Programação I

LUCAS SAMPAIO LEITE

Funções

- Uma função é uma **sequência nomeada de instruções que executa uma determinada operação**, sendo criada a partir de um nome e de instruções que a compõe, possibilitando ser utilizada posteriormente.
- Elas permitem dividir o código em partes menores e mais gerenciáveis, o que torna o código mais legível e facilita a manutenção.

Funções

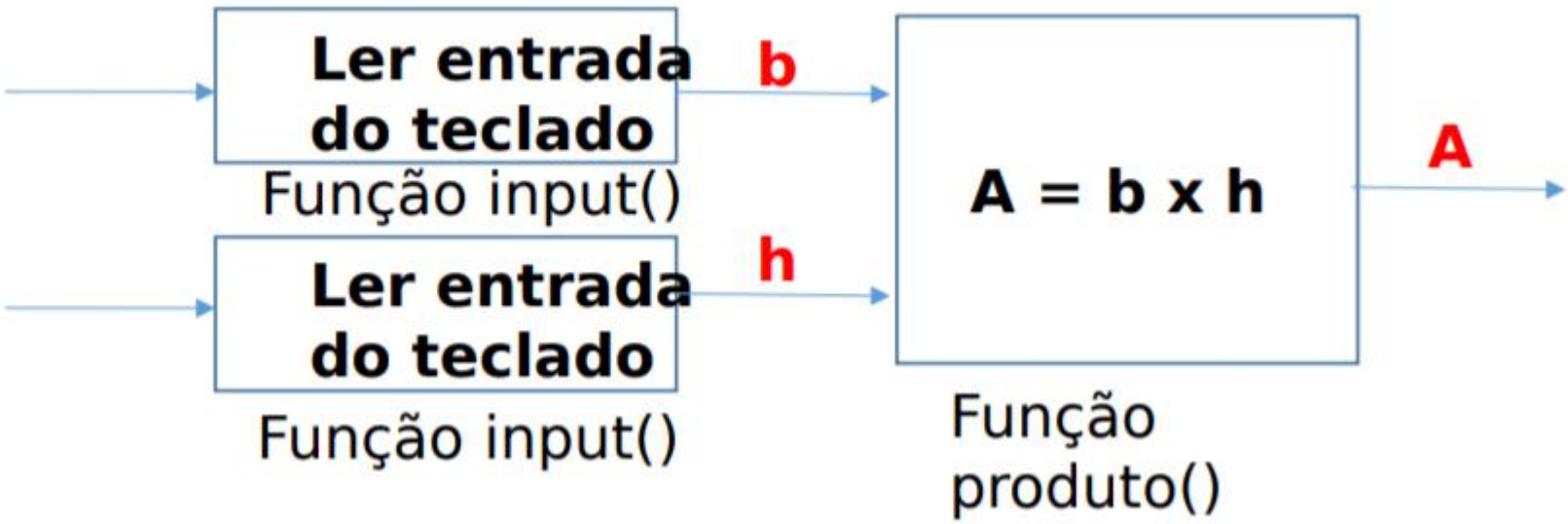
- Dividir para conquistar:
  - Abordagem comum para resolver um problema;
  - Dividir um problema em subproblemas menores;
  - Resolvemos cada subproblema;
  - Compomos as soluções de cada subproblema para chegar a uma solução do problema original.

Funções

- Na programação:
  - Dividimos a implementação de um algoritmo em funções;
  - Cada função resolve uma pequena parte do problema;
  - Uma mesma função pode ser usada para diversos outros problemas;
    - Ex: função multiplicação serve para calcular a área de um retângulo e de um triângulo;
    - Para resolver ambos os problemas precisamos calcular uma multiplicação.

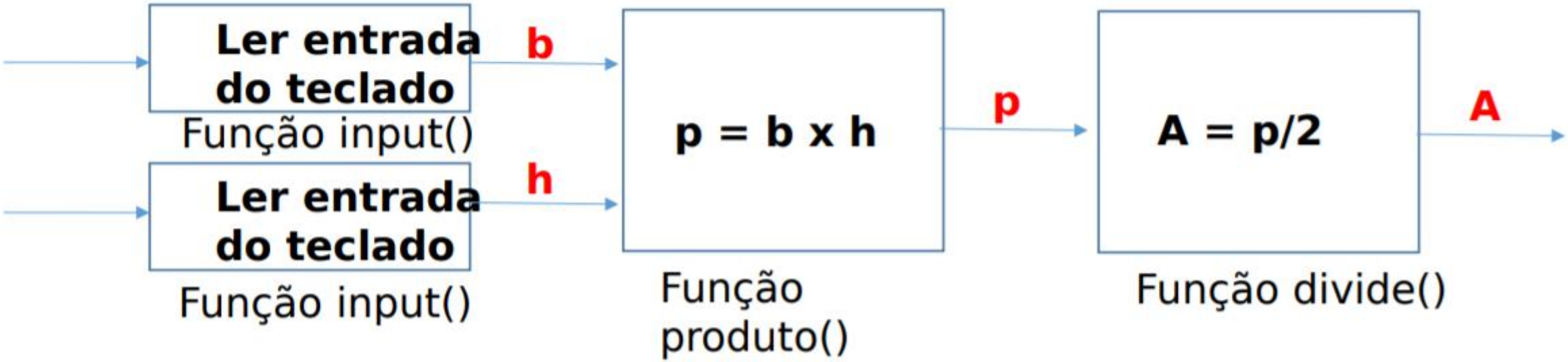
Funções

- Exemplo: cálculo área retângulo.



Funções

- Exemplo: cálculo área triângulo.



Funções

main.py > ...

```
1    def identificador(parametros):
2        #corpo da funcao
3        #o corpo da funcao pode ter várias linhas
4        #a indentação irá definir o corpo da funcao
5        return #Uma função pode ter ou não retorno
6    #este trecho está fora da função
```

- Def: define uma nova função.
- Identificador:
  - Serve para nomear a função;
  - É por meio do identificador da função que podemos invocá-la;
  - Ou seja, solicitar que ela seja executada para resolver parte de um problema.
- Parâmetros/argumentos (entre parênteses):
  - São valores de entrada usados na função;
  - Uma função pode ter 0 ou mais parâmetros.

Funções

```
main.py > ...
1    def identificador(parametros):
2        #corpo da funcao
3        #o corpo da funcao pode ter várias linhas
4        #a indentação irá definir o corpo da funcao
5        return #Uma função pode ter ou não retorno
6    #este trecho está fora da função
```

- Def: define uma nova função.
- Corpo:
  - Espaço que pode conter uma ou mais linhas de código;
  - Dentro do corpo podemos ter um comando return.
- return:
  - Retorna o resultado da função;
  - Algumas funções não tem retorno ou não retornam nenhum valor;
    - return None;
    - return
    - Ou o return pode não aparecer

Funções

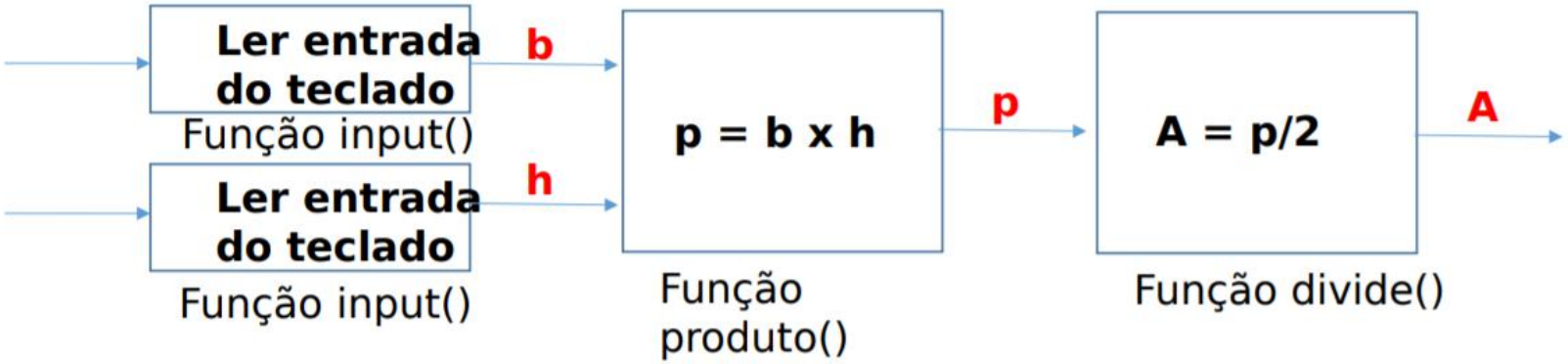
Fonte: https://br.pinterest.com/pin/790170697115762719/

Funções

```python
def nome_da_funcao(parametro):
    """Esta é a forma de declarar
    funções em python."""
    print('Esta função recebeu ', parametro, 'como parâmetro.')
    print('O tipo deste parâmetro é: ',type(parametro))
    return f'Estudamos funções na disciplina de {parametro}'
print(nome_da_funcao('Lógica de programação'))
```

```
lucas@lucas-Inspiron-5548:~/Dropbox/IFPE/2023.2/Logica/codes$
cas/Dropbox/IFPE/2023.2/Logica/codes/functions.py
Esta função recebeu  Lógica de programação como parâmetro.
O tipo deste parâmetro é:  <class 'str'>
Estudamos funções na disciplina de Lógica de programação
```

Funções



main.py > ...

```python
def produto (op1, op2):
    res = op1*op2
    return res

def divisao (op1, op2):
    if op2 == 0:
        return
    res = op1/op2
    return res

b=float(input('Digite o valor da base: '))
h=float(input('Digite o valor da altura: '))
p = produto(b,h)
area = divisao(p,2)
print ('Area do triângulo: ',area)
```

Funções

```
def calculo_IMC(peso, altura):
    return peso/altura**2

print(calculo_IMC(88, 1.72))
```

```
lucas@lucas-Inspiron-5548:~/Drop
cas/Dropbox/IFPE/2023.2/Logica/c
29.745808545159548
```

Funções

main.py > ...

```python
def calcularPagamento(qtd_horas, valor_hora):
    horas = float(qtd_horas)
    taxa = float(valor_hora)
    if horas <= 40:
        salario=horas*taxa
    else:
        hora_excedente = horas - 40
        salario = 40*taxa+(hora_excedente*(1.5*taxa))
    return salario

resultado = calcularPagamento(20,100)
print(resultado)
resultado = calcularPagamento(40,100)
print(resultado)
resultado = calcularPagamento(60,100)
print(resultado)
```

Funções

- Escopo de Variável: Bloco de código em que uma variável pode ser acessada;
- Escopo local:
  - Uma variável criada dentro de uma função pertence ao escopo local dessa função;
  - Logo, só pode ser usada dentro dessa função.
- Escopo global:
  - Variável criada no corpo principal de um arquivo python;
  - Podem ser acessadas tanto no escopo global quanto local
  - Uma variável criada fora de uma função pode ser acessada dentro e fora de uma função.

Funções

- Exemplo de uso:

```
var_global = 'Sou uma variável global \o/'
def show_me_the_code():
    var_local = 'Sou uma variável local o/'
    print(var_local)
    print(var_global)
show_me_the_code()
print(var_global)
print(var_local)
```

```
lucas@lucas-Inspiron-5548:~/Dropbox/IFPE/2023.2/Logica/codes$ /bin/pytho
s/Dropbox/IFPE/2023.2/Logica/codes/functions.py
Sou uma variável local o/
Sou uma variável global \o/
Sou uma variável global \o/
Traceback (most recent call last):
  File "/home/lucas/Dropbox/IFPE/2023.2/Logica/codes/functions.py", line
le>
    print(var_local)
NameError: name 'var_local' is not defined. Did you mean: 'var_global'?
```

Especificação de tipo de parâmetros e retorno

- Em Python, é possível especificar o tipo dos parâmetros e o tipo do retorno de uma função por meio de anotações (*annotations*) de tipo.
- As anotações de tipo são especificações de tipos opcionais que podem ser adicionadas à definição de uma função, indicando os tipos dos argumentos e do valor de retorno.
- Exemplo:

```
1  def soma(a: int, b: int) -> int:
2      resultado = a + b
3      return resultado
4
5  x=soma(5,6)
6  print(x)
```

Especificação de tipo de parâmetros e retorno

- As anotações de tipo em Python são opcionais e não afetam o comportamento da função em tempo de execução.
- No entanto, elas são uma ferramenta útil para ajudar os programadores a entender a interface da função e como ela deve ser usada.
- As anotações de tipo em Python são uma convenção e não são verificadas automaticamente pelo interpretador Python.
- Para garantir que os tipos dos argumentos e do valor de retorno correspondam às anotações de tipo, podem-se usar ferramentas como o mypy (-> https://mypy-lang.org/) que verificam automaticamente a consistência dos tipos em tempo de desenvolvimento.

Funções com lista de argumentos

- Python permite que uma função receba um número de argumentos não definidos previamente.
- Para isso, basta escrever * na frente do nome do parâmetro na função; neste caso, todos os parâmetros presentes após o asterisco serão considerados como uma tupla de elementos.

```
1   def recebeMultiplosArgumentos(*valores):
2       print(type(valores))
3       return valores
4   recebeMultiplosArgumentos(0, 5, 8, 7, 6, 4)
```

Funções com lista de argumentos

- Uma função que soma “n” números:

```
1    def soma(*valores):
2        return sum(valores)
3    print(soma(0, 5, 8, 7, 6, 4))
```

Funções com criação de dicionário de argumentos

- Para criar um dicionário com base em uma lista de argumentos, basta utilizar ** na frente do nome do parâmetro na função.
- Neste caso, cada valor no argumento deve ser indicado utilizando o operador = para indicar chaves e valores.

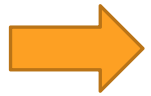
```
def funcao_recebe_dicionario(**dic):
    print(type(dic))
    return dic
print(funcao_recebe_dicionario(a=1, b=2, c=3))
```

```
lucas@lucas-Inspiron-5548:~/Drop
s/Dropbox/IFPE/2023.2/Logica/cod
<class 'dict'>
{'a': 1, 'b': 2, 'c': 3}
```

Funções com argumentos de valores padrão e ordem de passam de parâmetros

- Para valores padrão, a função aceita (mas não exige) que um valor seja adicionado à função no momento de sua chamada.
- Caso esse valor não seja indicado, a função assumirá o valor padrão indicado a ela.

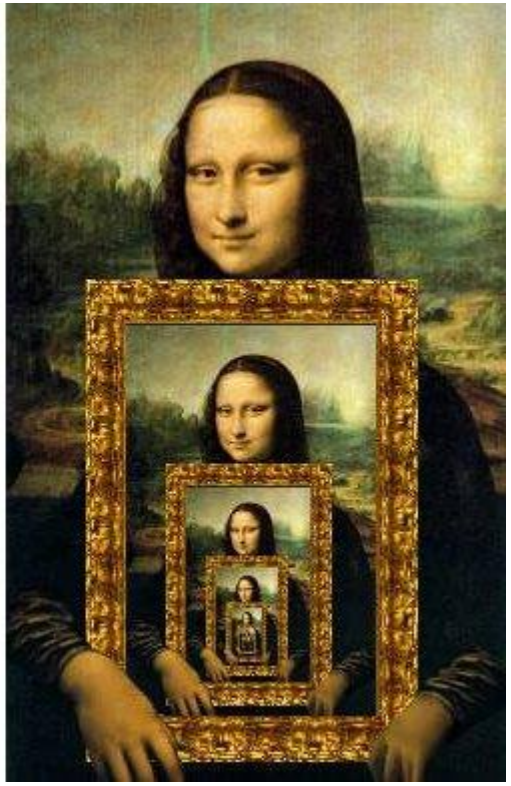
```
def soma_com_default(a, b=5):
    c = a + b
    return str(a) + ' + ' + str(b) + ' = ' + str(c)
print(soma_com_default(7))
print(soma_com_default(7, 7))
print(soma_com_default(b=10, a=12))
```



```
lucas@lucas-In
s/Dropbox/IFPE
7 + 5 = 12
7 + 7 = 14
12 + 10 = 22
```

Funções Recursivas?

RECURSÃO



https://panda.ime.usp.br/pensepy/static/pensepy/12-Recursao/recursionsimple-ptbr.html

Exercícios

1. Faça um programa, com uma função que necessite de três argumentos, e que forneça a soma desses três argumentos.
2. Faça um programa, com uma função que necessite de um argumento. A função retorna o valor de caractere ‘P’, se seu argumento for positivo, e ‘N’, se seu argumento for zero ou negativo.
3. Faça um programa para imprimir de acordo com a imagem abaixo para um n informado pelo usuário. Use uma função que receba um valor n inteiro e imprima até a n-ésima linha.

Exercícios

4. Reverso do número. Faça uma função que retorne o reverso de um número inteiro informado. Por exemplo: 127 -> 721.
5. Faça uma função que retorne a quantidade de dígitos de um determinado número inteiro informado pelo usuário. Não realize conversão de tipos.
6. Faça uma função que computa a potência de a^b para valores de a e b informados pelo usuário. Assuma valores de a e b como inteiros e não utilize o operador ** ou funções da biblioteca Math.
7. Utilizando funções, leia um número inteiro e retorne o seu equivalente em numeração romana.

Exercícios de revisão funções.

8. Escreva uma função que receba um número de parâmetros indefinido. Imprima a quantidade de parâmetros recebidos de cada tipo de dado. A função também deve imprimir o maior e o menor valor numérico recebido.
9. Escreva uma função para imprimir o valor absoluto de um número. (obs: utilize apenas operações aritméticas).
10. Escreva uma função que recebe um número como parâmetro e para cada número menor que o parâmetro, a função imprime "Fizz" se o número for múltiplo de três, imprime "Buzz" se o número for múltiplo de cinco, e imprime "FizzBuzz" se o número for múltiplo de três e cinco. Caso o número não seja múltiplo nem de três nem de cinco, ele deve ser impresso.

Exercícios de revisão funções.

11. Faça uma função recursiva que receba um número inteiro positivo N e imprima todos os números naturais de 0 até N em ordem decrescente.
12. A função fatorial duplo é definida como o produto de todos os números naturais ímpares de 1 até algum número natural ímpar N. Assim, o fatorial duplo de 5 é 5!! = 1 * 3 * 5 = 15. Crie uma função recursiva para calcular o fatorial duplo de um número n.
13. Escreva uma função que receba dois números e retorne True se o primeiro número for múltiplo do segundo. Valores esperados: múltiplo(8, 4) = True; múltiplo(7, 3) = False; múltiplo(5, 5) = True.
14. Escreva uma função que recebe como entrada um número n e imprime todas as potências de 2 menores ou iguais a n.

Exercícios de revisão funções.

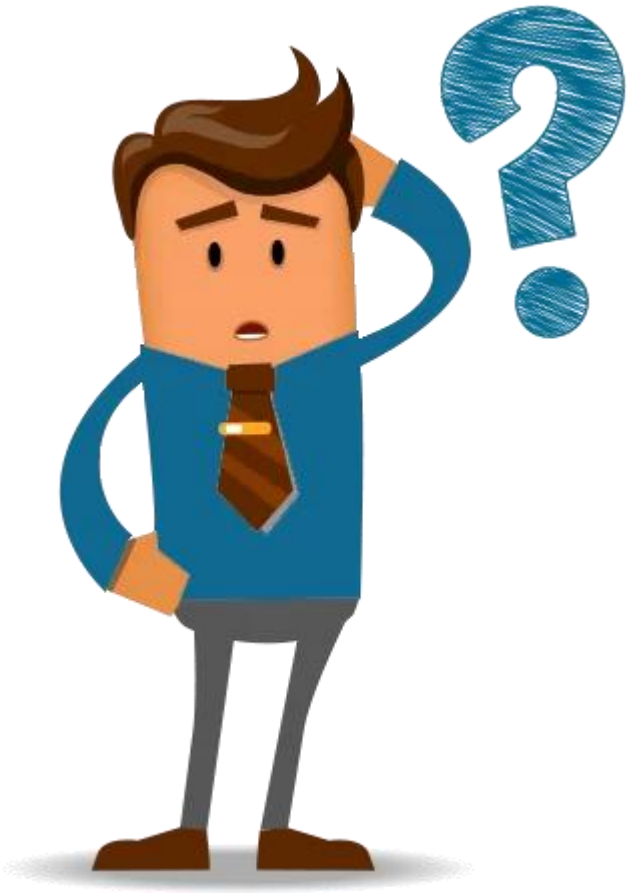
15. Crie uma função geradora da sequência de Fibonacci até o n-ésimo termo.
16. Escreva uma função que recebe como entrada um número ano e retorna True caso ano seja bissexto. Caso contrário, retorne False.

    Help: https://learn.microsoft.com/pt-br/office/troubleshoot/excel/determine-a-leap-year

Resolução de todos os exercícios trabalhados

https://github.com/lucassampaioleite/python-exercises

Dúvidas???



Fonte: https://institutoseculoxxi.com.br/duvidas-entramos-em-contato-com-voce/