Python

Primeiros passos

Prof. Dr. Dieval Guizelini

Analista e Desenvolvedor de Sistemas

Mestre em Bioinformática e Doutor em Ciências-Bioquímica

dieval at ufpr.br / dievalg at gmail.com

Python no

should be adopted when starting to build a new software syste

Feb 2022	Feb 2021	Change
1	3	▲
2	1	▼
3	2	▼
4	4	
5	5	
6	6	

https://www.tiobe.com/tiobe-index/

Language Ranking: IEEE Spectrum

Rank	Language	Type	Score
1	Python	web, desktop, system	100.0
2	Java	web, mobile, desktop	95.4
3	C	mobile, desktop, system	94.7
4	C++	mobile, desktop, system	92.4
5	JavaScript	web	88.1
6	C#	web, mobile, desktop, system	82.4
7	R	desktop	81.7

https://spectrum.ieee.org/top-programming-languages/

O ambiente de desenvolvimento

- On line
  - Python Shel
    https://www.python.org/shell/
  - Replit
    https://repl.it/languages/python3
  - Codingground
    https://www.tutorialspoint.com/execute_python_online.php

- Offline
  - Editores / não IDEs
    - Sublime text http://www.sublimetext.com/3
    - Textmate http://macromates.com/
    - Notepad++ https://notepad-plus-plus.org/download/v7.4.2.html
  - Integrated Developer Environment (IDE)
    - PyCharm http://www.jetbrains.com/pycharm/
    - WingIDE http://wingware.com/
    - Komodo IDE https://www.activestate.com/komodo-ide
    - IDLE distribuído com o Python... Shell gráfico.

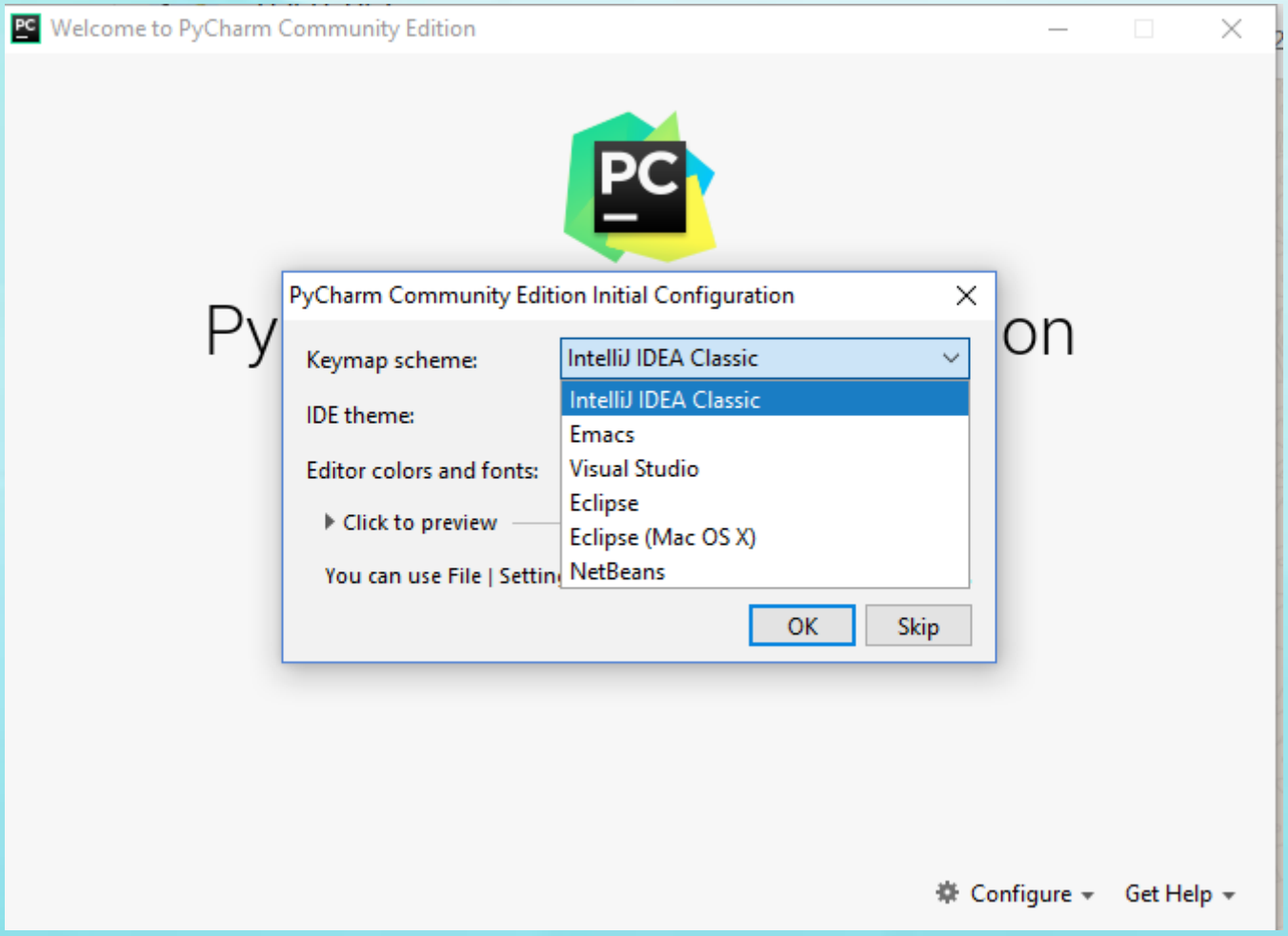
Baixe o Python

- Para MS-Windows, versão 3.9.5
  https://www.python.org/downloads/

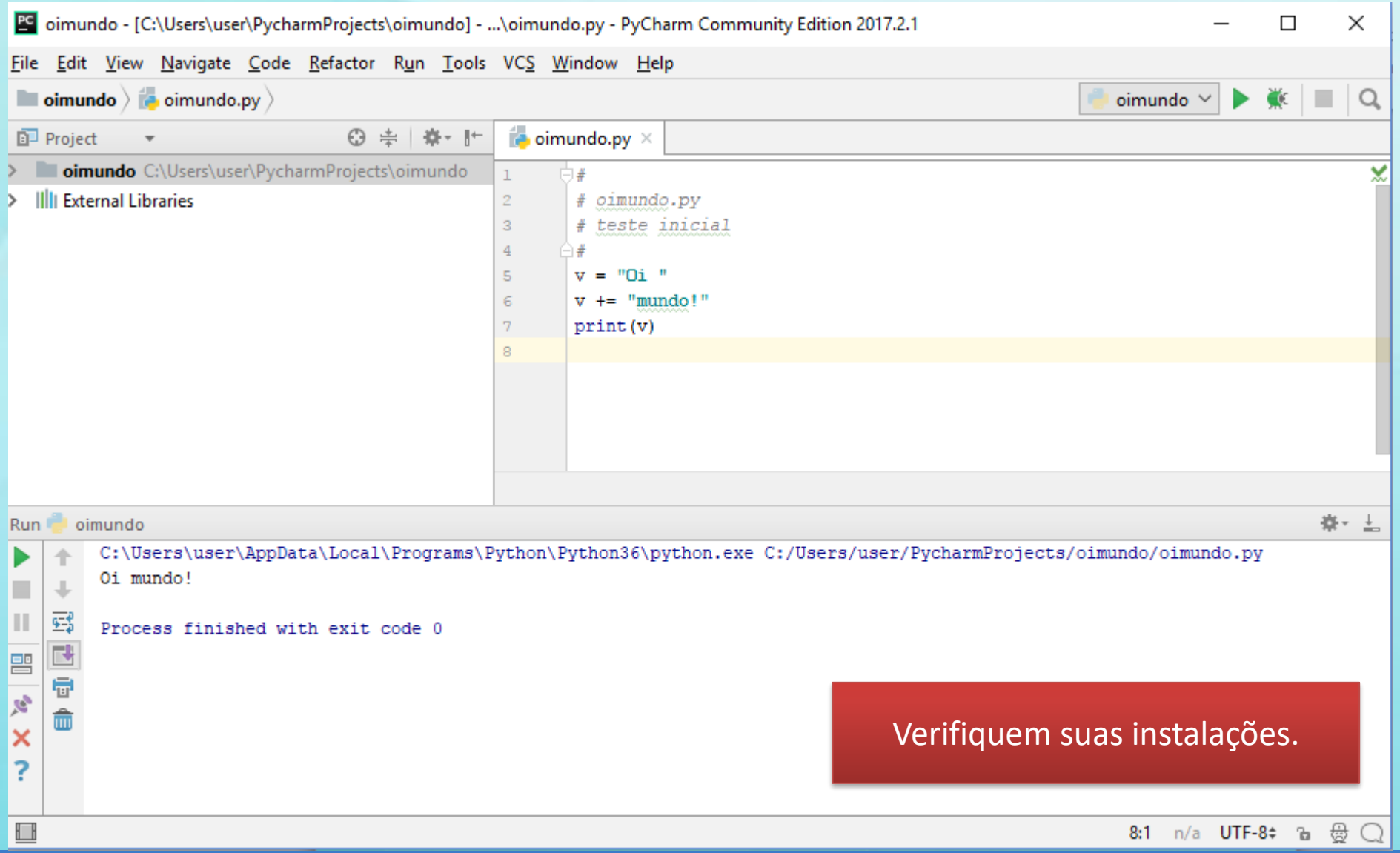
- Para Linux... Pacotes
  apt-cache search python*
  apt-get install python*

- Documentações, tutoriais etc
  https://wiki.python.org/moin/BeginnersGuide/Programmers

Após a instalação do PyCharm

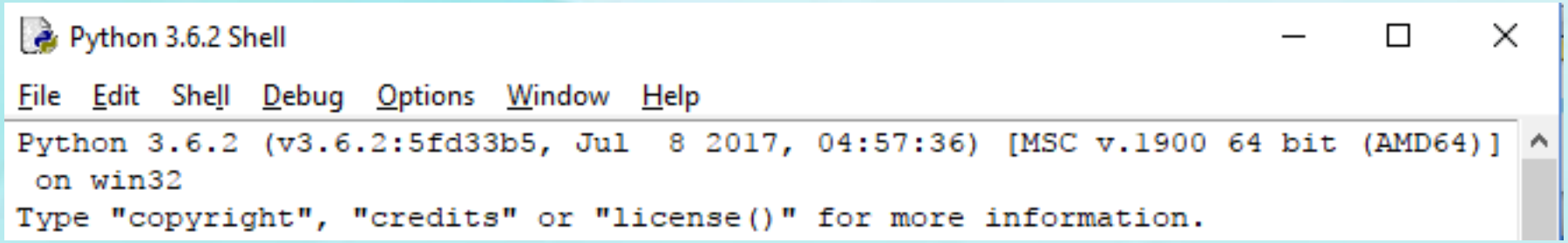


Oi mundo!



Verifiquem suas instalações.

O Shell do Python




```
dieval@biia:~$ python
Python 2.7.9 (default, Jun 29 2016, 13:08:31)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

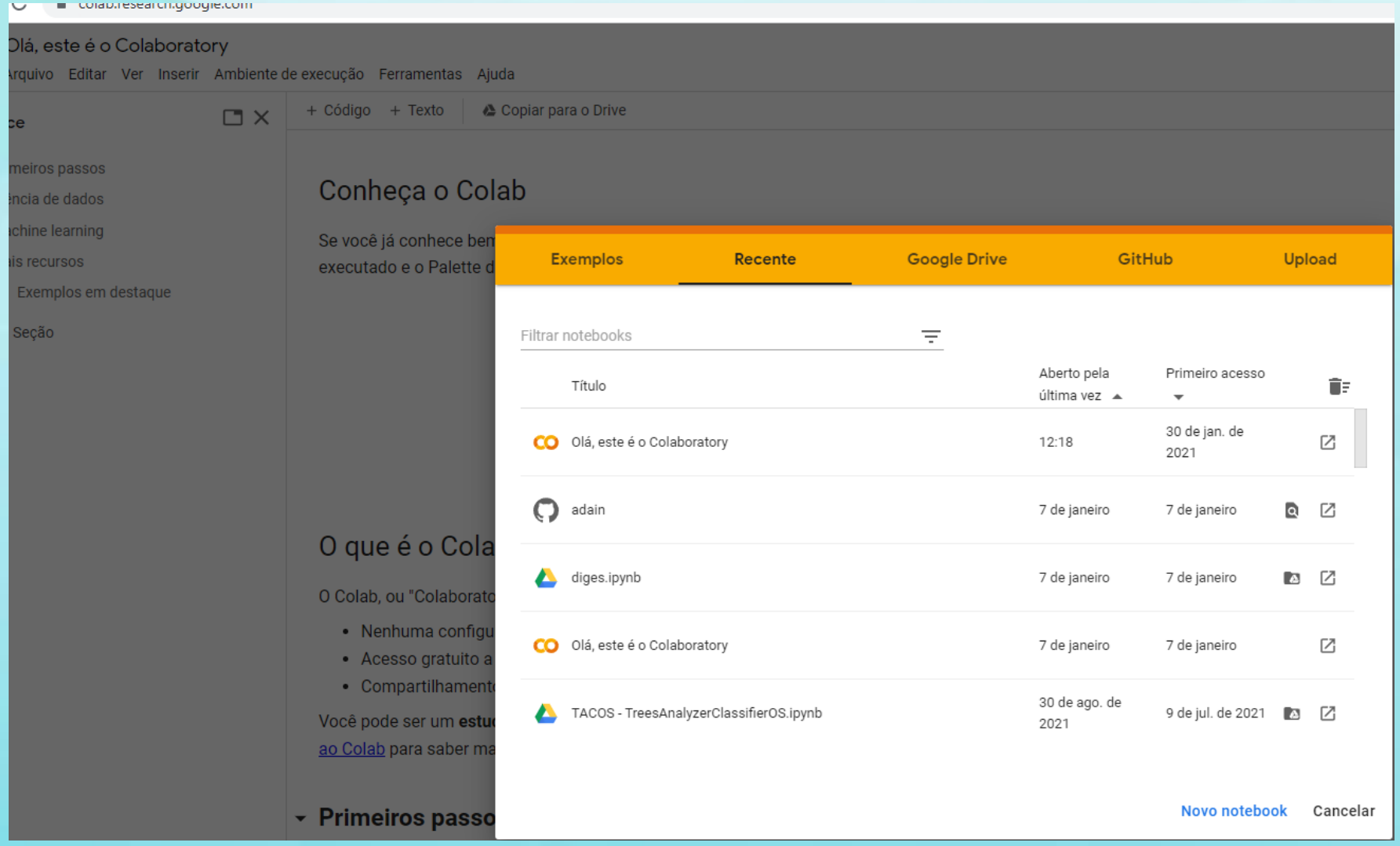
Esse é o modo “interativo” do Python, onde instruções são informadas uma a uma e imediatamente executas.

```
>>> a = 2
>>> b = 5
>>> a * b
10
>>>
```

```
>>> x = "oi "
>>> y = "mundo! "
>>> x + y
'oi mundo! '
>>>
```

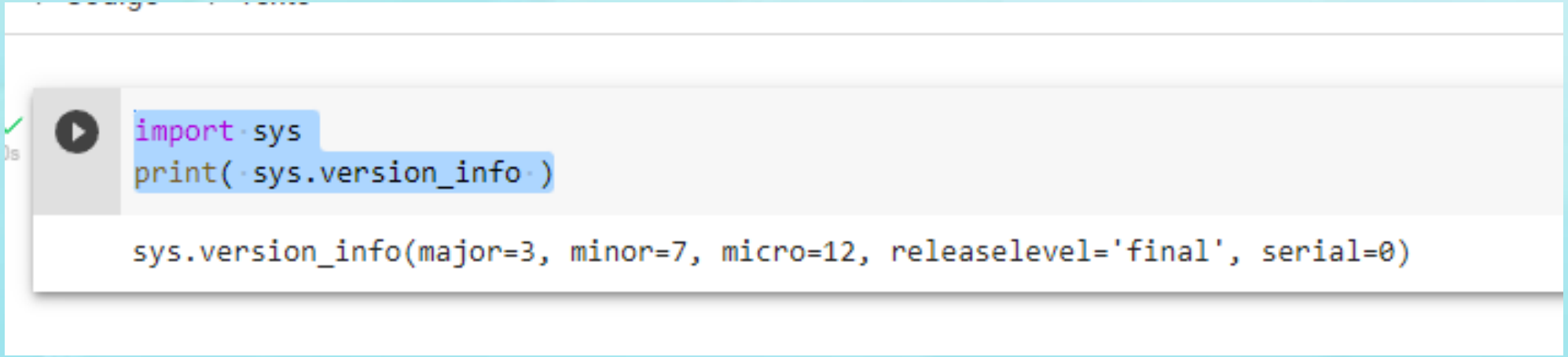
Google Colab

- https://colab.research.google.com/



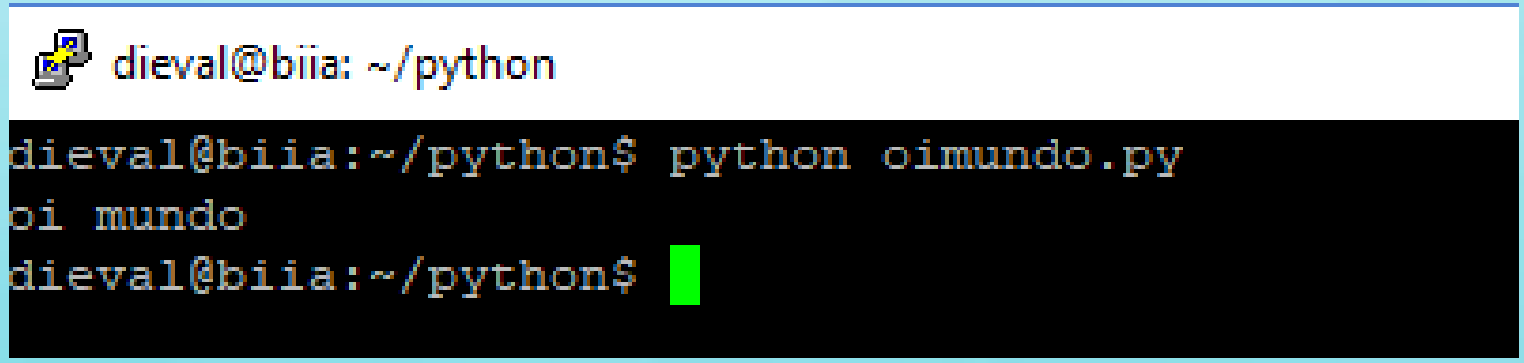
Versão no colab

```
import sys
print( sys.version_info )
```



Escrevendo e rodando em Python

- Em um editor de arquivo texto (vi, vim, notepad....), crie o arquivo oimundo.py e transcreva:
  s = “oi “
  s2 = “ mundo!”
  print(s + s2)
- Na linha de comando do S.O. (cmd.exe no Windows ou bash no linux) digite:
  python oimundo.py



Compilado x interpretado

- O Python compila o código-fonte (script) para uma versão intermediária chamada bytecode.
- O bytecode é interpretado pela VM do Python

Considerações:
1) Fácil de aprender
2) Pode-se codificar em diferentes paradigmas
3) “Extensível”
4) Comunidade aberta e ativa

Python is an interpreted language, which means the source code of a Python program is converted into *bytecode* that is then executed by the Python virtual machine. Python is different from major compiled languages, such as C and C + +, as Python code is not required to be *built* and *linked* like code for these languages. This distinction makes for two important points:

- **Python code is fast to develop**: As the code is not needed to be compiled and built, Python code can be readily changed and executed. This makes for a fast development cycle.
- **Python code is not as fast in execution**: Since the code is not directly compiled and executed and an additional layer of the Python virtual machine is responsible for execution, Python code runs a little slow as compared to conventional languages like C, C + +, etc.

https://www.sciencedirect.com/topics/computer-science/interpreted-language#:~:text=Python%20is%20an%20interpreted%20language,like%20code%20for%20these%20languages.

Antes de programar...

- Um programa é um algoritmo codificado segundo as regras de uma linguagem de programação.

- Algoritmo é um conjunto de instruções finita que descrevem como um problema pode ser resolvido.

- Todo programa/algoritmo manipulam dados, ou seja, o programa recebe dados (entrada), processa alguma coisa em decorrência desses dados e produz novos dados (saída). Os dados são sempre referenciados nos programas pelas variáveis.

Variáveis

- **Na matemática**:
  ax+by+c=0 ou f(x) = (ax+c) / b

- **Em informática**:
  as variáveis são “identificadas” por um nome.
  Para cada LP existem um conjunto de termos que não podem ser utilizados como identificados – palavras-reservadas.
  As LP podem ser CASE-SENSITIVE ou não.
- Associado a cada variável existe um “tipo” e um “valor/conteúdo”
  - Quando a LP exige a declaração da variável, geralmente, é chamada de fortemente tipada.
- O conteúdo/valor ou dado pode ser de diferentes naturezas e eventualmente indicar o tipo da variável. A natureza pode ser numérica, caracteres, lógicos, objetos etc.
  Ex: 12 não é o mesmo que “12”

Variáveis em Python

- Toda variável é um objeto.
- Python é CASE-SENSITIVE, ou seja, var e VAR são dois identificadores válidos e distintos.
- Python é considerada fortemente tipada, mas não precisa ter as variáveis declaradas antes de referenciá-las em expressões. Passam a existir no momento que recebem um valor/conteúdo.

Identificadores em Python

- Um identificador Python é um nome usado para identificar uma variável, função, classe, módulo ou outro objeto. Um identificador começa com uma letra A a Z ou a a z ou um sublinhado (_) seguido de zero ou mais letras, sublinhados e dígitos (0 a 9).
- Algumas convenções:
  - O nome das classes começam com a primeira letra em maiúsculas;
  - Identificadores em classes iniciados com um único (_) indica campos privados
  - Iniciando com _ _ indica que é fortemente privado
  - Se o identificador também terminar com dois __, o identificador será considerado especial e definido pela linguagem

Palavras reservadas

and	assert	break	class	continue
del	elif	else	except	exec
for	from	global	if	import
is	lambda	not	or	pass
raise	return	try	while	with
and	assert	break	class	continue

Tipos de dados

O Python 3 possui 5 tipos de dados:

- Números
  - int
  - long
  - float
  - complex
- String
- Bool (True/False)
- List
- Tuple
- Dictionary

int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAEl	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J
0x69	-4721885298529L	70.2-E12	4.53e-7j

Strings

Exemplos:

```
#!/usr/bin/python
str = 'Hello World!'
print( str )          # imprime o conteúdo de str (Hello World!)
print( str[0] )       # imprime apenas o 1o caractere (H)
print( str[2:5] )     # imprime da 3a a 5 letra (llo)
print( str[2:]  )     # imprime da 3a letra até o final
print( str * 2  )     # repete o conteúdo de str duas vezes
print( str + " test" ) # imprime o resultado da concatenação de str com test
```

Estrutura de Dados

- As estruturas disponíveis em Python são: listas (lists), registros (tuples) e dicionários (dictionaries – map)
- Conjuntos (sets) estão disponíveis na biblioteca sets (a partir da versão 2.5)
- As listas são como vetores/matrizes unidimensionais, mas pode-se utilizar listas de listas
- Os dicionários são matrizes associativas
- As tuplas são conjuntos de campos pré-definidos.

Conversão de tipos de dados

- int(str [,base]) converte string para int
- long(str [,base]) converte string para long
- float(str) converte string para float
- complex(real [,imag]) cria um n. complexo
- str(x) converte um obj para string
- repr(x)
- eval(str)
- chr(x) converte um int para um caractere
- unichr(x) converte um int para um caractere unicode
- ord(n) converte um char para um int
- hex(n), oct(x) converte um número para hexadecimal, octal.
- tuple(s) list(s) set(s) dict(s) frozenset(s)

Operadores

- Aritméticos: + - / * %
- Relacionais: < <= => > == !=
- Lógicos: and or
- Bit a bit << >> & | ~ ^

A	B	and	Or
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Sintaxe geral

- Não existem delimitadores de final de instrução
- Blocos são especificados por indentação (deslocamentos a direita, indicando subordinação a instrução anterior)
- Declarações que esperam um nível de indentação terminam com dois-pontos (:)
- O símbolo # é utilizado para identificar comentário de linha, tudo que é colocado após o símbolo não será reconhecido pelo compilador
- Os valores são atribuídos da direita para esquerda
- O símbolo de igual (=) é utilizado para atribuição, enquanto que o duplo símbolo de igual (==) é utilizado para comparação

Controle do fluxo de execução

- Bloco/Instrução condicional

```
if  expressão_lógica :
    comando1
else:
    comando2
```

```
>>> if True:
        print("condição verdadeira")


condição verdadeira
```

```
>>> if False:
        print("condição verdadeira")
else:
        print("condição falsa")


condição falsa
```

```
>>> a = 5
>>> if a%2==0 :
        print("par")
else:
        print("impar")


impar
```

Repetição (loops)

- while condição :
  comandos
- Exemplo 1:

```
conta = 1
while conta<10:
    print(conta)
    conta += 1
```

- Exemplo 2:

```
condicao = True
while(condicao):
    print("BLOCO while() e condicao==True")
    condicao = False
else:
    print("BLOCO ELSE e condicao==False")
```

Laço for

- **for** iterating_var **in** sequence/list **:**
  comandos

- Exemplo

```
#!/usr/bin/python

for letter in 'Python‘   :
    print( 'Current Letter :', letter )

fruits = ['banana', 'apple', 'mango']
for fruit in fruits    :
    print( 'Current fruit :', fruit )

print( “fim“ )
```

Laço for com índice

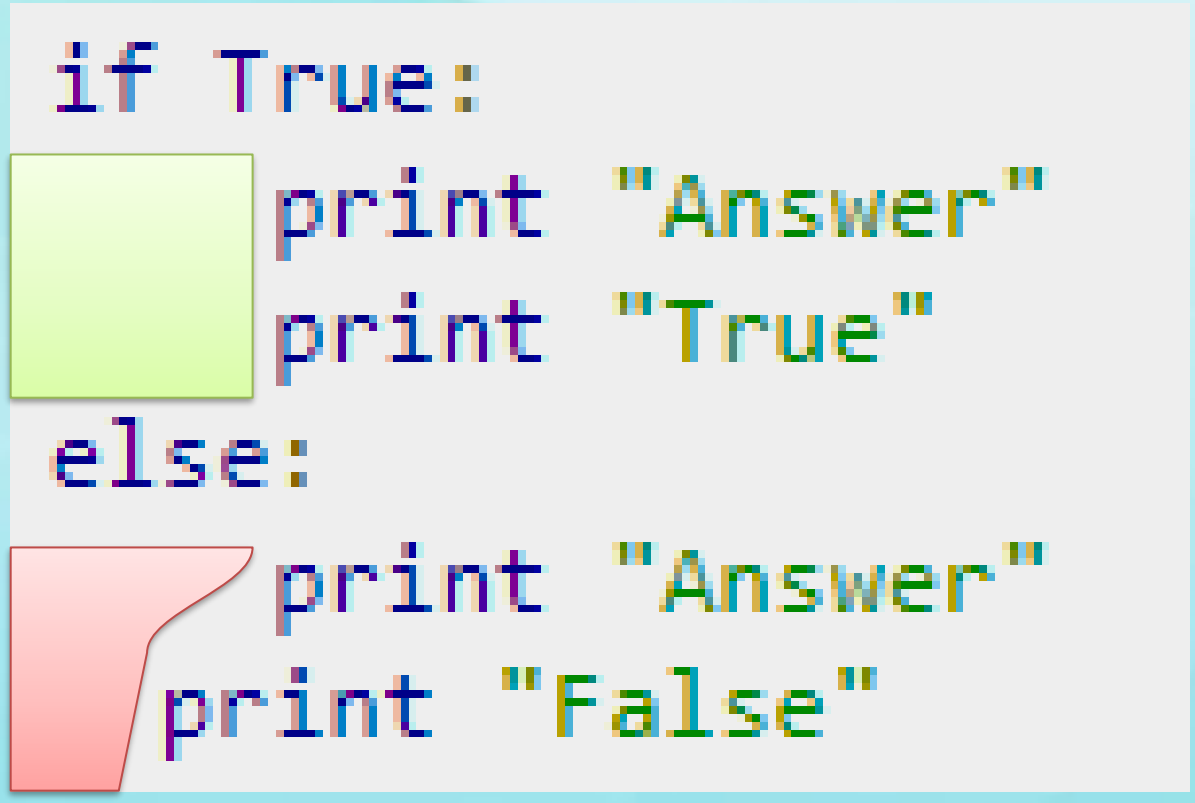
- Exemplo

```
#!/usr/bin/python

fruits = ['banana', 'apple', 'mango']
for index in range(len(fruits)):
      print ( 'Current fruit :', fruits[index]  )

print( "Good bye!“ )
```

Cuidado com a indentação...



Linhas grandes...

```
>>>total = \
      a + \
      b
>>> total
3
```

Funções de Entrada/Saída (E/S)

- print(obj)

  from help:
  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.

Funções de Entrada/Saída (E/S)

input(prompt=None, /)

From help:

- Read a string from standard input. The trailing newline is stripped.
  The prompt string, if given, is printed to standard output without a
  trailing newline before reading input.

  If the user hits EOF (*nix: Ctrl-D, Windows: Ctrl-Z+Return), raise EOFError.
  On *nix systems, readline is used if available.

Exemplos de E/S:

- Para ler String a partir do teclado:
  str = input()
- Para ler um inteiro a partir do teclado:
  num_int = int(input())
- Para ler um float a partir do teclado:
  num_float = float(input())

- Para ler um número:
  num = eval(input())

- Para descobrir o tipo de uma variável:
  type(nome)

Parâmetros da linha de comando

No shell o usuário pode entrar com os parâmetros, exemplo:

1. python test.py arg1 arg2 arg3
2. python test.py –i arq1.txt -o arq2.txt

No programa:

```
import sys
print 'Number of arguments:', len(sys.argv), 'arguments.'
print 'Argument List:', str(sys.argv)
```

Para tratar o caso 2, existe a opção “getopt.getopt ”

```
#!/usr/bin/python

import sys, getopt

def main(argv):
  inputfile = ''
  outputfile = ''
  try:
    opts, args = getopt.getopt(argv, \
        "hi:o:",["ifile=","ofile="])
  except getopt.GetoptError:
    print 'test.py -i <inputfile> -o <outputfile>'
    sys.exit(2)
  for opt, arg in opts:
    if opt == '-h':
      print 'test.py -i <inputfile> -o <outputfile>'
      sys.exit()
    elif opt in ("-i", "--ifile"):
      inputfile = arg
    elif opt in ("-o", "--ofile"):
      outputfile = arg
  print 'Input file is "', inputfile
  print 'Output file is "', outputfile
```

```
if __name__ == "__main__":
  main(sys.argv[1:])
```

Fim do arquivo test.py

Ao executar este programa:

```
$ python test.py  –h
usage: test.py -i <inputfile> -o
<outputfile>

$ test.py -i BMP -o
usage: test.py -i <inputfile> -o
<outputfile>

$ test.py -i inputfile
Input file is " inputfile
Output file is "
```

Referências

- Python Brasil
  http://python.org.br/

- TutoriaisPoint
  https://www.tutorialspoint.com/python/python_basic_syntax.htm

- Tutorial - Learn Python in 10 minutes
  https://www.stavros.io/tutorials/python/