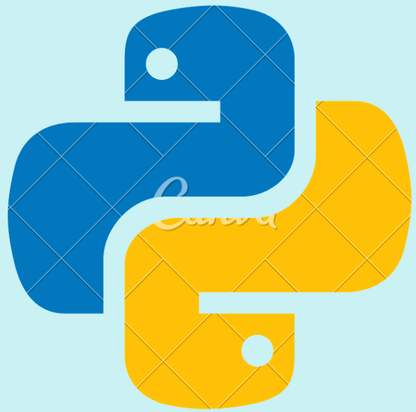
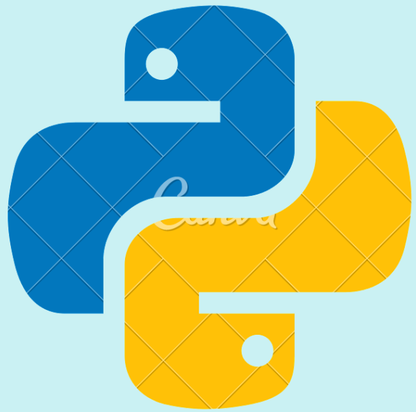


Minicurso de Python

da sintaxe básica à aplicação em processamento de imagens e visão computacional usando OpenCV

Profa. Daniela Costa Terra
Prof. Luiz Maurílio da Silva Maciel



Minicurso de Python

Módulo 1: Sintaxe Básica

Profa. Daniela Costa Terra
Prof. Luiz Maurílio da Silva Maciel

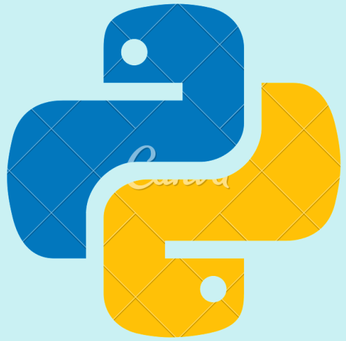
Linguagem Python

- Origem:

  Concebida por Guido van Rossum, programador e escritor holandês, criou a linguagem Python em 1996

  Guido nomeou a linguagem por ser fâ da série cômica da BBC "Monty Python's Flying Circus"

  Mantida atualmente pela Python Software Foundation, em um processo comunitário



Linguagem Python

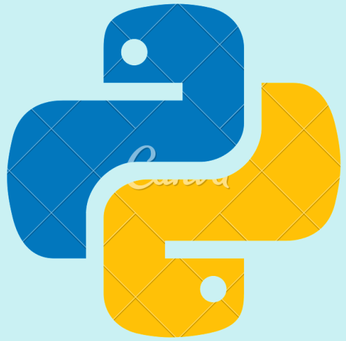
- Instalação do Python 3:

  Pode ser baixado de: https://www.python.org/downloads

  Para windows, vá em: https://www.python.org/downloads/windows/ e instale para a versão mais atual: "Windows x86-64 executable installer"

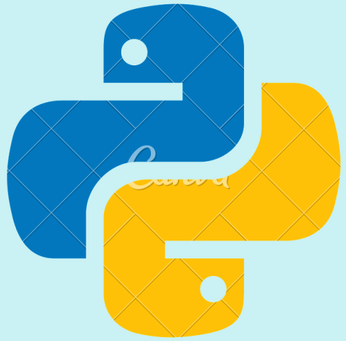
  Para usar o Python a partir da linha de comando:

  - durante a instalação marque uma das opções: "AddPython 3.7 PATH" ou "Add Python to environment variables"
  - após a instalação configura a variável de ambiente "Path" do Windows



Linguagem Python

- Simples e fácil de aprender
- Orientada a Objetos
- FLOSS (free and open source software)
- Extensas bibliotecas
- Portável
- Interpretada
- Extensível e embutível



Multi-paradigma

- Python suporta construções nos paradigmas de programação:

  Imperativa (estruturada)

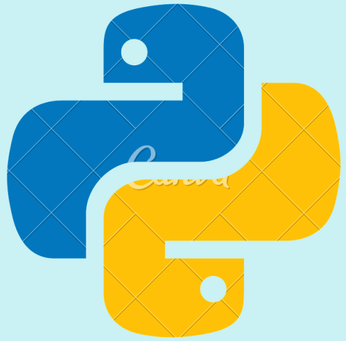
  - Funções, estruturas de controle, módulos •

  Orientada a objetos

  - Classes, objetos

  Funcional

  - Manipulação de listas



Multi-propósito

- Python pode ser usado para definir aplicações para:

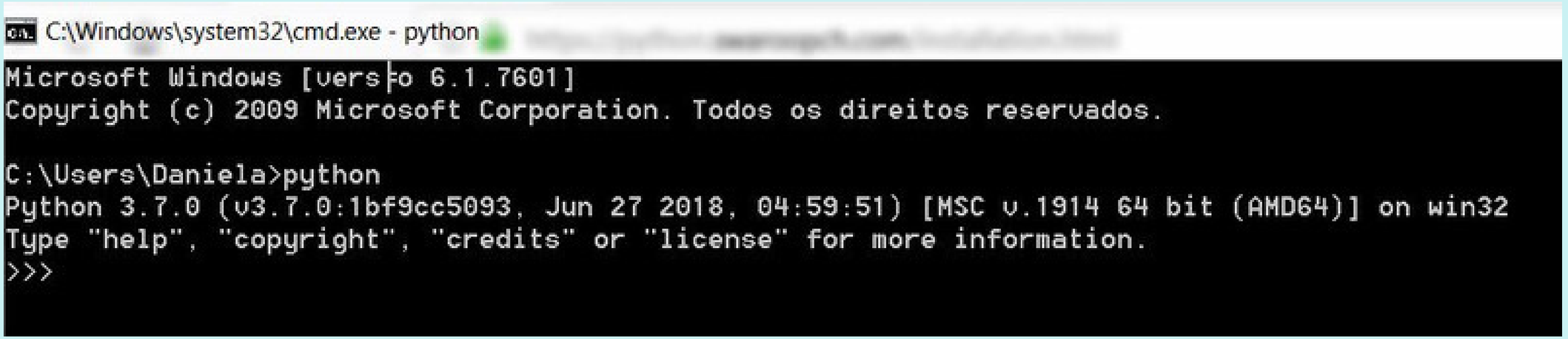
  Aplicativos desktop

  Aplicativos web (Django, Grok, etc.)

  Web Services

- A linguagem é usada hoje no Youtube, Google, Globo.com, etc.

Python no modo interativo



- Após o prompt ">>>" basta digitar sentenças Python seguidas de <ENTER>

Python no modo interativo

- Use o help(comando) para tirar suas dúvidas sobre Python

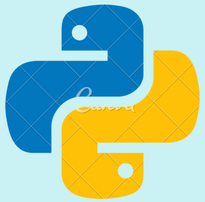
```
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.

>>> print("um", "dois", "tres", sep=':')
um:dois:tres
>>>
```

- Lembre-se que Python é case sensitive : 'print' é diferente de 'Print'

Python no modo interativo

```
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.

>>> print("um", "dois", "tres", sep=':')
um:dois:tres
>>>
```

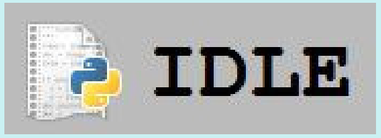
Saia do prompt com exit() ou ctrl+Z seguindo de <ENTER>



Codificando um programa Python

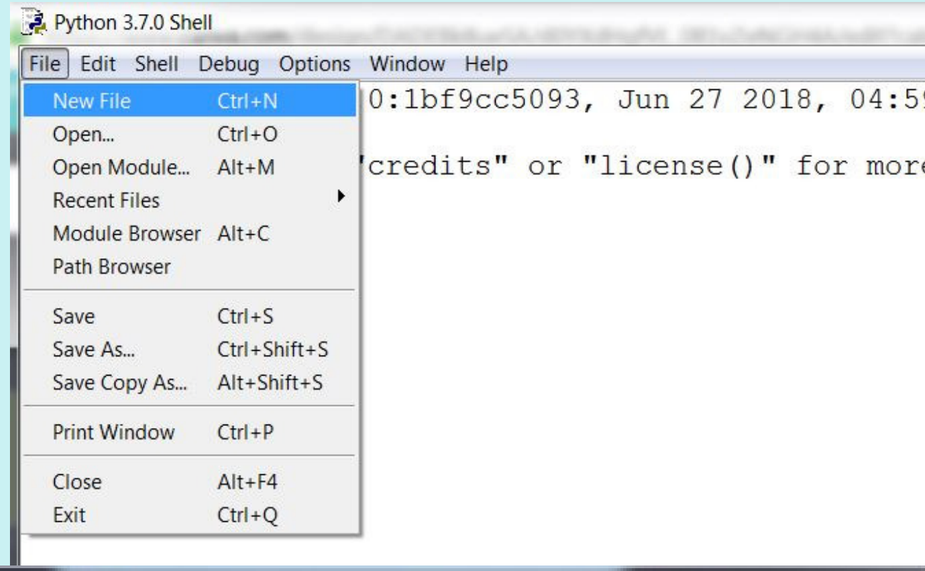
- Não faz sentido redigitar seu programa no modo interativo sempre que desejar executá-lo
- Salve seu arquivo fonte Python com extensão .py em qualquer editor de texto

Há inúmeros editores de código e ambientes de programação (IDE) para Python, para citar alguns:



Programando no Idle

- Inicie o IDLE e crie um novo arquivo...



- Digite o programa abaixo

Untitled

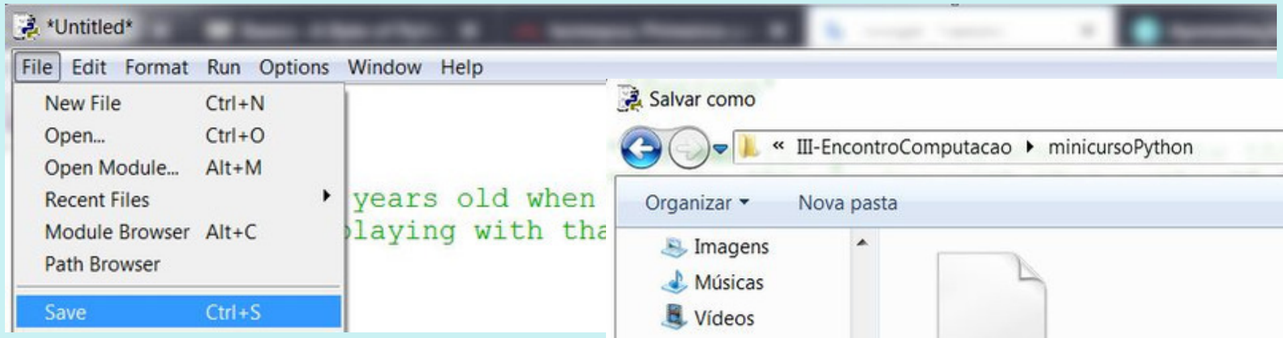
File Edit Format Run Options Window Help

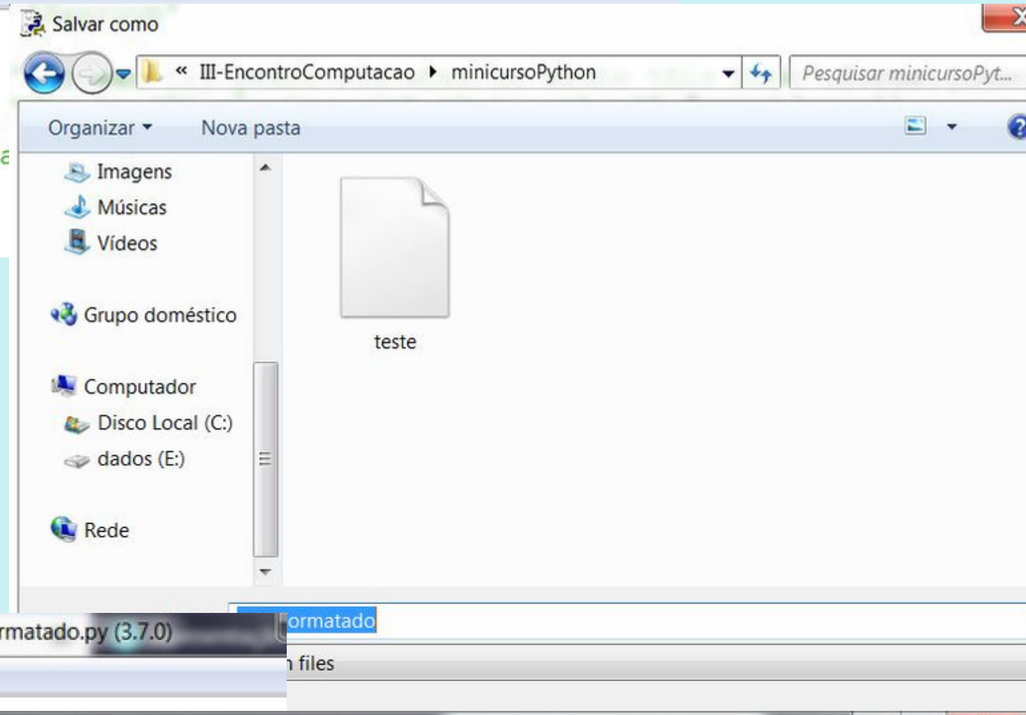
```
age = 9
name = 'Swaroop'

print('{0} was {1} years old when he wrote this book'.format(name, age))
print('Why is {0} playing with that python?'.format(name))
```

Delimite cadeia de caracteres com aspas dupla " " ou aspas simples ' '

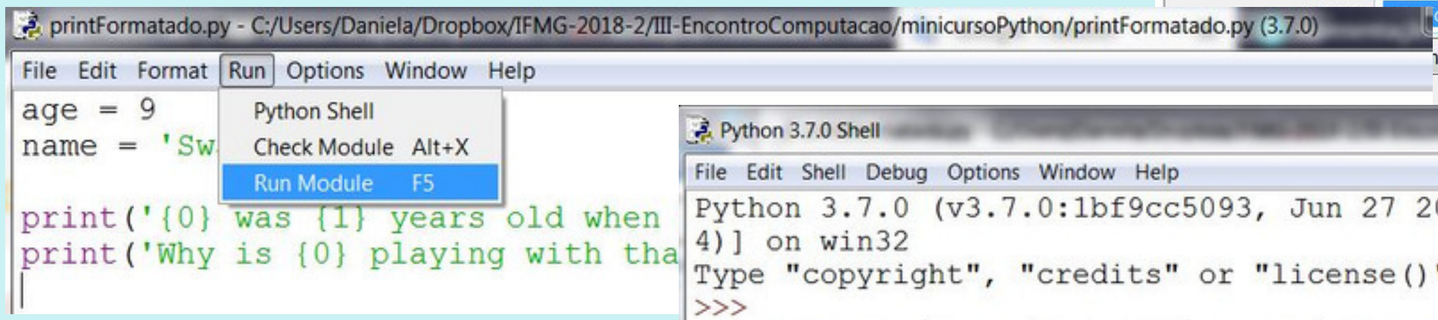
Programando no Idle

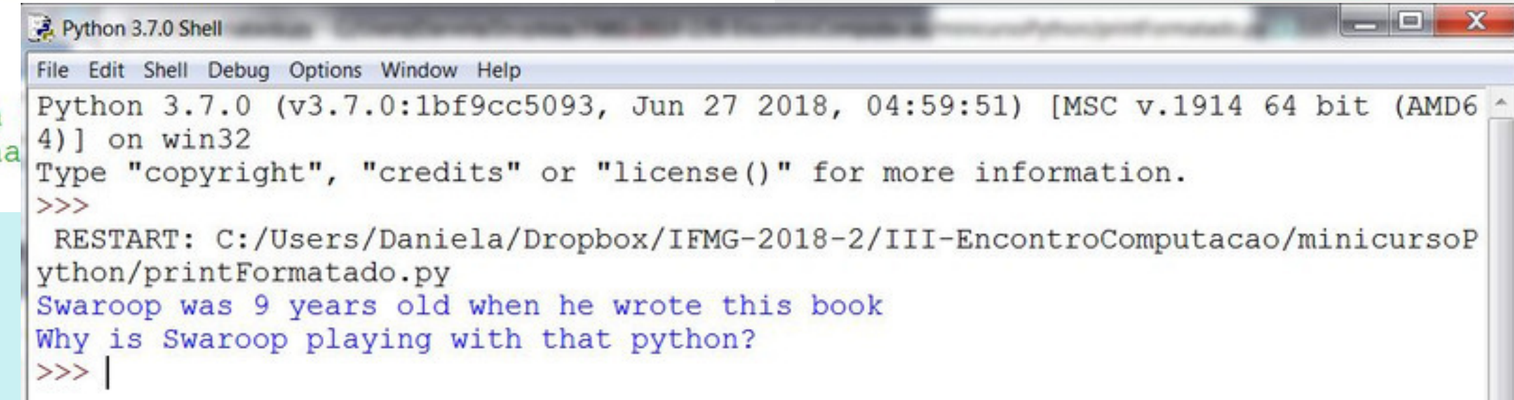
- Salve o arquivo...





- Execute com Run

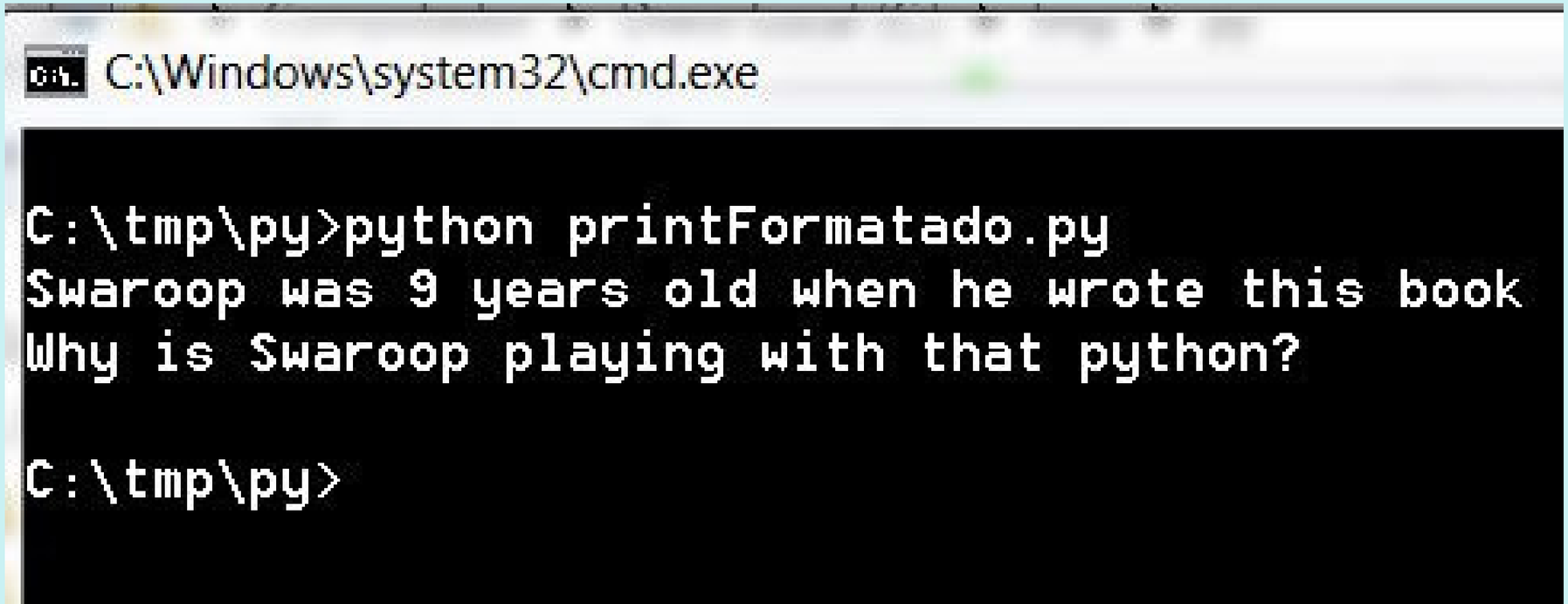


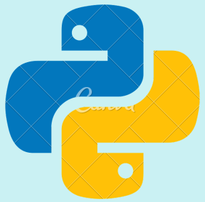


Executando do Prompt

- Abra o shell do SO (ex. cmd do Windows)
- Vá para o diretório onde está seu arquivo fonte .py e faça:

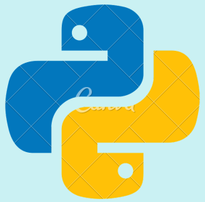
python <nomeDoPrograma>.py





Sintaxe básica: identificadores

- Para nomear variáveis e outros identificadores:
    - o 1o. caractere deve ser uma letra do alfabeto (uppercase ASCII ou lowercase ASCII ou um caractere Unicode) ou um underscore (_)
    - o 2o. caractere em diante pode ser uma letra (uppercase ASCII ou lowercase ASCII ou Unicode character), underscores (_) ou números (0-9)



Sintaxe básica: tipos de dados

- No Python a tipificação é dinâmica
  - nenhuma declaração ou definição de tipo é necessária
  - o tipo de uma variável é definido pelo conteúdo atribuído
- Há tipos básicos embutidos:
  - números (inteiro e ponto flutuante), strings e booleanos
- Há tipos embutidos para estruturas de dados:
  - listas, tuplas, conjuntos, dicionários
- Os tipos não basicos são classes

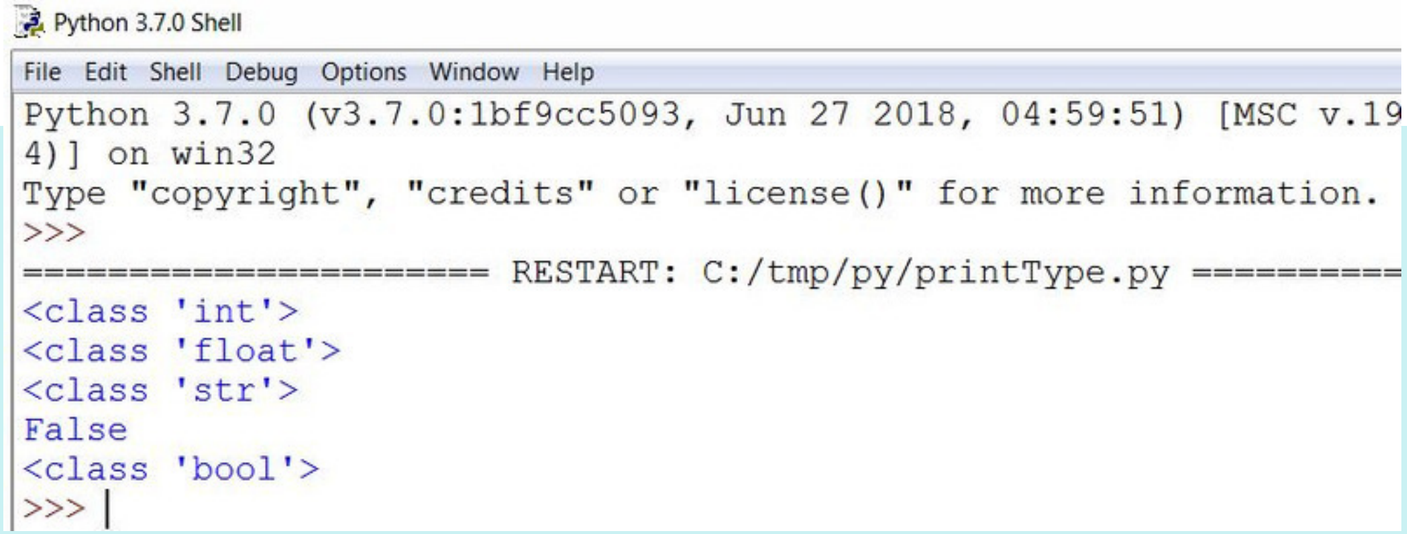
Sintaxe básica: variáveis

- Observe os tipos exibidos pela função 'type' no programa abaixo:

```
"""
O trecho abaixo imprime o nome dos tipos de dados
das variáveis usando a função type
"""
number = 9
print(type(number))

float_number = 9.0
print(type(float_number))

caractere = 'A';  print(type(caractere))  # para inserir mais de um comando na mesma linha use ';'

flag = False;
print(flag)
print(type(flag))
```

Sintaxe básica: variáveis

```
"""
O trecho abaixo imprime o nome dos tipos de dados
das variáveis usando a função type
"""
number = 9
print(type(number))

float_number = 9.0
print(type(float_number))

caractere = 'A';  print(type(caractere))  # para inserir mais de um comando na mesma linha use ';'

flag = False;
print(flag)
print(type(flag))
```



Sintaxe básica: operadores

- Operadores comuns:

Operador Aritmético	
soma	+
subtração	-
multiplicação	*
potência	**
divisão	/
divide e acha o piso: divisão inteira	//
módulo	%
shift left	<<

Operadores bit-a-bit, relacionais e lógicos	
and bit-a-bit	&
or bit-a-bit	|
or exclusivo	^
inverte bit	~
menor que	<
maior que	>
menor ou igual	<=
maior ou igual	>=
igual	==
diferente	!=
NOT booleano	not
AND booleano	and
OR booleano	or

Sintaxe básica: operadores

- Operadores de identidade e pertinência :

operador identidade		
retorna ***true*** se ambas as variáveis são o mesmo objeto	**is**	x = ["apple", "banana"] y = ["apple", "banana"] z = x print(x is z) # **True**
retorna ***true*** se as variáveis não são o mesmo objeto	**is not**	print(x is not y) # **True**

operador de pertinência		
retorna ***true*** se a sequência está presente no objeto	**in**	x = ["apple", "banana"] print("banana" in x) # **True**
retorna ***true*** se a sequência não está presente no objeto	**not in**	print("ban" not in x) # **False**

Sintaxe básica: indentação

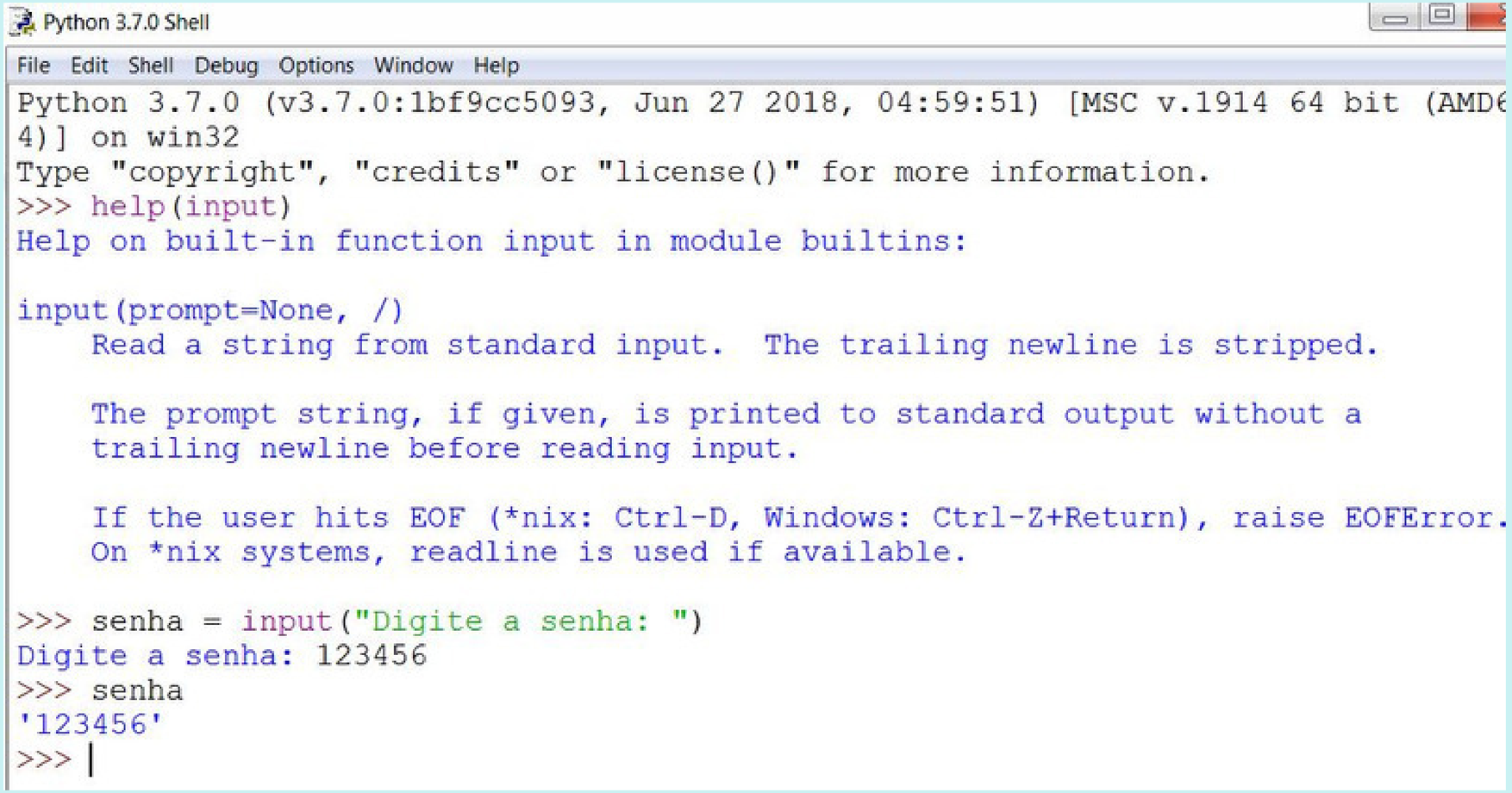
- A indentação no início da linha é usado para determinar o agrupamento de sentenças de um bloco

```
if True:
    print('Yes, it is true')
```

Sintaxe básica: entrada de dados

- Para ler dados do console use a função 'input':

use help(comando) para saber como se usa



Sintaxe básica: casting

- int( ): converte para inteiro
- float( ): converte para float
- str( ): converte para string

```
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3

x = float(1)     # x will be 1.0
y = float(2.8)   # y will be 2.8
z = float("3")   # z will be 3.0
w = float("4.2") # w will be 4.2

x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

Sintaxe básica: if-elif-else

- Observe a sintaxe:

insira ':'
e inclua
o bloco de
comandos

```
number = 23
guess = int(input('Enter an integer : '))

if guess == number:
    # New block starts here
    print('Congratulations, you guessed it.')
    print('(but you do not win any prizes!)')
    # New block ends here
elif guess < number:
    # Another block
    print('No, it is a little higher than that')
    # You can do whatever you want in a block ...
else:
    print('No, it is a little lower than that')
    # you must have guessed > number to reach here

print('Done')
```

Sintaxe básica: if-elif-else

- Observe a sintaxe:

```
number = 23
guess = int(input('Enter an integer : '))

if guess == number:
    # New block starts here
    print('Congratulations, you guessed it.')
    print('(but you do not win any prizes!)')
    # New block ends here
elif guess < number:
    # Another block
    print('No, it is a little higher than that')
    # You can do whatever you want in a block ...
else:
    print('No, it is a little lower than that')
    # you must have guessed > number to reach here

print('Done')
```

Saída:

```
Enter an integer : 50
No, it is a little lower than that
Done

$ python if.py
Enter an integer : 22
No, it is a little higher than that
Done

$ python if.py
Enter an integer : 23
Congratulations, you guessed it.
(but you do not win any prizes!)
Done
```

Sintaxe básica: while

- Observe que o 'while' pode incluir um 'else ' adicional:

use 'break' (para sair do loop) e 'continue' (para passar para a próxima iteração)

```
number = 23
running = True

while running:
    guess = int(input('Enter an integer : '))

    if guess == number:
        print('Congratulations, you guessed it.')
        # this causes the while loop to stop
        running = False
    elif guess < number:
        print('No, it is a little higher than that.')
    else:
        print('No, it is a little lower than that.')
else:
    print('The while loop is over.')
    # Do anything else you want to do here

print('Done')
```

Saída:

```
$ python while.py
Enter an integer : 50
No, it is a little lower than that.
Enter an integer : 22
No, it is a little higher than that.
Enter an integer : 23
Congratulations, you guessed it.
The while loop is over.
Done
```

Sintaxe básica: for

- Usado para operar sobre sequências (lista, tuplas, dicionários, conjuntos ou uma string):

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Saída:

```
apple
banana
cherry
```

Sintaxe básica: for

- Usado para operar sobre sequências (lista, tuplas, dicionários, conjuntos ou uma string):

Saída:

```
for x in "banana":
  print(x)
```

```
b
a
n
a
n
a
```

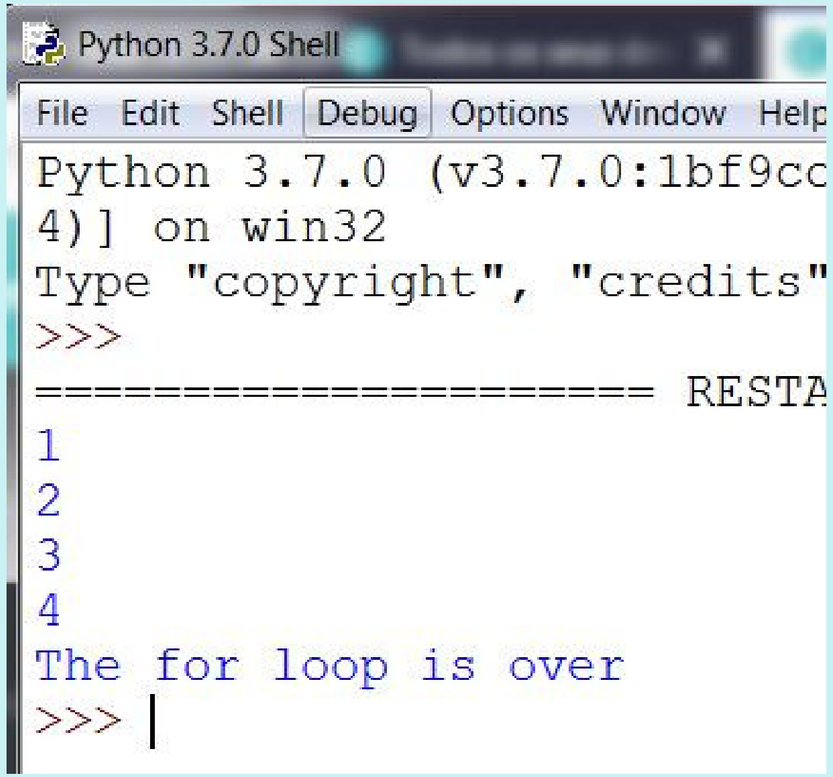
Sintaxe básica: for

- Usado para operar sobre sequências (lista, tuplas, dictionários, conjuntos ou uma string):

```
for i in range(1, 5):
    print(i)
else:
    print('The for loop is over')
```

range(1, 5) gera a lista de inteiros: [1, 2, 3, 4]

Saída:



Listas, tuplas, conjuntos e dicionários

- Há 4 tipos de dados embutidos (built-in) para estruturas de dados em Python:
    - lista: coleção ordenada, indexada e mutável. Permite membros duplicados
    - tupla: coleção ordenada e imutável. Permite membros duplicados
    - conjunto: coleção não ordenada e não indexada. Não permite membros duplicados
    - dicionário: coleção não ordenada, mutável e indexada. Não permite membros duplicados

Listas

- Coleção ordenada, indexada e mutável
- Permite membros duplicados
- Em Python, listas são escritas entre colchetes '[' e ']'

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

Saída:

```
C:\Users\My Name>python demo_list.py
['apple', 'banana', 'cherry']
```

Listas

Saídas:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

```
C:\Users\My Name>python demo_list_access.py
banana
```

```
thislist[1] = "blackcurrant"

print(thislist)
```

```
['apple', 'blackcurrant', 'cherry']
```

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

```
apple
banana
cherry
```

Listas

Saídas:

```python
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

```
3
```

```python
thislist = ["apple", "banana", "cherry"]

thislist.append("orange")

print(thislist)
```

```
['apple', 'banana', 'cherry', 'orange']
```

```python
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```

```
['apple', 'orange', 'banana', 'cherry']
```

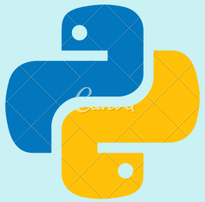
```python
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

```
['apple', 'cherry']
```

Listas

- Outros métodos úteis da classe List:

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list



Listas: pratique

- Faça um programa para criar uma lista de itens de compra com dados lidos do usuário
- Exiba a lista em ordem crescente e em ordem descrecente de nomes de produtos

Dica: para criar uma lista vazia faça:

```
listaDeCompras = []
```

Listas: pratique

- Programa 'Lista de compra':

pratica1.py - C:/Users/Daniela/Dropbox/IFMG-2018-2/III-EncontroComputacao/minicursoPython/pratica1.py (3

File Edit Format Run Options Window Help

```
listaDeCompra = []
i = 0
continua = True
resposta = ''
while(continua):
    listaDeCompra.append(input("De o item["+str(i)+"]:"))
    i+=1
    print("\nMais itens? (S/N):")
    resposta = input().strip().upper()
    continua = (resposta[0]) == 'S'

print('\nLista em ordem alfabetica: ')
listaDeCompra.sort()
print(listaDeCompra)

print('\nLista em ordem alfabetica invertida: ')
listaDeCompra.reverse()
print(listaDeCompra)
```

Tuplas

- Coleção ordenada e imutável

  Permite membros duplicados

  Em Python, tuplas são definidas entre parênteses '(' e ')'

exTuplas.py - C:/Users/Daniela/Dropbox/IFMG-2018-2/III-EncontroComputacao/minicursoPython/exTuplas.py (3.7.0)

File Edit Format Run Options Window Help

```
thistuple = ("apple", "banana", "cherry")
print(thistuple) # imprime os itens da tupla

print(thistuple[1]) # imprime "banana"

#thistuple[1] = "blackcurrant"  #Erro. Alteraçao nao suportada

print(len(thistuple)) # imprime o tamanho da tupla

for x in thistuple:   # imprime cada item da tupla
    print(x)
```

Tuplas

exTuplas.py - C:/Users/Daniela/Dropbox/IFMG-2018-2/III-EncontroComputacao/minicursoPython/exTuplas.py (3.7.0)

File Edit Format Run Options Window Help

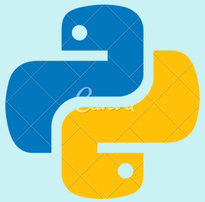
```
thistuple = ("apple", "banana", "cherry")
print(thistuple) # imprime os itens da tupla

print(thistuple[1]) # imprime "banana"

#thistuple[1] = "blackcurrant"  #Erro. Alteraçao nao suportada

print(len(thistuple)) # imprime o tamanho da tupla

for x in thistuple:    # imprime cada item da tupla
    print(x)
```

Saída:

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:5
4)] on win32
Type "copyright", "credits" or "license()" for mor
>>>
 RESTART: C:/Users/Daniela/Dropbox/IFMG-2018-2/III
ython/exTuplas.py
('apple', 'banana', 'cherry')
banana
3
apple
banana
cherry
>>>
```



Tuplas

- Tuplas são imutáveis, não é possível inserir nem remover elementos
- Outros métodos disponíveis:

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Tuplas: pratique

- Altere o programa da 'Lista de compras' representando cada item como uma tupla :

(<nome do item>, <qtde>)

Ex.: cada tupla será inserida à lista:

[("arroz", 5.6), ("feijao", 10.2), ("carne", 5.2)]

Tuplas: pratique

- Programa 'Lista de compras' :

pratica2.py - C:/Users/Daniela/Dropbox/IFMG-2018-2/III-EncontroComputacao/minicursoPython/p

File Edit Format Run Options Window Help

```
resposta = ''
item = ''
qtde = 0.0
itemCompra = ()
while(continua):
    item = input("De o item["+str(i)+"]:")
    qtde = float(input("Qtde["+str(i)+"]:"))
    itemCompra = (item, qtde)
    listaDeCompra.append(itemCompra)
    i+=1
    print("\nMais itens? (S/N):")
    resposta = input().strip().upper()
    continua = (resposta[0]) == 'S'

print('\nLista em ordem alfabetica: ')
listaDeCompra.sort()
print(listaDeCompra)

print('\nLista em ordem alfabetica invertida: ')
listaDeCompra.reverse()
print(listaDeCompra)
```

Saída:

```
De o item[0]:arroz
Qtde[0]:5

Mais itens? (S/N):
sim
De o item[1]:feijao
Qtde[1]:3

Mais itens? (S/N):
sim
De o item[2]:carne
Qtde[2]:2.80

Mais itens? (S/N):
nao

Lista em ordem alfabetica:
[('arroz', 5.0), ('carne', 2.8), ('feijao', 3.0)]

Lista em ordem alfabetica invertida:
[('feijao', 3.0), ('carne', 2.8), ('arroz', 5.0)]
>>>
```

Conjuntos (set)

- Coleção que não é ordenada nem indexada
- Conjuntos em Python são informados entre chaves '{' e '}'

exConjunto.py - C:/Users/Daniela/Dropbox/IFMG-2018-2/III-EncontroComputacao/minicursoPython/exConjunto.py (3.7.0)

File Edit Format Run Options Window Help

```
thisset = {"apple", "banana", "cherry"}
print(thisset)

'''
Não é possível acessar um item do conjunto
informando seu indice pois conjunto não são
ordenados.

Para acessar um item do conjunto usando o 'for'
'''
for x in thisset:
  print(x)

# ou pelo uso do 'in'
print("banana" in thisset)
```

Conjuntos (set)

- Observe a manipulação e uso de funções para sets abaixo:

exConjunto.py - C:/Users/Daniela/Dropbox/IFMG-2018-2/III-EncontroComputacao/minicursoPython/exConjunto.py (3.7.0)

File Edit Format Run Options Window Help

```
thisset = {"apple", "banana", "cherry"}
print(thisset)

'''
Não é possível acessar um item do conjunto
informando seu indice pois conjunto não são
ordenados.

Para acessar um item do conjunto usando o 'for'
'''
for x in thisset:
  print(x)

# ou pelo uso do 'in'
print("banana" in thisset)

thisset.add("orange")  # Adiciona um item ao conjunto
print(thisset)

thisset.update(["orange", "mango", "grapes"]) # Adiciona multiplos itens
print(thisset)

print(len(thisset)) # Imprime o tamanho do conjunto
```

Saída:

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MS
4)] on win32
Type "copyright", "credits" or "license()" for more informa
>>>
 RESTART: C:/Users/Daniela/Dropbox/IFMG-2018-2/III-Encontro
ython/exConjunto.py
{'apple', 'cherry', 'banana'}
apple
cherry
banana
True
{'apple', 'cherry', 'banana', 'orange'}
{'apple', 'banana', 'mango', 'cherry', 'grapes', 'orange'}
6
>>>
```

Conjuntos (set)

- Observe ao lado funções (métodos) úteis para operar sobre conjuntos:

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes the specified element
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

Dicionário

- Coleção não ordenada, imutável e indexada
- Em Python inserimos as entradas no dicionários entre chaves '{' e '}'
- As entradas são pares do tipo: 'chave' e 'valor'

exDicionario.py - C:/Users/Daniela/Dropbox/IFMG-2018-2/III-EncontroComputacao/minicursoPy

File Edit Format Run Options Window Help

```
thisdict =      {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)

# Acessando uma entrada do dicionário pela chave:
x = thisdict["model"]

# Ou, então use get():
x = thisdict.get("model")
```

Dicionário

exDicionario.py - C:/Users/Daniela/Dropbox/IFMG-2018-2/III-EncontroComputacao/minicursoPy

File Edit Format Run Options Window Help

```
thisdict =      {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)

# Acessando uma entrada do dicionário pela chave:
x = thisdict["model"]

# Ou, então use get():
x = thisdict.get("model")

# Alterando o valor (value) da entrada:
thisdict["year"] = 2018

# Imprime cada chave (key):
for x in thisdict:
  print(x)

# Imprime cada valores (value):
for x in thisdict:
  print(thisdict[x])

# Imprime key e value:
for x, y in thisdict.items():
  print(x, y)
```

- Observe a manipulação de um dicionário e usando funções disponíveis:

Saída:

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:5
4)] on win32
Type "copyright", "credits" or "license()" for more i
>>>
 RESTART: C:/Users/Daniela/Dropbox/IFMG-2018-2/III-Er
ython/exDicionario.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
brand
model
year
Ford
Mustang
2018
brand Ford
model Mustang
year 2018
>>>
```

Dicionário

- É possível inserir itens ao dicionário após sua criação
- Teste o trecho abaixo e verifica a inserção da entrada: "color": "red"

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["color"] = "red"
print(thisdict)
```

Dicionário: pratique

- Refaça o mesmo programa 'Lista de compras' usando entradas do tipo:

Ex.: itemDeCompra = {"item": "",
"qtde": 0.0}

Dicionário: pratique

- Programa 'Lista de compra':

pratica3.py - C:/Users/Daniela/Dropbox/IFMG-2018-2/III-EncontroComputacao/minicursoPython/pratica3.py (3.7.0)

File Edit Format Run Options Window Help

```
listaDeCompra = []
i = 0
continua = True
resposta = ''
itemDeCompra = {"descricao": "",
                "qtde": 0.0}
while(continua):
    itemDeCompra["descricao"] = input("De o item["+str(i)+"]:")
    itemDeCompra["qtde"] = float(input("Qtde["+str(i)+"]:"))

    listaDeCompra.append(itemDeCompra)
    i+=1
    print("\nMais itens? (S/N):")
    resposta = input().strip().upper()
    continua = (resposta[0]) == 'S'

print('\nLista de Compra: ')
print(listaDeCompra)
```

Saída:

```
De o item[0]:arroz
Qtde[0]:5.90

Mais itens? (S/N):
sim
De o item[1]:feijao
Qtde[1]:3.9

Mais itens? (S/N):
sim
De o item[2]:carne
Qtde[2]:4.5

Mais itens? (S/N):
nao

Lista de Compra:
[{'descricao': 'carne', 'qtde': 4.5}, {'descricao': 'carne', 'qtde': 4.5}, {'des
cricao': 'carne', 'qtde': 4.5}]
>>>
```

Operando matrizes

- Observe no código a seguir um exemplo de manipulação de uma lista de listas, para representar uma matriz de inteiros:

File Edit Format Run Options Window Help

```
''' observe que devemos definir uma lista vazia
    que nesse caso é uma listas de listas
'''
m = [[]]

for i in range(3):
    m.append([])    #cria nova linha vazia
    for j in range(3):
         m[i].append(j)  # cria nova coluna
else:
      m.remove([])

print(m)

"""
   Varrendo a matriz
"""
for i in range(len(m)):    # len(m) retorna o n° de linhas
    for j in range(len(m[i])): #len(m[i]) retorna o n° de colunas
        print(m[i][j])
```

Operando matrizes

```
File  Edit  Format  Run  Options  Window  Help
''' observe que devemos definir uma lista vazia
    que nesse caso é uma listas de listas
'''
m = [[]]

for i in range(3):
    m.append([])   #cria nova linha vazia
    for j in range(3):
        m[i].append(j)  # cria nova coluna
else:
      m.remove([])

print(m)

"""
   Varrendo a matriz
"""
for i in range(len(m)):   # len(m) retorna o n° de linhas
    for j in range(len(m[i])): #len(m[i]) retorna o n° de colunas
        print(m[i][j])
```

Saída:

```
Python 3.7.0 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun
4)] on win32
Type "copyright", "credits" or "licen
>>>
 RESTART: C:/Users/Daniela/Dropbox/IF
ython/teste3.py
[[0, 1, 2], [0, 1, 2], [0, 1, 2]]
0
1
2
0
1
2
0
1
2
>>>
```

Definindo Funções

- Uma função é um bloco de código que é executado somente quando é chamado
- É possível passar dados para funções (os parâmetros)

exFuncao.py - C:/Users/Daniela/Dropbox/IFMG-2018-2/III-

File Edit Format Run Options Window Help

```
def my_function(x):
  return 5 * x

for i in range(10):
    print(my_function(i), "\n")
```

Definindo Funções

exFuncao.py - C:/Users/Daniela/Dropbox/IFMG-2018-2/III-

File Edit Format Run Options Window Help

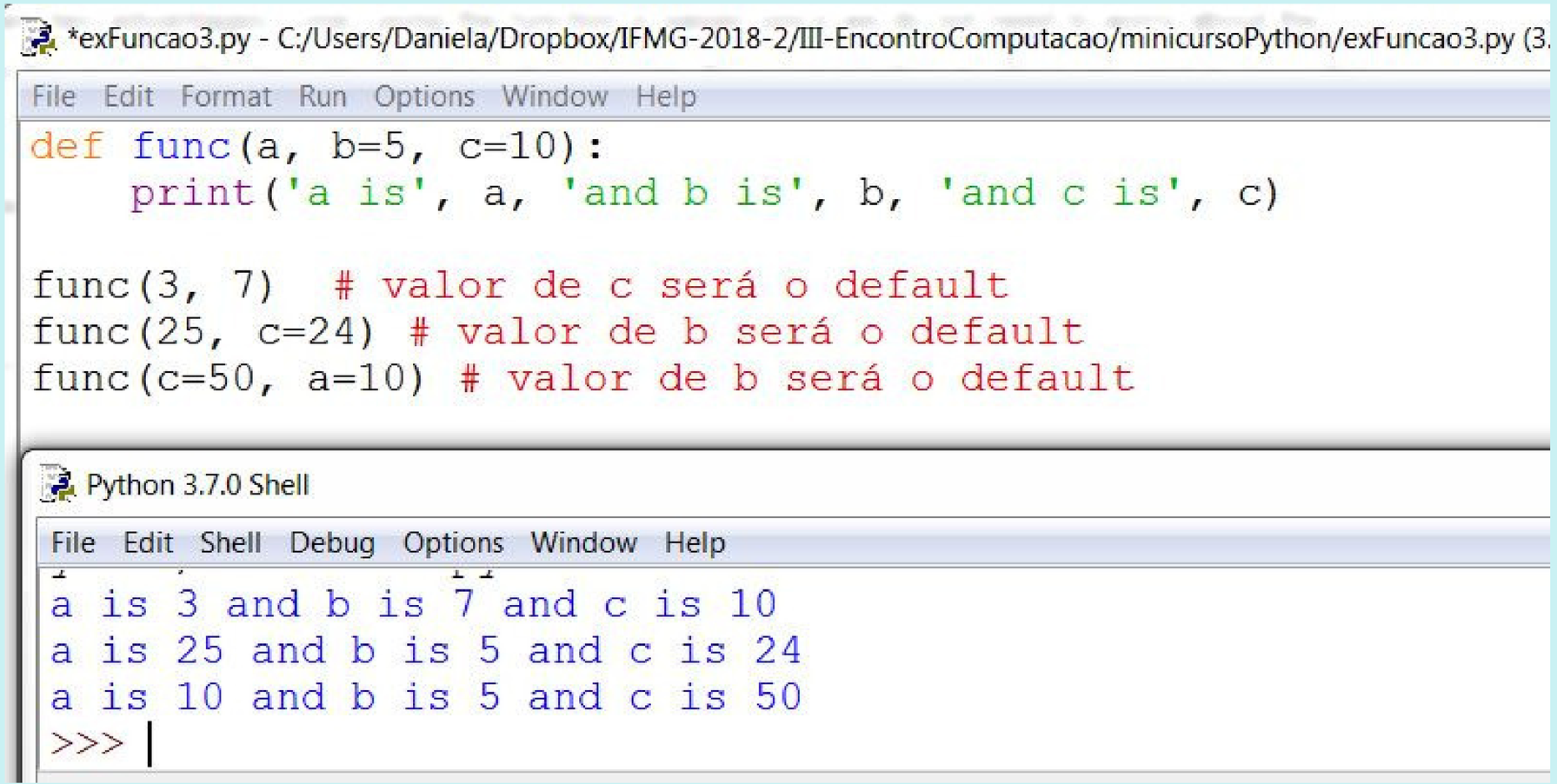
```
def my_function(x):
  return 5 * x

for i in range(10):
    print(my_function(i), "\n")
```

- Saída:

```
0

5

10

15

20

25

30

35

40

45

>>>
```

Definindo Funções

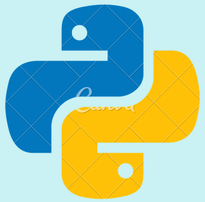
- Há como definir valores default para parâmetros e nomear os parâmetros passados:



Funções: pratique

- Complete o código da função fib() abaixo:

```
def sum_two_numbers(a, b):
    return a + b              # retorna o resultado nas chamadas

c = sum_two_numbers(3, 12)   # associa o resultado da função à  variável 'c'


def fib(n):
    """Retorna uma lista contendo os termos da série
    de fibonacci até n."""
    result = []
    a = 1
    b = inicialize a variável b
    while a < n:
        result.append(a)
        tmp_var = b
        atualize a variável b
        atualize a  variável a
    return result

print(fib(10))
```



Funções: pratique

```
def sum_two_numbers(a, b):
    return a + b                # retorna o resultado nas chamadas

c = sum_two_numbers(3, 12)  # associa o resultado da função à  variável 'c'


def fib(n):
    """Retorna uma lista contendo os termos da série
    de fibonacci até n."""
    result = []
    a = 1
    b = 1
    tmp_var = 0
    while a < n:
        result.append(a)
        tmp_var = b
        b = a
        a = sum_two_numbers(tmp_var, a)
    return result

print(fib(10))
```

Módulos

- Agrupa funções e dados para reutilização
- Todo programa em Python é considerado um módulo
- Observe um exemplo de importação de módulos da biblioteca padrão do Python: math e sys

```
import sys
import math
from math import sqrt  # from ___ import ___ dispensou o uso de math.sqrt(4)

print(sqrt(4))
print(math.ceil(3.4))

print('Os argumentos de linha são:')

#imprime os argumentos de linha de comando
for i in sys.argv:
    print(i)

#imprime os caminhos de busca do Python
print('\n\nO PYTHONPATH é ', sys.path, '\n')

#imprime de caminhos em cada linha
for x in sys.path:
    print(x)
```

Referências para Python

- A byte of Python. Disponível em: https://python.swaroopch.com/stdlib.html (Ebook gratuito)
- Um byte de Python. Disponível em: https://www.homeyou.com/~edu/introducao (Ebook gratuito, em Português)
- Python Standard Library. Disponível em: https://docs.python.org
- Tutorial de Python da w3schools.com. Disponível em: https://www.w3schools.com/python/
- Comunidade Python Brasil. Link: https://wiki.python.org.br/PythonBrasil