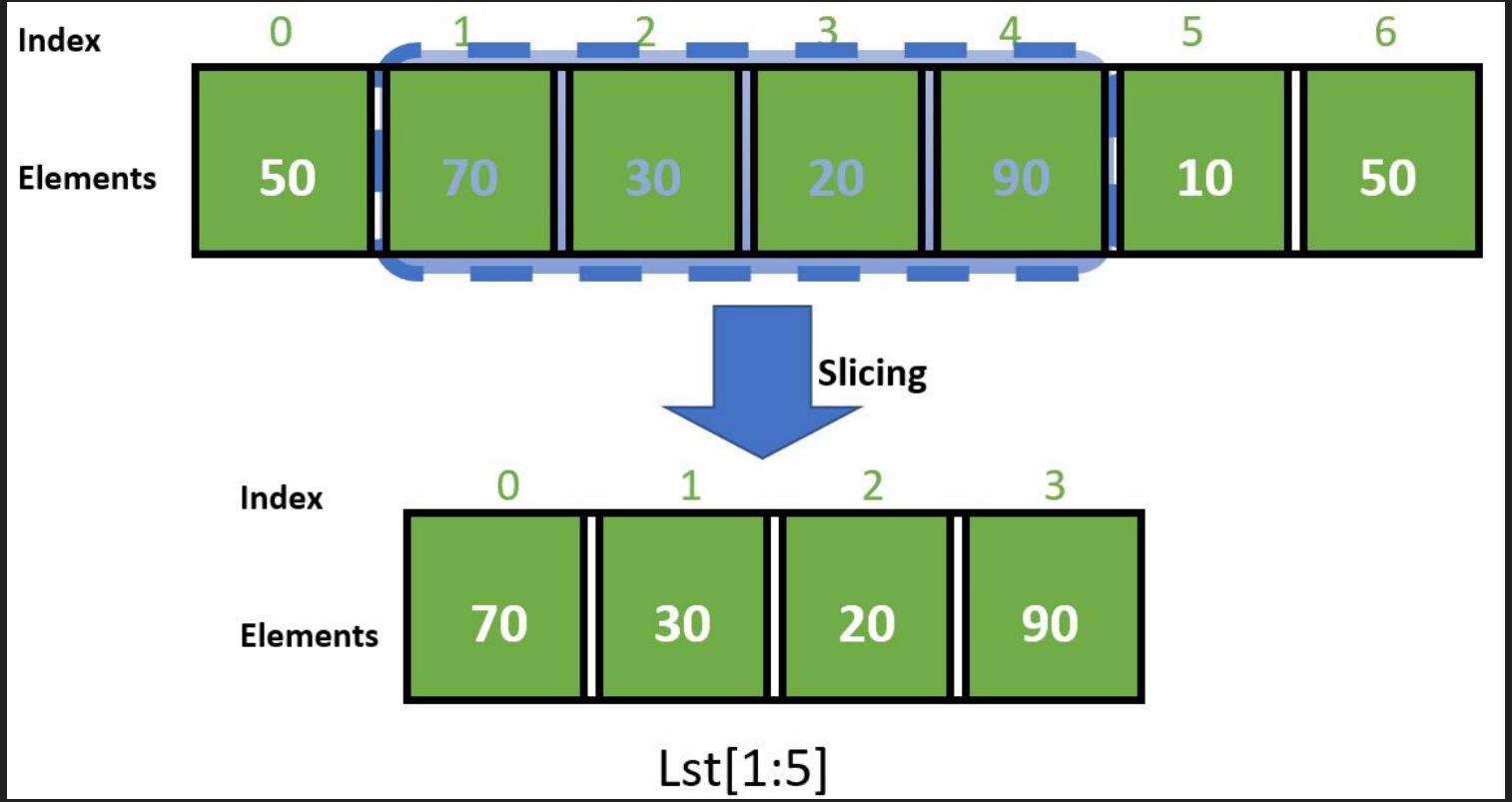
SSC0800 - Introdução à Ciência de Computação I

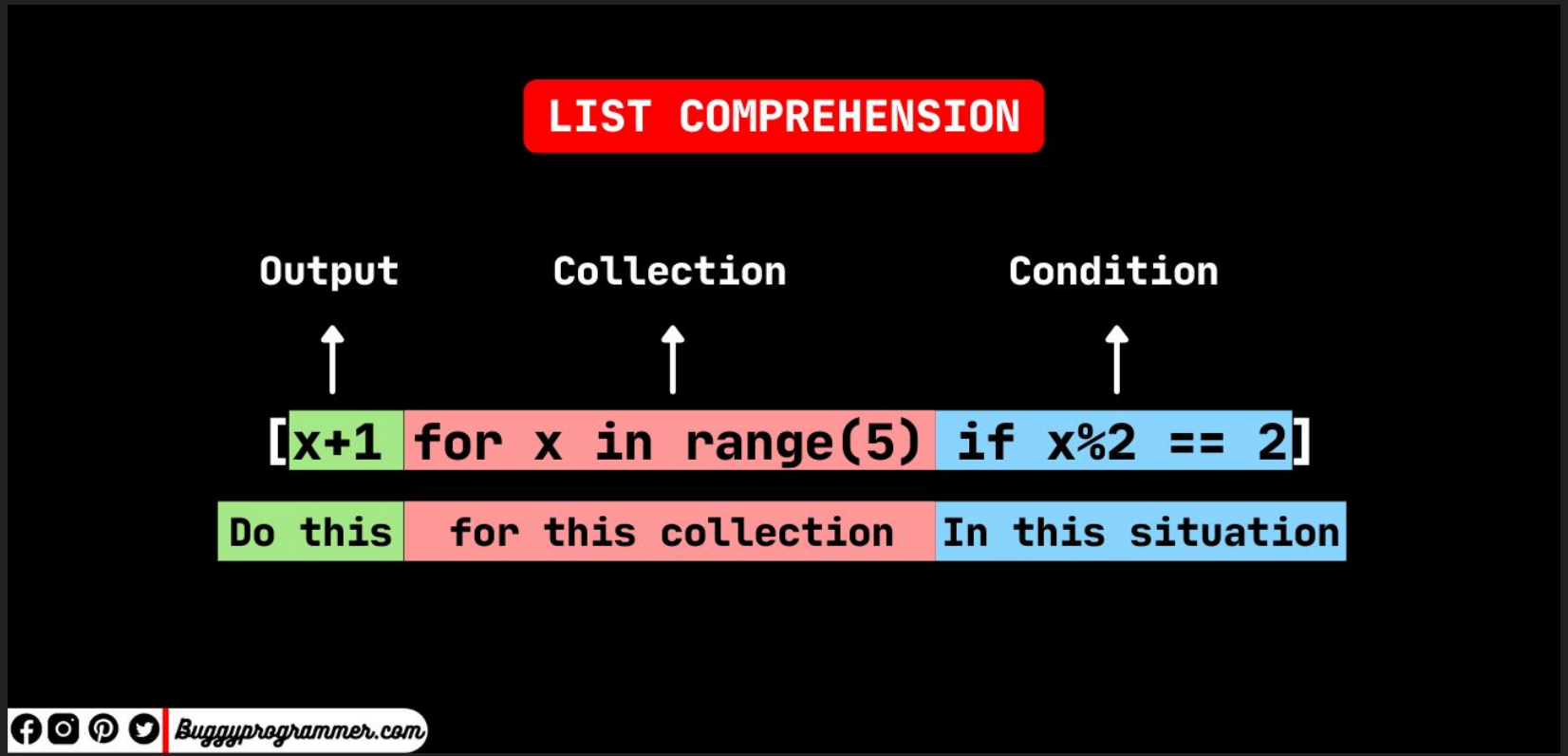
Funções

Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br
Baseado no material do prof Cesar Henrique Comin

Na aula passada...



Fonte: https://www.geeksforgeeks.org/python-list-slicing/



Fonte: https://www.freecodecamp.org/news/list-comprehension-in-python/

`namedtuple()`	factory function for creating tuple subclasses with named fields
`deque`	list-like container with fast appends and pops on either end
`ChainMap`	dict-like class for creating a single view of multiple mappings
`Counter`	dict subclass for counting hashable objects
`OrderedDict`	dict subclass that remembers the order entries were added
`defaultdict`	dict subclass that calls a factory function to supply missing values
`UserDict`	wrapper around dictionary objects for easier dict subclassing
`UserList`	wrapper around list objects for easier list subclassing
`UserString`	wrapper around string objects for easier string subclassing

Fonte: https://docs.python.org/3/library/collections.html

O que vamos aprender hoje?

Objetivos

- ➔ Conhecer o conceito de funções
- ➔ Entender sua sintaxe
- ➔ Entender o que é uma pilha de chamadas
- ➔ Estudar exemplos de uso

Tópicos da Aula

- Turtle
- Funções
-

Turtle

Turtle

- Python turtle
  - Um módulo para movimentar uma tartaruga pela tela
- No colab, é preciso usar um módulo próprio

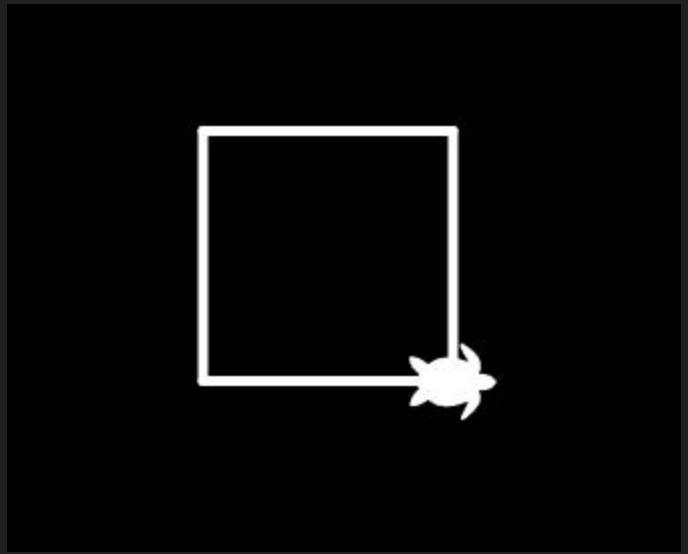
Turtle

```
!pip3 install ColabTurtle # instalar pacote
from ColabTurtle.Turtle import * # importar pacote
initializeTurtle() # inicializar módulo
forward(valor) # mover para frente
left(valor) #virar para esquerda
right(valor) # virar para direita
```

Fazendo um Quadrado

```
from ColabTurtle.Turtle import *
square_size = 100
initializeTurtle()
forward(square_size)
left(90)
forward(square_size)
left(90)
forward(square_size)
left(90)
forward(square_size)
```

Turtle

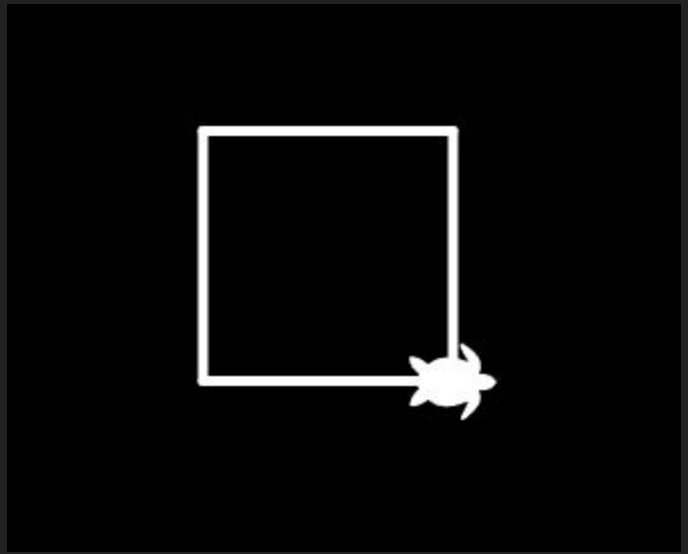


Fazendo um Quadrado - For

```
from ColabTurtle.Turtle import *

square_size = 100
initializeTurtle()
for i in range(4):
  forward(square_size)
  left(90)
```

Turtle



Reposicionando (sem desenhar)

```
from ColabTurtle.Turtle import *
square_size = 100
initializeTurtle()
penup()
setpos(200, 120)
pendown()
for i in range(4):
  forward(square_size)
  left(90)
```

Exemplo

Funções

Funções

- E se eu fizesse um código que sempre desenha o quadrado
  - E só preciso passar as variáveis principais?
- Isso é uma função!

Funções

- Grupo de declarações com um nome, e que pode ser chamado de algum local do programa.
- Permite estruturar programas em segmentos de códigos que fazem tarefas individuais
- Sintaxe:
  - def nome (parametro1, parametro2, ...):
  - declarações
  - return valor

Funções

- O tipo de retorno da função é dinâmico
  - Não é preciso declarar
- Nome
  - Identificador pelo qual a função é chamada

Funções

- *Parâmetros (quantos forem necessários)*
  - Cada um tem seu nome
  - Alguns podem ter um valor padrão definido (opcionais)
  - Agem como variáveis locais à função
  - Passa argumentos às funções do local em que foram chamadas - Ordem importa!
  - Normalmente, tem um “_” no começo

Funções

- *Declarações*
  - Corpo da função
  - Bloco de declarações que especificam o que a função faz

Exemplo

Funções

- Para chamar uma função, é preciso usar seu nome e valores (ou variáveis com tais valores) correspondentes a seus parâmetros
  - Caso ela retorne um valor, é desejável receber esse retorno (ou realizar alguma operação com ele)

Funções

- ➔ Em python, é possível passar valores diretamente para cada parâmetro dando seu nome explicitamente!
- ➔ Também é possível dar valores “padrões” para variáveis

Funções

- A palavra-chave *return* indica o fim da função, retornando o controle de volta ao ponto na qual ela foi chamada
- Não é preciso “receber” o retorno em uma variável! Ele pode ser usado diretamente como parâmetro de uma função, por exemplo.

Exemplo

Funções

- A função *main* é sempre chamada primeiro
  - O código de qualquer outra função só é executado se for chamado pela *main*
    - Direta ou indiretamente
  - Em Python, a main é opcional, mas pode existir
    - É obrigatória para scripts compilados e módulos

Funções

- Ao chamar uma função, a execução da função que a chamou (e.g.: *main*) é parado e o controle é passado à função chamada.
  - O controle é retornado após a função terminar
- No caso da passagem de argumentos por parâmetros, eles são copiados para as variáveis locais da função na hora da chamada
  - Em nosso exemplo: *a* e *b*

Funções

- Funções devem, preferencialmente, ser declaradas no início do código
- E, então, serem usadas na main
  - Ou em funções que a main chama
    - Mas que ficam mais “abaixo” no código
- Funções podem chamar outras funções!
- E elas são executadas apenas quando chamadas

IMPORTANTE!

- ➔ Funções são chamadas com parênteses ()
- ➔ Elementos de coleções são chamados com colchetes []

Exemplo

Funções

- Funções complexas devem ser documentadas
- Porém!!!
  - Um bom nome substitui qualquer documentação para códigos limpos
  - Comentários em funções simples e bem nomeadas pode ser redundante
    - E ainda ficar desatualizado e trazer erros de entendimento

Funções

- Falando em nomes...
- Funções e variáveis não devem ter o mesmo nome
  - Função é substituída pela variável
- Caso vá documentar, prefira docstring
  - Adicionar string com 3 aspas no começo

Funções

- Solução mais interessante:
  - Type hinting (Python 3.5+)
- def hello_name(name: str) -> str:

  return(f"Hello {name}")

Funções

- Para forçar tipos
  - Mypy (Python 3.8+)
  - Biblioteca que funciona como um compilador para reforçar tipagem no Python
  - Checador estático de tipos

Exemplo

Referências

Referências

1. https://www.learnpython.org/
2. https://www.w3schools.com/python/
3. https://panda.ime.usp.br/cc110/static/cc110/index.html
4. https://www.youtube.com/playlist?list=PLcoJJSvnDgcKpOi_UeneTNTIVOigRQwcn
5. https://mypy.readthedocs.io/en/stable/getting_started.html#getting-started
6. https://mypy-lang.org/
7. https://docs.python.org/3/library/typing.html
8. https://www.treinaweb.com.br/blog/tipagem-no-python-com-type-hints