



FACVLDDE MAURÍCIO DE  
**NASSAU**



# Aprenda a programar em Python

PROF. DR. DIOGO RODRIGUES

010117368@PROF.UNINASSAU.EDU.BR



**Diogo Francisco Borba Rodrigues**

Engenheiro de Software PL



**Postdoctoral Researcher at Technology and Geosciences Center**

Universidade Federal de Pernambuco

jun de 2017 - fev de 2019 · 1 ano 9 meses

Recife, Pernambuco, Brasil



**Universidade Federal Rural de Pernambuco**

Doutorado em Engenharia Agrícola, Sensoriamento Remoto; Desenvolvimento de Aplicativos

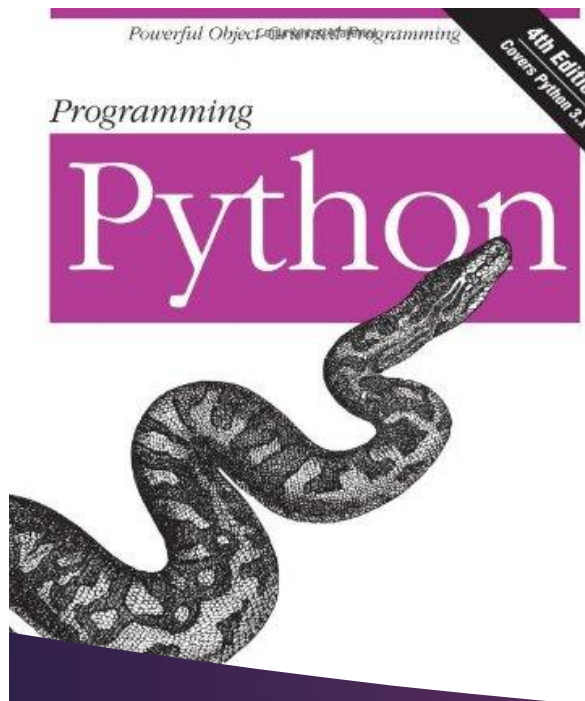
2013 - 2017



DIOGOBRODRIGUES

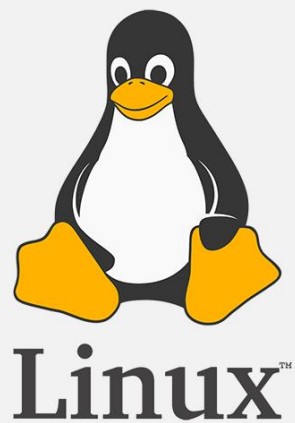


# Professor



## A origem

- ▶ 1982 – Amsterdam Holanda –Guido Van Hossum
- ▶ Monty Python



Sistemas com Python pré-instalado

# Características gerais



LINGUAGEM DE  
PROPOSITO  
GERAL



SIMPLES FÁCIL E  
INTUITIVO



MULTIPLATAFORMA



BATERIAS  
INCLUÍDAS



LIVRE



ORGANIZADA



ORIENTADA A  
OBJETOS

# Principais áreas

Inteligência artificial

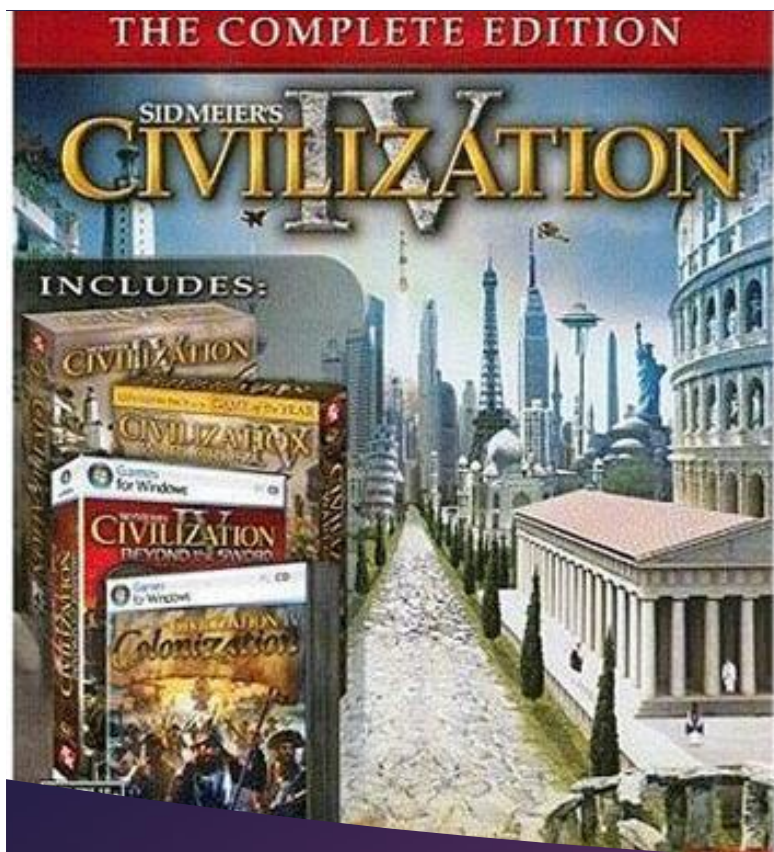
Biotecnologia

Computação 3d



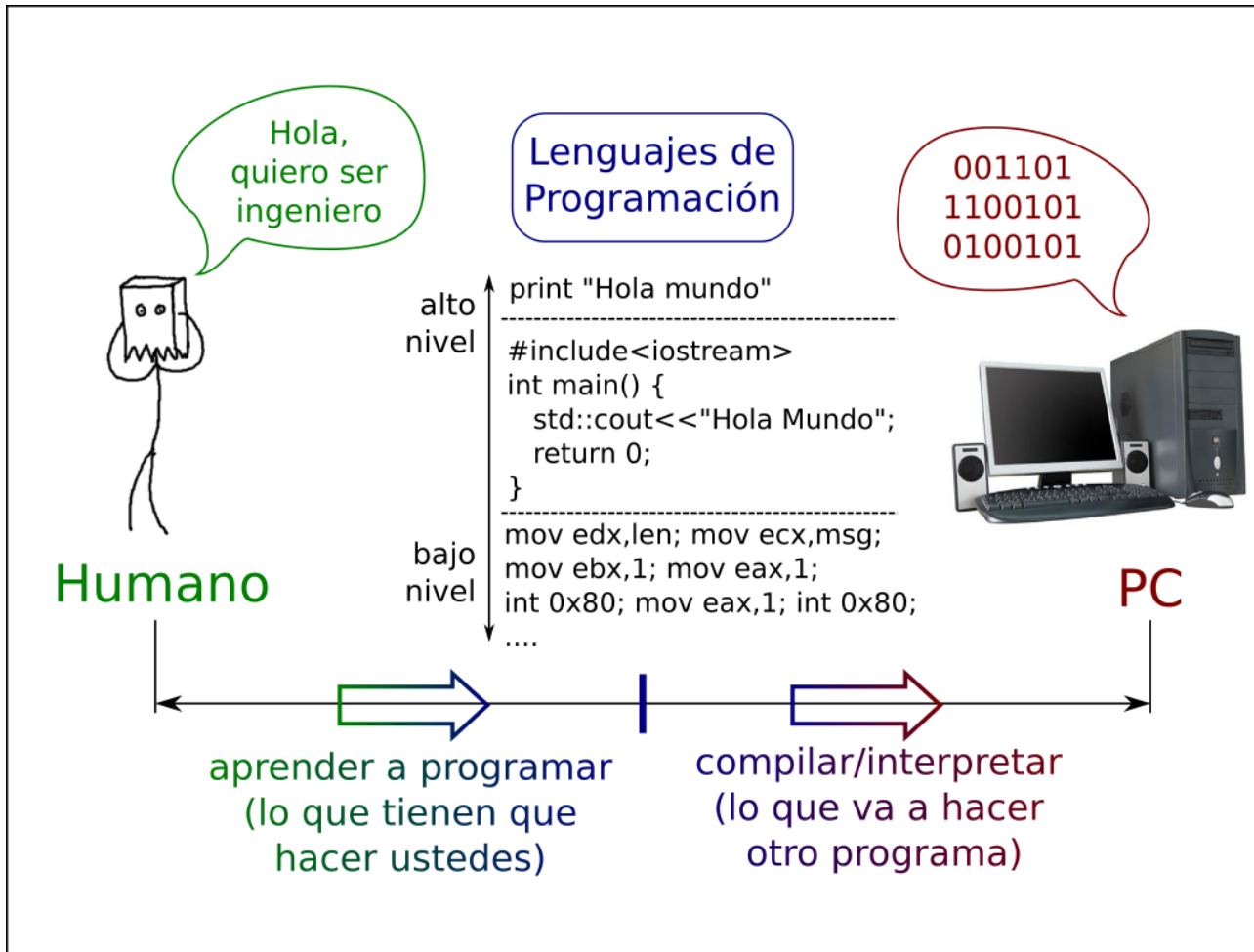
# Aplicações



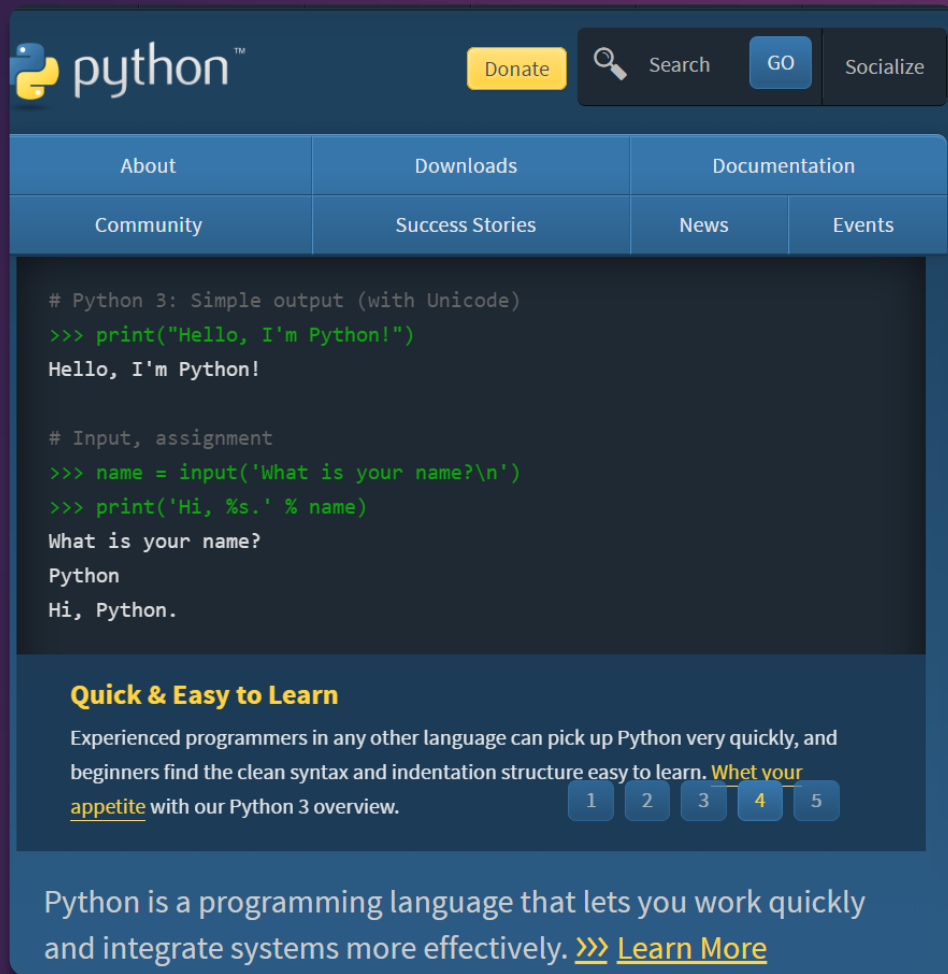


Aplicações





Se  
comunique  
com uma  
máquina

A screenshot of the Python.org homepage. The header features the Python logo, a 'Donate' button, a search bar with a 'GO' button, and a 'Socialize' button. Below the header is a navigation menu with links to 'About', 'Downloads', 'Documentation', 'Community', 'Success Stories', 'News', and 'Events'. The main content area displays a code snippet for a simple Python program that prints 'Hello, I'm Python!' and a program that takes user input and prints a greeting. Below the code is a section titled 'Quick & Easy to Learn' with a paragraph about Python's syntax and a link to a Python 3 overview. At the bottom, there is a paragraph about Python's capabilities and a link to 'Learn More'.

python™

Donate Search GO Socialize

About Downloads Documentation

Community Success Stories News Events

```
# Python 3: Simple output (with Unicode)
>>> print("Hello, I'm Python!")
Hello, I'm Python!

# Input, assignment
>>> name = input('What is your name?\n')
>>> print('Hi, %s.' % name)
What is your name?
Python
Hi, Python.
```

**Quick & Easy to Learn**

Experienced programmers in any other language can pick up Python very quickly, and beginners find the clean syntax and indentation structure easy to learn. [Whet your appetite](#) with our Python 3 overview.

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. [>>> Learn More](#)

# Instalando o Python

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

[Donate](#)[GO](#)[Socialize](#)[About](#)[Downloads](#)[Documentation](#)[Community](#)[Success Stories](#)[News](#)[Events](#)

```
# Python 3: Fib
>>> def fib(n):
>>>     a, b =
>>>     while a
>>>         pri
>>>         a,
>>>     print()
>>>     fib(1000)
0 1 1 2 3 5 8 1
```

[All releases](#)[Source code](#)[Windows](#)[Mac OS X](#)[Other Platforms](#)[License](#)[Alternative Implementations](#)

### Download for Windows

[Python 3.8.5](#)

**Note that Python 3.5+ *cannot* be used on Windows XP or earlier.**

Not the OS you are looking for? Python can be used on many operating systems and environments.

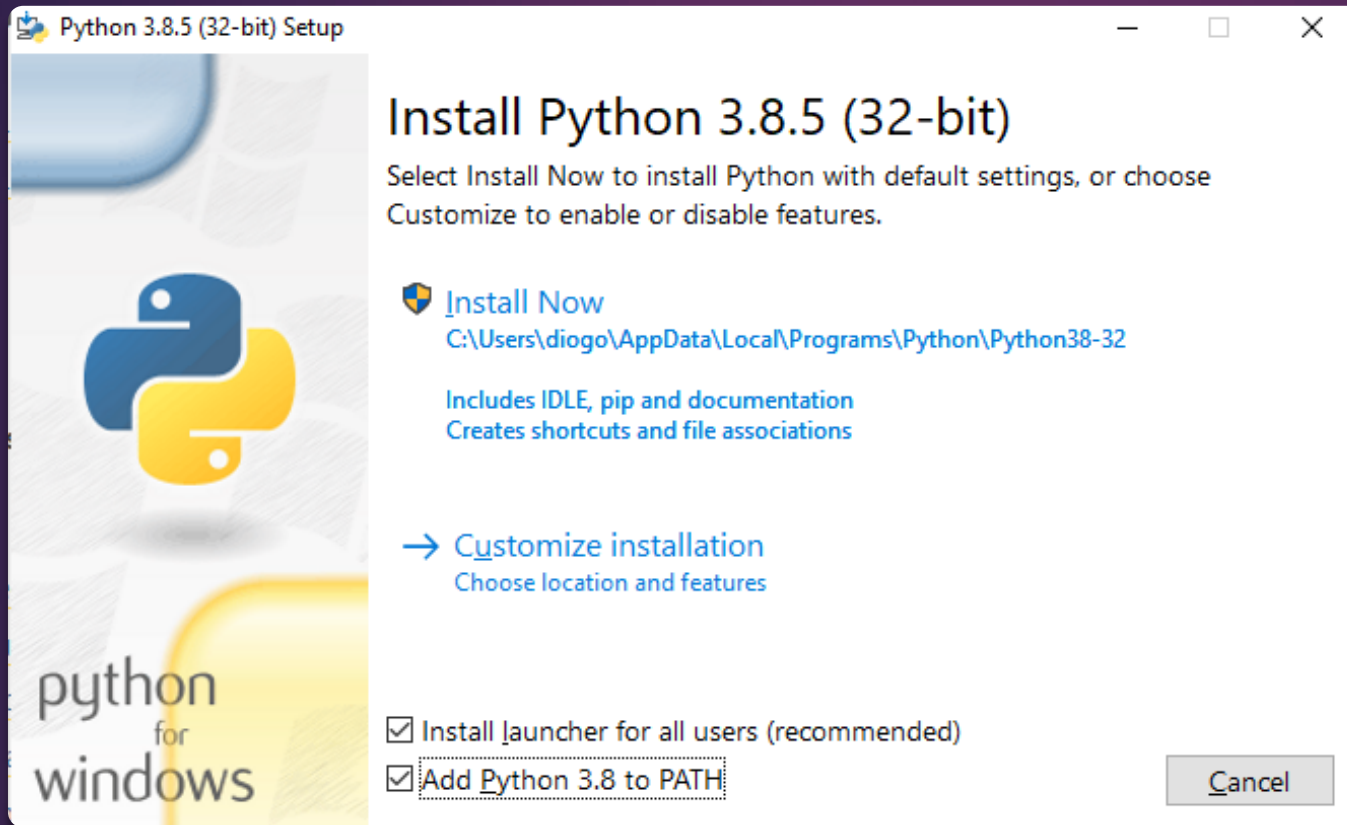
[View the full list of downloads.](#)

ing functions.

ments, keyword

s. [More about](#)

Python is a programming language that lets you work quickly



Add  
Python  
3.8 path



IDLE (Python 3.8 32-bit)

Aplicativo



Python 3.8.5 Shell



File Edit Shell Debug Options Window Help

Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>



# Comandos básicos - Print

- ▶ `print ( 'Olá Mundo!')`
- ▶ `print('Olá'+ ' Mundo!')`
- ▶ `print ( 3 + 7)`
- ▶ `print('3' + '7')`

# Variáveis

Nome

=

Idade

=

Peso

=

# Funções de Saída x Entrada



print



input

# Exercícios

1

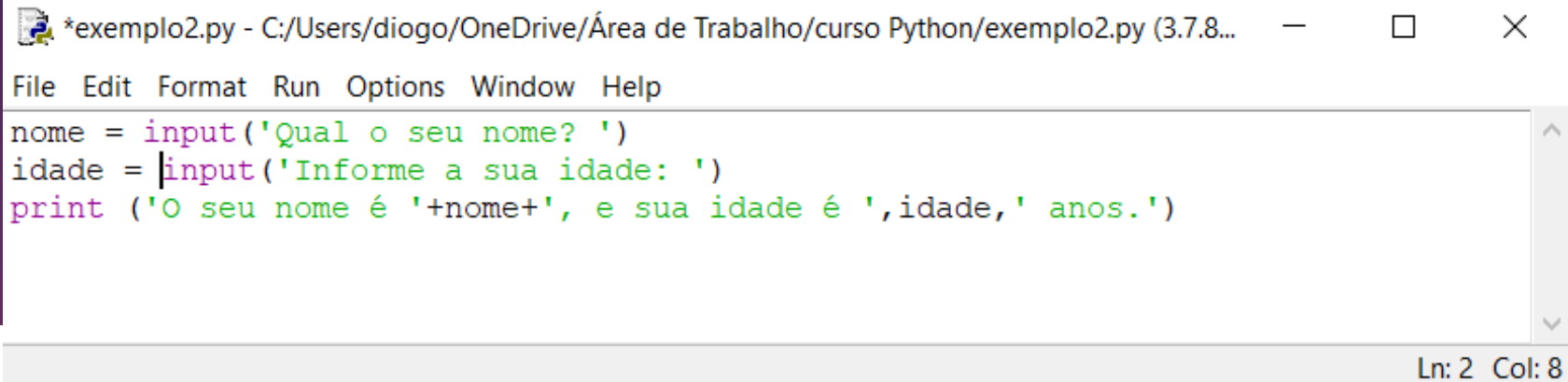
```
Qual o seu nome?Diogo
Seja bem vindo Diogo ao curso de Python
```

2

```
Dia = 3
Mes = Abr
Ano = 1900
Você nasceu no dia 3 de Abr de 1900
```

3

```
Digite um número3
Digite mais um número2
a soma vale 5
```

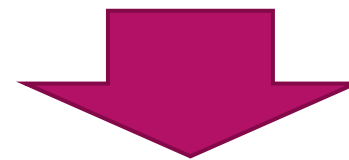


```
*exemplo2.py - C:/Users/diogo/OneDrive/Área de Trabalho/curso Python/exemplo2.py (3.7.8...
File Edit Format Run Options Window Help
nome = input('Qual o seu nome? ')
idade = input('Informe a sua idade: ')
print ('O seu nome é '+nome+', e sua idade é ',idade,' anos.')
Ln: 2 Col: 8
```

# Tipos primitivos

- ▶ int → 7; -4 ; 0; 1900
- ▶ float → 4.5; 0.085; -12.325; 7.0
- ▶ bool → True; False
- ▶ str → 'Diogo' '8.5'

Outras formas de fazer o print



```
print('A soma vale', s)  
print('|A soma vale{}'.format(s))
```



## Voltando ao ex 3

Digite um valor:2

Digite outro valor10

A soma entre 2 e 10 é 12

```
n1 = int(input('Digite um valor:'))  
n2 = int(input('Digite outro valor'))  
soma = n1+n2  
print('A soma entre {} e {} é {}'.format(n1,n2,soma))
```

# Operações Aritméticas

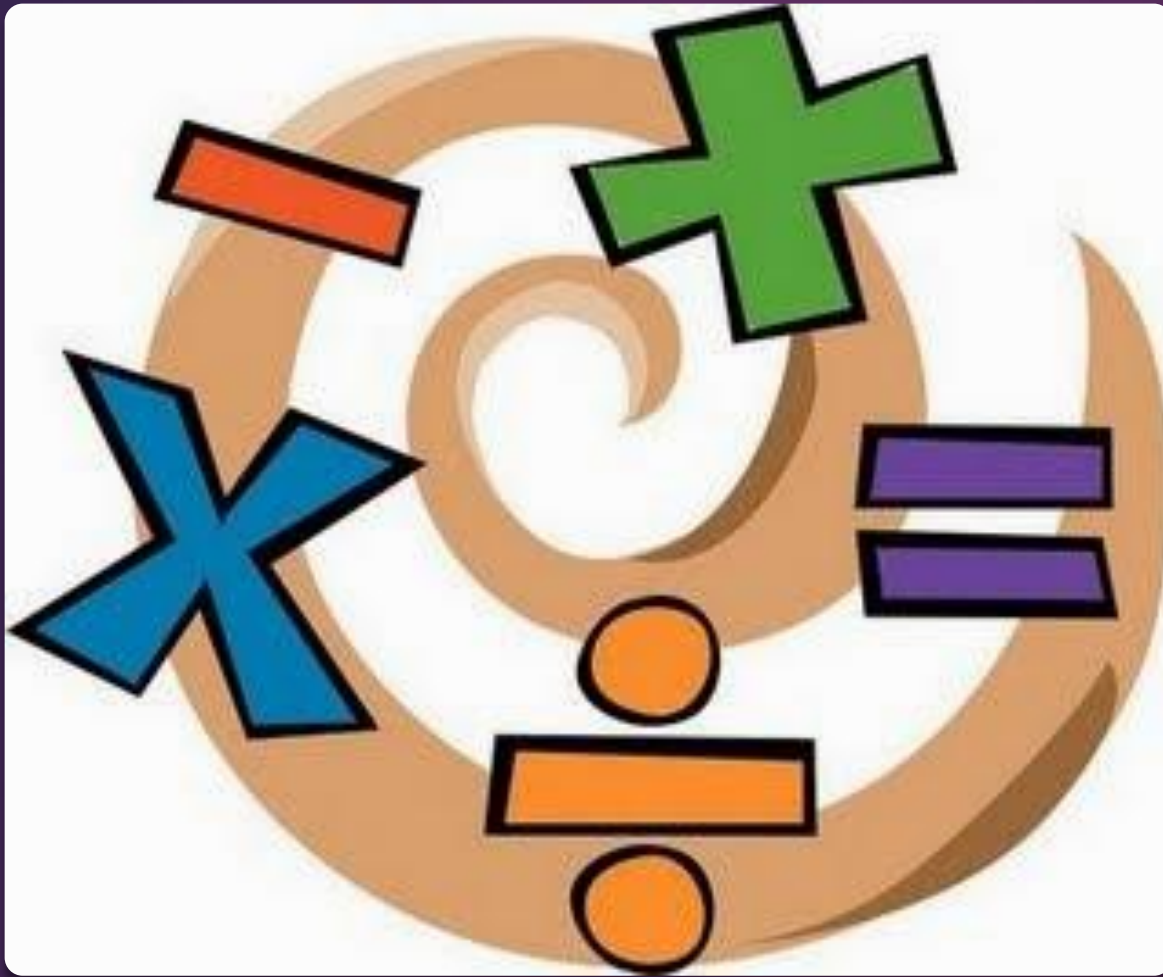
- ▶ + (adição)
- ▶ - (subtração)
- ▶ \* (multiplicação)
- ▶ / (divisão)
- ▶ \*\* (potência)
- ▶ // (divisão inteira)
- ▶ % (resto da divisão)


$$2 + 2 = 4$$

# Ordem de precedência

- ▶  $1 \rightarrow ()$
- ▶  $2 \rightarrow **$
- ▶  $3 \rightarrow * ; / ; // ; \%$
- ▶  $4 \rightarrow + ; -$





# Módulos

MATH → CEIL; FLOOR;  
TRUNC; POW; SQRT;  
FACTORIAL...

# Importar Bibliotecas em Python

- ▶ Import
- ▶ From import
- ▶ Ex:
- ▶ Import math
- ▶ From math import sqrt



```
tempo = int(input('Quantos anos tem seu carro?'))
if tempo <= 3:
    print('carro novo')
else:
    print('carro Velho')
```

# Estruturas condicionais – if e else

# Calculadora Simples

```
primeiro = input("Digite o primeiro número: ")
segundo = input("Digite o segundo número: ")
operacao = input("Digite a operação: ")

resultado = None
if operacao == "+":
    resultado = float(primeiro) + float(segundo)
```

# Estrutura de repetição

Estrutura de repetição for—contador—intervalo (início, fim, incremento)

```
for c in range(1,6):
```

```
for c in range(1,6,2):
```

```
s = 0
for c in range(0, 4):
    n = int(input('Digite um valor: '))
    s += n
print('O somatório de todos os valores foi {}'.format(s))
```

# Estrutura de repetição

Estrutura de repetição while – estrutura de repetição sem intervalo definido

Sintaxe:

`while num !=0:`

# Lista de Dados

---

Estrutura pré-definidas em Python:

- Listas
- Dicionários
- Tuplas
- Conjuntos

Seria muuuuito mais fácil  
trabalhar com esses dados...  
se eu os organizasse como  
uma lista.





# Objeto

Em **pythonn** tudo é **objeto**, e pode ser atribuído a uma variável.

Como em outras linguagens de programação, os objetos podem ter estado (*atributos* ou *valores*) e comportamento(*métodos*).

## Baseado em Objetos

Qualquer coisa pode ser atribuída qualquer variável

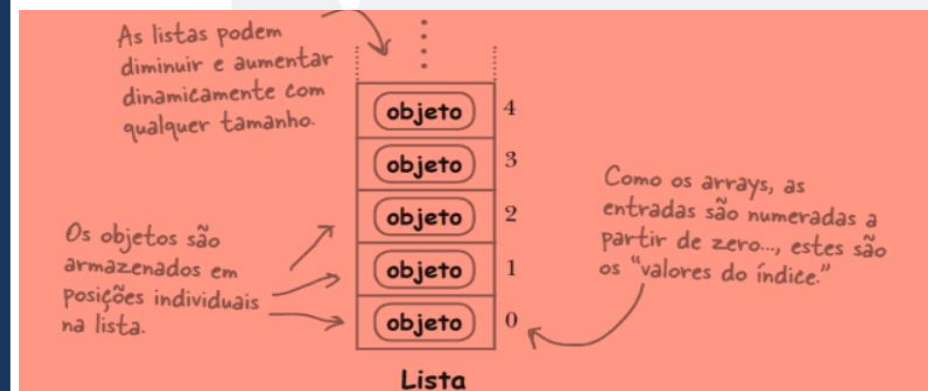


# Lista:

Uma **coleção variável e ordenada de objetos**, similar ao **array** nas outras linguagens de programação.

Uma **coleção indexada de objetos afins**, com dada posição na lista numerada a partir do zero.

As listas em python, diferente dos arrays em outras linguagens, pode aumentar ou diminuir de tamanho **dinamicamente**.



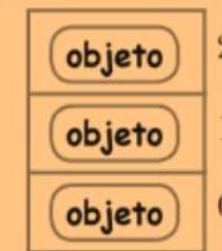
# Tupla:

Quando uma lista é **invariável** (ou seja, não pode mudar) é chamada de **tupla**.

Uma coleção **invariável** e **ordenada** de objetos.

Pode-se considerar como uma lista constante.

As tuplas são como listas, exceto que, uma vez criadas, NÃO PODEM mudar. As tuplas são constantes.



As tuplas usam valores de índice também (como as listas).

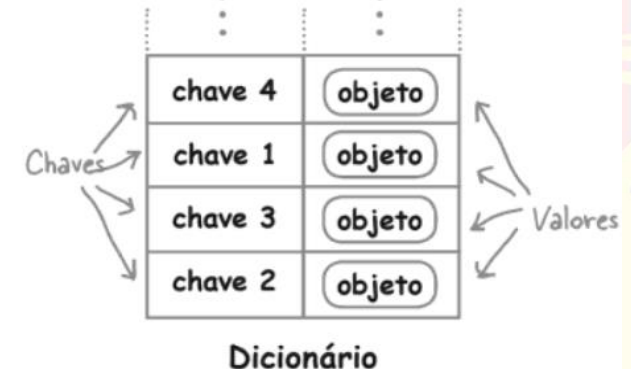
# Dicionário :

Manter os dados em uma ordem específica não é importante, mas a estrutura sim.

O dicionário do Python permite armazenar uma coleção de pares **chave/valor**.

Cada chave única tem um valor associado no dicionário.

Os dicionários associam chaves a valores e (como as listas) podem diminuir e aumentar dinamicamente com qualquer tamanho.



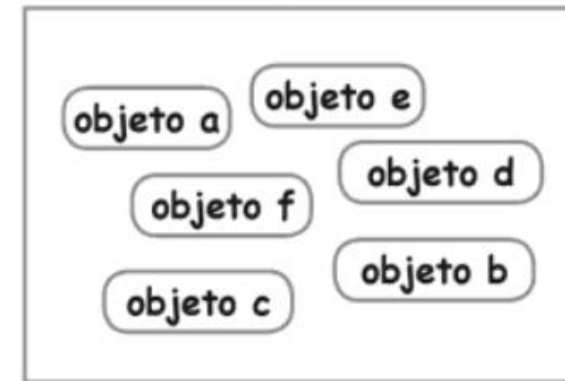
# Conjunto:

Uma estrutura de dados que **evita duplicatas**.

*Permite a realização de uniões, interseções e diferenças.*

*Como os dicionários, os conjuntos são desordenados, portanto não é possível fazer suposições sobre a ordem dos objetos nele.*

Pense em um conjunto como uma coleção de itens únicos desordenados — sem duplicatas.

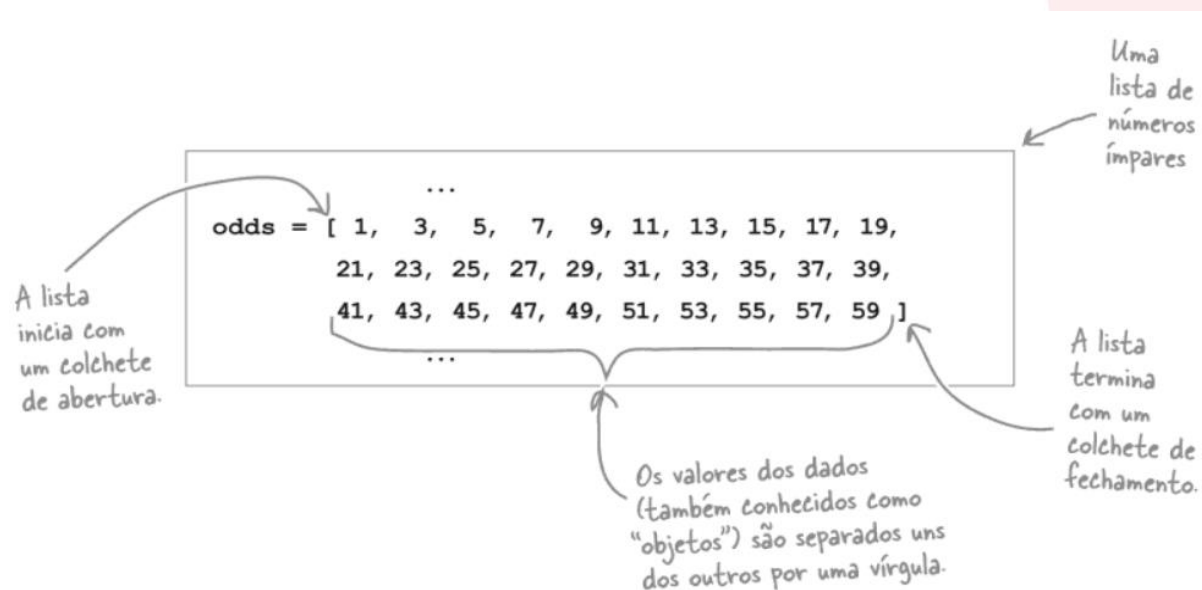


Conjunto

# Voltando para Lista

Array de objetos diferentes, sem restrições, além disso são dinâmicas, podendo aumentar ou diminuir de tamanho a qualquer momento.

As listas podem ser criadas literalmente (criada e preenchida) ou aumentadas no código.



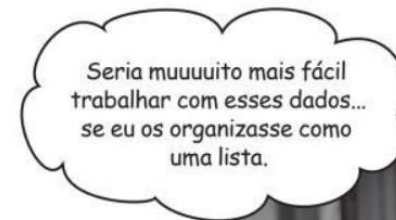


# Retirando um objeto de uma lista

---

O método `remove` é ótimo quando você conhece o valor do objeto que deseja remover.

Mas, geralmente, você deseja remover um objeto de uma posição



# Exercícios – Lista 2

- ▶ Escreva um programa que faça o computador “pensar em um número inteiro entre 0 e 5, e peça para o usuário tentar descobrir qual foi o número que o computador escolheu. Se o usuário acertar escreva na tela : “Você venceu”, caso ele erre escreva “Você perdeu”;
- ▶ Crie um programa que leia um número inteiro qualquer, ele deve indicar se o número é par ou ímpar
- ▶ Faça um programa que leia 3 números e mostre qual o maior e qual o menor número digitado.
- ▶ Faça um programa que leia duas notas do aluno e verifique se o mesmo está aprovado ou não .

```
import random  
numero = (random.randint(0, 9))  
print(numero)
```



# Respostas Lista 2

#ex1

*#Escreva um programa que faça o computador “pensar em um número inteiro entre 0 e 5, e peça para o usuário tentar descobrir qual foi o número que o computador escolheu. Se o usuário acertar escreva na tela : “Você venceu”, caso ele erre escreva “Você perdeu”;*

```
'''import random
num = int(input('Tente adivinhar o número que o computador está pensando de 0 e 5: '))
pcNumber = random.randint(0, 5)
if(num == pcNumber):
    print('parabéns você acertou!')
else :
    print('O número pensando foi {}, você perdeu!'.format(pcNumber))'''
```

#ex2

*# Crie um programa que lei um número inteiro qualquer, ele deve indicar se o número é par ou ímpar*

```
'''num = int(input('Digite um número: '))
resp = num % 2
if resp == 0 :
    print('O número é par! ')
else :
    print('O número é ímpar! ')'''
```

#ex 3

*# Faça um programa que leia 3 números e mostre qual o maior e qual o menor número digitado.*

```
'''print('Programa de ordenação, digite três números e vamos verificar qual o maior e qual o menor número digitado')
num1 = int(input('o primeiro número: '))
num2 = int(input('segundo número: '))
num3 = int(input('terceiro número: '))
```

```
if num1 < num2 :
    if num1 < num3 :
        print('o número {} é o menor número digitado'.format(num1))
```

```
if num2 < num3 :
    if num2 < num1 :
        print('o número {} é o menor número digitado'.format(num2))
```

```
if num3 < num2 :
    if num3 < num1 :
        print('o número {} é o menor número digitado'.format(num3))
```

```
if num1 > num2 :
    if num1 > num3 :
        print('o número {} é o maior número digitado'.format(num1))
```

```
if num2 > num3 :
    if num2 > num1 :
        print('o número {} é o maior número digitado'.format(num2))
```

```
if num3 > num2 :
    if num3 > num1 :
        print('o número {} é o maior número digitado'.format(num2))'''
```

# Respostas Lista 2

*#ex4*

*# Faça um programa que leia duas notas do aluno e verifique se o mesmo está a provado ou não .*

```
"""nota1 = float(input('Digite a primeira nota: '))
nota2 = float(input('Digite a segunda nota: '))
media = (nota1 + nota2) / 2
if media >= 7:
    print('O aluno foi aprovado, com nota {}'.format(media))
else:
    print('O aluno foi reprovado, e sua média foi {}'.format(media))
"""
```

# Aula de hoje

---

- Até então só consideramos teclado e monitor como mecanismos de entrada e saída



- Veremos como ler e escrever em arquivos



# Motivação

---

- Em algumas situações é desejado ler dados de arquivos e escrever dados em arquivos
  - Não é necessário digitar via teclado os dados a cada execução do programa
- Os resultados do programa podem ser impressos ou enviados para outras pessoas com mais facilidade
  - O estado do programa (jogo, por exemplo) pode ser salvo e recarregado em outro momento



# Operações Básicas

---

- Abertura do arquivo
  - Liga uma variável do programa com o arquivo físico
  - Essa variável deve ser usada no programa para manipular o arquivo (ler e escrever no arquivo)
- Fechamento do arquivo
  - Encerramento da conexão da variável com o arquivo físico
- Leitura do conteúdo do arquivo
- Escrita no arquivo

# Escrita de arquivos

---

- É muito parecido com escrita no monitor, só que é necessário conectar com o arquivo antes (abrir o arquivo)
- Vamos ver um exemplo...



# Exemplo: escrevendo números aleatórios em um arquivo

*O arquivo aparecerá na raiz do projeto do PyCharm*

```
import random
```

```
def escreverNumerosAleatorios(qtdNumeros, nomeArquivo):  
    arquivoNumeros = open(nomeArquivo, 'w')  
    for i in range(qtdNumeros):  
        arquivoNumeros.write(str(random.randint(0,100)))  
        arquivoNumeros.write("\n")  
    arquivoNumeros.close()
```

```
escreverNumerosAleatorios(100, 'aleatorios.txt')
```

*Abertura do arquivo para escrita e posterior fechamento*



# Exemplo: escrevendo números aleatórios em um arquivo

---

```
import random
```

```
def escreverNumerosAleatorios(qtdNumeros, nomeArquivo):  
    arquivoNumeros = open(nomeArquivo, 'w')  
    for i in range(qtdNumeros):  
        arquivoNumeros.write(str(random.randint(0,100)))  
        arquivoNumeros.write("\n")  
    arquivoNumeros.close()
```

```
escreverNumerosAleatorios(100, 'aleatorios.txt')
```

*Escrita no arquivo*





# Leitura de arquivos

---

- Novamente, é muito parecido com leitura do teclado, só que é necessário conectar com o arquivo antes (abrir o arquivo)
- Vamos ver um exemplo...



## Detalhes do comando de leitura

---

- Necessário que o arquivo tenha sido aberto em modo leitura ou leitura/escrita
- `varString = varArquivo.readline()`
  - Lê uma linha do arquivo e a retorna como string
- `varListaString = varArquivo.readlines()`
  - Lê o arquivo do ponto atual até o final, e retorna o conteúdo em uma lista de strings
  - Cada linha do arquivo é guardada em uma posição da lista

# Arquivos Texto: Sempre String

---

- Para inserir valores em um arquivo, primeiro é necessário convertê-los para strings

```
>>>arq.write(str(12.3))  
>>>arq.write(str([1, 2, 3]))
```

- Quando você lê esses valores de volta, você obtém uma string. O tipo original do dado foi perdido...

```
>>> arq.readline()  
'12.3[1, 2, 3]'
```

# Exercícios

---

1. Faça um programa que leia um número N e gere um arquivo com N nomes e idades aleatórios

- Faça uso de duas listas criadas na mão: uma que contenha 20 nomes e outra que contenha 20 sobrenomes
- Cada linha do arquivo resultante deve conter um nome completo e a sua idade

2. Estenda o exemplo do cadastro para considerar também a altura da pessoa



# Exercícios

---

3. Escreva uma função que recebe dois nomes de arquivos e copia o conteúdo do primeiro arquivo para o segundo arquivo. Considere que o conteúdo do arquivo de origem é um texto. Sua função não deve copiar linhas comentadas (que começam com `//`)
4. Faça um programa contendo uma função que recebe como argumentos os nomes de dois arquivos. O primeiro arquivo contém nomes de alunos e o segundo arquivo contém as notas dos alunos. No primeiro arquivo, cada linha corresponde ao nome de um aluno e no segundo arquivo, cada linha corresponde às notas dos alunos (uma ou mais). Assuma que as notas foram armazenadas como strings, e estão separadas umas das outras por espaços em branco. Leia os dois arquivos e gere um terceiro arquivo que contém o nome do aluno seguido da média de suas notas.



## Exercícios

---

5. Faça um programa para alterar uma das notas de um aluno (usando os arquivos do exercício anterior). O programa deve ter uma função que recebe o nome do aluno, a nota velha e a nova nota. A função deve fazer a alteração no arquivo.

6. Faça uma função que leia um arquivo texto contendo uma lista de endereços IP e gere dois outros arquivos, um contendo os endereços IP válidos e outro contendo os endereços inválidos. O formato de um endereço IP é **num1.num.num.num**, onde **num1** vai de 1 a 255 e **num** vai de 0 a 255.