





# BEM-VINDOS AO PROGRAMA DE TALENTOS 2022!

Git / Python / FastAPI

## APRESENTAÇÃO



Fernando Celmer Fernando Celmer Site: https://fernandocelmer.com/

Github: https://github.com/FernandoCelmer

Linkedin: https://www.linkedin.com/in/fernando-celmer/

Discord: fernando.celmer#0679

#### Cronograma

- Git (Github)
  - Dia 1 15-08-2022
- Python Básico
  - Dia 2 16-08-2022
- Framework FastAPI
  - Dia 3 17-08-2022
  - Dia 4 18-08-2022
  - Dia 5 19-08-2022

# AVISOS

https://discord.gg/MrYP3XRb

- Grupo Discord
- Git (Github)
- Configuração Python
- Explicação atividades

# DIA 2 PYTHON BÁSICO

## 0 QUE É PYTHON?

Python é uma linguagem de programação de alto nível, interpretada de script, imperativa, orientada <u>a objetos</u>, funcional, de tipagem dinâmica e forte. Foi lançada por <mark>Guido van</mark>

Rossum em 1991.



## POR QUE PYTHON?

- Python funciona em diferentes plataformas (Windows, Mac, Linux, etc).
- Python tem uma sintaxe simples semelhante ao idioma inglês.
- Python tem sintaxe que permite aos desenvolvedores escrever programas com menos linhas do que algumas outras linguagens de programação.
- Python é executado em um sistema interpretador, o que significa que o código pode ser executado assim que for escrito. Isso significa que a prototipagem pode ser muito rápida.
- Python pode ser tratado de forma procedural, orientada a objetos ou funcional.

### SINTAXE DO PYTHON

- O Python foi projetado para facilitar a leitura e possui algumas semelhanças com o idioma inglês.
- Python usa novas linhas para completar um comando, ao contrário de outras linguagens de programação que geralmente usam ponto e vírgula ou parênteses.

#### print("Hello, World!")

• Python depende de recuo, usando espaço em branco, para definir o escopo; como o escopo de loops, funções e classes. Outras linguagens de programação costumam usar colchetes para essa finalidade.



## VARIÁVEIS

Python não tem comando para declarar uma variável. Uma variável é criada no momento em que você atribui um valor a ela.

x = 5

y = "John"

print(x)

print(y)

As variáveis não precisam ser declaradas com nenhum tipo específico e podem até mudar de tipo depois de terem sido definidas.

### VARIÁVEIS - CONVERSÃO

Se você deseja especificar o tipo de dados de uma variável, isso pode ser feito com conversão.

```
x = str(3)  # x will be '3'
y = int(3)  # y will be 3
z = float(3)  # z will be 3.(
```

### VARIÁVEIS - NOMES

Uma variável pode ter um nome curto (como x e y) ou um nome mais descritivo (idade, name, total\_volume). Regras para variáveis Python:

#### Nomes de variáveis legais:

myvar = "John"

my\_var = "John"

\_my\_var = "John"

myVar = "John"

MYVAR = "John"

myvar2 = "John"

#### Nomes de variáveis ilegais:

2myvar = "John"

my-var = "John"

my var = "John"

### TIPOS DE DADOS

Na programação, o tipo de dados é um conceito importante.

Variáveis podem armazenar dados de diferentes tipos, e diferentes tipos podem fazer coisas diferentes.

O Python tem os seguintes tipos de dados integrados por padrão, nestas categorias:

- Tipo de texto: str
- Tipos Numéricos: int, float
- Tipos de sequência: list, tuple, range
- Tipo de mapeamento: dict
- Tipos de conjunto: set
- Tipo booleano: bool
- Tipos binários: bytes
- Nenhum Tipo: NoneType

### OBTENDO O TIPO DE DADOS

Você pode obter o tipo de dados de qualquer objeto usando a type() função:

x = 5

print(type(x))

## STRINGS

#### Strings:

Em python são cercadas por aspas simples ou aspas duplas.

'olá' é o mesmo que "olá" .

print(a)

Atribuir String a uma Variável

a = "Hello"

print(a)

### STRINGS - SEPARANDO

Você pode retornar um intervalo de caracteres usando a sintaxe de fatia. Especifique o índice inicial e o índice final, separados por dois pontos, para retornar uma parte da string.

b = "Hello, World!"

print(b[2:5])

### STRINGS - MODIFICANDO

Python tem um conjunto de métodos integrados que você pode usar em strings.

O método upper() retorna a string em maiúsculas:

a = "Hello, World!"

print(a.upper())

O método lower() retorna a string em letras minúsculas:

a = "Hello, World!"

print(a.lower())

### VALORES BOOLEANOS

Booleanos são representados por dois valores: True ou False.

É muito utilizado na programação nas vezes você precisa saber se uma expressão é <mark>True</mark> ou False.

Você pode avaliar qualquer expressão em Python e obter uma das duas respostas, True ou False.

Quando você compara dois valores, a expressão é avaliada e o Python retorna a resposta booleana:

print(10 > 9)

print(10 == 9)

print(10 < 9)

### VALORES BOOLEANOS

print("b não é maior que a")

```
Quando você executa uma condição em uma instrução if, o Python retorna True ou False:
Ex:
a = 200
b = 33
if b > a:
     print("b é maior que a")
else:
```

#### OPERADORES PYTHON

Os operadores são usados para realizar operações em variáveis e valores.

No exemplo abaixo, usamos o + operador para somar dois valores:

#### print(10 + 5)

#### Python divide os operadores nos seguintes grupos:

- Operadores aritméticos
- Operadores de atribuição
- Operadores de comparação
- Operadores lógicos
- Operadores de identidade
- Operadores de associação
- Operadores bit a bit



As listas são usadas para armazenar vários itens em uma única variável.

As listas são um dos 4 tipos de dados internos do Python usados para armazenar coleções de dados, os outros 3 são Tuple , Set e Dictionary , todos com qualidades e usos diferentes.

As listas são criadas usando colchetes:

thislist = ["apple", "banana", "cherry"]

print(thislist)

### LISTAS – ACESSAR ITENS

Os itens da lista são indexados e você pode acessá-los consultando o número do índice:

thislist = ["apple", "banana", "cherry"]

print(thislist[1])

#### LISTAS – ALTERAR VALOR DO ITEM

Para alterar o valor de um item específico, consulte o número do índice:

thislist = ["apple", "banana", "cherry"]

thislist[1] = "blackcurrant"

print(thislist)

### LISTAS - ADICIONAR ITENS DE LISTA

Para adicionar um item ao final da lista, use o método append():

thislist = ["apple", "banana", "cherry"]

thislist.append("orange")

print(thislist)

### LISTAS – PERCORRER UMA LISTA

```
Você pode percorrer os itens da lista usando um for loop:
thislist = ["apple", "banana", "cherry"

for x in thislist:
    print(x)
```

## TUPLAS

Tuplas são usadas para armazenar vários itens em uma única variável.

Uma tupla é uma coleção ordenada e imutável . Tuplas são escritas com colchetes.()

thistuple = ("apple", "banana", "cherry")

#### print(thistuple)

#### Itens de Tupla:

Os itens de tupla são ordenados, imutáveis e permitem valores duplicados. Os itens de tupla são indexados, o primeiro item possui índice [0], o segundo item possui índice [1]etc.

#### Imutável:

As tuplas são imutáveis, o que significa que não podemos alterar, adicionar ou remover itens após a criação da tupla.

### TUPLAS – ACESSAR ITENS

Você pode acessar os itens da tupla consultando o número do índice, entre colchetes:

thistuple = ("apple", "banana", "cherry")

print(thistuple[1])

## DICIONÁRIOS

Os dicionários são usados para armazenar valores de dados em pares chave:valor. Um dicionário é uma coleção ordenada, mutável e que não permite duplicatas.

Os dicionários são escritos com colchetes e possuem chaves e valores:

```
thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
```

print(thisdict)

### DICIONÁRIOS – ACESSAR ITENS

Você pode acessar os itens de um dicionário consultando o nome da chave, entre colchetes:

thisdict = {

"model": "Mustang"

}

x = thisdict["model"]

Existe também um método chamado get() que lhe dará o mesmo resultado:

x = thisdict.get("model")

### DICIONÁRIOS – ALTERAR ITENS

Você pode alterar o valor de um item específico consultando seu nome de chave:

```
thisdict = {
     "brand": "Ford",
     "model": "Mustang",
    "year": 1964
thisdict["year"] = 2018
O método update() atualizará o dicionário com os itens do argumento fornecido.
thisdict.update({"year": 2020})
```

#### DICIONÁRIOS – ADICIONAR ITENSS

A adição de um item ao dicionário é feita usando uma nova chave de índice e atribuindo um valor a ela:

```
thisdict = {
    "brand": "Ford",
    "model":"Mustang",
    "year": 1964
}
thisdict["color"] = "red"
print(thisdict)
```

### DICIONÁRIOS - REMOVER ITENS

Existem vários métodos para remover itens de um dicionário:

O método pop() remove o item com o nome de chave especificado:

#### thisdict.pop("model")

O método popitem() remove o último item inserido (nas versões anteriores à 3.7, um item aleatório é removido):

#### thisdict.popitem()

A del palavra-chave remove o item com o nome de chave especificado:

#### del thisdict["model"]

O método clear() esvazia o dicionário:

#### thisdict.clear()

## If ... ELSE

#### Condições do Python e instruções If:

Python suporta as condições lógicas usuais da matemática:

- Igual a: a == b
- Diferentes: a != b
- Menor que: a < b</p>
- Menor ou igual a: a <= b</p>
- Maior que: a > b
- Maior ou igual a: a >= b

## If ... ELSE

Essas condições podem ser usadas de várias maneiras, mais comumente em "instruções if" e loops.

a = 33

b = 200

if b > a:

print("b is greater than a")

## WHILE LOOPS

Python tem dois comandos de loop primitivos:

- while loops
- for loops

Com o loop while podemos executar um conjunto de instruções desde que uma condição seja verdadeira.

i = 1

while i < 6:

print(i) i += 1

## FOR LOOPS

Um loop for é usado para iterar sobre uma sequência (que é uma lista, uma tupla, um dicionário, um conjunto ou uma string).

Isso é menos parecido com a palavra-chave for em outras linguagens de programação e funciona mais como um método iterador, conforme encontrado em outras linguagens de programação orientadas a objetos.

Com o loop for podemos executar um conjunto de instruções, uma vez para cada item de uma lista, tupla, conjunto etc.

fruits = ["apple", "banana", "cherry"]

for x in fruits:

print(x)

## FUNÇÕES

Uma função é um bloco de código que só é executado quando é chamado. Você pode passar dados, conhecidos como parâmetros, para uma função.

Uma função pode retornar dados como resultado.

Criando uma função:

def my\_function():

print("Hello from a function")

Chamando uma função:

my\_function()

## FUNÇÕES – ARGUMENTOS

As informações podem ser passadas para funções como argumentos. Os argumentos são especificados após o nome da função, dentro dos parênteses.

Você pode adicionar quantos argumentos quiser, basta separá-los com uma vírgula.

O exemplo a seguir tem uma função com um argumento (fname). Quando a função é chamada, passamos um primeiro nome, que é usado dentro da função para imprimir o nome completo:

def my\_function(fname):

print(fname + " Refsnes")

my\_function("Emil")

my\_function("Tobias")

my\_function("Linus")

# CLASSE

Python é uma linguagem de programação orientada a objetos. Quase tudo em Python é um objeto, com suas propriedades e métodos. Uma classe é como um construtor de objetos, ou um "projeto" para criar objetos.

Para criar uma classe, deve-se usar a palavra-chave class:

#### class MyClass:

x = 5

Criar objeto:

p1 = MyClass()

print(p1.x)

## CLASSE - A FUNÇÃO \_\_INIT\_\_()

Para entender o significado das classes, temos que entender a função \_\_init\_\_() embutida.

Todas as classes possuem uma função chamada \_\_init\_\_(), que sempre é executada quando a classe está sendo iniciada.

## CLASSE - A FUNÇÃO \_\_INIT\_\_()

A função \_\_init\_\_() é usada para atribuir valores às propriedades do objeto ou outras operações que são necessárias quando o objeto está sendo criado:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

p1 = Person("John", 36)

print(p1.name)

print(p1.age)
```



A herança nos permite definir uma classe que herda todos os métodos e propriedades de outra classe.

A classe pai é a classe que está sendo herdada, também chamada de classe base.

A classe filha é a classe que herda de outra classe, também chamada de classe derivada.

## HERANÇA – CLASSE PAI

Qualquer classe pode ser uma classe pai, então a sintaxe é a mesma da criação de qualquer outra classe:

#### class Person:

```
def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

def printname(self):
    print(self.firstname, self.lastname)
```

```
x = Person("John", "Doe")
x.printname()
```

## HERANÇA – CLASSE FILHA

Para criar uma classe que herde a funcionalidade de outra classe, envie a classe pai como parâmetro ao criar a classe filha:

class Student(Person):

pass

# TRY EXCEPT

- O try bloco que permite testar um bloco de código quanto a erros.
- O except bloco que permite que você lide com o erro.
- O else bloco permite executar código quando não há erro.
- O finally bloco permite que você execute código, independentemente do resultado dos blocos try e except.

## MANIPULAÇÃO DE EXCEÇÃO

Quando ocorre um erro, ou exceção, como chamamos, o Python normalmente para e gera uma mensagem de erro. Essas exceções podem ser tratadas usando a try:

Este bloco de try irá gerar uma exceção, pois x não está definido:

try:

print(x)

except:

print("An exception occurred")

Como o bloco try gera um erro, o bloco except será executado. Sem o bloco try, o programa irá travar e gerar um erro:

# PYTHON PIP

PIP é um gerenciador de pacotes para pacotes Python, ou módulos, se preferir.

#### O que é um Pacote?

Um pacote contém todos os arquivos necessários para um módulo. Módulos são bibliotecas de código Python que você pode incluir em seu projeto.

### PYTHON - AMBIENTE VIRTUAL

O ambiente virtual ou módulo venv irá isolar as dependências do projeto de modo onde cada projeto terá suas bibliotecas de maneira separada do sistema principal.

# ATIVIDADE

eeeeeh!

# ATIVIDADES

Link: <a href="https://github.com/CakeERP/cakeerp-talent-program-2022">https://github.com/CakeERP/cakeerp-talent-program-2022</a>

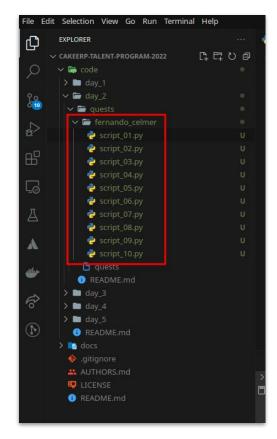
#### Onde fazer as atividades?

As atividades devem ser desenvolvidas dentro do repositório cakeerp-talent-program-2022 no seguinte local. code > day\_2 > quests > <seu\_nome>.

#### Como enviar as atividades?

Para envio das atividades deve-se realizar todo fluxo referente ao Github (add/commit/push/pull request).

OBS: Não commitar o ambiente virtual!



# ATIVIDADES

Link: <a href="https://github.com/CakeERP/cakeerp-talent-program-2022">https://github.com/CakeERP/cakeerp-talent-program-2022</a>

#### O que é preciso fazer?

- 1 Faça um Programa que leia um vetor de 5 números inteiros e mostre-os.
- 2 Faça um Programa que peça a idade e a altura de 5 pessoas, armazene cada informação no seu respectivo vetor. Imprima a idade e a altura na ordem inversa a ordem lida.
- 3 Utilizando listas faça um programa que faça 5 perguntas para uma pessoa sobre um crime. As perguntas são:
  - A. "Telefonou para a vítima?"
  - B. "Esteve no local do crime?"
  - C. "Mora perto da vítima?"
  - D. "Devia para a vítima?"
  - E. "Já trabalhou com a vítima?"

O programa deve no final emitir uma classificação sobre a participação da pessoa no crime. Se a pessoa responder positivamente a 2 questões ela deve ser classificada como "Suspeita", entre 3 e 4 como "Cúmplice" e 5 como "Assassino". Caso contrário, ele será classificado como "Inocente".

- 4 Faça um programa, com uma função que necessite de três argumentos, e que forneça a soma desses três argumentos.
- 5 Faça um programa com uma função chamada somaImposto. A função possui dois parâmetros formais: taxaImposto, que é a quantia de imposto sobre vendas expressa em porcentagem e custo, que é o custo de um item antes do imposto. A função "altera" o valor de custo para incluir o imposto sobre vendas.

# ATIVIDADES

- 6 Faça um Programa que peça dois números e imprima o maior deles.
- 7 Faça um Programa que verifique se uma letra digitada é "F" ou "M". Conforme a letra escrever: F Feminino, M Masculino, Sexo Inválido.
- 8 Faça um programa para a leitura de duas notas parciais de um aluno. O programa deve calcular a média alcançada por aluno e apresentar:
  - A. A mensagem "Aprovado", se a média alcançada for maior ou igual a sete;
  - B. A mensagem "Reprovado", se a média for menor do que sete;
  - C. A mensagem "Aprovado com Distinção", se a média for igual a dez.
- 9 Faça um Programa que leia três números e mostre-os em ordem decrescente.
- 10 As Organizações Tabajara resolveram dar um aumento de salário aos seus colaboradores e lhe contraram para desenvolver o programa que calculará os reajustes. Faça um programa que recebe o salário de um colaborador e o reajuste segundo o seguinte critério, baseado no salário atual:
  - A. salários até R\$ 280,00 (incluindo) : aumento de 20%
  - B. salários entre R\$ 280,00 e R\$ 700,00 : aumento de 15%
  - C. salários entre R\$ 700.00 e R\$ 1500.00 : aumento de 10%
  - D. salários de R\$ 1500,00 em diante : aumento de 5% Após o aumento ser realizado, informe na tela:
  - E. o salário antes do reajuste;
  - F. o percentual de aumento aplicado;
  - G. o valor do aumento;
  - H. o novo salário, após o aumento.



- Introduction to Python W3Schools
- Python Operators
- <u>Criando ambientes virtuais para projetos Python com o Virtualenv</u>
- Ambientes virtuais em Python