MODULARIZAÇÃO DE ALGORITMOS

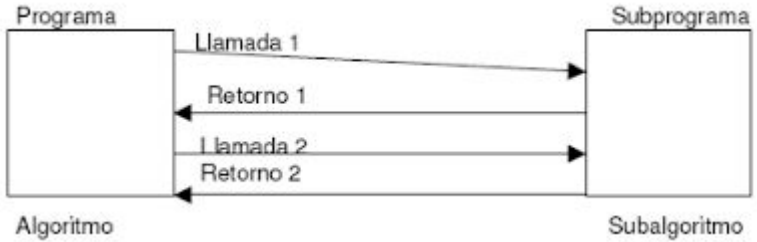
Slide baseado no trabalho do Professor Ailton Cruz, Algoritmo e Estrutura de Dados I

Sumário

- Subalgoritmos
  - Módulos
  - Definição - parâmetros, retorno
  - Variáveis locais e variáveis globais
  - Imports

Subalgoritmo

- Pequeno bloco de código inserido no algoritmo principal
- Ocorre mediante a ativação (chamada)
- Pode ou não retornar uma resposta



https://xdoc.mx/preview/programacion-estructurada-5f399a8e37291

Subalgoritmo

Em resumo, os subalgoritmos são importantes:

- a subdivisão de algoritmos complexos, **facilita** o seu entendimento;
- na estruturação de algoritmos, **facilitando**, principalmente, a **detecção de erros** e a documentação de sistemas;
- na **modularização** de sistemas, que facilita a **manutenção** de softwares e **reutilização** de subalgoritmos já implementados.

https://docente.ifrn.edu.br/placidoneto/disciplinas/2014.1/poo/algoritmo05.pdf

Assim ...

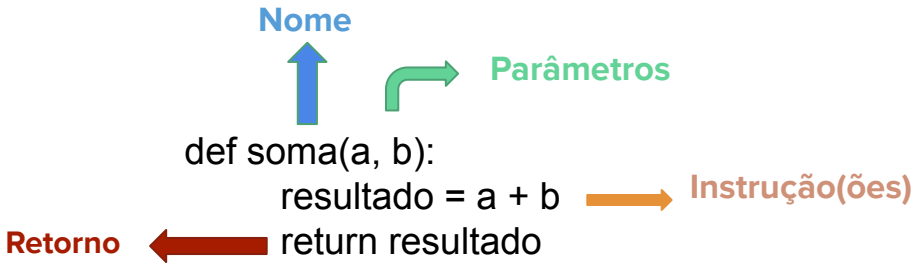
- Divisão do problema em partes;
- Reaproveitamento de código **(Curiosidade, pesquisem sobre DRY)**;
- Facilita na análise e correção;
- Facilita teste de software;
- ...

Slide baseado no tutorial do Professor Ailton Cruz, Algoritmo e Estrutura de Dados I

Elementos de um subalgoritmo

A definição de um **subalgoritmo** consta de:

- **cabeçalho**: onde estão definidos o **nome** [...], os seus **parâmetros**
- **corpo do subalgoritmo**: onde se encontram **as instruções**, **que serão executadas cada vez que ele é chamado**.



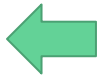
https://docente.ifrn.edu.br/placidoneto/disciplinas/2014.1/poo/algoritmo05.pdf

Parâmetros e Retorno

“Os subalgoritmos **realizam tarefas específicas "devolvendo", ou não, um valor para o algoritmo principal**. Quando um subalgoritmo devolve um valor para o programa principal, este **valor é chamado de retorno**. [...] Um subalgoritmo sem retorno é chamado na forma de um comando propriamente, e é denominado de **procedimento**“

Função

```
def soma(a, b):
    resultado = a + b
    return resultado
```



Procedimento

```
def exibir_mensagem(nome):
    print(f"Olá, {nome}! Bem-vindo.")
```

Slide baseado no tutorial do Professor Ailton Cruz, Algoritmo e Estrutura de Dados I

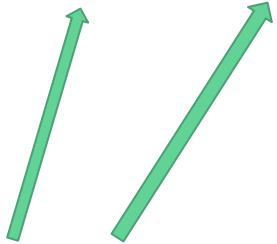
Em resumo ...

```
Defina nome_da_função (parâmetros):
        corpo da função
Fim_função
```

Slide baseado no tutorial do Professor Ailton Cruz, Algoritmo e Estrutura de Dados I

Passando parâmetros ...

```
def minha_funcao(parametro1, parametro2):
    return True
```



```
x = minha_funcao(“a”, “b”)
```

Chamando[Ativando] a função

Slide baseado no tutorial do Professor Ailton Cruz, Algoritmo e Estrutura de Dados I

Passando parâmetros ...

```
def minha_funcao(parametro1, parametro2):
    return True


x = minha_funcao(“a”, “b”)
```

Chamando[Ativando] a função

Slide baseado no tutorial do Professor Ailton Cruz, Algoritmo e Estrutura de Dados I

Passando parâmetros ...

def minha_funcao(**parametro1**, **parametro2**):

    return True

x = minha_funcao**(**“a”, “b”**)**

Chamando[Ativando] a função

Slide baseado no tutorial do Professor Ailton Cruz, Algoritmo e Estrutura de Dados I

Para chamar (ativar) ...

Com retorno

```
def soma(a, b):
    resultado = a + b
    return resultado

total = soma(1,2)
```

Sem retorno

```
def say_hello(name):
    print(f“Hello {name}”

say_hello("Teste")
```

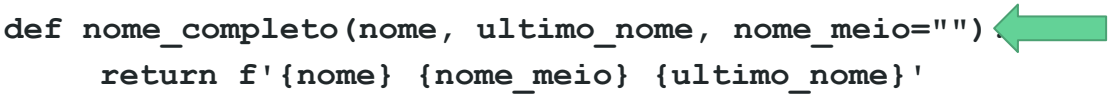
CODE!

Vamos criar a função **soma(x,y) retornando x + y**

Parâmetros opcionais

Posso ter parâmetros opcionais!

“nome_meio”



```
def nome_completo(nome, ultimo_nome, nome_meio=""):
    return f'{nome} {nome_meio} {ultimo_nome}'


nome = nome_completo("a", "b")

print(nome)
```

Boa Práticas

- Nomes de funções devem ser descritivos!!

Nada de: def a(...), def a1(), def b1(), etc

Lembre-se, você está desenvolvendo para outras pessoas também!

Variáveis
Globais x Locais

Variáveis globais (perspectiva de função)

- Variáveis que são criadas fora de uma função;
- Variáveis globais podem ser usadas por todos, tanto dentro das funções quanto fora.

https://www.w3schools.com/python/python_variables_global.asp

```
a ="Hello World" ->>> ESCOPO GLOBAL
def exibir_mensagem():
    print(a)


exibir_mensagem()
```

Variáveis Locais

- Variáveis que são criadas, por exemplo, dentro de uma função;
- Pode ser utilizada dentro da função

https://www.w3schools.com/python/python_variables_global.asp

Variáveis Locais x Globais

```
def exibir_mensagem():
    a ="Hello World"
```

ESCOPO LOCAL

```
exibir_mensagem()
print(a)
```

name 'a' is not defined

Slide baseado no tutorial do Professor Ailton Cruz, Algoritmo e Estrutura de Dados I

Palavra GLOBAL

“Normalmente, quando você **cria uma variável dentro de uma função**, essa variável é local e só pode ser usada dentro dessa função. **Para criar uma variável global dentro de uma função, você pode usar a palavra-chave global**.”

```
def exibir_mensagem():
    global a
    a ="Hello World"

exibir_mensagem()
print(a)
```

https://www.w3schools.com/python/python_variables_global.asp

Importando de outros arquivos/pacotes

Tudo no mesmo arquivo?

https://thewebforbusiness.com/mind-blowing-website-tools/

Criando em outro arquivo e importando

1: **Vamos criar um arquivo chamado calculadora.py**

Dentro do arquivo as funções:

- **somar(numero1, numero2)**
- **dividir(dividendo, divisor)**
- **elevar(base, expoente) -> Expoente deve ser opcional com default 2**
- **subtrair(...)**
- **multiplicar(...)**

Criando em outro arquivo e importando

2: **Vamos importar as funções**

No arquivo main.py:

from calculadora import soma, subtrair, ...

Nosso import ficou grande? E se houver mais operações?

Package:

Conhecendo: __init__.py

O arquivo __init__.py indica ao Python que o **diretório onde ele está localizado deve ser tratado como um pacote**. Isso significa que Python irá reconhecer a pasta como um módulo que pode conter outros módulos ou subpacotes.

https://docs.python.org/pt-br/3/tutorial/modules.html

Criando em outro arquivo e importando

1: Crie uma pasta /utils/

2: Dentro dela crie um arquivo chamado __init__.py

3: Coloque a calculadora.py dentro de /utils/

4: Troque

- from calculadora import somar, subtrair ... por: import utils

Curiosidades Python

Dicas adicionais!

- Em python minha função pode ter 2 retornos

```
def funcao():
    return "retorno um", "retorno dois"
```

Exemplo Prático

```
from datetime import datetime

def criar_registro(nome, telefone):
    objeto = {
        'nome': nome,
        'telefone': telefone
    }
    data_hora_criacao = datetime.now()
    return objeto, data_hora_criacao

objeto_criado, data_criacao = criar_registro('Joao', 'Neto')

print("Objeto Criado:", objeto_criado)
print("Data de Criação:", data_criacao)
```

Dicas adicionais!

- 2015 - Posso dizer o tipo (Typing Hint) - Python 3.5+ (PEP 484)

```
def funcao(parametro1: str, parametro2: str) -> str:
    return parametro1 + parametro2
```

Referências

1. https://educapes.capes.gov.br/bitstream/capes/176522/2/TextoAEDI_SI_UFAL_AiltonCruz.pdf

Slide baseado no trabalho do Professor Ailton Cruz, Algoritmo e Estrutura de Dados I