

Módulo 01 - Introdução a Linguagem Python

CENTRO UNIVERSITÁRIO DE JOÃO PESSOA

PRÓ-REITORIA ACADÊMICA (PROAC)



Componente Curricular: Programação de Computadores

Professor: Álvaro George R. de A. Júnior

E-mail: alvaro.george@unipe.edu.br

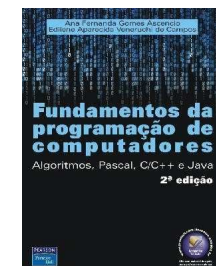
Referências Bibliográficas:

Básica:

- PERKOVIC, Ljubomir. **Introdução à computação usando Python: um foco no desenvolvimento de aplicações**, Rio de Janeiro: LTC, 2016.
- MANZANO, José Augusto N. G.; **Estudo dirigido de algoritmos**, São Paulo: Erica, 2011.
- DOBRUSHKIN, Vladimir A.; **Métodos para análise de algoritmos**, Rio de Janeiro: LTC, 2012
- ASCENCIO, Ana Fernanda Gomes; CAMPOS, Edilene Aparecida Veneruchi. **Fundamentos da programação de computadores: algoritmos, PASCAL, C/C++ (padrão ANSI) e JAVA**. 3ª ed. São Paulo: Pearson Education do Brasil, 2012.

Complementar:

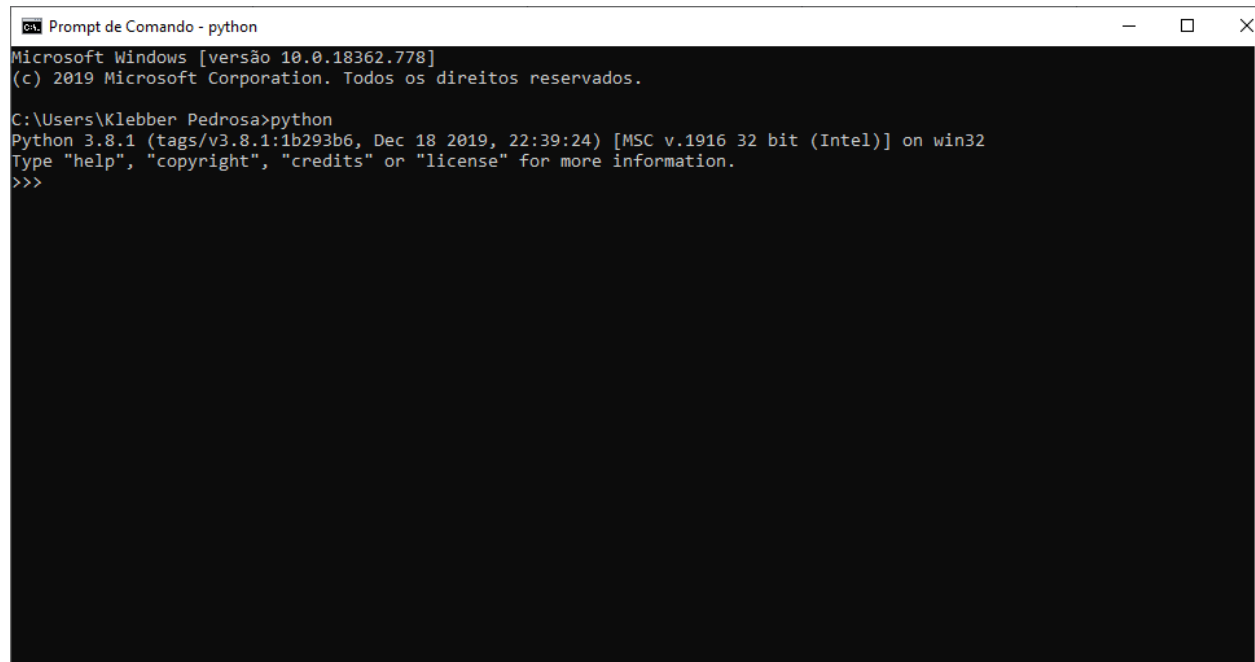
- PAPADIMITRIOU, Christos. VAZIRANI, Umesh. **Algoritmos**, Porto Alegre: AMGH, 2011.
- AGUILAR, Luis Joyanes. **Fundamentos de programação : algoritmos, estruturas de dados e objetos**, Porto Alegre: AMGH, 2008.
- MANZANO, José Augusto N. G.; **Algoritmos: técnicas de programação**, São Paulo: Erica, 2016.



- Página oficial: <https://www.python.org/>
- Criado por Guido van Rossum em 1991;
- Tipagem Dinâmica;
- Fracamente tipada? *
- Fortemente tipada? *
- Orientada a objetos;
- Endentação define blocos.

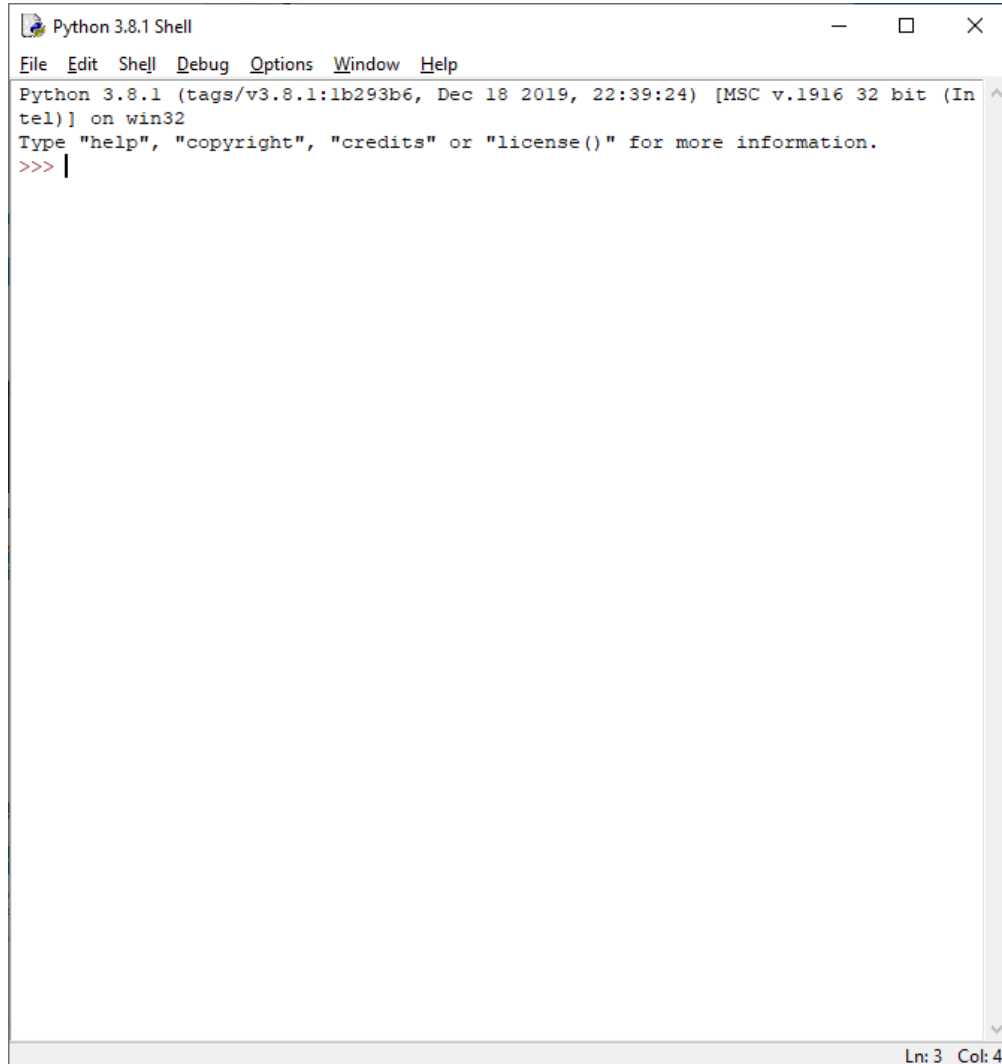


- Windows
 - ✓ <https://www.python.org/downloads/>
- Linux e MacOS
 - ✓ Já vem instalado;



```
Prompt de Comando - python
Microsoft Windows [versão 10.0.18362.778]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.

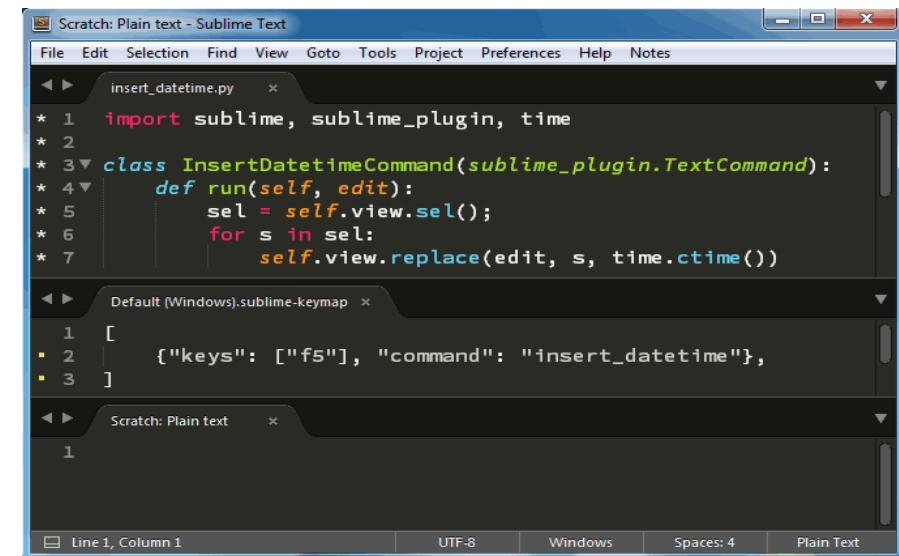
C:\Users\Klebbler Pedrosa>python
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



A screenshot of a Python 3.8.1 Shell window. The window has a title bar "Python 3.8.1 Shell" and a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following output:

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

The status bar at the bottom right indicates "Ln: 3 Col: 4".



A screenshot of a Sublime Text editor window. The window has a title bar "Scratch: Plain text - Sublime Text" and a menu bar with "File", "Edit", "Selection", "Find", "View", "Goto", "Tools", "Project", "Preferences", "Help", and "Notes". The main text area shows the following code:

```
insert_datetime.py
* 1 import sublime, sublime_plugin, time
* 2
* 3 class InsertDatetimeCommand(sublime_plugin.TextCommand):
* 4     def run(self, edit):
* 5         sel = self.view.sel()
* 6         for s in sel:
* 7             self.view.replace(edit, s, time.ctime())
```

The status bar at the bottom indicates "Line 1, Column 1", "UTF-8", "Windows", "Spaces: 4", and "Plain Text".

Windows/Linux
CTRL + B

- Dois tipos de programas processam linguagens de alto nível, traduzindo-as para linguagens de baixo nível: **interpretadores** e **compiladores**.
- **Interpretador:** lê um programa escrito em linguagem de alto nível e o executa, ou seja, faz o que o programa diz.



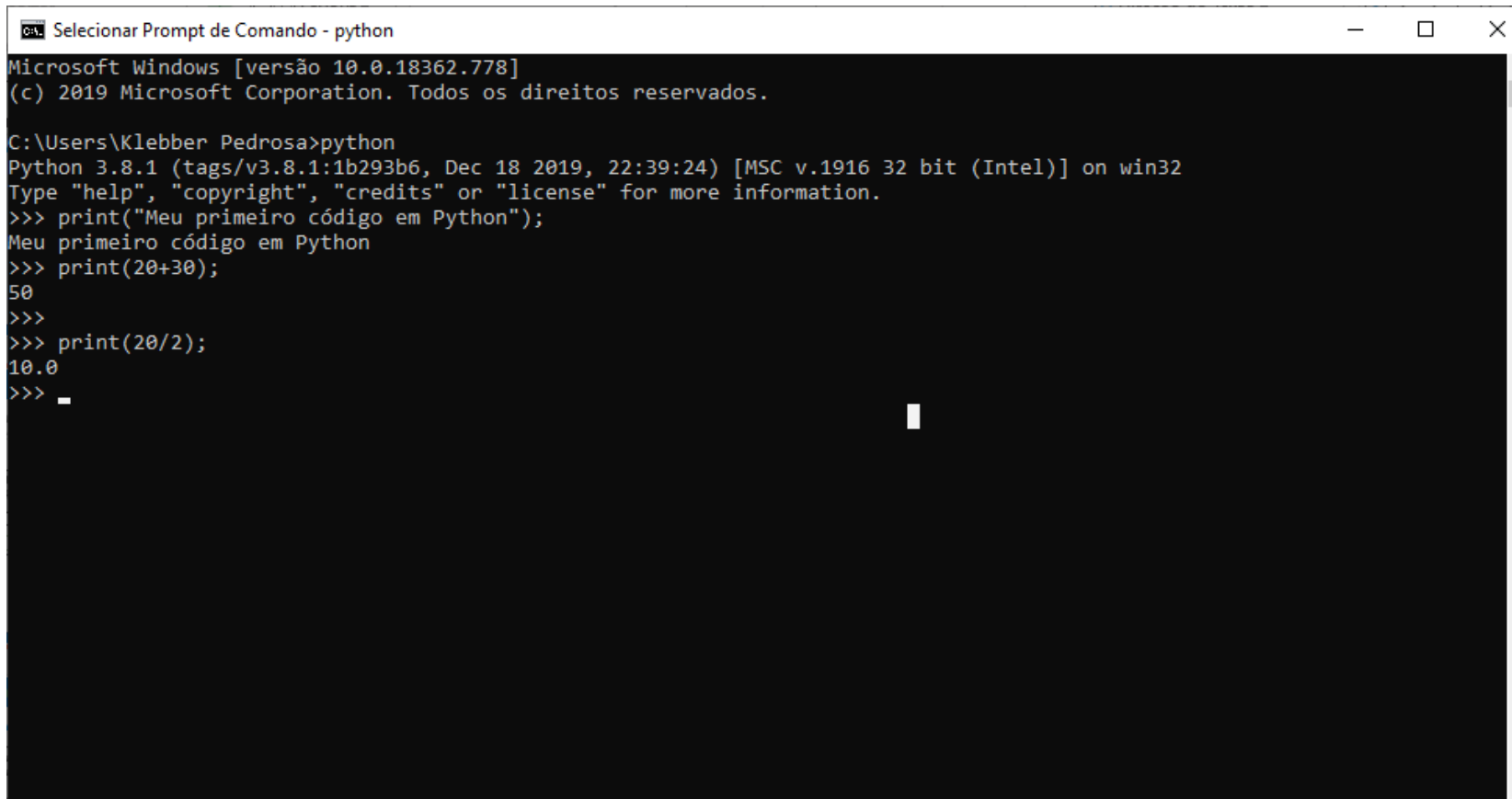
- **Compilador:** lê o programa e o traduz completamente antes que o programa comece a rodar.
- O programa traduzido é chamado de **código objeto** ou **executável**.



- O Python usa ambos os processos, mas ela é em geral considerada uma linguagem interpretada.

- Existem duas maneiras de usar o interpretador: no modo linha de comando (“**shell mode**”) e no modo de script (“**program mode**”).
- **Linha de comando**: você digita comandos em Python eo interpretador mostra o resultado.

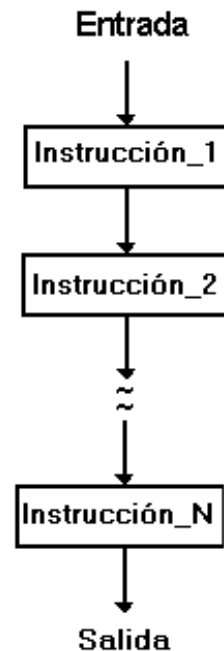
- Por convenção, arquivos que contêm programas em Python tem nomes que terminam com a extensão **.py**, ex: programa1.py



```
Selecionar Prompt de Comando - python
Microsoft Windows [versão 10.0.18362.778]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Klebber Pedrosa>python
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Meu primeiro código em Python");
Meu primeiro código em Python
>>> print(20+30);
50
>>>
>>> print(20/2);
10.0
>>> _
```

- Como visto no início da disciplina Programação de Computadores, um **programa** é uma sequência de comandos que serão executados.



- O programa deve ter **um comando por linha**. Os comandos serão executados nesta ordem, **de cima para baixo**, um por vez.

Estrutura Básica de um Programa em Python

```
>>>  
>>> print("Centro Universitário de João Pessoa");  
Centro Universitário de João Pessoa  
>>> print("Linguagem Python")  
Linguagem Python  
>>>  
>>>  
>>> print("Primeiros passos na linguagem");  
Primeiros passos na linguagem  
>>>
```

```
>>>  
>>> print("Cuidado")      print("Isso vai dar um erro!");  
File "<stdin>", line 1  
    print("Cuidado")      print("Isso vai dar um erro!");  
                          ^  
SyntaxError: invalid syntax  
>>>
```

*Este programa gera um **erro** pois temos dois comandos em uma mesma linha.*

Você pode usar um ponto e vírgula ao final de cada comando para usar **vários comandos em uma mesma linha**.

```
>>>  
>>> print("Agora sim");      print("Isso pode acontecer!");  
Agora sim  
Isso pode acontecer!  
>>>
```

- Qualquer dado em Python é um objeto, que é de um certo tipo específico.
- O tipo de um objeto especifica quais operações podem ser realizadas sobre o objeto.

Exemplos:

- O número 20 é representado com um objeto do tipo **int** em Python.

```
>>>  
>>> print(type(20));  
<class 'int'>  
>>>
```

- O texto “Unipê” é representado com um objeto do tipo **str** (*string ou cadeia de textos*) em Python.

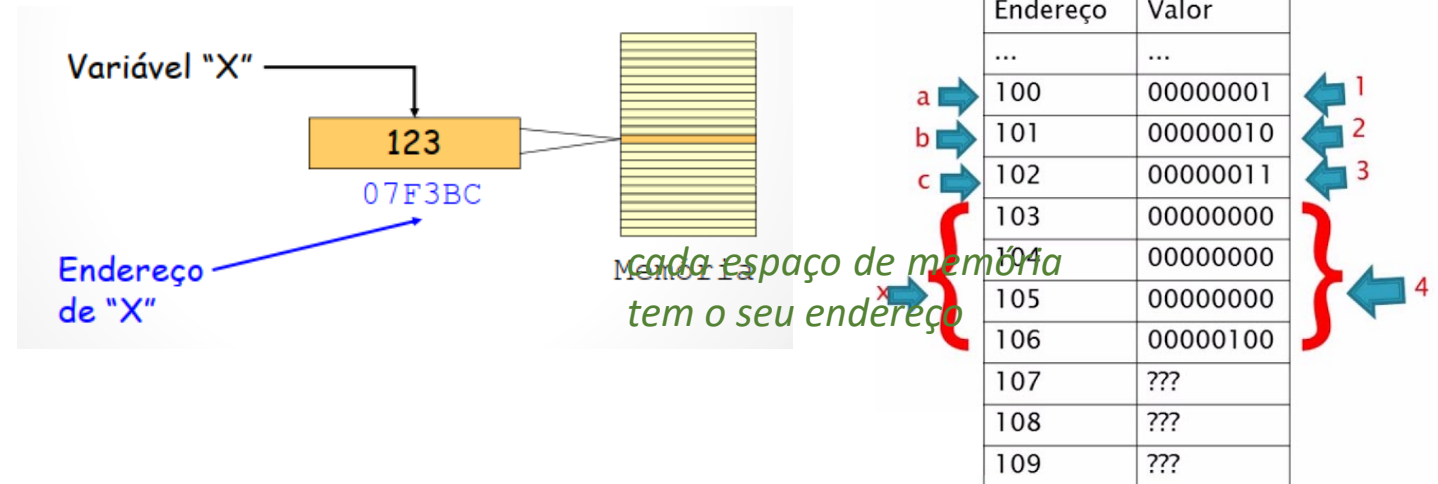
```
>>>  
>>> print(type("Unipê"));  
<class 'str'>  
>>>
```

- Observe que 20 é um número inteiro, porém entre aspas é um texto.

```
>>>  
>>> print(type("20"));  
<class 'str'>  
>>>
```

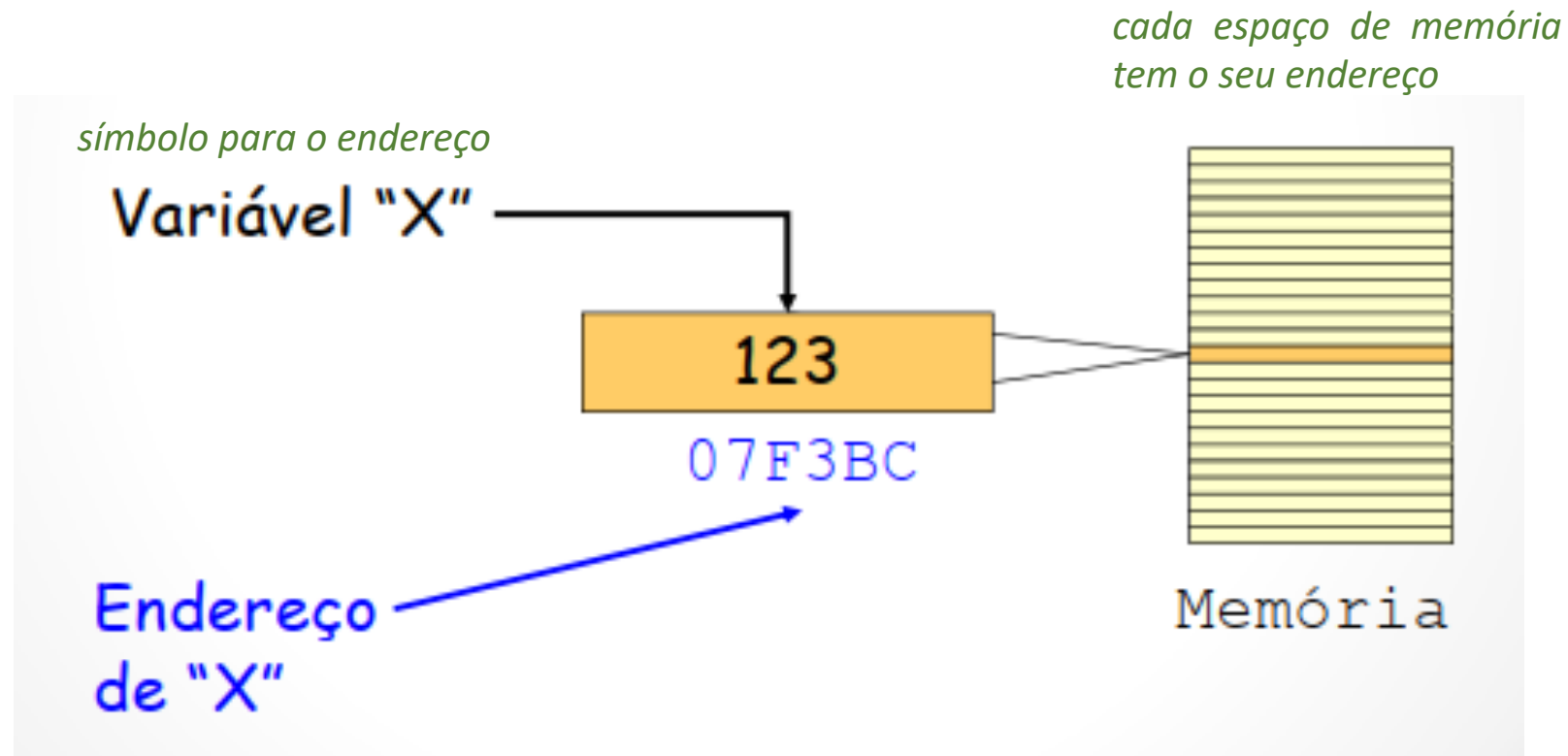
- Variáveis permitem o armazenamento de informação na memória do computador.
- São associados a nomes, chamados identificadores;
- Identificadores são usados para referenciar e diferenciar as variáveis em algoritmos;
- É uma forma de se associar um nome dado pelo programador com um objeto.
símbolo para o endereço
- No exemplo abaixo associamos os nomes **nota1**, **nota2** e **media** com os valores 9.0, 10.0, respectivamente.

```
>>>  
>>> nota1 = 9.0;  
>>> nota2 = 10.0;  
>>>  
>>> print( (nota1+nota2)/2 );  
9.5  
>>>
```



- Visão geral das variáveis:

```
>>>  
>>> x = 123;  
>>>
```



- **Deve** começar com uma letra (maiúscula ou minúscula) ou underscore “_”;
- **Nunca** pode começar com um número;
- Pode conter letras maiúsculas, minúsculas, números e subscrito;
- Não deve conter caracteres especiais (: { (+ -* / \ n ; . , ?) e nem espaço;
- É **case sensitive**, ou seja, letras maiúsculas e minúsculas são diferentes: c = 4 C = 3.

```
nomeVariavel = valor
```

= - operador de atribuição.

valor - define o tipo da variável.

- O operador = do Python é o operador de atribuição.
- Ele associa a variável do lado esquerdo do operador com o objeto do lado direito do operador.

```
>>>
>>> 123minhaVariavel = 123;
      File "<stdin>", line 1
        123minhaVariavel = 123;
          ^
SyntaxError: invalid syntax
>>>
>>>
>>> valorem$ = 100;
      File "<stdin>", line 1
        valorem$ = 100;
          ^
SyntaxError: invalid syntax
>>>
>>> import = "estrangeiro";
      File "<stdin>", line 1
        import = "estrangeiro";
          ^
SyntaxError: invalid syntax
>>>
```

- A variável **123minhaVariavel** não pode, pois não começa **com uma letra**;
- A variável **valorEm\$** não pode, pois **\$** é um **caracter especial**;
- A variável **import** não pode, pois **import** é uma **palavra reservada**.

- As **palavras reservadas** definem a *sintaxe da linguagem* e sua estrutura e não podem ser usadas como nomes de variáveis.
 - ✓ Python tem pouco mais de trinta palavras reservadas (e uma vez ou outra melhorias em Python introduzem ou eliminam uma ou duas).

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

- Se uma variável for usada sem estar associada com nenhum objeto, um erro ocorre.
- No exemplo abaixo não podemos usar a variável **w**, pois esta não foi definida (associada com algum objeto).

```
>>>  
>>> x = 5  
>>> y = 10  
>>> z = x + y  
>>> z  
15  
>>> x = z + w  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
NameError: name 'w' is not defined  
>>>
```

- Melhora a organização e explicação do código;
- Não são processados;

Comentários de linha:

```
# Python é case sensitive  
var = 'João'  
Var = 'João'
```

Comentários de bloco:

```
'''  
Comentários multilinhas  
'''
```

- Python possui os seguintes tipos básicos que veremos nesta aula:
 - ✓ **int**: Corresponde aos números inteiros.
 - Ex: 10, -24.
 - ✓ **float**: Corresponde aos números racionais.
 - Ex: 2.4142, 3.141592.
 - ✓ **str** ou **string**: Corresponde a textos.
 - Ex: "Ola turma".
- Outros tipos, tais como **booleanos**, **bytes**, **listas**, **tuplas** e **dicionários** serão vistos ao longo do curso.

Observação:

- ✓ Por padrão, o Python não reconhece acentos.
- ✓ Basta incluir o comentário:

```
1 # -*- coding: utf-8 -*-
```

- Objetos do tipo `int` armazenam valores inteiros.
- Literais do tipo `int` são escritos comumente como escrevemos inteiros.
- O tipo `int` possui precisão arbitrária (limitado a memória do seu computador).

Exemplos: 3, 1034 e -512.

- Objetos do tipo `float` armazenam valores “**reais**”.
- Literais do tipo `float` são escritos com um ponto para separar a parte inteira da parte decimal.
Exemplos: 3.1415 e 9.8.
- Possuem problemas de precisão pois há uma quantidade limitada de memória para armazenar um número real no computador.

- Objetos do tipo `string` armazenam textos.
- Um literal do tipo `string` deve estar entre aspas simples ou aspas duplas.

Exemplos:

- `'Ola Brasil!'` ou `"Ola Brasil"`.
- Veremos posteriormente nesta disciplina diversas operações que podem ser realizadas sobre objetos do tipo `string`.

- Uma variável em Python possui o tipo correspondente ao objeto que ela está associada **naquele instante**.
- Python não possui tipagem forte como outras linguagens.
 - Isto significa que você pode atribuir objetos de diferentes tipos para uma mesma variável.
 - Como uma variável não possui tipo pré-definido, dizemos que Python tem **tipagem fraca**.
 - Em outras linguagens cria-se variáveis de tipos específicos e elas só podem armazenar valores daquele tipo para o qual foram criadas.
 - Estas últimas linguagens possuem **tipagem forte**.

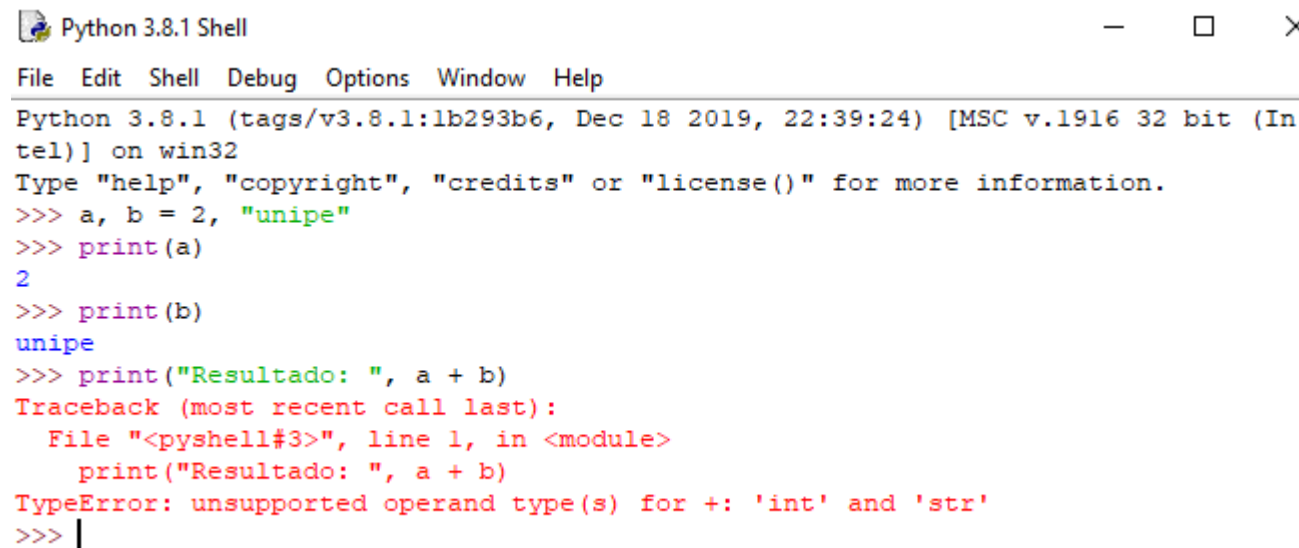
Fracamente tipada

- Considera-se a consequência da tipagem dinâmica.

```
>>>  
>>> x = 10;  
>>> print(x);  
10  
>>>  
>>> x = 50.25  
>>> print(x);  
50.25  
>>>  
>>> x = "Linguagem Python";  
>>> print(x);  
Linguagem Python  
>>>
```

Fortemente tipada

- Considera-se a o fato do interpretador do Python avaliar as expressões e não faz coerções automáticas entre tipos não compatíveis (conversões de valores) .



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a, b = 2, "unipe"
>>> print(a)
2
>>> print(b)
unipe
>>> print("Resultado: ", a + b)
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    print("Resultado: ", a + b)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> |
```

- No código acima definimos a como 2 e b como “unipe”. Com isso, a é um inteiro e b é uma string.
- Ao executar este código, temos um erro em virtude de fazer uma operação com tipos incompatíveis.

- Qual o valor armazenado na variável `c` no fim do programa?
- Qual o tipo da variável `c` ao final do código?

```
>>>  
>>> a = 10  
>>> b = 5  
>>> c = 10  
>>> d = a + c  
>>> c = d/2  
>>> c = c/2  
>>> print(c)  
?  
>>>  
>>> print(type(c));  
<class ' ? '>  
>>>
```

- Qual o valor armazenado na variável `c` no fim do programa?
- Qual o tipo da variável `c` ao final do código?

```
>>>  
>>> a = 10  
>>> b = 5  
>>> c = 10  
>>> d = a + c  
>>> c = d/2  
>>> c = c/2  
>>> print(c)  
5.0  
>>>  
>>> print(type(c));  
<class 'float'>  
>>>
```

- Para entrarmos com valores via teclado, devemos fazer:

```
nome = input("Digite o seu nome: ")

print(type(nome))

idade = input("Digite a sua idade: ")

print(type(idade))

novaIdade = int(idade)

print(novaIdade)

outraIdade = int(input("Digite a sua idade: "))

print(type(outraIdade))
```

Shell

```
Digite o seu nome: José
<class 'str'>
Digite a sua idade: 20
<class 'str'>
20
Digite a sua idade: 20
<class 'int'>
>>>
```

- **Questão 01.** Faça um programa em Python que receba um valor que é o **valor pago**, um segundo valor que é o **preço do produto** e imprima na tela do computador o troco a ser dado.
- **Questão 02.** A imobiliária **Imóbilis** vende apenas terrenos retangulares. Faça um programa em Python para ler as *dimensões de um terreno* e depois exibir a *área do terreno*.
- **Questão 03.** Faça um programa em Python que lê a idade de uma pessoa expressa em anos, meses e dias e *mostre a sua idade expressa apenas em dias* (considere 365 dias para cada ano e 30 dias para cada mês).
- **Questão 04.** Escrever um algoritmo em Python que lê o *nome de um aluno e as notas das três provas* que ele obteve no semestre. No final informar o nome do aluno e a sua *média aritmética*.