do Python, escreva um programa que sorteie os seis números da Mega-Sena e os apresente em ordem crescente. Os números não podem ser repetidos.

EXERCÍCIO 2

Faça um programa que use a função valorPagamento para determinar o valor a ser pago por uma prestação de uma conta.

O programa deverá solicitar ao usuário o valor da prestação e o número de dias em atraso e passar estes valores para a função valorPagamento, que calculará o valor a ser pago e devolverá este valor ao programa que a chamou. O programa deverá então exibir o valor a ser pago na tela.

Após a execução o programa deverá voltar a pedir outro valor de prestação e assim continuar até que seja informado um valor igual a zero para a prestação. Neste momento o programa deverá ser encerrado, exibindo o relatório do dia, que conterá a quantidade e o valor total de prestações pagas no dia.

O cálculo do valor a ser pago é feito da seguinte forma: para pagamentos sem atraso, cobrar o valor da prestação; quando houver atraso, cobrar 3% de multa, mais 0,1% de juros por dia de atraso.

OBRIGADA!