Estruturas Simples e Funções

Paulo Almeida
André Grégio

JUSTIÇA 4.0: INOVAÇÃO E EFETIVIDADE NA REALIZAÇÃO DA JUSTIÇA PARA TODOS
PROJETO DE EXECUÇÃO NACIONAL BRA/20/015

1.
Listas

Listas

Lista de compras

Precisamos comprar: maçãs, leite, arroz, frango

Listas

Lista de compras

Precisamos comprar: maçãs, leite, arroz, frango

Isso é uma lista!

Em Python

Python possui **nativamente** uma **estrutura de dados** para listas

Para criar uma lista em Python:

```
nome_lista = [item1, item2, item3, …]
```

Questão de Ordem!

Uma lista é uma coleção de elementos **em uma ordem particular.**

Os itens vão se manter na ordem que você definiu.

Exceto se você explicitamente pedir para que a ordem seja alterada

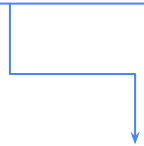
Exemplo

```
lista_compras = ["maçãs", "leite", "arroz", "frango"]
```

Exemplo

Cada item da lista é uma string. Então não esqueça das aspas.

```
lista_compras = ["maçãs", "leite", "arroz", "frango"]
```



Boa prática: como listas possuem múltiplos itens, utilize um nome no plural!

Iterando

Existem **várias formas** para acessar (iterar) os itens da lista

Uma simples é usando uma versão especial do loop *for*

```
for nome_variavel in nome_lista:
    faz algo com nome_variavel
```

Exemplo

```
lista_compras = ["maçãs", "leite", "arroz", "frango"]
for item in lista_compras:
    print("Preciso comprar:", item)
```

Índices

Uma lista nada mais é do que um vetor indexado

E um nome melhor para as listas do Python talvez fosse vetor

Veja uma breve explicação aqui: docs.python.org/3/faq/design.html#how-are-lists-implemented-in-cpython

Toda lista é indexada.

O primeiro elemento é o 0

O segundo elemento é o 1

O terceiro elemento é o 2...

Acesso por índice

Você pode acessar a lista pelos seus índices. Para isso utilize o operador []

```
lista_compras = ["maçãs", "leite", "arroz", "frango"]

print("O primeiro item que preciso comprar é:",
lista_compras[0])
```

Qual o tamanho da lista?

A função len() retorna o tamanho de uma lista

```
lista_compras = ["maçãs", "leite", "arroz", "frango"]

tamanho = len(lista_compras)
print("A lista de compras possui:", tamanho, "itens")
```

Outra forma de iterar

Dessa forma, podemos iterar também de outras formas em uma lista:

```
for i in range(len(lista_compras)):
     print("O elemento", i, "da lista é", lista_compras[i])
```

Ou usando um loop while:

```
i = 0
while i < len(lista_compras):
     print("O elemento", i, "da lista é", lista_compras[i])
     i = i + 1
```

Intervalos

Podemos acessar um intervalo da lista utilizando [inicio:fim] onde

Início é o índice do início da sublista, que é incluído na sub lista
Fim é o índice de fim da sub lista, que não é incluído na sub lista

Intervalos

Podemos acessar um intervalo da lista utilizando [inicio:fim] onde

Início é o índice do início da sublista, que é incluído na sub lista
Fim é o índice de fim da sub lista, que não é incluído na sub lista

Exemplo

```
lista_compras = ["maçãs", "leite", "arroz", "frango", "macarrão"]
sublista = lista_compras[1:4]
for item in sublista:
     print(item)

print("Outra forma")
for item in lista_compras[2:6]:
     print(item)
```

Alterando

Você pode alterar o valor de um elemento da lista, acessando-o via []. Veja um exemplo:

```
lista_compras = ["maçãs", "leite", "arroz", "frango"]
lista_compras[1] = "açúcar"

for item in lista_compras:
     print(item)
```

2.
Funções de listas

Funções de listas

Existem diversas funções que podem ser usadas com listas

Vamos ver apenas algumas

Veja uma lista completa em
docs.python.org/pt-br/3/tutorial/datastructures.html

Índices e contagens

```
lista_compras = ["maçãs", "leite", "arroz", "frango", "leite", "trigo"]

qtde_vezes = lista_compras.count("leite")
idx = lista_compras.index("leite")

print("Quantidade: ", qtde_vezes)
print("Primeiro idx: ", idx)
```

Índices e contagens

```
lista_compras = ["maçãs", "leite", "arroz", "frango", "leite", "trigo"]

qtde_vezes = lista_compras.count("leite")  Quantas vezes “leite” aparece na lista?
idx = lista_compras.index("leite") Qual o primeiro índice de “leite” na lista?

print("Quantidade: ", qtde_vezes)
print("Primeiro idx: ", idx)
```

Ordenando

Você pode ordenar os elementos da lista usando sort()

```
lista_compras = ["maçãs", "leite", "arroz", "frango", "trigo"]
for item in lista_compras:
    print(item)

print("Ordenado")
lista_compras.sort()
for item in lista_compras:
    print(item)

print("Ordenado Decrescente")
lista_compras.sort(reverse = True)
for item in lista_compras:
    print(item)
```

Excluindo por índice

Para excluir um elemento da lista por índice, utilize `del nome_lista[idx]`

```
lista_compras = ["maçãs", "leite", "arroz", "frango", "trigo"]

del lista_compras[1]

for item in lista_compras:
        print(item)
```

Excluindo por valor

A função remove(conteúdo) remove o primeiro elemento com o conteúdo especificado

A função pop() remove o último elemento

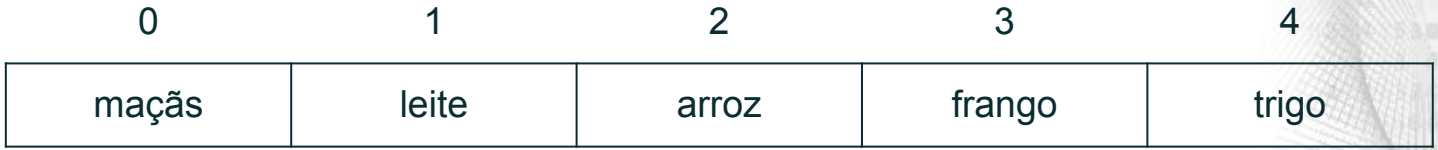
```
lista_compras = ["maçãs", "leite", "arroz", "frango", "trigo"]

lista_compras.remove("arroz")
lista_compras.pop()

for item in lista_compras:
     print(item)
```

Inserindo

`insert(idx, item)` insere o item na posição

`append(item)` insere o item no final

```
lista_compras = ["maçãs", "leite", "arroz", "frango", "trigo"]

lista_compras.insert(2, "feijão")
lista_compras.append("tomate")
```

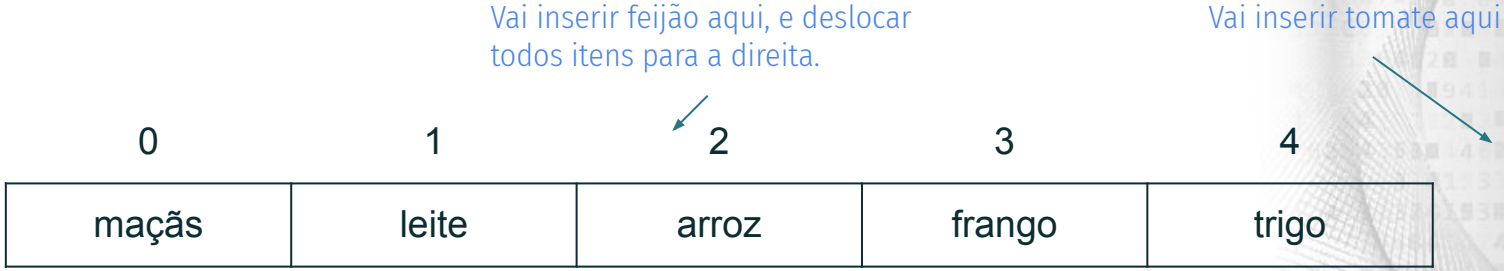


Inserindo

`insert(idx, item)` insere o item na posição

`append(item)` insere o item no final

```
lista_compras = ["maçãs", "leite", "arroz", "frango", "trigo"]

lista_compras.insert(2, "feijão")
lista_compras.append("tomate")
```



Exemplo completo

```
lista_compras = [] #uma lista vazia
item = input("Digite um item ou sair: ")
while item != "sair":
    lista_compras.append(item)
    item = input("Digite um item ou sair: ")

for it in lista_compras:
    print(it)
```

Listas são heterogêneas

Em python, uma lista é heterogênea

Pode misturar todo tipo de item

Ter inteiros, strings, outras listas, ...

Listas são heterogêneas

Em python, uma lista é heterogênea

Pode misturar todo tipo de item

Ter inteiros, strings, outras listas, ...

Exemplo

```
lista_inteiros = [1,2,3]

elementos = ["casa", 1, "banana", lista_inteiros]

for item in elementos:
    print(item, type(item))
```

Arrays

Python provê arrays (vetores) homogêneos

Operam de maneira similar a lista mas em um **array, todos elementos precisam ser do mesmo tipo**

(Muito) Mais eficiente do ponto de vista computacional

Veja mais sobre arrays em docs.python.org/3/library/array.html

3.
Tuplas

Tuplas

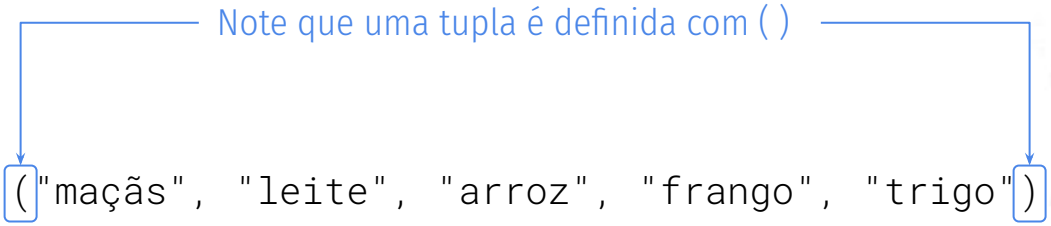
Tuplas em Python são similares a listas, no entanto tuplas são **imutáveis**

Depois de criada uma tupla, você não pode fazer modificações

Exemplo

```
tupla_compras = ("maçãs", "leite", "arroz", "frango", "trigo")

for item in tupla_compras:
    print(item)

print("O item 1 é", tupla_compras[1])
```

Exemplo




```
tupla_compras = ("maçãs", "leite", "arroz", "frango", "trigo")

for item in tupla_compras:
    print(item)

print("O item 1 é", tupla_compras[1])
```

Exemplo

```
tupla_compras = ("maçãs", "leite", "arroz", "frango", "trigo")

for item in tupla_compras:
    print(item)

print("O item 1 é", tupla_compras[1])
```

```
tupla_compras[0] = "tomate"
```

TypeError: 'tuple' object does not support item assignment

Você não pode modificar uma tupla!

Quando usar

Sempre que você precisar de uma “lista” e garantir que ela não deve ser modificada, use tuplas.

Garantir que os itens não sejam modificados por acidente.

Tuplas são mais eficientes do que listas.

Por curiosidade...

```
import sys

lista = ["maçãs", "leite", "arroz", "frango", "trigo"]
tupla = ("maçãs", "leite", "arroz", "frango", "trigo")

print("Tamanho na memória da tupla:", sys.getsizeof(tupla))
print("Tamanho na memória da lista:", sys.getsizeof(lista))
```

Tamanho na memória da tupla: 80
Tamanho na memória da lista: 96

Mais sobre tuplas

Veja mais sobre tuplas na bibliografia recomendada, e em docs.python.org/3/tutorial/datastructures.html#tuples-and-sequences

4.
Funções

Funções

Estamos usando funções o tempo todo!

print() é uma função
math.sqrt() é uma função
input() é uma função
...

Funções

Uma função é uma sequência de instruções para realizar uma dada tarefa

Conceito similar a uma função matemática

Funções

Uma função é uma sequência de instruções para realizar uma dada tarefa

Conceito similar a uma função matemática

Podemos criar nossas próprias funções!

Funções

Uma função é uma sequência de instruções para realizar uma dada tarefa

Conceito similar a uma função matemática

Podemos criar nossas próprias funções!

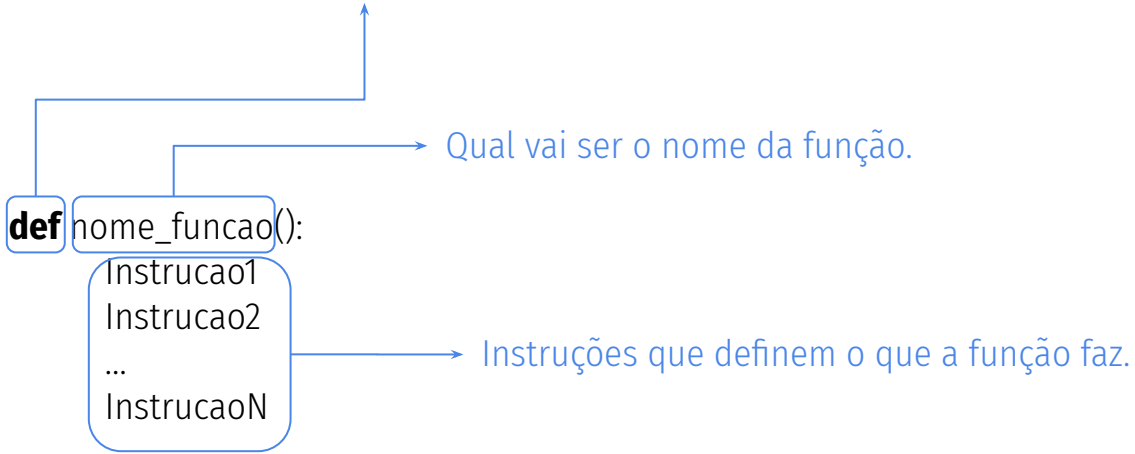
Depois de criadas, podemos usá-las em qualquer lugar do programa, sem precisar fazer control+c / control+v de código

Formato

```
def nome_funcao():
        Instrucao1
        Instrucao2
        ...
        InstrucaoN
```

Formato

Palavra reservada def indica que estamos criando uma função.



Exemplo

```
def saudacao():
    print("Ola, bem vindo.")
    print("Esse é o curso de Python.")


saudacao()
```

Exemplo

```
def saudacao():
    print("Ola, bem vindo.")
    print("Esse é o curso de Python.")
```

```
saudacao()
```

→ Chamando a função

Parâmetros

Uma função comumente possui parâmetros

Passamos esses parâmetros para a função, indicando o que ela precisa fazer

Formato

```
def nome_funcao(nome_par1, nome_par2, ...):
    Instrucao1
    Instrucao2
    ...
    InstrucaoN
```

Exemplo

```
def saudacao(nome, periodo):
    if (periodo == 'm'):
        print("Bom dia,", nome + '.')
    elif(periodo == 't'):
        print("Boa tarde,", nome + '.')
    elif(periodo == 'n'):
        print("Boa noite,", nome + '.')
    else:
        print("Ops, período inválido.")

    print("Esse é o curso de Python.")


saudacao("Paulo", 'n')
```

Valores default

Os parâmetros da função podem ter valores *default* (padrão)

Ao chamar a função, caso o programador não passe esse parâmetro, o interpretador vai assumir o parâmetro *default*

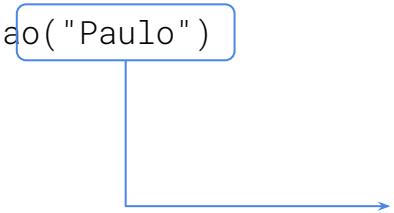
Para adicionar um valor padrão ao parâmetro, basta usar `nome_par = valor_padrao`

Exemplo

```
def saudacao(nome, periodo = 'm'):
     if (periodo == 'm'):
          print("Bom dia,", nome + '.', "Essa é a saudação padrão.")
     elif(periodo == 't'):
          print("Boa tarde,", nome + '.')
     elif(periodo == 'n'):
          print("Boa noite,", nome + '.')
     else:
          print("Ops, período inválido.")

     print("Esse é o curso de Python.")


saudacao("Paulo")
```

Exemplo

```
def saudacao(nome, periodo = 'm'):
     if (periodo == 'm'):
          print("Bom dia,", nome + '.', "Essa é a saudação padrão.")
     elif(periodo == 't'):
          print("Boa tarde,", nome + '.')
     elif(periodo == 'n'):
          print("Boa noite,", nome + '.')
     else:
          print("Ops, período inválido.")

     print("Esse é o curso de Python.")


saudacao("Paulo")
```



Não foi definido um valor para período na chamada da função. Então o valor default 'm' será usado.

Fatorial

```
def fatorial(n):
    res = 1
    while(n > 1):
        res = res*n
        n = n -1
    print("O fatorial é", res)



valor = int(input("Digite um valor ou -1 para sair: "))
while(valor != -1):
    fatorial(valor)
    valor = int(input("Digite um valor ou -1 para sair: "))
```

5.
Retorno e boas práticas

Fatorial

A função fatorial imprime o resultado

Isso (geralmente) é uma má prática. Idealmente, essa função deveria **retornar** (devolver) o resultado da conta.

Quem chamou a função decide o que fazer com esse resultado.

```
def fatorial(n):
    res = 1
    while(n > 1):
        res = res*n
        n = n -1
    print("O fatorial é", res)
```

return

Para retornar um resultado, utilize a palavra chave **return** dentro de uma função

Uma função retorna (termina) imediatamente ao encontrar um return.

Formato

```
return valor_a_ser_retornado
```

Fatorial melhorado

```
def fatorial(n):
        if (n < 0):
                return -1 #retornando -1 para indicar um erro
        res = 1
        while(n > 1):
                res = res*n
                n = n -1
        return res

valor = int(input("Digite um valor ou -1 para sair: "))
while(valor != -1):
        resultado = fatorial(valor)
        if (resultado != -1):
                print("O fatorial de", valor, "é", resultado)
        else:
                print("Impossível calcular o fatorial de", valor)
        valor = int(input("Digite um valor ou -1 para sair: "))
```

Funções - O caminho completo

Uma função então

Pode receber **entradas** (parâmetros) indicando o que fazer

Faz algum **processamento**, possivelmente envolvendo os parâmetros

Gera uma possível **saída**, na forma de um retorno

Cópias e Referências

Variáveis simples (e.g., ints e floats) e também strings são passadas por **cópia**

Isso significa que se a função alterar o parâmetro, a variável original não vai ser **alterada**

```
def altera_simples(val_int, val_float, texto):
    val_int =  val_int + 1
    val_float = val_float + 1
    texto = texto + " teste"
    print("Dentro da função: ", val_int, val_float, texto)

meu_int = 10
meu_float = 3.14
minha_str = "aula"
altera_simples(meu_int, meu_float, minha_str)
print("Originais: ", meu_int, meu_float, minha_str)
```

```
Dentro da função:  11 4.140000000000001 aula teste
Originais:  10 3.14 aula
```

Cópias e Referências

Objetos compostos, como listas e arrays, são passados por **referência**

Alterar o objeto na função também **altera o original**

```
def altera_lista(lista):
    lista[0] = 0
    lista[1] = 1

def imprime_lista(lista):
    for item in lista:
        print(item)

minha_lista = [3,4,5]
print("Antes de alterar")
imprime_lista(minha_lista)
altera_lista(minha_lista)
print("Depois de alterar")
imprime_lista(minha_lista)
```

```
Antes de alterar
3
4
5
Depois de alterar
0
1
5
```

Boas práticas

Para evitar que seu programa vire uma salada, é uma boa prática criar arquivos separados para suas funções

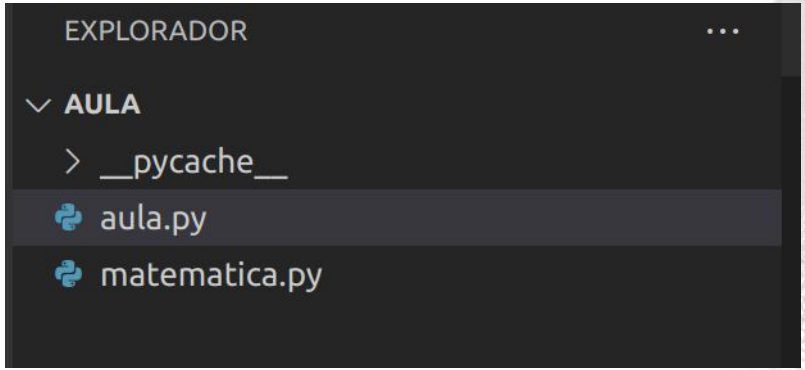
Crie os arquivos normalmente, como os criados até agora, com a extensão .py

Você vai aumentar a **modularização** do seu programa.

Exemplo

Crie um arquivo chamado matematica.py

Coloque a função fatorial dentro desse arquivo e salve



Importando

Para importar a sua função para outro arquivo, basta fazer no início

```
import matematica
```

Para usar as funções definidas dentro desse arquivo, faça

```
matematica.nome_funcao()
```

Exemplo

```
import matematica

valor = int(input("Digite um valor ou -1 para sair: "))
while(valor != -1):
    resultado = matematica.fatorial(valor)
    if (resultado != -1):
        print("O fatorial de", valor, "é", resultado)
    else:
        print("Impossível calcular o fatorial de", valor)
    valor = int(input("Digite um valor ou -1 para sair: "))
```

É isso...

Agora você pode criar suas próprias funções, organizá-las em um arquivo, e usar esse arquivo em vários projetos, ou ainda repassá-lo para seus colegas.

6.
Teste seus conhecimentos

Executando o script

1. Replique tudo que foi ensinado durante as aulas no seu computador para fixar os conhecimentos.
2. Melhore o exemplo da lista de compras dado em aula. O programa deve solicitar novos itens do usuário até ele digitar “sair”. Se o usuário digitar “remover”, o programa deve perguntar qual o item ele quer remover da lista, e removê-lo. Se o usuário digitar “substituir”, o programa deve perguntar o nome do item a ser substituído, e substituí-lo por outro digitado pelo usuário.

Obrigado!

Bons Estudos!!!