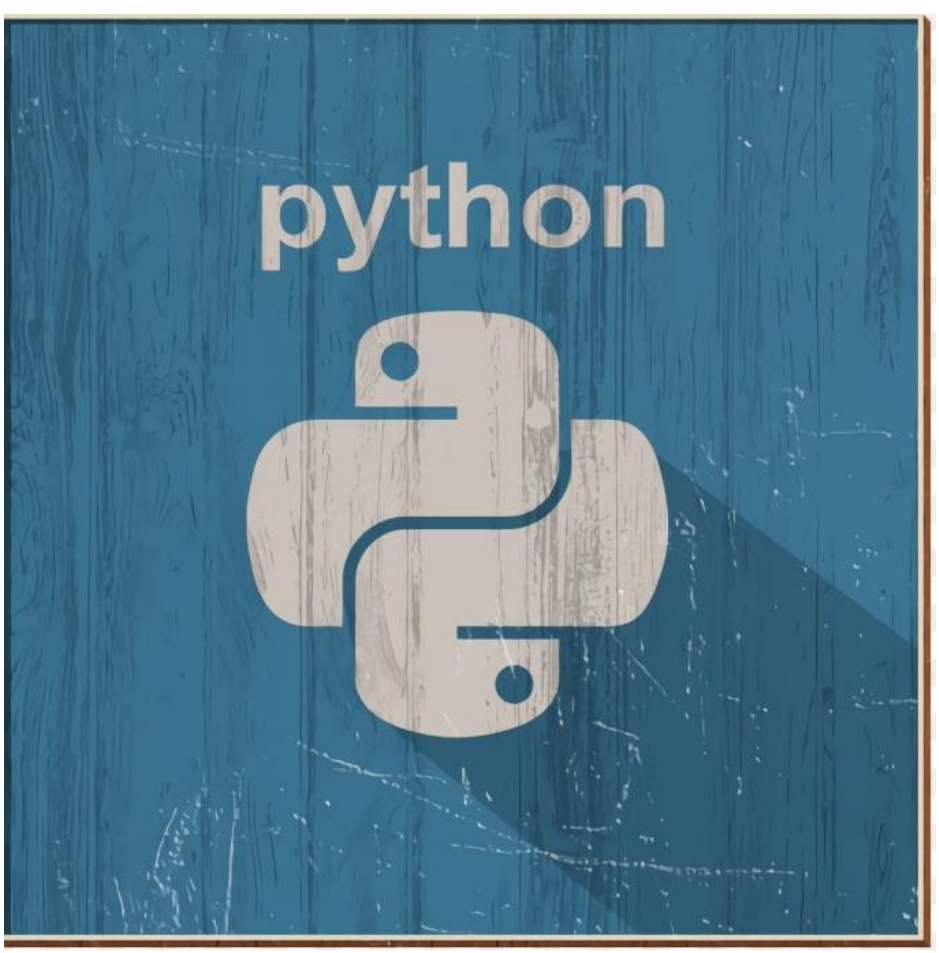
@Odemir Depieri Jr

Guia rápido de Python



Variáveis e Texto (String)

Variáveis e Strings

Variáveis são usadas para armazenar valores.
Uma **String** é uma série de caracteres, entre aspas simples ou duplas.

```
[3]  # Mostrando uma frase
     print('Olá Mundo!')
```

```
Olá Mundo!
```

```
[4]  # Frase definida com uma variavel
     Mensagem = 'Olá Mundo!'
     print(Mensagem)
```

```
Olá Mundo!
```

```
[6]  # Juntando ou combinando textos
     Primeiro_Nome = 'Odemir'
     Ultimo_Nome = 'Depieri'
     Nome_Completo = Primeiro_Nome + ' ' + Ultimo_Nome
     print( Nome_Completo )
```

```
Odemir Depieri
```

Listas

Listas

Uma lista armazena uma **série de itens** em uma ordem específica.
Pode acessar os itens usando um índice ou dentro de um loop de repetição.

```
[7]  # Criando uma lista
     Frutas = ['Maça', 'Banana', 'Pera']
     Frutas
```

```
['Maça', 'Banana', 'Pera']
```

```
[8]  # Acessando a 1º primeira posição da lista
     Frutas[0]
```

```
'Maça'
```

```
[9]  # Acessando a ultima posição da lista
     Frutas[-1]
```

```
'Pera'
```

```
[10] # Acessando a lista atraves de um loop de repetição
     for Qual_Fruta in Frutas:
       print( Qual_Fruta )
```

```
Maça
Banana
Pera
```

```
[11] # Adicionando itens em uma lista
     Lista_Vazia = []

     Lista_Vazia.append( 'Morango' )
     Lista_Vazia.append( 'Laranja' )
     Lista_Vazia.append( 'Limão' )

     print( Lista_Vazia )
```

```
['Morango', 'Laranja', 'Limão']
```

```
[14] # Criando uma lista numerica
     Serie_Numerica = []

     # Loop para repetir 10x
     for x in range(1,11):
       # multiplicando cada loop por 2x
       Serie_Numerica.append(x*2)

     print( Serie_Numerica )
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

```
[16] # Criando uma lista numerica de uma forma resumida
     # Mesma situação do exemplo de cima

     Serie_Numerica = [ x*2 for x in range(1, 11) ]
     Serie_Numerica
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

```
[12] # Fatiando uma lista
     Serie_Numerica[2:]
```

```
[6, 8, 10, 12, 14, 16, 18, 20]
```

```
[15] # Fatiando uma lista
     Serie_Numerica[2:-1]
```

```
[6, 8, 10, 12, 14, 16, 18]
```

Tuplas

Tuplas

Tuplas são semelhantes a listas, mas os itens em uma tuplas não podem ser modificado.

```
[18] # Criando uma tupla
     Frutas  =( 'Maça', 'Pera', 'Uva ')
     Frutas
```

```
     ('Maça', 'Pera', 'Uva ')
```

Condições

Condições

Declarações são usadas para testar condições particulares e responder apropriadamente..

Condições

```
[20] # Igual
     x == 1

     # Não é igual
     x != 1

     # Maior que, Maior e igual a
     x > 0
     x >= 0

     # Menor que, Menor e igual a
     x < 0
     x <= 0
```

Condições em uma lista

```
[21] # Validação em uma lista
     Frutas = ['Maça', 'Banana', 'Pera']
     'Banana' in Frutas
```

```
     True
```

Condições em Booleano

```
[22] # Condições Booleanos ( Verdadeiro ou Falso )
     Acessar_Sistema = True
     Cadastrar_Usuario = False
```

Condição usando SE

```
[23] # Condição usando SE, Forma simples

     Idade = 20

     if Idade > 18:
       print('Maior de Idade ')
```

```
Maior de Idade
```

```
[26] # Condição usando SE, ELIF, ELSE

     Idade = 66

     if Idade < 18:
       print( 'Menor de Idade - Jovem ')

     elif Idade > 18 and Idade < 65:
       print ('Maior de Idade - Adulto')

     else:
       print('3º Idade - Idoso')
```

```
3º Idade - Idoso
```

```
[29] # Condição SE com Boleano
     # Exemplo 1

     Acessar_Sistema = True

     if Acessar_Sistema:
       print('Entrando')

     else:
       print('Sem acesso')
```

```
Entrando
```

```
[31] # Condição SE com Boleano
     # Exemplo 2

     Acessar_Sistema = False

     if Acessar_Sistema:
       print('Entrando')

     else:
       print('Sem acesso')
```

```
Sem acesso
```

Dicionários

Dicionários

Dicionários armazenam conexões entre peças de
em formação. Cada item em um dicionário é um par de valores-chave.

```
[35] # Criando um Dicionario
     Dados  = { 'Nome': 'Odemir', 'Sobrenome': 'Depieri', 'Idade': 29 }
     Dados
```

```
{'Idade': 29, 'Nome': 'Odemir', 'Sobrenome': 'Depieri'}
```

```
[36] # Acessando valores
     print('Meu nome é ' + Dados['Nome'] + ' ' + Dados['Sobrenome'] )
```

```
Meu nome é Odemir Depieri
```

```
[38] # Adicionando um valor no dicionario
     Dados['Linguaguem de Programação'] = 'Python'
     Dados
```

```
{'Idade': 29,
 'Linguaguem de Programação': 'Python',
 'Nome': 'Odemir',
 'Sobrenome': 'Depieri'}
```

```
[48] # Loop para entrar no dicionaio
     Dados = { 'Nome': 'Odemir', 'Sobrenome': 'Depieri' }
     for nome, sobrenome in Dados.items():
      print(nome ,sobrenome )
```

```
Nome Odemir
Sobrenome Depieri
```

```
[52] # Loop para entrar no dicionaio - Acessando as chaves
     Dados = { 'Nome': 'Odemir', 'Sobrenome': 'Depieri' }
     for consulta in Dados.keys():
      print( consulta )
```

```
Nome
Sobrenome
```

```
[53] # Loop para entrar no dicionaio - Acessando os Valores
     Dados = { 'Nome': 'Odemir', 'Sobrenome': 'Depieri' }
     for consulta in Dados.values():
      print( consulta )
```

```
Odemir
Depieri
```

Loops de Repetições

For

Um laço ‘FOR’ é utilizado para a iteração através de uma sequência (isto é, quer uma lista, uma tupla, um dicionário, um conjunto, ou uma cadeia).

```
[1]  # For --- aplicado em uma lista
     Frutas = [ 'Maça', 'Banana', 'Amora']
     for Consulta in Frutas:
       print(Consulta)
```

```
Maça
Banana
Amora
```

```
[2]  # For --- aplicando em um texto
     for x in 'paralelepípedo':
       print(x)
```

```
p
a
r
a
l
e
l
e
p
í
p
e
d
o
```

```
[3]  # For --- aplicando em uma sequencia
     for x in range(6):
       print(x)
```

```
0
1
2
3
4
5
```

```
[4]  # For --- aplicando em um sequencia
     for c in range(2,6):
       print(c)
```

```
2
3
4
5
```

```
[5]  # For --- aplicando em sequencia pulando valores
     for d in range(2, 30, 3):
       print(d)
```

```
2
5
8
11
14
17
20
23
26
29
```

```
[9]  # For --- Aplicando em uma sequencia com condição.
     for x in range(6):
       if x == 5:
         print(x)
     else:
       print('Acabou')
```

```
5
Acabou
```

While

While é um ciclo que podemos executar um conjunto de instruções enquanto uma condição for verdadeira.

```
[11] # While --- aplicando a uma sequencia
     Contador = 1
     while Contador < 6:
       print( Contador )
       Contador += 1
```

```
1
2
3
4
5
```

```
[12] # While --- aplicando a um sequencia com condições
     Contador = 1
     while Contador < 6:
       print( Contador )
       if Contador == 3:
         break
       Contador += 1
```

```
1
2
3
```

```
[12] # While --- aplicando a um sequencia com condições
     Contador = 1
     while Contador < 6:
       print( Contador )
       if Contador == 3:
         break
       Contador += 1
```

```
1
2
3
```

```
[15] # While --- aplicando usando o metodo continue
     Contador = 0
     while Contador < 6:
       Contador += 1
       if Contador == 3:
         continue
       print( Contador )
```

```
1
2
4
5
6
```

Funções

Função

Uma função é um bloco de código que só é executado quando é chamado.

```
[16] # Função Simples
     def Minha_Funcao():
       print('Função, funfoo !!')

     Minha_Funcao()
```

```
Função, funfoo !!
```

```
[17] # Função com arguneto
     def Minha_Funcao(Argumento):
       print('Meu nome é ' + Argumento)

     Minha_Funcao('Odemir')
```

```
Meu nome é Odemir
```

```
[19] # Função com varios argumentos
     def Minha_Funcao(Nome, Sobrenome, Idade):
       print('Meu nome é ' + Nome + ' ' + Sobrenome)
       print('Tenho ' + str( Idade ) + 'anos')

     Minha_Funcao('Odemir', 'Depieri', 25)
```

```
Meu nome é Odemir Depieri
Tenho 25anos
```

```
[21] # Função quando o argumento é desconhecido
     def Minha_funcao(*Argumento_Desconhecido):
       print( 'Meu nome é ' + Argumento_Desconhecido[2] )

     Minha_funcao('Marina', 'Julia', 'Zoe')
```

```
Meu nome é Zoe
```

```
[25] # Função com argumento padrão
     def Minha_Funcao( bebida = 'Cerveja'):
       print('Eu Adoro ' + bebida )

     Minha_Funcao()
     Minha_Funcao('Coca-Cola')
```

```
Eu Adoro Cerveja
Eu Adoro Coca-Cola
```

```
[26] # Função com loop de repetição
     def Minha_Funcao(Comidas):
       for x in Comidas:
         print(x)

     Lista_Comidas = ['Arroz', 'Feijão', 'Milho', 'Batata']

     Minha_Funcao( Lista_Comidas )
```

```
Arroz
Feijão
Milho
Batata
```

Classes

Classes

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 objeto ou um "projeto" para a criação de objetos.

```
[29] # Criando uma classe simples
     class Pessoa:
       def __init__(self, Nome, Idade):
         self.Nome = Nome
         self.Idade = Idade

       def Minha_funcao(self):
         print('Meu nome é ' + self.Nome)

     Argumento = Pessoa( 'Odemir', 29)
     Argumento.Minha_funcao()
```

```
Meu nome é Odemir
```

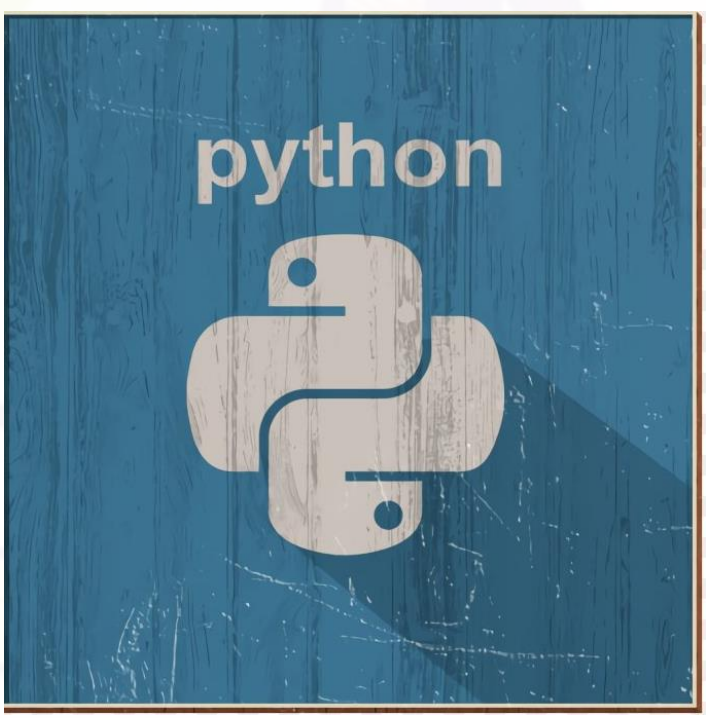
A função __init __ ()

O exemplo acima é uma classe e objeto em sua forma mais simples e não são realmente úteis em aplicativos da vida real.
Para entender o significado das classes, temos que entender a função interna __init __ ().
Todas as classes possuem uma função chamada __init __ (), que sempre é executada quando a classe está sendo iniciada.

Final

Esse guia é uma abordagem rápida sobre a linguagem Python.

Guia da documentação caso queira mais detalhes
https://www.python.org/doc/



Odemir Depieri Jr

Software Engineer Sr
Tech Lead
Specialization AI