



Python Básico

**Programação para
Internet II**

Professor: Ritomar Torquato

Python Básico: Invocação, Tipos, Operadores e Estruturas

Objetivo: Conhecer e instalar o interpretador de comandos Python; Reconhecer a versão do Python instalada; O ambiente de desenvolvimento; IDLE; Ajuda Python; Comandos básico da linguagem Python e execução de Scripts; .

Usando o Interpretador

- Obtendo e Instalando o Interpretador

Download the latest version for Windows

Download Python 3.4.3

Download Python 2.7.9

Wondering which version to use? [Here's more about the difference between Python 2 and 3.](#)

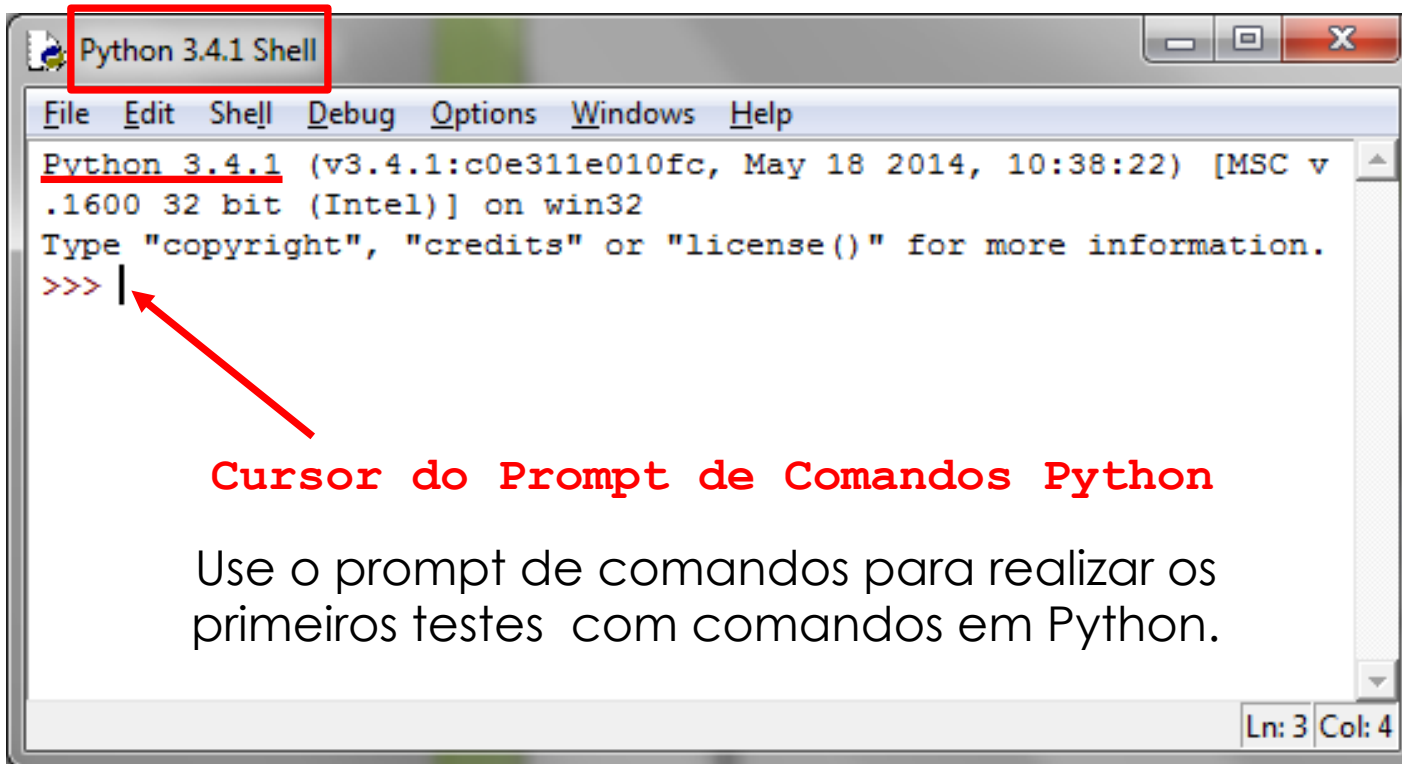
Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Pre-releases](#)

<https://www.python.org/downloads/>

Usando o Interpretador

- Shell de Comandos IDLE (Python GUI)



```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v
.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

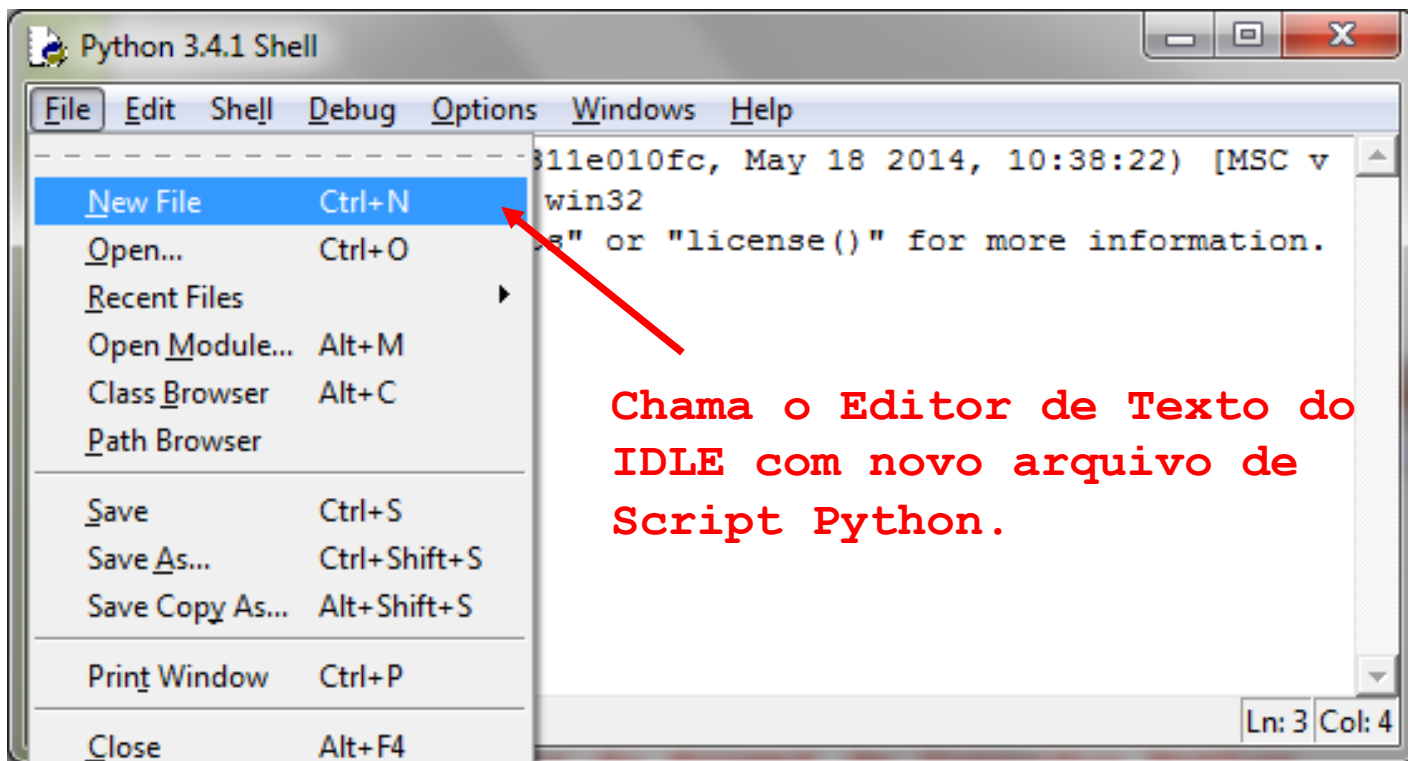
Cursor do Prompt de Comandos Python

Use o prompt de comandos para realizar os primeiros testes com comandos em Python.

Ln: 3 Col: 4

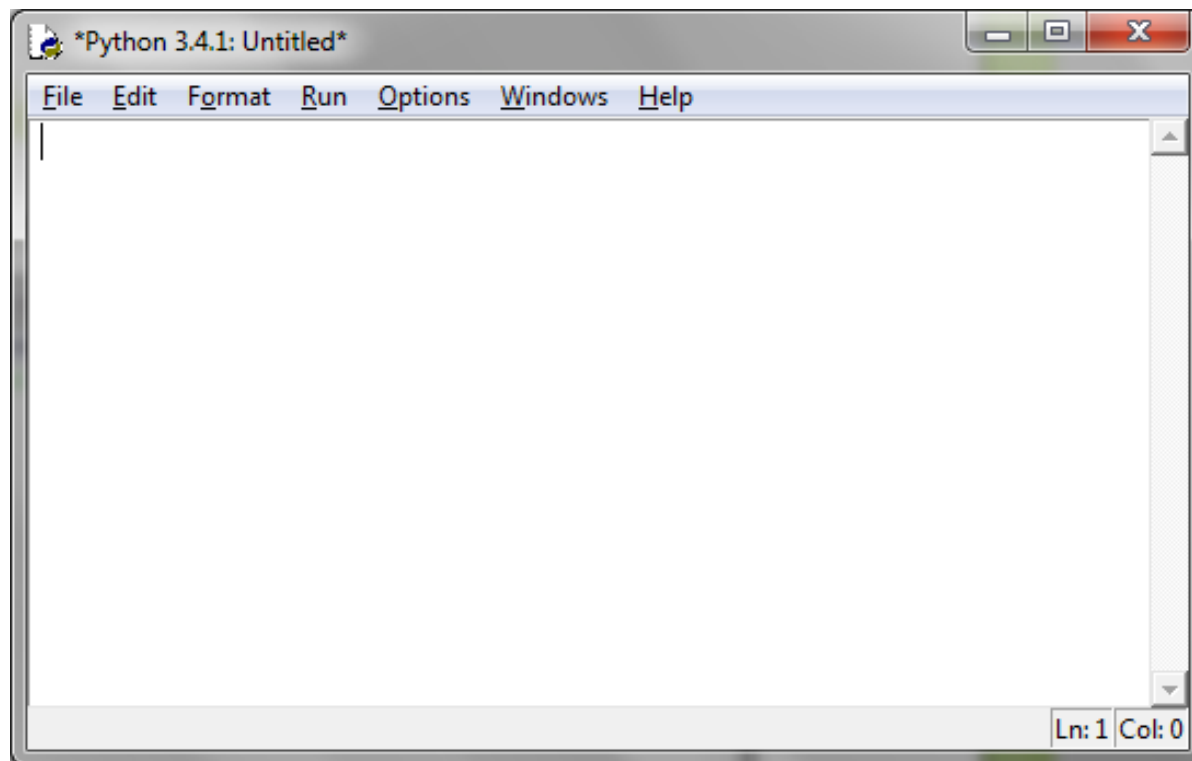
Usando o Interpretador

- Shell de Comandos IDLE (Python GUI)



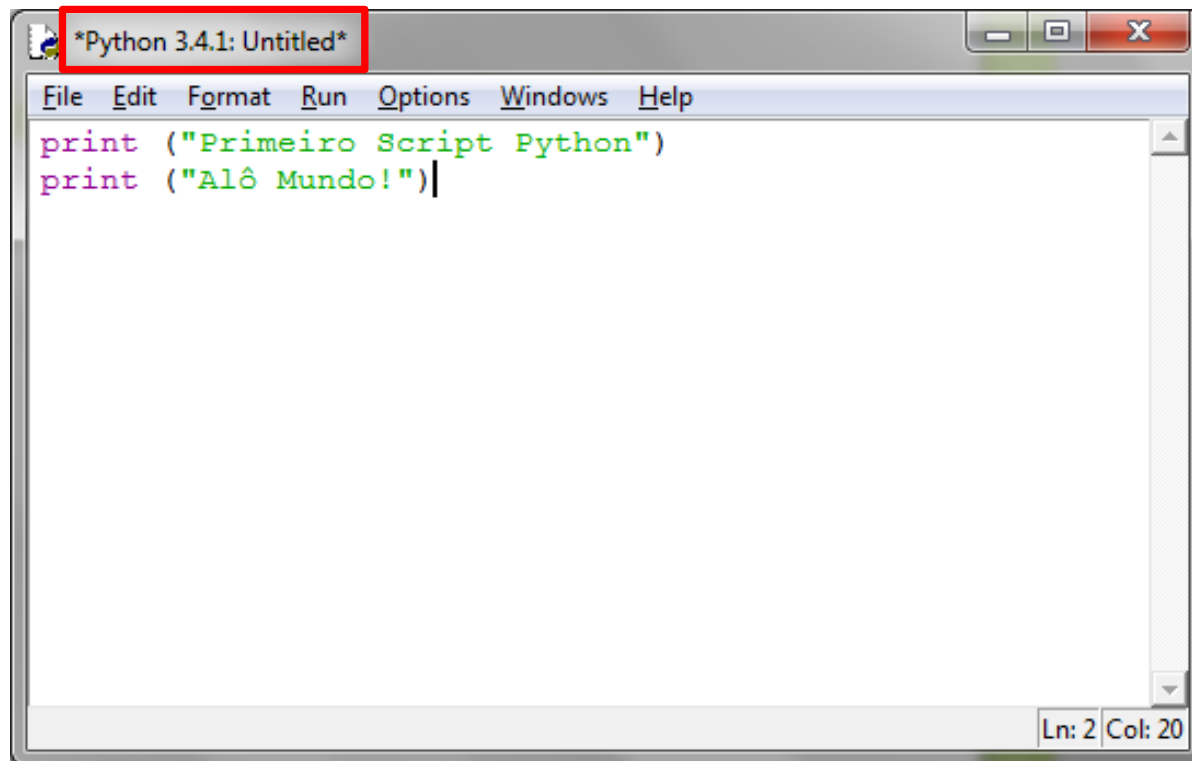
Usando o Interpretador

- O editor do IDLE



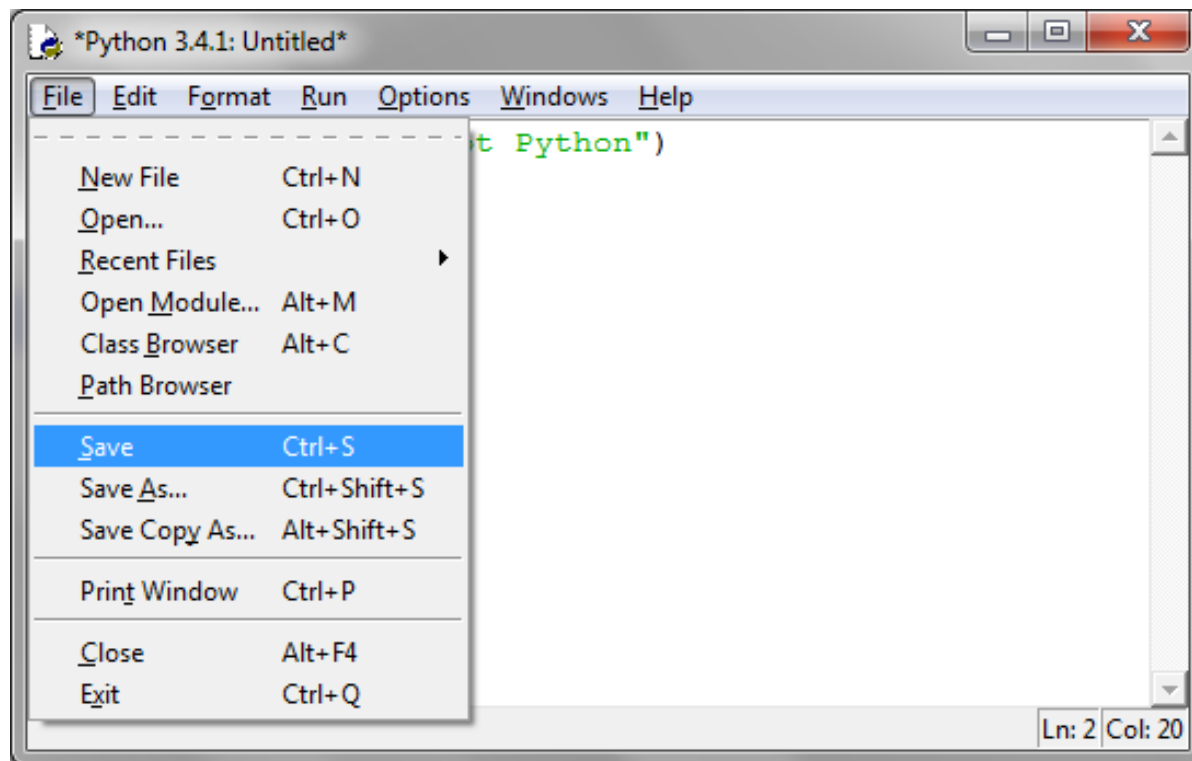
Usando o Interpretador

- Script Python



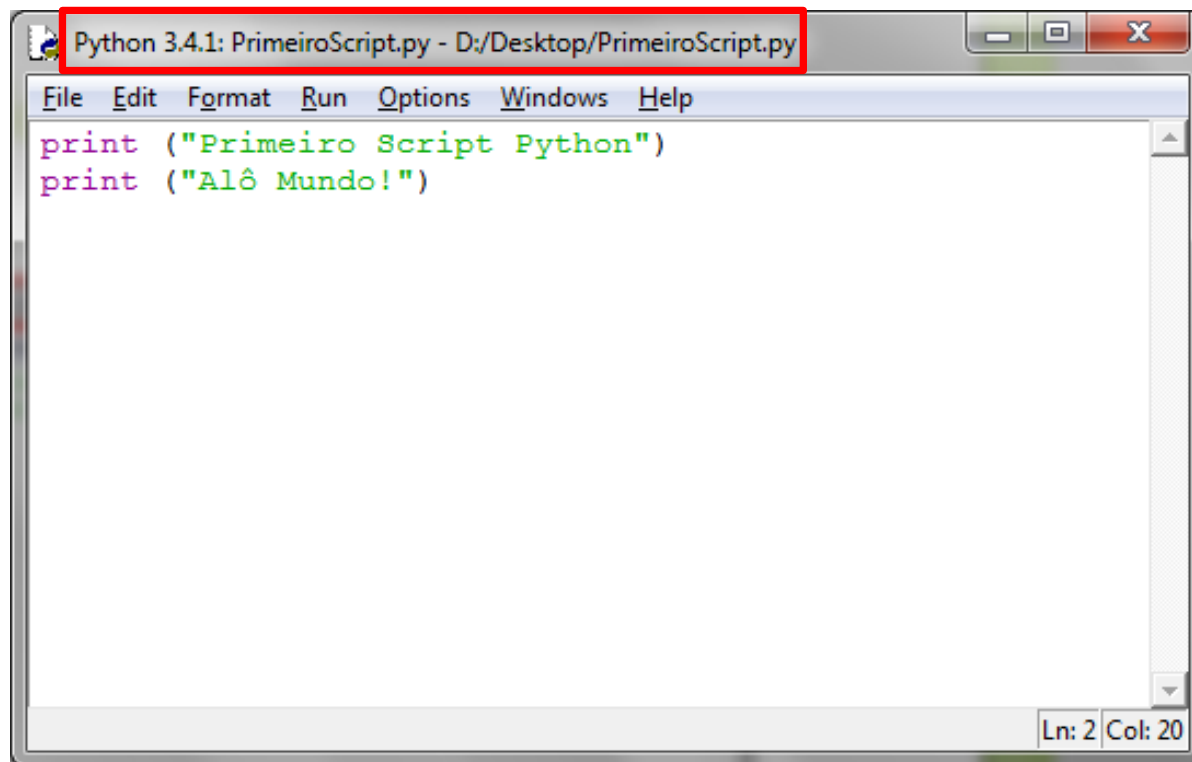
Usando o Interpretador

- Salvando o Script



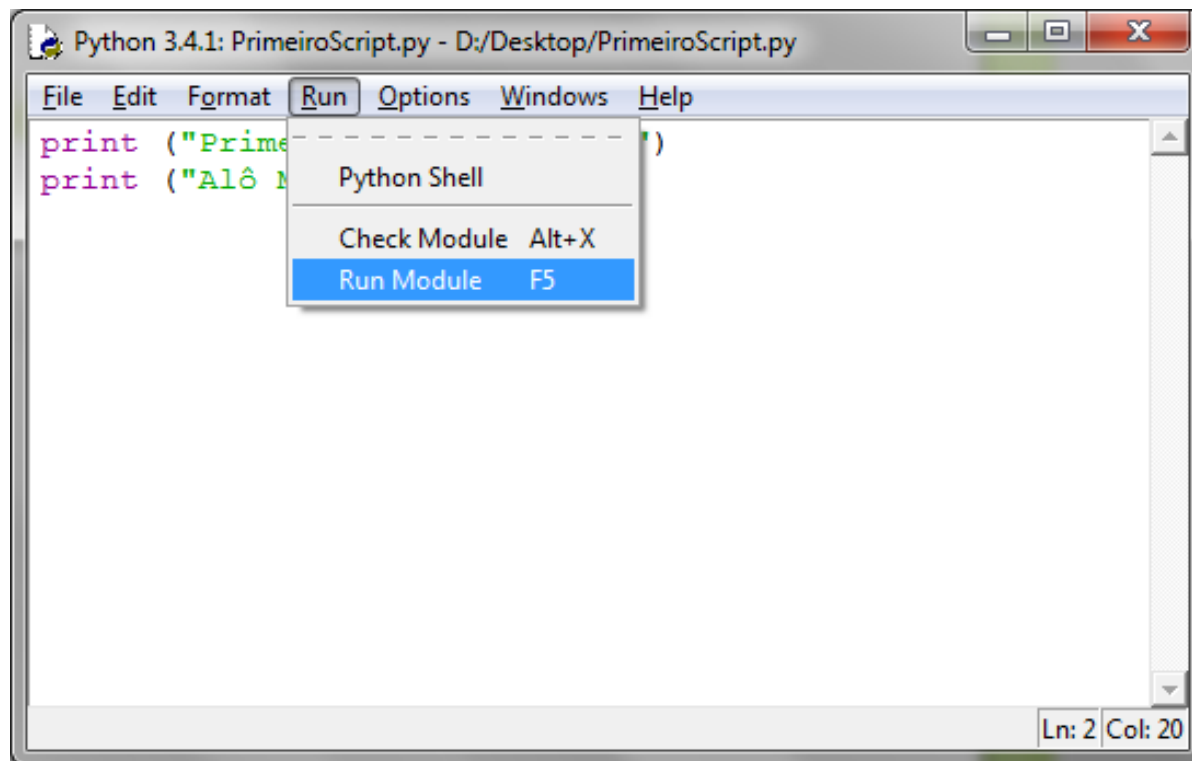
Usando o Interpretador

- Salvando o Script



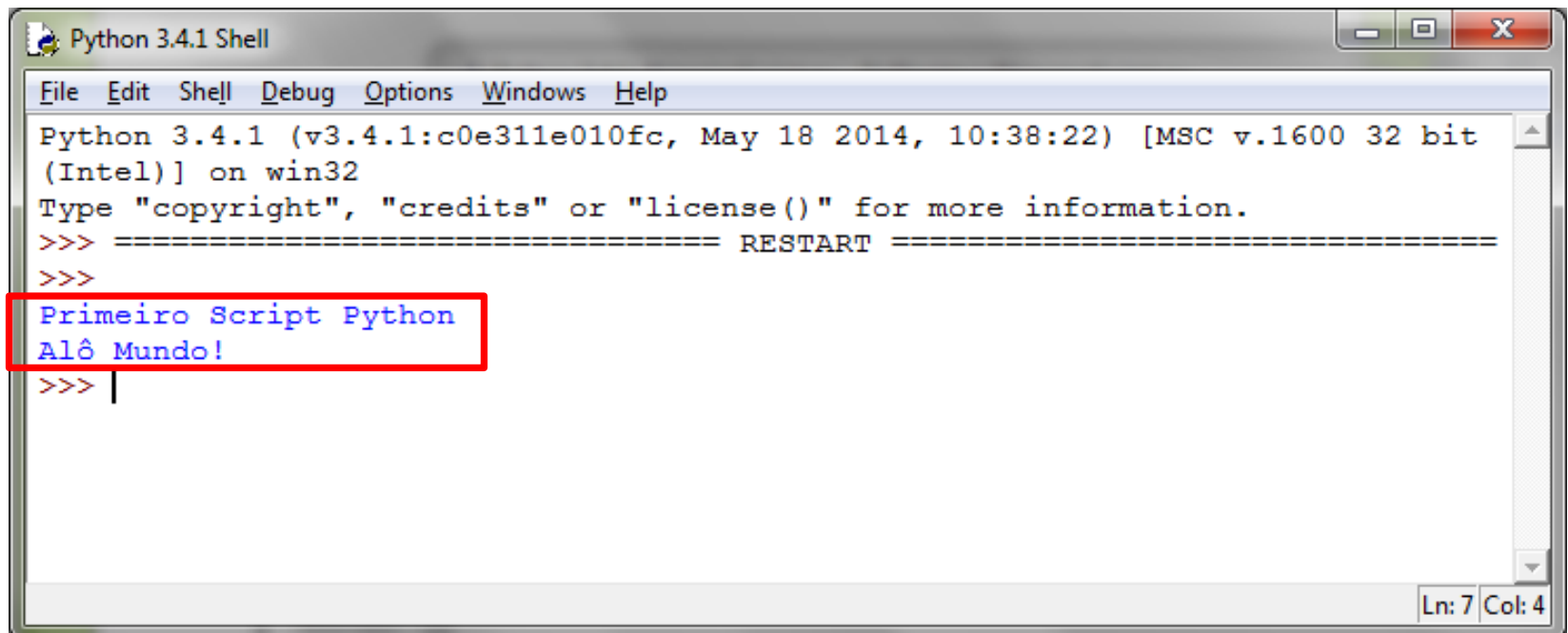
Usando o Interpretador

- Executando o Script



Usando o Interpretador

- Saída do Script no Shell de Comandos



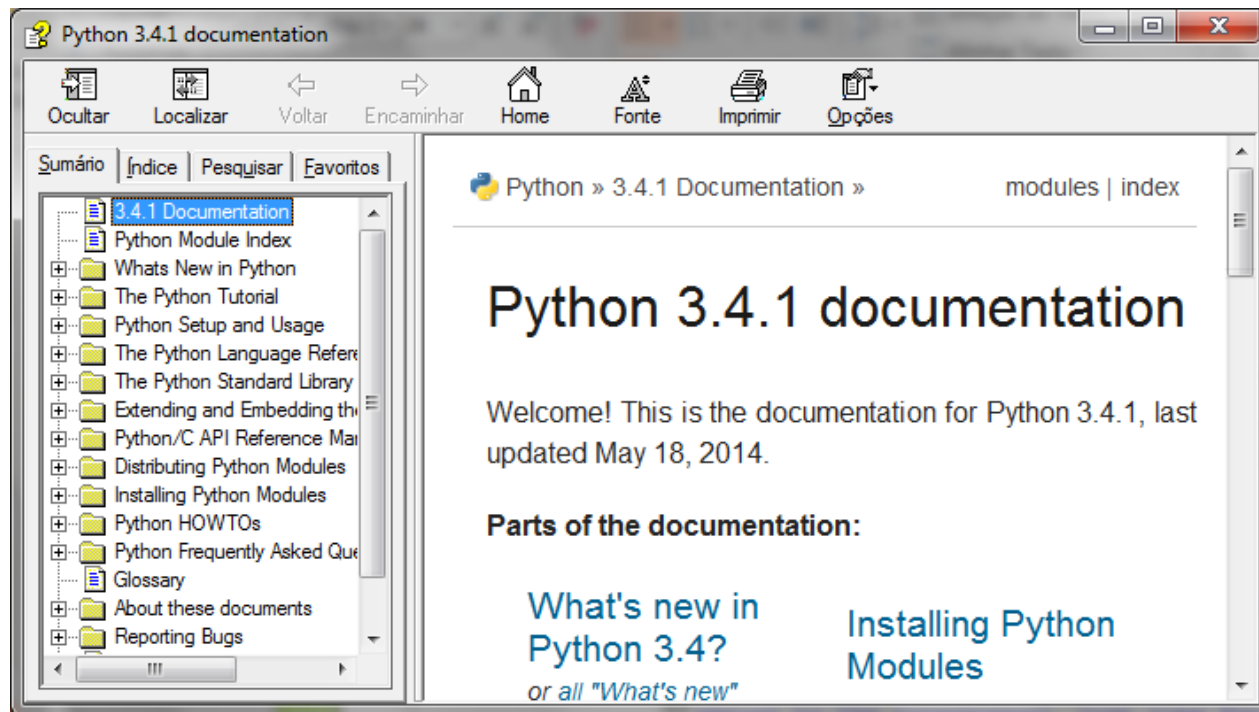
The screenshot shows a window titled "Python 3.4.1 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The main text area displays the following content:

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit  
(Intel)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>> ===== RESTART =====  
>>>  
Primeiro Script Python  
Alô Mundo!  
>>> |
```

The lines "Primeiro Script Python" and "Alô Mundo!" are highlighted with a red rectangular box. The status bar at the bottom right indicates "Ln: 7 Col: 4".

Ajuda no Python

- Offline



Shell IDLE :: Menu Help:: Python Docs (F1)

Ajudando Python

Online

Python » 3.4.3 » Documentation » modules | index

Download
Download these documents

Docs for other versions
Python 2.7 (stable)
Python 3.3 (stable)
Python 3.5 (in development)
Old versions

Other resources
PEP Index
Beginner's Guide
Book List
Audio/Visual Talks

Quick search

Enter search terms or a module, class or function name.

Python 3.4.3 documentation

Welcome! This is the documentation for Python 3.4.3, last updated Mar 02, 2015.

Parts of the documentation:

- [What's new in Python 3.4?](#)
or all "What's new" documents since 2.0
- [Tutorial](#)
start here
- [Library Reference](#)
keep this under your pillow
- [Language Reference](#)
describes syntax and language elements
- [Python Setup and Usage](#)
how to use Python on different platforms
- [Python HOWTOs](#)
in-depth documents on specific topics
- [Installing Python Modules](#)
installing from the Python Package Index & other sources
- [Distributing Python Modules](#)
publishing modules for installation by others
- [Extending and Embedding](#)
tutorial for C/C++ programmers
- [Python/C API](#)
reference for C/C++ programmers
- [FAQs](#)
frequently asked questions (with answers!)

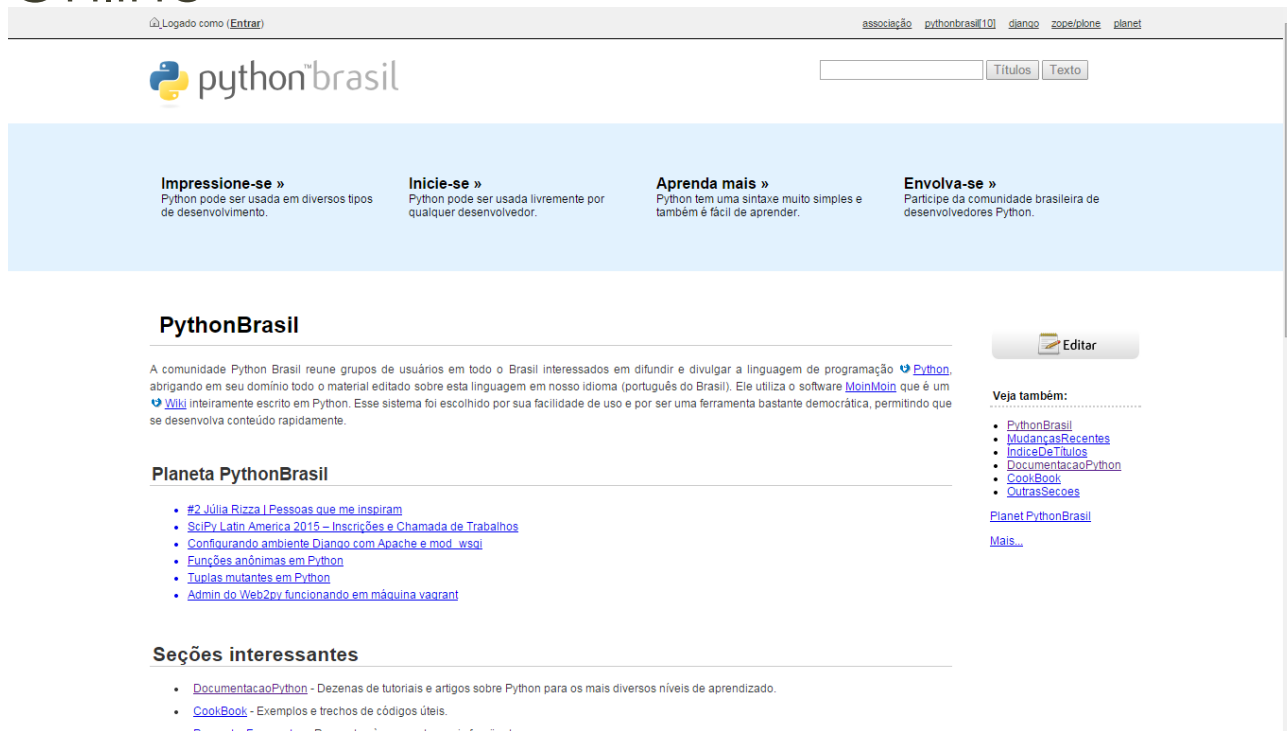
Indices and tables:

- [Global Module Index](#)
quick access to all modules
- [Search page](#)
search this documentation

docs.python.org


Ajudando no Python

Online



The screenshot shows the PythonBrasil website. At the top, there's a navigation bar with links: associação, pythonbrasil[10], django, zope/plone, and planet. Below this is the PythonBrasil logo and a search bar with buttons for 'Títulos' and 'Texto'. A light blue banner contains four sections: 'Impressione-se »' (Python can be used in various development types), 'Inicie-se »' (Python can be used freely by any developer), 'Aprenda mais »' (Python has a very simple syntax and is easy to learn), and 'Envolve-se »' (Join the Brazilian Python community). Below the banner, the 'PythonBrasil' section describes the community's goal to spread Python knowledge in Portuguese, mentioning the use of MoinMoin. The 'Planeta PythonBrasil' section lists various resources like tutorials, conferences, and books. The 'Seções interessantes' section highlights documentation, code examples, and a mailing list.

Logado como (Entrar) associação pythonbrasil[10] django zope/plone planet

 pythonbrasil

Títulos Texto

Impressione-se »
Python pode ser usada em diversos tipos de desenvolvimento.

Inicie-se »
Python pode ser usada livremente por qualquer desenvolvedor.

Aprenda mais »
Python tem uma sintaxe muito simples e também é fácil de aprender.

Envolve-se »
Participe da comunidade brasileira de desenvolvedores Python.

PythonBrasil


A comunidade Python Brasil reúne grupos de usuários em todo o Brasil interessados em difundir e divulgar a linguagem de programação [Python](#), abrigando em seu domínio todo o material editado sobre esta linguagem em nosso idioma (português do Brasil). Ele utiliza o software [MoinMoin](#) que é um [Wiki](#) inteiramente escrito em Python. Esse sistema foi escolhido por sua facilidade de uso e por ser uma ferramenta bastante democrática, permitindo que se desenvolva conteúdo rapidamente.

Planeta PythonBrasil

- #2 Júlia Rizza | Pessoas que me inspiram
- SciPy Latin America 2015 – inscrições e Chamada de Trabalhos
- Configurando ambiente Django com Apache e mod_wsgi
- Funções anônimas em Python
- Tuplas mutantes em Python
- Admin do Web2py funcionando em máquina vagrant

Seções interessantes

- [DocumentaçãoPython](#) - Dezenas de tutoriais e artigos sobre Python para os mais diversos níveis de aprendizado.
- [CookBook](#) - Exemplos e trechos de códigos úteis.
- [DiscussãoEmPortugues](#) - Espaço para discussão mais formal.

 Editar

Veja também:

- [PythonBrasil](#)
- [MudançasRecentes](#)
- [ÍndiceDeTítulos](#)
- [DocumentaçãoPython](#)
- [CookBook](#)
- [OutrasSeções](#)

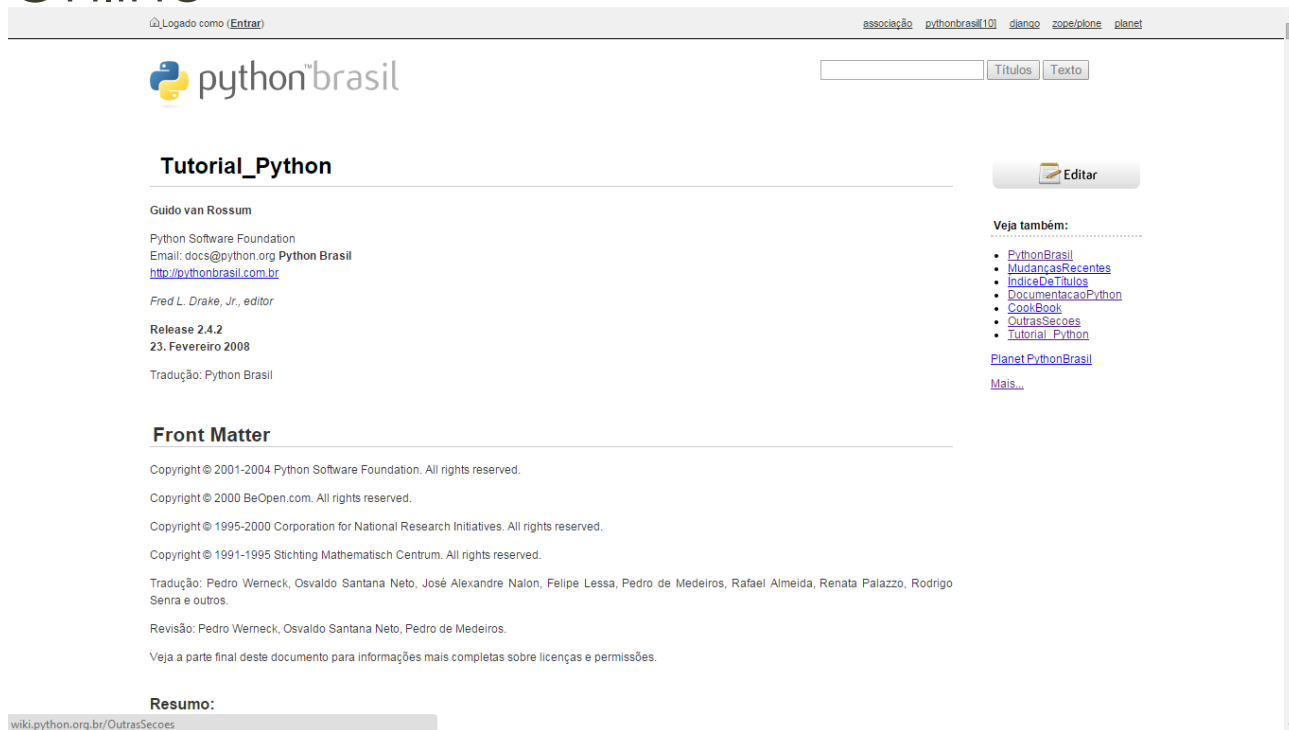
[Planet PythonBrasil](#)

[Mais...](#)

wiki.python.org.br

Ajudar no Python

Online

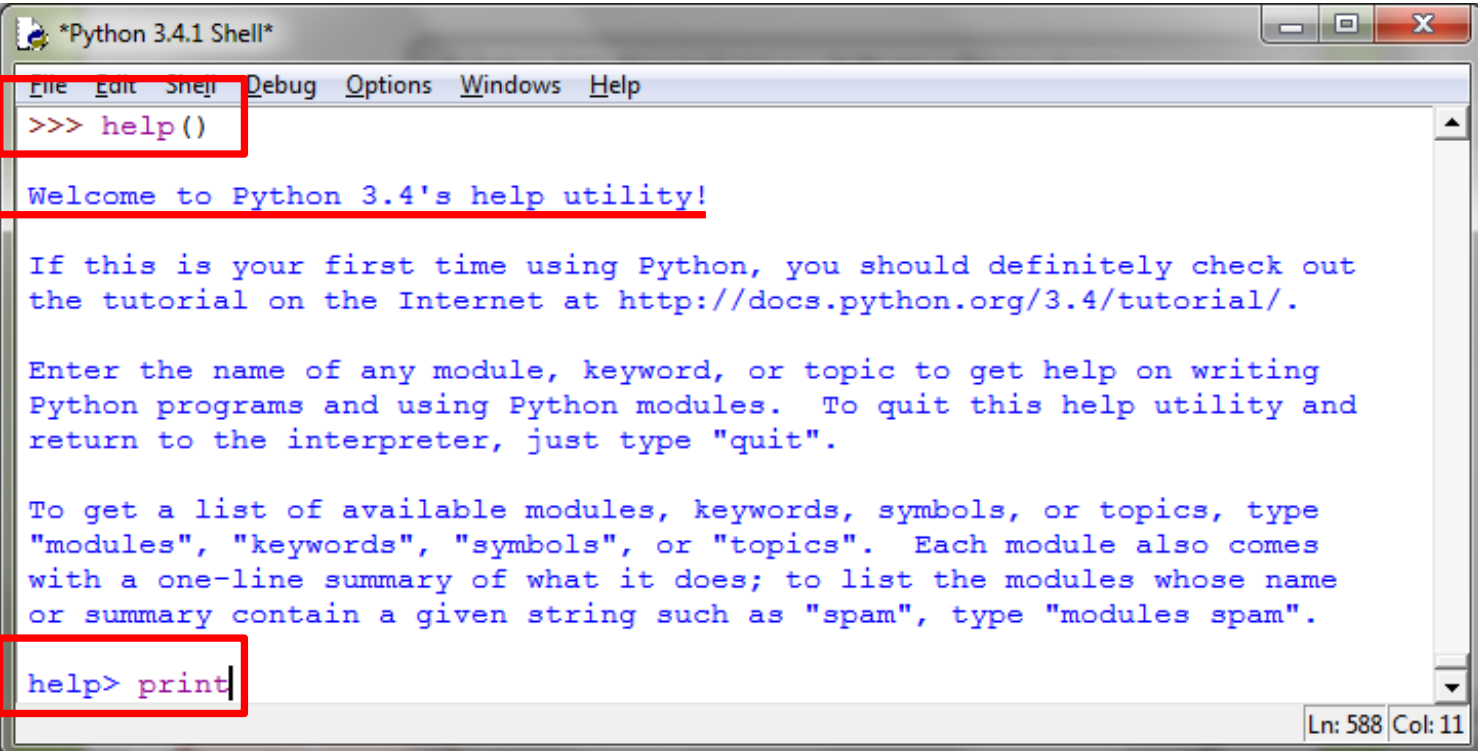


The screenshot shows the Python Brasil website. At the top, there is a navigation bar with links: [associação](#), [pythonbrasil\[10\]](#), [django](#), [zope/plone](#), and [planet](#). Below the navigation bar is the Python logo and the text "pythonbrasil". To the right of the logo is a search bar with a "Títulos" button and a "Texto" button. The main content area is titled "Tutorial_Python" and features a "Editar" button. The text on the page includes: "Guido van Rossum", "Python Software Foundation", "Email: docs@python.org Python Brasil", "<http://pythonbrasil.com.br>", "Fred L. Drake, Jr., editor", "Release 2.4.2", "23. Fevereiro 2008", "Tradução: Python Brasil", "Front Matter", "Copyright © 2001-2004 Python Software Foundation. All rights reserved.", "Copyright © 2000 BeOpen.com. All rights reserved.", "Copyright © 1995-2000 Corporation for National Research Initiatives. All rights reserved.", "Copyright © 1991-1995 Stichting Mathematisch Centrum. All rights reserved.", "Tradução: Pedro Werneck, Osvaldo Santana Neto, José Alexandre Nalon, Felipe Lessa, Pedro de Medeiros, Rafael Almeida, Renata Palazzo, Rodrigo Senra e outros.", "Revisão: Pedro Werneck, Osvaldo Santana Neto, Pedro de Medeiros.", "Veja a parte final deste documento para informações mais completas sobre licenças e permissões.", "Resumo:", and a link to wiki.python.org.br/OutrasSecoes. On the right side, there is a "Veja também:" section with links to [PythonBrasil](#), [MudancasRecentes](#), [IndiceDeTitulos](#), [DocumentacaoPython](#), [CookBook](#), [OutrasSecoes](#), [Tutorial_Python](#), [PlanetPythonBrasil](#), and [Mais...](#).

wiki.python.org.br/Tutorial_Python

Ajuda no Python

- Linha de comando



The screenshot shows a window titled "*Python 3.4.1 Shell*" with a menu bar (File, Edit, Shell, Debug, Options, Windows, Help). The main text area displays the output of the `help()` command. The first line, `>>> help()`, is highlighted with a red box. The output text includes a welcome message, a tutorial link, and instructions on how to use the help utility. The prompt `help> print` is also highlighted with a red box. The status bar at the bottom right shows "Ln: 588 Col: 11".

```
*Python 3.4.1 Shell*
File Edit Shell Debug Options Windows Help
>>> help()

Welcome to Python 3.4's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/3.4/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

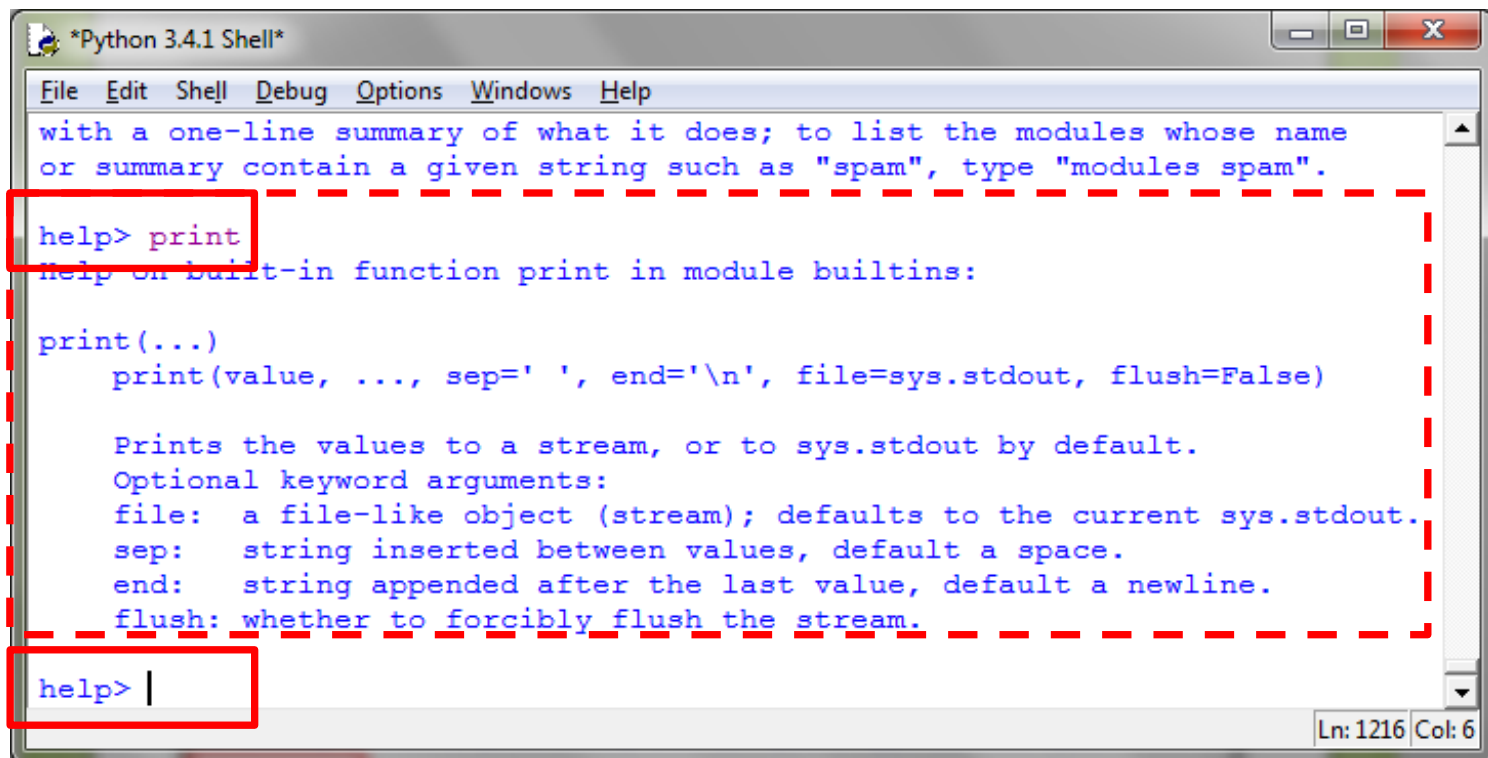
To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> print
```

Ln: 588 Col: 11

Ajuda no Python

- Linha de comando



The screenshot shows a Python 3.4.1 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Windows, Help). The main text area displays the help for the `print` function. A red dashed rectangle highlights the command `help> print` and the subsequent help text. Another red dashed rectangle highlights the prompt `help> |` at the bottom. The status bar at the bottom right indicates 'Ln: 1216 Col: 6'.

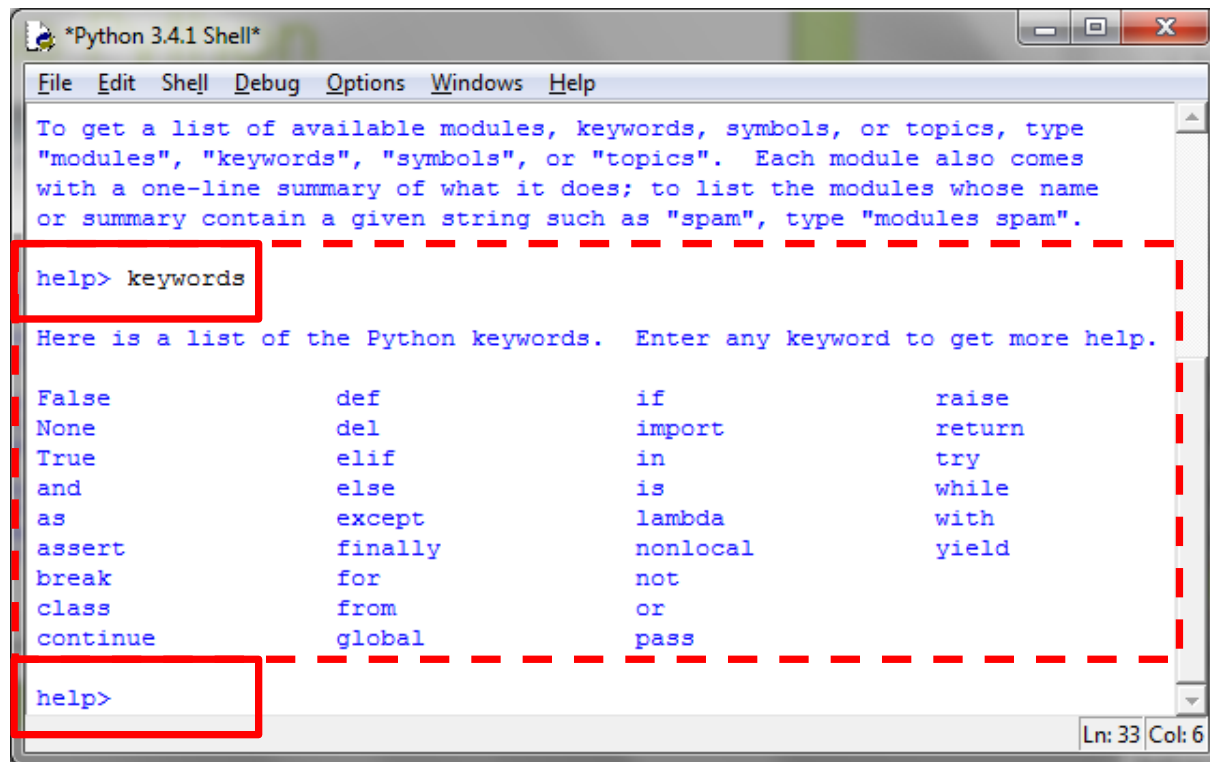
```
*Python 3.4.1 Shell*
File Edit Shell Debug Options Windows Help
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".
help> print
help on built-in function print in module builtins:

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.
help> |
Ln: 1216 Col: 6
```

Ajuda no Python

- Linha de comando



The screenshot shows a terminal window titled '*Python 3.4.1 Shell*'. The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The main text area contains a blue instruction: 'To get a list of available modules, keywords, symbols, or topics, type "modules", "keywords", "symbols", or "topics". Each module also comes with a one-line summary of what it does; to list the modules whose name or summary contain a given string such as "spam", type "modules spam".' Below this, the command 'help> keywords' is entered and highlighted with a red box. The output is a list of Python keywords arranged in four columns, also enclosed in a red dashed box. At the bottom, the command 'help>' is entered and highlighted with a red box. The status bar at the bottom right shows 'Ln: 33 Col: 6'.

```
*Python 3.4.1 Shell*
File Edit Shell Debug Options Windows Help

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics". Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> keywords

Here is a list of the Python keywords. Enter any keyword to get more help.

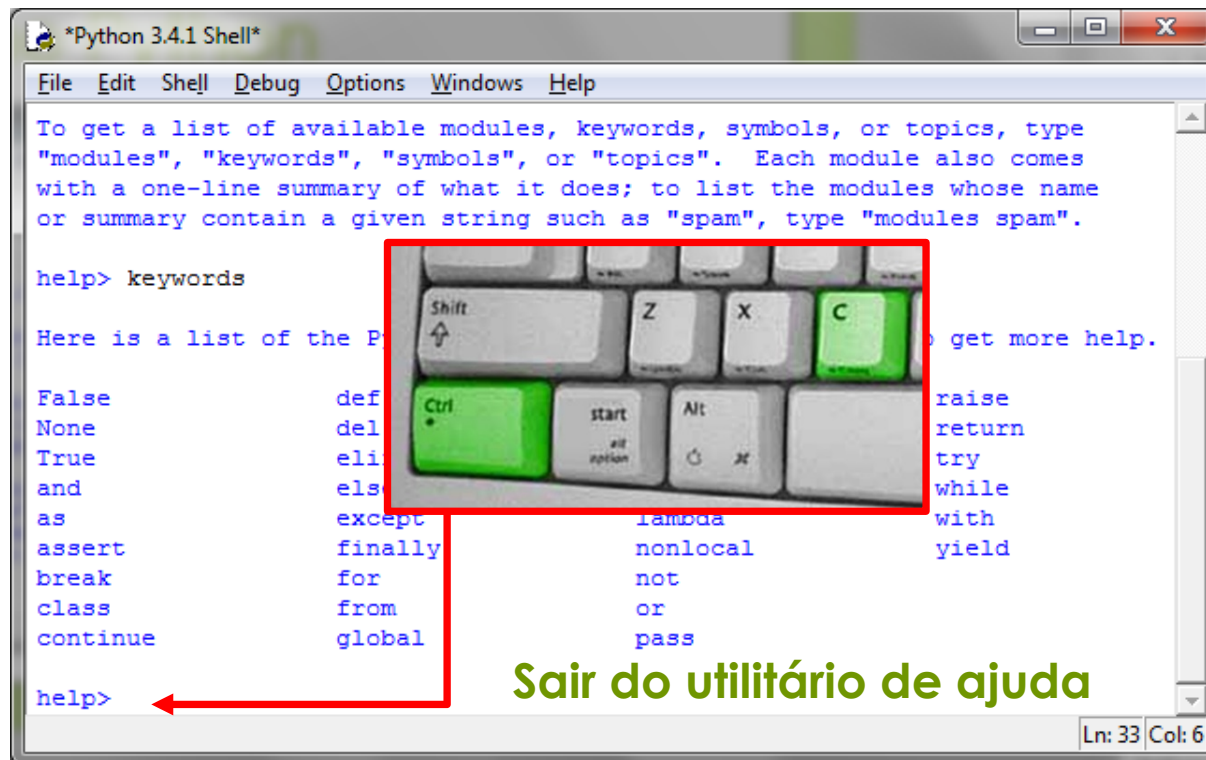
False      def         if          raise
None       del         import      return
True       elif        in          try
and        else        is          while
as         except     lambda     with
assert     finally   nonlocal   yield
break     for        not
class     from       or
continue  global    pass

help>
```

Ln: 33 Col: 6

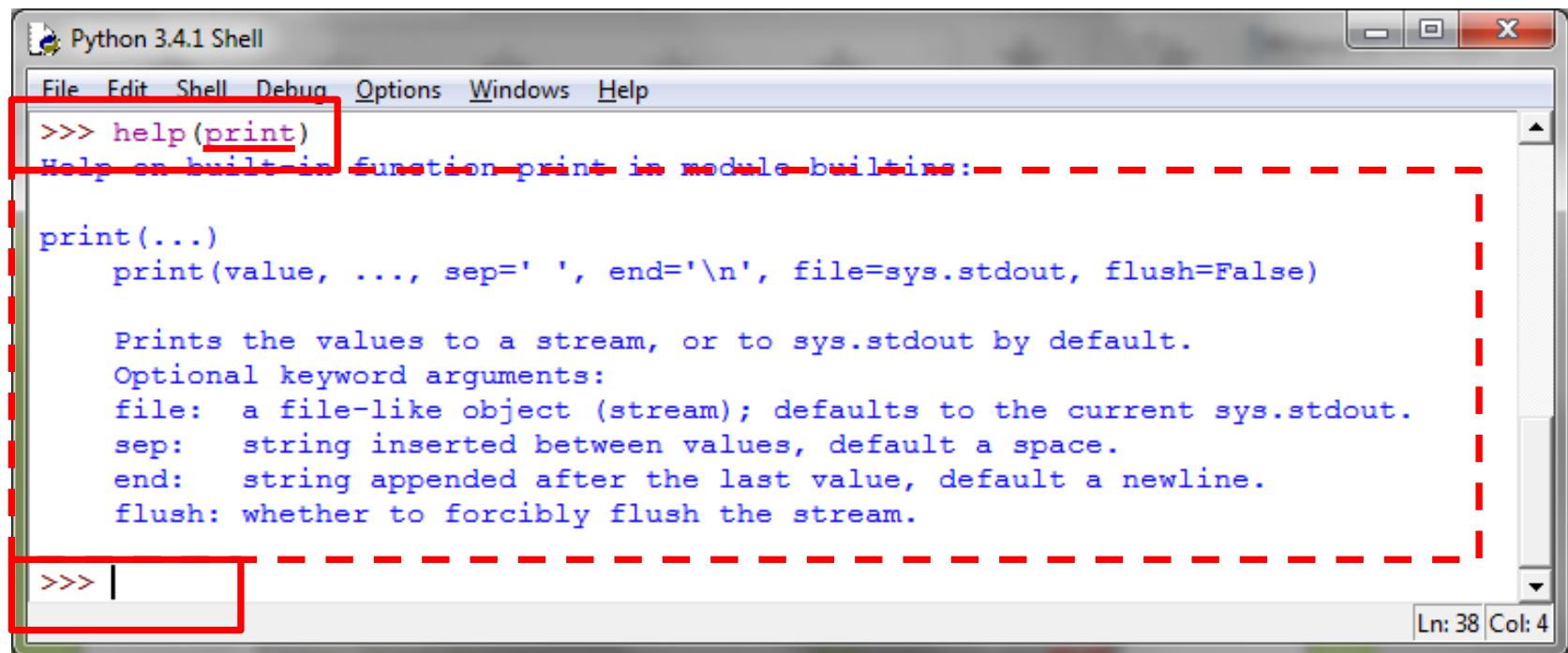
Ajuda no Python

- Linha de comando



Ajuda no Python

- Linha de comando



The screenshot shows a Python 3.4.1 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Windows, Help). The command prompt shows `>>> help(print)` and the output displays the help text for the `print` function. A red dashed box highlights the entire help output area. A red solid box highlights the command prompt input area at the bottom left, showing `>>> |`. The status bar at the bottom right indicates 'Ln: 38 Col: 4'.

```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
>>> 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.

>>> |
```

Ln: 38 Col: 4

Comentários no Python

- Comentários de uma linha

```
print ("Aqui não é comentário.") #Aqui é um comentário de linha  
#Aqui também é um comentário de linha
```

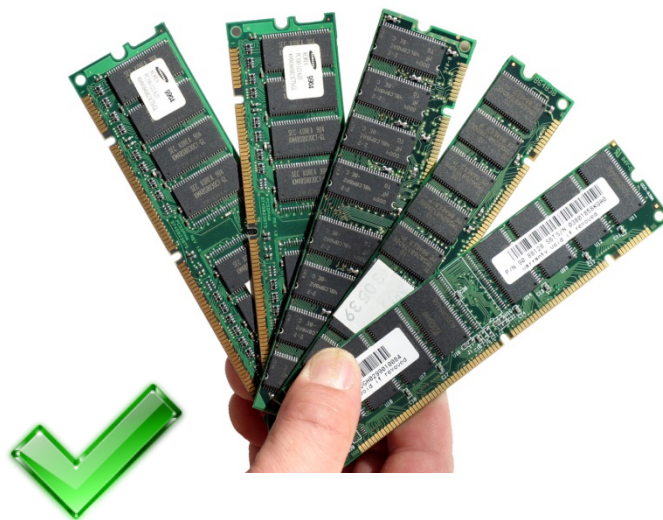
- Comentários de várias linhas (DocStrings)

```
"""  
    Este é um comentário  
    com várias linhas ou DocStrings.  
"""
```

Não iremos detalhar DocStrings, por hora basta saber que podem ser usadas como comentários de multi-linha.

Variáveis

- São espaços identificados na memória para armazenar um dado de um determinado tipo, previamente especificados.



Variáveis

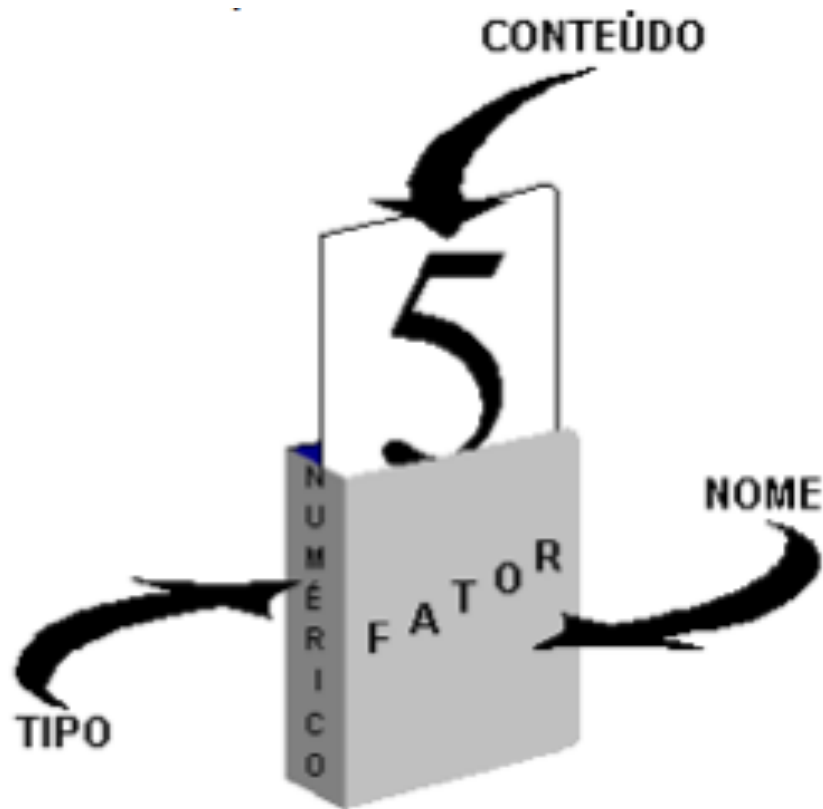
- Como a memória é organizada?

Endereço	Dado	Endereço	Dado	Endereço	Dado
0000		0008		0016	
0001		0009		0017	
0002		0010		0018	
0003		0011		0019	
0004		0012 ←	8 bits	→ 0020	
0005		0013		0021	
0006		0014		0022	
0007		0015		...	

Qual o maior valor inteiro que pode ser armazenado em cada espaço?

Variáveis

- Espaço na memória para armazenar um conteúdo.
- Identificador
- Tipo de dado
- Conteúdo



Python tem **tipagem dinâmica**, isso significa que o tipo na caixa muda de acordo com o seu conteúdo.

Variáveis

`nome = conteúdo`

`fator = 5`

Endereço	Dado
...	?
0065	?
0066	5
0067	
0068	
0069	
0070	?
...	?

} Fator

Comandos de Entrada e Saída

- Print: Exibe uma informação na tela. No Python 3 deixou de fazer parte da linguagem (keyword) e tornou-se uma **função**. Sendo assim, é obrigatório o uso de parênteses.

```
#Python 2  
print "Alô Mundo!"
```

```
#Python 3  
print ("Alô Mundo!")
```

Comandos de Entrada e Saída

- Input: função que ler informações digitadas pelo teclado e guarda sempre como sendo do tipo **string**.

```
>>> nome = input("Nome: ")
Nome: Maria
>>> type(nome)
<class 'str'>
>>> idade = input("Idade: ")
Idade: 20
>>> type(idade)
<class 'str'>
```

Comandos de Entrada e Saída

- Input: função que ler informações digitadas pelo teclado e guarda sempre como sendo do tipo **string**.

```
>>> idade = idade + 5
Traceback (most recent call last):
  File "<pyshell#59>", line 1, in <module>
    idade = idade + 5
TypeError: Can't convert 'int' object to str implicitly
>>> idade = idade + "5"
>>> print(idade)
205
>>> type(idade)
<class 'str'>
```

Comandos de Entrada e Saída

- Conversão Explícita

```
>>> idade = input("Idade: ")
Idade: 20
>>> type(idade)
<class 'str'>
>>> idade = int(idade)
>>> type(idade)
<class 'int'>
>>> idade = int(input("Idade: "))
Idade: 25
>>> type(idade)
<class 'int'>
```

Comandos de Entrada e Saída

- Tratando erros de Entrada de Dados

```
try:
```

```
    idade = int(input("Digite sua idade: "))  
    print("Sua idade em 5 anos será:", idade + 5)
```

```
except:
```

```
    print("Você não digitou um valor numérico.")
```

Comandos de Entrada e Saída

- Tratando erros de Entrada de Dados

`try:` Tenta fazer tudo que estiver neste alinhamento (endentação).

```
idade = int(input("Digite sua idade: "))  
print("Sua idade em 5 anos será:", idade + 5)
```

`except:`

```
print("Você não digitou um valor numérico.")
```

Comandos de Entrada e Saída

- Tratando erros de Entrada de Dados

try:

```
idade = int(input("Digite sua idade: "))  
print("Sua idade em 5 anos será:", idade + 5)
```

except:

Se algo der errado, executa essas linhas.

```
print("Você não digitou um valor numérico.")
```


Comandos de Entrada e Saída

- Tratando erros de Entrada de Dados

```
>>>
```

```
Digite sua idade: 20
```

```
Sua idade em 5 anos será: 25
```

```
>>>
```

```
Digite sua idade: Vinte
```

```
Você não digitou um valor numérico.
```

Prática

- Leia dois números inteiros e imprima de volta na tela
- Leia um valor em real (R\$), calcule e escreva 70% deste valor.

Prática

- Leia um número inteiro e imprima na tela seu antecessor e o seu sucessor.
- Mostre o produto entre dois números informados pelo usuário.
- Mostre o triplo de um número informados pelo usuário.
- Leia nome, endereço e telefone e imprima na tela.

Tipos de dados

- São categorias de valores que são processados de forma semelhante:
 - Por exemplo, números inteiros são processados de forma diferente dos números de ponto flutuante (decimais) e dos números complexos.

Tipos de dados

- Tipos primitivos: são aqueles já embutidos no núcleo da linguagem
 - Simples: numéricos (int, float, bool e complex) e cadeias de caracteres (strings);
 - Compostos: listas, dicionários, tuplas e conjuntos;
- Tipos definidos pelo usuário: são correspondentes a classes (orientação objeto).

Tipos Numéricos

- Tipos numéricos Inteiros:
 - **Inteiros (int)**: número integral de tamanho ilimitado, sujeito apenas a disponibilidade de memória virtual.
 - -30, -5, 0, 1 , 2 , 15 , 19
 - **Boleanos (bool)**: False e True.
 - igual a zero: False;
 - diferente de zero: True.

Tipos Numéricos

- Tipo Real:

- **Ponto Flutuante(float)**: número real de precisão dupla.

- -30.6, -5.0, 0.0, 1.2 , 2.6 , 15.0 , 19.2

Strings

- São cadeias de caracteres
- Constantes string são escritas usando aspas simples ou duplas
 - Exemplo: "a" ou 'a'
- O operador "+" pode ser usado para concatenar strings
 - Exemplo: "a"+"b" é o mesmo que "ab"
- O operador "*" pode ser usado para repetir strings
- Exemplo: "ab"*3 é o mesmo que "ababab"

Strings

- Uma String é uma sequência imutável, não pode ter parte do seu valor alterado, como mostrado abaixo:

```
>>> TesteString = "Sistemas"
>>> print (TesteString)
Sistemas
>>> TesteString[3] = "T"
Traceback (most recent call last):
  File "<pyshell#46>", line 1, in <module>
    TesteString[3] = "T"
TypeError: 'str' object does not support item
assignment
```

Strings

- Python usa a tabela de caracteres default do S.O.
Exemplo: ASCII, UTF8
- Caracteres não imprimíveis podem ser expressos usando notação “barra invertida” (\)
 - \n é o mesmo que nova linha
 - \r é o mesmo que retorno de carro
 - \t é o mesmo que tabulação
 - \b é o mesmo que backspace
 - \\ é o mesmo que \
 - \x41 é o mesmo que o caractere cujo código hexadecimal é 41 (“A” maiúsculo)

Strings

- Uma String é uma sequência de letras endereçadas de tal forma que você possa requisitar qualquer uma delas.

```
>>> palavra = "laranja"
>>> palavra[2]
'r'
>>> palavra[2]*3
'rrr'
```

Strings

- Podemos entender uma string como uma sequencia de blocos, onde cada letra ocupa uma posição:

L	A	R	A	N	J	A
0	1	2	3	4	5	6

Strings

- Você também pode solicitar um intervalo de uma sequência, por exemplo, para solicitar o intervalo de 3 a 7:

```
>>> palavra = "laranja"  
>>> palavra[3:7]  
'anja'
```

Importante: O intervalo selecionado é ABERTO no final, sendo assim, o último caractere não é retornado.

Determinando o tipo de uma Variável

- Deve-se utilizar a função `type()`

```
>>> a = 10
>>> type (a)
<class 'int'>
>>> s = "IFPI"
>>> type (s)
<class 'str'>
>>> b = False
>>> type (b)
<class 'bool'>
```

Operadores

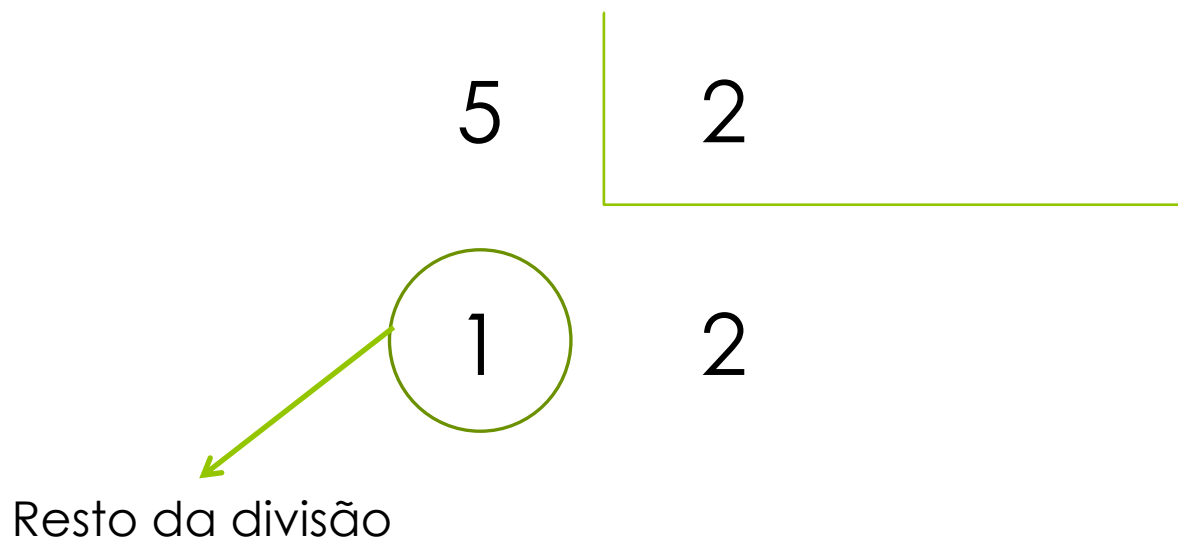
- Atuam sobre operandos e produzem um resultado. Exemplo: $3 + 2$
 - Os números (3 e 2) são os operandos
 - O operador (+) que representa adição

Operadores Aritméticos

Operador	Função
+	Adição
-	Subtração
*	Multiplicação
/	Divisão (Real)
//	Divisão (Inteira)
**	Exponenciação
%	Módulo (Resto da Divisão)

Operadores Aritméticos

- Operador Módulo



Operadores Aritméticos

- Operador Módulo: retorna o resto da divisão inteira ou real. Por exemplo:
 - `numInt = 11 % 2`
 - `numFloat = 8.5 % 3.2`
- Resulta em:
 - `num = 1`
 - `numDouble = 2.09999999999999996`

Operador de Atribuição

- Usado para definir o conteúdo de uma variável
- Em Python, o operador de atribuição é representado pelo símbolo “=” (igual).
- A sintaxe básica de um comando de atribuição é:
 - **variavel = expressão**

Leia: variável recebe expressão

Operador de Atribuição

- Exemplos de atribuição em Python:

```
# a variável ch recebe o valor 'a'
```

```
ch = 'a'
```

```
# a variável b recebe o valor 10
```

```
b = 10
```

```
# a variável s recebe o valor 155
```

```
s = 155
```

```
# a variável i recebe o valor 100
```

```
i = 100
```

Atribuição Composta ou Sobrecarregada

Operador	Função
+=	Atribuição com adição
-=	Atribuição com subtração
*=	Atribuição com multiplicação
/=	Atribuição com divisão
%=	Atribuição com módulo

Atribuição Composta

- Em resumo, os operadores aritméticos possuem um operador de atribuição correspondente:
 - Exemplo: $A = A + 2 \leftrightarrow A += 2$
- Outros Exemplos

Expressão	Forma compacta
$x = x + y$	$x += y$
$x = x - y$	$x -= y$
$x = x * y$	$x *= y$
$x = x / y$	$x /= y$
$x = x \% y$	$x \% = y$

Operadores Relacionais

Operador	Função
==	Igual
!=	Diferente
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a

Operadores Relacionais

```
a = 10  
b = 12  
c = 10
```

```
b1 = (a == b)  
b2 = (a == c)  
b3 = (a != b)
```

```
print("a =", a)           # a = 10  
print("b =", b)           # b = 12  
print("c =", c)           # c = 10
```

```
print("a == b :", b1)     # a == b : False  
print("a == c :", b2)     # a == c : True  
print("a != b :", b3)     # a != b : True
```


Operadores Relacionais

```
a = 25  
b = 25.0  
c = -37
```

```
b1 = (a > b)  
b2 = (a >= c)  
b3 = (a <= b)
```

```
print("a =", a)           # a = 25  
print("b =", b)           # b = 25.0  
print("c =", c)           # c = -37
```

```
print("a > b :", b1)       # a > b : False  
print("a >= c :", b2)     # a >= c : True  
print("a <= b :", b3)     # a <= b : True
```

Operadores Lógicos

Operador	Função
and	E lógico
or	OU lógico
not	NÃO lógico

Operadores Lógicos

- Tabela Verdade

E (and)	V	F
V	V	F
F	F	F

OU (or)	V	F
V	V	V
F	V	F

NÃO (not)	V	F
-	F	V

Operadores Lógicos

```
x = True  
y = False  
z = True
```

```
b1 = (x and y)  
b2 = (x and z)  
b3 = (x or z)
```

```
print("x =", x)           # x = True  
print("y =", y)           # y = False  
print("z =", z)           # z = True
```

```
print("x and y :", b1)    # x and y : False  
print("x and z :", b2)    # x and z : True  
print("x or z  :", b3)    # x or z  : True
```

Operadores Lógicos

- Agrupamentos de operadores lógicos

```
>>> nome = "Maria"
```

```
>>> idade = 30
```

```
>>> nome == "Maria" and idade == 30  
True
```

```
>>> nome == "Maria" and idade != 30  
False
```

Prioridade dos Operadores

- primeiro as operações aritméticas, depois as de comparação, por fim as lógicas

Nível	Categoria	Operadores
7 (alto)	exponenciação	**
6	multiplicação	* / // %
5	adição	+ -
4	relação	== != <= >= > <
3	negação	not
2	conjunção	and
1 (baixo)	disjunção	or

Prática

- Faça um algoritmo que receba dois números e, ao final, mostra a soma, a subtração, a multiplicação e a divisão dos números lidos.
- A Loja Mamão com Açúcar está vendendo seus produtos em 5 (cinco) prestações sem juros. Faça um algoritmo que receba o valor de uma compra e mostre o valor das prestações.

Prática

- Faça um algoritmo que receba o preço de custo de um produto e mostre o valor da venda. Sabe-se que o preço de custos receberá um acréscimo de acordo com um percentual informado pelo usuário.