Funções

Joyce Teixeira

Introdução

- Funções são utilizadas para reduzir a complexidade de um algoritmo;
- Ou seja: dividir um problema grande em vários problemas menores;
- Redução de complexidade → Facilidade de compreensão;
- Chamadas de:
  - Sub-programas
  - Sub-rotinas
  - Procedimentos

Introdução

- Você saberia fazer um algoritmo que pedisse para digitar um número?
- E para imprimir um número na tela?
- Não são algoritmos fáceis de desenvolver, então, como você os utilizam?
- Existem comandos prontos que executam essas ações;
- Só é necessário solicitar a execução desses algoritmos dentro do nosso;
- Esse tipo de comando que utilizamos é chamado de funções.

Funções

Funções são:

- Blocos de códigos bem definidos, que possuem início, meio e fim;
- Separados do programa principal ;
- Servem para programar modularmente;
- Possuem um identificador (nome) que é utilizado em um programa maior, como se fosse um comando.

Funções

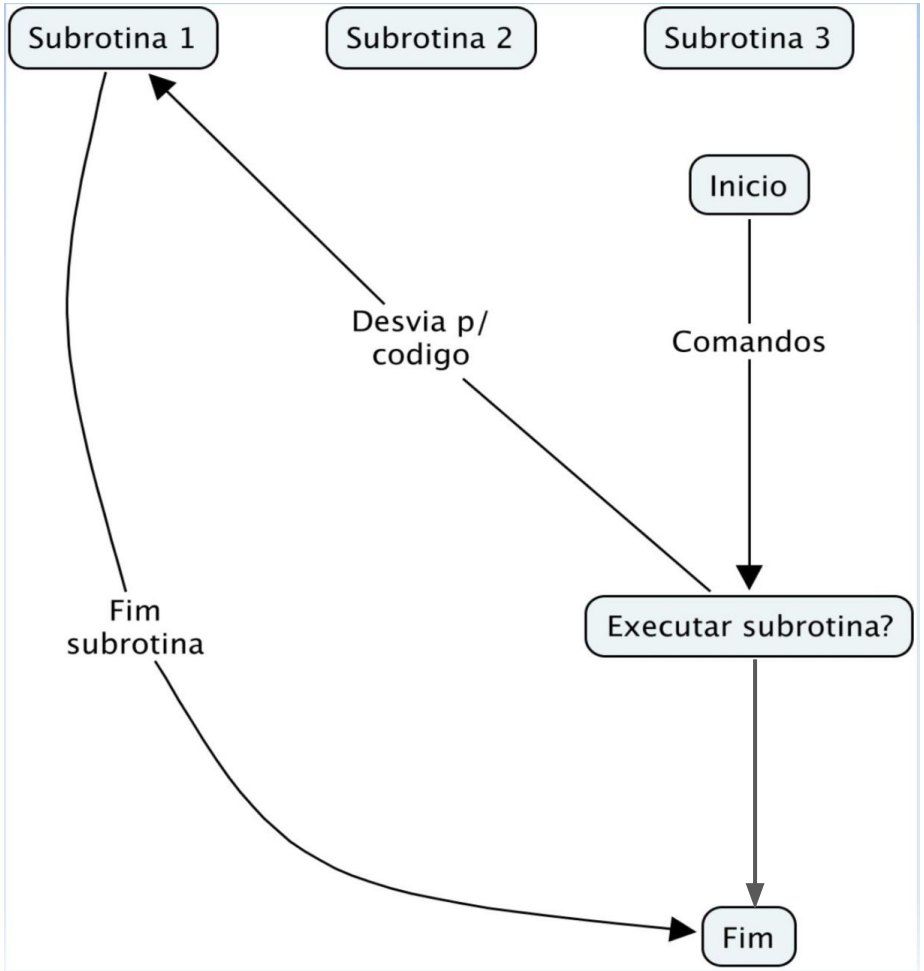
Sintaxe:

```
def <NOME_DA_FUNCAO> ( <PARAMETROS_DA_FUNCAO> ):
    <COMANDOS>
    return <RETORNO_DA_FUNCAO>

### Programa Principal ###
```

Funções

Sintaxe:

```
def <NOME> ( <PARAM> ):
    <COMANDOS>
    [return <ALGO>]

## Programa Principal ##
```



Funções

- Existem várias vantagens em utilizar funções:
  - Maior controle sobre a complexidade;
  - Estrutura lógica mais clara;
  - Maior facilidade de testar o código;
  - Possibilidade de reutilização do código.

Funções - Nomes

Nomes para funções:

- Mesmas regras utilizadas para montar nomes de variáveis.
- São identificadores:
  - *Não pode haver espaços em branco, símbolos ou caracteres de pontuação;*
  - *Precisam começar por letras ou underline;*
  - *Não pode ser uma palavra chave/reservada.*

Funções - Outras definições

Como um **sub-programa**, pode ter quase tudo que existe em um programa:

- Variáveis: serão todas locais;
- Atribuição, desvio condicional, estruturas de repetição, ...
- Uso de funções pré-definidas;
- Chamadas a outros sub-programas.

Funções - Outras definições

- Mas se eu posso escrever tudo no programa principal, qual a utilidade das funções?
- Vamos para o PyCharm!

Funções - Exemplo

```
nome = input('Digite seu nome: ')
print('Olá, boa noite!')
print(f'Bem-vindo(a), {nome}')
nome = input('Digite seu nome: ')
print('Olá, boa noite!')
print(f'Bem-vindo(a), {nome}')
nome = input('Digite seu nome: ')
print('Olá, boa noite!')
print(f'Bem-vindo(a), {nome}')
```

```
def boa_noite():
    print('Olá, boa noite!')


# Programa principal
nome = input('Digite seu nome: ')
boa_noite()
print(f'Bem-vindo(a), {nome}')
nome = input('Digite seu nome: ')
boa_noite()
print(f'Bem-vindo(a), {nome}')
nome = input('Digite seu nome: ')
boa_noite()
print(f'Bem-vindo(a), {nome}')
```

```
Digite seu nome: Maria
Olá, boa noite!
Bem-vindo(a), Maria
Digite seu nome: Carlos
Olá, boa noite!
Bem-vindo(a), Carlos
Digite seu nome: Thiago
Olá, boa noite!
Bem-vindo(a), Thiago
```

Funções - Exemplo

- Faça uma função que imprima a seguinte assinatura:

```
Atenciosamente,
Maria Silva
Gerente de Vendas
Cel. 99999-8888
```

- No programa principal, solicite o nome de três pessoas e envie e-mail de boas-vindas a cada uma, com a assinatura no final.

Funções - Argumentos

Como é possível "preencher" os parênteses do cabeçalho de uma função?

- **Argumentos / parâmetros:** Uma lista de tipos separados por vírgula

Em Python, todos os argumentos/parâmetros são passados por valor:

- Apenas uma cópia da variável original é passada para a função. Mudanças a essa variável no interior da função **não** acarretam em mudanças na variável original

Funções - Exemplo

```python
def boa_noite(nome):
    print('Olá, boa noite!')
    print(f'Bem-vindo(a), {nome}')


# Programa principal
boa_noite('Maria')
boa_noite('Thiago')
boa_noite('Carlos')
```

```
Olá, boa noite!
Bem-vindo(a), Maria
Olá, boa noite!
Bem-vindo(a), Thiago
Olá, boa noite!
Bem-vindo(a), Carlos
```

Funções - Exemplo

```
x = 5
y = 10
m = (x + y) / 2
print(m)

x = 7
y = 4
m = (x + y) / 2
print(m)

x = 9
y = 8
m = (x + y) / 2
print(m)
```

```
def media(x, y):
    m = (x + y) / 2
    print(m)


# Programa Principal
media(10, 5)
media(7, 4)
media(8, 9)
```

```
7.5
5.5
8.5
```

Funções - Exemplo

```python
def media(x, y):
    m = (x + y) / 2
    print(f'A média é: {m}')


# Programa Principal
n1 = int(input('Digite num 1: '))
n2 = int(input('Digite num 2: '))
media(n1, n2)
```

```
Digite num 1: 7
Digite num 2: 10
A média é: 8.5
```

Funções - Argumentos

- Na verdade, n1 e n2 podem receber quaisquer nomes dentro da função
- Apenas o valor de n1 e n2 são passados para a função
- Nomes dentro da função não importam
- Para argumentos, é importante garantir três características:
  - Tipo
  - Quantidade
  - Ordem

Funções - Outras definições

- **Tipo:** preferencialmente, os tipos dos parâmetros devem coincidir dentro e fora da função (parâmetros podem ser heterogêneos)

```
def media(x, y):
    m = (x + y) / 2
    print(f'A média é: {m}')


# Programa Principal
n1 = int(input('Digite num 1: '))
n2 = int(input('Digite num 2: '))
media(n1, n2)
```

```
Digite num 1: 10
Digite num 2: sete
Traceback (most recent call last):
ValueError: invalid literal for int() with base 10: 'sete'
```

Funções - Outras definições

- **Tipo:** preferencialmente, os tipos dos parâmetros devem coincidir dentro e fora da função (parâmetros podem ser heterogêneos)

```
def media(nome, x, y):
    m = (x + y) / 2
    print(f'{nome} sua média é: {m}')


# Programa Principal
n = input('Digite seu nome: ')
n1 = int(input('Digite num 1: '))
n2 = int(input('Digite num 2: '))
media(n, n1, n2)
```

```
Digite seu nome: Maria
Digite num 1: 10
Digite num 2: 7
Maria sua média é: 8.5
```

Funções - Outras definições

- **Quantidade:** a chamada da função e o cabeçalho da função devem ter a mesma quantidade de elementos, a não ser que tenha parâmetros opcionais informados na definição da função.

```
def media(nome, x, y):
    m = (x + y) / 2
    print(f'{nome} sua média é: {m}')


# Programa Principal
media('Maria', 10)
```

```
TypeError: media() missing 1 required positional argument: 'y'
```

Definindo valor default

```
def media(nome, x, y=0):
    m = (x + y) / 2
    print(f'{nome} sua média é: {m}')


# Programa Principal
media('Maria', 10)
```

```
Maria sua média é: 5.0
```

Funções - Outras definições

- **Ordem:** a ordem dos argumentos no momento de chamada é a ordem de atribuição dentro da função.

```python
def media(nome, x, y):
    m = (x + y) / 2
    print(f'{nome} sua média é: {m}')


# Programa Principal
media('Maria', 10, 7)
```

```
Maria sua média é: 8.5
```

```python
def media(nome, x, y):
    m = (x + y) / 2
    print(f'{nome} sua média é: {m}')


# Programa Principal
media(10, 7, 'Maria')
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Funções - Variáveis

*Uma variável é considerada **global** quando sua declaração acontece no algoritmo principal, enquanto que uma variável é considerada **local** quando sua declaração acontece em um subprograma (função ou procedimento).*

- **Variáveis globais:**
  - Inicializada/utilizada no algoritmo principal;
  - Todos os subprogramas podem enxergar a variável global.

- **Variáveis locais:**
  - Inicializada/utilizada no subprograma;
  - Somente o subprograma pode enxergá-la e utilizá-la.

Funções - Exemplo

```
# Programa principal
x = 10
print(f'No programa principal x é {x}')
```

```
No programa principal x é 10
```

```
def verifica():
    print(f'Na função verifica x é {x}')


# Programa principal
x = 10
print(f'No programa principal x é {x}')
verifica()
```

```
No programa principal x é 10
Na função verifica x é 10
```

- **x** está no **escopo global**, por esse motivo a função verifica o enxerga.

Funções - Exemplo

```
def verifica():
    y = 5
    print(f'Na função verifica x é {x}')
    print(f'Na função verifica y é {y}')


# Programa principal
x = 10
print(f'No programa principal x é {x}')
verifica()
print(f'No programa principal y é {y}')
```

```
NameError: name 'y' is not defined
```

- **y** está no **escopo local**, por esse motivo o programa principal **não** o enxerga.

Funções - Exemplo

```
def verifica():
    x = 50
    print(f'Na função verifica x é {x}')


# Programa principal
x = 10
print(f'No programa principal x é {x}')
verifica()
```

```
No programa principal x é 10
Na função verifica x é 50
```

- Há **duas variáveis** com o nome **x**, no escopo **global** e no escopo **local**.

Funções - Exemplo

```
def verifica(y):
    x = 2
    y += 5
    print(f'Na função verifica x é {x}')
    print(f'Na função verifica y é {y}')


# Programa principal
x = 10
verifica(x)
print(f'No programa principal x é {x}')
```

```
Na função verifica x é 2
Na função verifica y é 15
No programa principal x é 10
```

```
def verifica(y):
    global x
    x = 2
    y += 5
    print(f'Na função verifica x é {x}')
    print(f'Na função verifica y é {y}')


# Programa principal
x = 10
verifica(x)
print(f'No programa principal x é {x}')
```

```
Na função verifica x é 2
Na função verifica y é 15
No programa principal x é 2
```

- Quando a **palavra reservada global** é utilizada, é possível modificar o valor da variável global dentro do escopo local.

Funções - Return

- O "**return**" interrompe a função;
- Pode estar em qualquer parte da função;
- Após execução do "**return**", todo o restante do código é ignorado;
- Uso "opcional" (depende da lógica);
- Pode haver mais de um;
- Pode receber como argumento: (a) Um valor constante; (b) Uma expressão; (d) Uma variável; (e) Uma lista de variáveis separadas por vírgula.

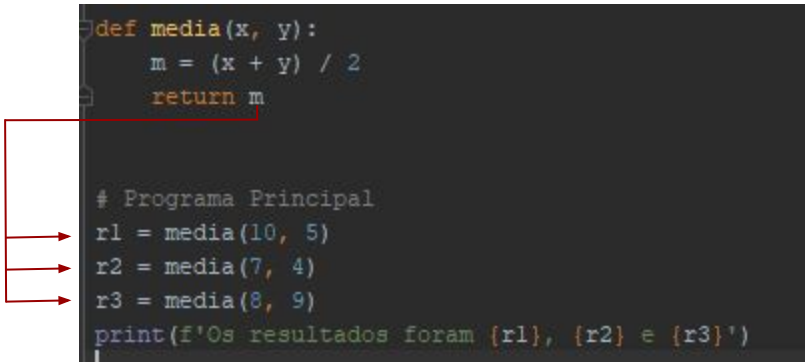
Funções - Return

- Como seria possível exibir “As médias foram 7.5, 5.5 e 8.5” ?

```
def media(x, y):
    m = (x + y) / 2
    print(m)


# Programa Principal
media(10, 5)
media(7, 4)
media(8, 9)
```

Funções - Return

- Quando a função possui retorno, é necessário armazenar e/ou usar este retorno.



```python
def media(x, y):
    m = (x + y) / 2
    return m


# Programa Principal
r1 = media(10, 5)
r2 = media(7, 4)
r3 = media(8, 9)
print(f'Os resultados foram {r1}, {r2} e {r3}')
```

```
Os resultados foram 7.5, 5.5 e 8.5
```

Funções - Return

- Faça uma função que receba por parâmetro uma idade e retorne True se apto a votar e False caso contrário. No programa principal solicite a idade ao usuário e chame a função, exibindo: ‘Pode votar’ ou ‘Ainda não pode votar’.

```
def votar(i):
    if i >= 16:
        return True
    else:
        return False


# Programa principal
idade = int(input('Digite sua idade: '))
if votar(idade):
    print('Pode votar')
else:
    print('Ainda não pode votar')
```

```
Digite sua idade: 10
Ainda não pode votar
```

Importação de Módulo e Funções

Para funções, é importante garantir três características:

- Para usar um módulo, ele precisa ser importado:
  - `import module`
- Assim as funções daquele módulo podem ser utilizadas:
  - `module.function( )`
- Também é possível importar tudo de um módulo de uma só vez:
  - *`from module import *`*
  - Isso permite que você chame as funções do módulo sem referenciá-lo

Importação de Módulo e Funções

pessoa.py

```python
def idade_func(nome):
    ano_atual = int(input('Qual o ano atual? '))
    ano_nasc = int(input('Qual o ano do seu nascimento? '))
    i = ano_atual - ano_nasc
    return i

```

cadastro.py

```python
import pessoa
nome = input('Qual seu nome? ')
idade = pessoa.idade_func(nome)
print(f'Sua idade é: {idade}')
```

```python
from pessoa import idade_func
nome = input('Qual seu nome? ')
idade = idade_func(nome)
print(f'Sua idade é: {idade}')
```

```python
from pessoa import *
nome = input('Qual seu nome? ')
idade = idade_func(nome)
print(f'Sua idade é: {idade}')
```

Importação de Módulo e Funções

pessoa.py

```
def idade_func(nome):
    ano_atual = int(input('Qual o ano atual? '))
    ano_nasc = int(input('Qual o ano do seu nascimento? '))
    i = ano_atual - ano_nasc
    return i

```

cadastro.py

```
from pessoa import idade_func

pessoa.idade_func('Joyce')
```

```
NameError: name 'pessoa' is not defined
```

Importação de Módulo e Funções

- Então por que não usar *from somemodule import* * ?
  - O uso da importação de todas as funções do módulo é desencorajado, pois desorganiza o escopo global e, se você importar itens de vários módulos, provavelmente entrará em conflito e substituirá as classes/funções existentes.

Função “Main” no Python?

Função “Main” no Python?

- Não existe uma função pré-definida chamada *main* no Python
- Existe uma variável nativa chamada __name__. Com ela, é possível saber se o módulo (arquivo.py, por exemplo), está sendo executado diretamente ou se está sendo importado por outro módulo
- Como diferenciar se o arquivo está sendo executado diretamente () ou por meio de outro arquivo (from module import *)?

Função “Main” no Python?

- *if __name__ == “__main__”*

executando diretamente o arquivo pessoa.py

```
def idade_func(nome):
    ano_atual = int(input('Qual o ano atual? '))
    ano_nasc = int(input('Qual o ano do seu nascimento? '))
    i = ano_atual - ano_nasc
    return i


if __name__ == '__main__':
    n = input('Olá! Digite seu nome: ')
    idade = idade_func(n)
    print(f'{n}, você tem {idade} anos')
    print(__name__)
```

```
Olá! Digite seu nome: Joyce
Qual o ano atual?2023
Qual o ano do seu nascimento?1989
Joyce, você tem 34 anos.
__main__
```

Isso permite que partes do código sejam reutilizadas em outros scripts, se necessário, sem serem executadas automaticamente quando importadas.

Função “Main” no Python?

- *if __name__ == “__main__”*

pessoa.py

```python
def idade_func(nome):
    ano_atual = int(input('Qual o ano atual? '))
    ano_nasc = int(input('Qual o ano do seu nascimento? '))
    i = ano_atual - ano_nasc
    return i


if __name__ == '__main__':
    n = input('Olá! Digite seu nome: ')
    idade = idade_func(n)
    print(f'{n}, você tem {idade} anos')
    print(__name__)
else:
    print('O else não é necessário')
    print('Está aqui apenas para mostrar que o __name__ não é __main__')
    print(f'__name__ é {__name__}')
```

Importando o módulo

cadastro.py

```python
from pessoa import idade_func
nome = input('Qual seu nome? ')
idade = idade_func(nome)
print(f' Sua idade é: {idade}')
```

Função “Main” no Python?

Importando o módulo

- *if __name__ == “__main__”*

```
O else não é necessário
Está aqui apenas para mostrar que o __name__ não é __main__
__name__ é: pessoa
Qual seu nome? Joyce
Qual o ano atual?2023
Qual o ano do seu nascimento?1989
Sua idade é: 34
```

Função “Main” no Python?

- Geralmente, para não deixar código “solto” no arquivo, é definida uma função main() que é chamada dentro do

  if __name__ == “__main__”

Função “Main” no Python?

ANTES

```
def idade_func(nome):
    ano_atual = int(input("Qual o ano atual?"))
    ano_nasc = int(input("Qual o ano do seu nascimento?"))
    idade = ano_atual - ano_nasc
    return idade


if __name__ == '__main__':
    nome = input("Olá! Digite seu nome:")
    idade = idade_func(nome)
    print(nome, ", você tem", idade, "anos")
```

Função “Main” no Python?

DEPOIS

```
def main():
    nome = input("Olá! Digite seu nome:")
    idade = idade_func(nome)
    print(nome, ", você tem", idade, "anos")

def idade_func(nome):
    ano_atual = int(input("Qual o ano atual?"))
    ano_nasc = int(input("Qual o ano do seu nascimento?"))
    idade = ano_atual - ano_nasc
    return idade


if __name__ == '__main__':
    main()
```

Exercícios

1. Crie uma função que realize a conversão de Horas para Minutos, onde o tempo em horas é passado como parâmetro e o tempo em minutos é retornado. Crie também um algoritmo para testar tal função.

2. Faça uma função que recebe duas notas de um estudante e retorna se aprovado (>=7) ou reprovado (<7). Crie também um algoritmo para testar tal função.

3. Altere a função acima para não retornar informações e sim exibir a situação na própria função. O algoritmo principal deve receber as duas notas e testar a função.

Funções Pré-definidas

No Python, há inúmeras funções já pré-definidas, que podem ser utilizadas no nosso código. Alguns módulos externos (de sistema) que existem...

- math
  - *ceil(x), cos(x), degrees(x), exp(x), fabs(x), factorial(x), floor(x), log(x[, base]), log10(x), pow(x,y), radians(x), sin(x), sqrt(x), tan(x), e, pi, ...*
- string
  - *format(), ...*
- random
  - *seed([a]), randrange(start, stop[, step]), random(), randint(a, b), ...*

Funções Pré-definidas: Math

Função	Descrição	Uso
ceil()	Arredondamento para cima	ceil(x)
cos()	Cosseno	cos(x)
degrees()	Converte de radianos para graus	degrees(x)
exp()	Potenciação com base e	exp(x)
fabs()	Valor absoluto (módulo)	fabs(x)
factorial	Fatorial	factorial(x)
floor()	Arredondamento para baixo	floor(x)
log()	Logaritmo natural ou neperiano	log(x[, base]
log10()	Logaritmo na base 10	log10(x)
pow()	Potência	pow(x,y)
radians()	Converte de graus para radianos	radians(x)
sin()	Seno	sin(x)
sqrt()	Raiz quadrada	sqrt(x)
tan()	Tangente	tan(x)
e	Número de Euler (e = 2,718281...)	e
pi	Constante PI (3,141592...)	pi

Exemplos

```
import math

print("A raiz quadrada de 36 é: ", math.sqrt(36))
```

```
from math import *

print("A raiz quadrada de 36 é: ", sqrt(36))
```

```
A raiz quadrada de 36 é:  6.0
```

Exercícios

1. Faça um algoritmo que solicite o raio ao usuário e informe a medida da circunferência, utilizando o valor de PI via função pré-definida.

2. Em um sítio há um determinado número de árvores que possui exatamente o mesmo número de galhos e em cada galho há exatamente o mesmo número de maçãs. Solicite este número ao usuário, utilize uma função pré-definida e descubra quantas maçãs existem no sítio.

Funções

- Lista de Exercício 08 disponível!