

# Introdução à Programação Python

[github.com/kaylani2/iHUB\\_Python](https://github.com/kaylani2/iHUB_Python)

Kaylani Bochie

iHUB

# Conteúdo

- 1 Conceitos Básicos
  - Introdução ao Pensamento Computacional
  - Programação em Linguagens Estruturadas
- 2 Entrada e Saída de Dados
- 3 Estruturas de Controle e Repetição
  - if / else
  - for e while
- 4 Funções, Recursividade e Modularização de Código
- 5 Estruturas de Dados
  - Listas
  - Tuplas
  - Sets
  - Dicionários
- 6 Manipulação de Arquivos
- 7 Programação Orientada a Objetos
- 8 Tratamento de Exceção

- Já usou um computador antes?
- Tem computador em casa com acesso à internet?
- Já programou antes?
- Entende inglês?
- Qual seu objetivo ao aprender a programar?

- Internet Explorer, Firefox;
- Windows Explorer, Microsoft Word;
- iTunes, Windows Media Player;
- Angry Birds, WhatsApp.

**Como escrever os programas para que o computador entenda?**

## Níveis de linguagem

- Linguagem natural;
- **Linguagens de alto nível;**
- Linguagem de montagem (*Assembly*);
- Linguagem de máquina;

## ***High Level Language*** (HLL)

- Fácil para o programador entender;
- Menos eficiente (memória);
- Simples de "debugar";
- É portátil;
- Independe da plataforma;
- Fácil de manter;
- Precisa de um compilador ou interpretador;

## ***Low Level Language*** (LLL)

- Fácil para a máquina entender;
- Mais eficiente (memória);
- Difícil de "debugar";
- Não é portátil;
- Depende da plataforma;
- Difícil de manter;
- Precisa de um *assembler*;

```
print ('Hello, World!')
```

```
#include <stdio.h>

int main ()
{
    printf ("Hello, World!\n");
    return 0;
}
```



# Assembly

```
global _start

section .text

_start:
    mov rax, 1
    mov rdi, 1
    mov rsi, msg
    mov rdx, msglen
    syscall

    mov rax, 60
    mov rdi, 0
    syscall

section .rodata
    msg: db "Hello, World!", 10
    msglen: equ $ - msg
```

# Hierarquia de Dados

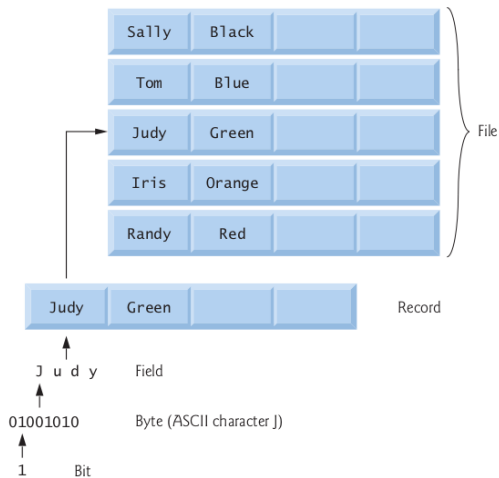


Figura 1: Hierarquia de dados [1].

# Linguagens Compiladas X Linguagens Interpretadas

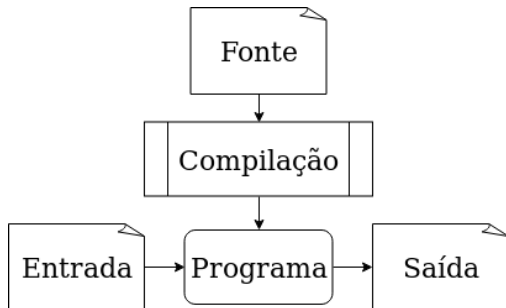


Figura 2: Compiladas

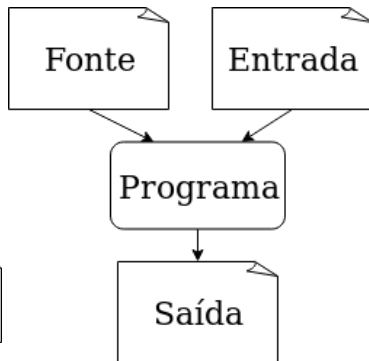


Figura 3: Interpretadas

# Paradigmas

- Imperativo:
  - **Procedural** (COBOL, C, Python [2, 3, 4]);
  - **Orientado a objetos** (Python, C++, Java).
- Declarativo:
  - Funcional (Lisp, Python);
  - Lógico;
  - Matemático.

## Linguagens multi-paradigmas



# Hello, World!



# Interagindo com o Programa



# Tipos de Dados

- Inteiros (int);
- Reais (float);
- Texto (string);
- Condicionais (boolean, bool);
- Grupos "homogêneos"(arrays, vectors, lists);
- Compostos (dictionary, struct, object).



- Hello, Usuário.
- Calcular a área de um retângulo.
- Calcular o rendimento de um investimento (juros simples).

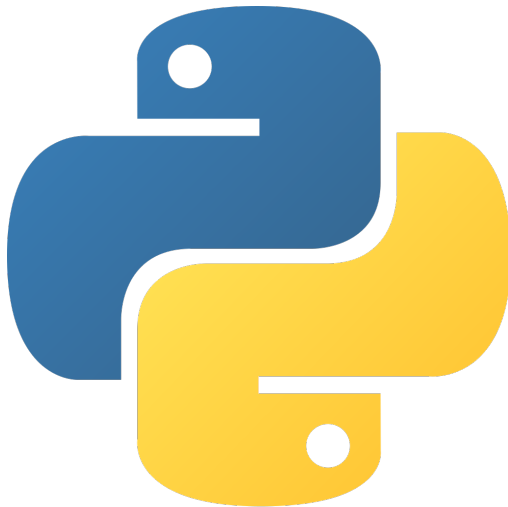


## ***Automate the Boring Stuff with Python***, Capítulo 1 - Python Basics [2]

- Compiladores e interpretadores;
- Níveis de linguagem;
- Boas práticas de programação;
- Tipos de dados;
- Solução de problemas.



if / else



- Dada a temperatura, dizer se a sala está muito quente, quente, agradável, fria ou muito fria.
- A partir das notas e faltas, dizer a condição do aluno.



# Repetição for e while



## Usar os dois laços de repetição!

- Calcular a média de N notas.
- Calcular o fatorial de um número N.



## help (tipo)

- help (str)
- len ('string')
- str.upper ('string')
- str.lower ('string')
- str.count (umaString, elemento, inicio, fim)
- str.index ('string', 'i')



## ***Automate the Boring Stuff with Python***, Capítulo 2 - Flow Control [2]

### ● **BOAS PRÁTICAS DE PROGRAMAÇÃO!**

- Indentação;
  - Identificadores com nomes descritivos;
  - Consistência;
  - Legibilidade.
- Controle de fluxo.







## Escreva uma função que:

- Calcula o fatorial de N utilizando *for*, *while* e recursão.
- Calcula área de um retângulo.
- Calcula o rendimento de um investimento (juros simples).
- Calcula o enésimo termo da Sequência de Fibonacci.
- Calcula a combinação simples de N elementos P a P.

$$C_p^n = n! / p!(n - p)!$$



***Automate the Boring Stuff with Python***, Capítulo 3 - Functions [2]

***Python Cookbook***, Capítulo 7 - Functions [2]

## ● **BOAS PRÁTICAS DE PROGRAMAÇÃO!**

- Escrita de funções;
  - Funções não interagem com o usuário;
  - Modularização de código;
  - Indentação;
  - Identificadores com nomes descritivos (*camelCase*);
  - Consistência;
  - Legibilidade.
- 
- Funções;
  - Recursividade:
    - Recursividade:
      - Recursividade...

- **Listas** (*list [ ]*);
- Vetores e matrizes;
- Pilhas;
- Filas;
- **Tuplas** (*tuple ( )*);
- **Dicionários** (*dict { : }*);
- **Sets** (*set { }*);
- Grafos.
- Indexação;
- Concatenação;
- Replicação;
- Iteração;
- Fatiamento (*slicing*);
- Métodos.



## Escreva uma função que:

- Calcula a média de uma turma, onde a lista recebida pela função contém dicionários do tipo:

```
{'nome': 'Nome do aluno', 'registro': 001, 'notas': [0,  
5, 10]}
```

- Identifica o maior elemento de uma lista.
- Remove elementos duplicados de uma lista usando apenas o método `.pop()`.
- Ordena uma lista sem utilizar o método `.sort()`.
- Calcula a soma de todos os números primos até 900000.
- Multiplica duas matrizes.



***Python Docs - The Python Tutorial***, Capítulo 5 - Data Structures [5]

## ● **BOAS PRÁTICAS DE PROGRAMAÇÃO!**

- Modularização de código;
  - Indentação;
  - Identificadores com nomes descritivos (*camelCase*);
  - Consistência;
  - Legibilidade.
- 
- Vetores e matrizes (para dados homogêneos);
  - Dicionários (para dados heterogêneos);
  - Imutabilidade de tuplas;
  - Conjuntos desordenados através de *sets*.

# JavaScript Object Notation (JSON)

```
{  
  "id": "108",  
  "info":  
  {  
    "stock": 10,  
    "colors": [  
      "red",  
      "green",  
      "blue"],  
    "price": 40,  
    "location": [3, 120],  
    "available": true  
  }  
}
```



- `f = open ('nome', 'modo')`
- `f.read ()`, `f.readline ()`, `f.write ('string')`, `f.close ()`
- `with open ('nome') as f:`
- *JavaScript Object Notation* (JSON)
  - `json.load (handle)`
  - `json.dump (dict, handle)`

***Automate the Boring Stuff with Python***, Capítulo 8 - Reading and Writing Files [2]

## ● **BOAS PRÁTICAS DE PROGRAMAÇÃO!**

- Complexidade de algoritmos;
- Manipulação de arquivos;
- Módulos de terceiros;
- Prova.

Objetivo: Abstração

Classes:

- Conceito estendido de estrutura de dados;
- Além de organizar dados, oferecem funções de manipulação;
- Encapsulamento;
- Elementos públicos e privados;

Objetos:

- Instâncias de classes;

- Métodos construtores.
- Atributos privados;
- *getattr*, *hasattr*, *setattr*, *delattr*;
- Atributos de classe: *\_\_dict\_\_*, *\_\_doc\_\_*, *\_\_name\_\_*...
- Métodos *set* e *get*;
- Testes de validade (*set*).



## ***Python Docs - The Python Tutorial***, Capítulo 9 - Classes [5]

- Classes e objetos;
- Encapsulamento;
- Herança;
- Construtores e destrutores;
- Escopo;
- Atributos estáticos.



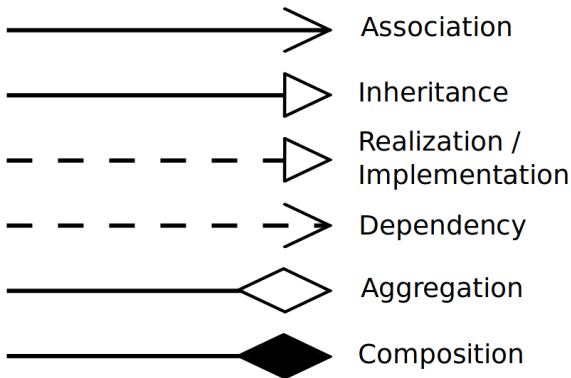


Figura 4: Relações entre classes [6].

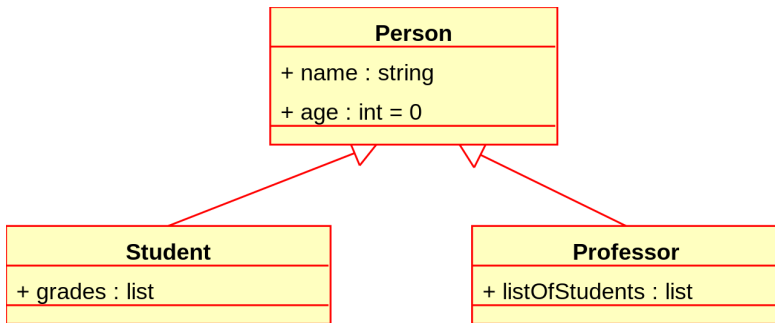







Figura 5: Diagrama de classes [7].





# Trabalho Final!



-  P. Deitel and H. Deitel, *C++ How to Program*. Pearson, 9 ed., 2014.
-  D. Beazley and B. K. Jones, *Automate the Boring Stuff with Python*. William Pollock, 2015.
-  L. Ramalho, *Fluent Python*. O'Reilly Media, Inc., 2015.
-  D. Beazley and B. K. Jones, *Python Cookbook*. O'Reilly Media, Inc., 2013.
-  P. S. Foundation, "The python tutorial; python 3.7.5rc1 documentation." <https://docs.python.org/3/tutorial/>, 2019. Acessado em 13/10/2019.



Yampas, “Uml notation for oop.”

<https://commons.wikimedia.org/w/index.php?curid=48015014>,  
2016.

Acessado em 30/10/2019.



K. Pietrzak, “Class diagram generalization.”

[https://upload.wikimedia.org/wikipedia/commons/6/66/  
KP-UML-Generalization-20060325.svg](https://upload.wikimedia.org/wikipedia/commons/6/66/KP-UML-Generalization-20060325.svg), 2006.

Acessado em 30/10/2019.