Glossário de Python

Indentação

Indentação refere-se aos espaços no início de uma linha de código. Em Python, a indentação é usada para definir blocos de código.

```
if True:
    print("Indentação correta")
```

Comentários

Comentários são linhas de código que não serão executadas. Usam o caractere

```
#
```

.

```
# Este é um comentário
```

Comentários Multilinha

Comentários multilinha são feitos usando três aspas duplas ou simples.

```
"""
Este é um comentário
multilinha em Python
"""
```

Criando Variáveis

Variáveis são contêineres para armazenar valores de dados.

```
nome = "Maria"
idade = 30
```

Nomes de Variáveis

Como nomear suas variáveis: use letras, números e sublinhados. Não comece com números.

```
nome_usuario = "Ana"
idade_usuario = 25
```

Atribuir Valores a Múltiplas Variáveis

Como atribuir valores a várias variáveis de uma vez.

```
x, y, z = 1, 2, 3
```

Saída de Variáveis

Use a função

```
print()
```

para exibir variáveis.

```
nome = "Carlos"
print(nome)
```

Concatenação de Strings

Como combinar strings usando o operador

```
+
```

.

```
nome = "João"
saudacao = "Olá, " + nome
print(saudacao)
```

Variáveis Globais

Variáveis que pertencem ao escopo global.

```
nome = "Global"

def mostrar_nome():
    print(nome)

mostrar_nome()
```

Tipos de Dados Incorporados

Python tem um conjunto de tipos de dados incorporados, como

```
int
```

,

```
float
```

, e

```
str
```

.

Obter Tipo de Dados

Use

`type()`

para obter o tipo de um objeto.

```
x = 5
print(type(x))
```

Definir Tipo de Dados

Python é uma linguagem tipada dinamicamente, então o tipo é definido automaticamente.

Números

Existem três tipos numéricos em Python:

`int`

,

`float`

, e

`complex`

.

Int

Tipo de número inteiro.

```
num = 10
```

Float

Tipo de número flutuante.

```
num = 10.5
```

Complex

Tipo de número complexo.

```
num = 3 + 4j
```

Conversão de Tipo

Como converter de um tipo numérico para outro.

```
num = 10
num_float = float(num)
```

Número Aleatório

Como criar um número aleatório usando o módulo

```
random
```

.

```
import random
num = random.randint(1, 100)
print(num)
```

Especificar um Tipo de Variável

Como especificar um tipo de dado para uma variável (em Python, isso é feito automaticamente).

Literais de String

Como criar literais de string.

```
string = "Olá, mundo!"
```

Atribuindo uma String a uma Variável

Como atribuir uma string a uma variável.

```
mensagem = "Bem-vindo!"
```

Strings Multilinha

Como criar uma string multilinha usando aspas triplas.

```
texto = """Esta é uma
string multilinha"""
print(texto)
```

Strings são Arrays

Strings em Python são arrays de bytes representando caracteres Unicode.

Fatiamento de String

Como fatiar uma string.

```
texto = "Python"
print(texto[0:3])  # "Pyt"
```

Indexação Negativa em String

Como usar indexação negativa para acessar caracteres da string.

```
texto = "Python"
print(texto[-1])  # "n"
```

Comprimento da String

Como obter o comprimento de uma string.

```
texto = "Python"
print(len(texto))  # 6
```

Verificar na String

Como verificar se uma string contém uma frase especificada.

```
texto = "Python"
print("Py" in texto)  # True
```

Formatar String

Como combinar duas strings.

```
nome = "Ana"
saudacao = f"Olá, {nome}!"
print(saudacao)
```

Caracteres de Escape

Como usar caracteres de escape em strings.

```
texto = "Linha 1\nLinha 2"
print(texto)
```

Valores Booleanos

Os valores

```
True
```

ou

```
False
```

.

Avaliar Booleanos

Avalie um valor ou expressão para retornar

```
True
```

ou

```
False
```

.

```
valor = 5 > 3
print(valor)  # True
```

Retornar Valor Booleano

Funções podem retornar valores booleanos.

```
def é_maior(x, y):
    return x > y

print(é_maior(10, 5))  # True
```

Operadores Booleanos

Operadores para combinar valores booleanos:

```
and
```

,

```
or
```

, e

```
not
```

.

Operadores Lógicos

Como usar

```
and
```

,

```
or
```

, e

```
not
```

para combinar expressões booleanas.

```
a = True
b = False
print(a and b)  # False
```

Funções Built-in

Funções integradas como

```
print()
```

,

```
len()
```

, etc.

Função

`len()`

Retorna o comprimento de um objeto.

```
texto = "Python"
print(len(texto))  # 6
```

Função

`type()`

Retorna o tipo de um objeto.

```
numero = 10
print(type(numero))  #
```

Função

`isinstance()`

Verifica se um objeto é uma instância de uma classe ou tipo específico.

```
numero = 10
print(isinstance(numero, int))  # True
```

Função

`isinstance()`

com Vários Tipos

Verifica se um objeto é uma instância de um dos vários tipos.

```
valor = "texto"
print(isinstance(valor, (int, str)))  # True
```

Funções Built-in Personalizadas

Como criar suas próprias funções e utilizá-las.

```
def saudacao(nome):
    return f"Olá, {nome}!"

print(saudacao("Lucas"))  # Olá, Lucas!
```

Definir Funções

Como definir uma função usando

```
def
```

.

```
def adicionar(a, b):
    return a + b

print(adicionar(5, 3))  # 8
```

Função Sem Parâmetros

Função que não recebe parâmetros.

```
def mensagem():
    print("Olá, mundo!")

mensagem()
```

Função com Parâmetros

Função que recebe parâmetros.

```
def saudacao(nome):
    return f"Olá, {nome}!"

print(saudacao("Ana"))
```

Funções e Variáveis Locais

Como usar variáveis locais dentro de uma função.

```
def exemplo():
    x = 10  # variável local
    print(x)

exemplo()
# print(x)  # Isso gerará um erro
```

Funções e Variáveis Globais

Como usar variáveis globais dentro de uma função.

```
x = 20  # variável global

def exemplo():
    global x
    x = 10  # altera a variável global

exemplo()
print(x)  # 10
```

Parâmetros e Argumentos

Como passar parâmetros e argumentos para funções.

```
def saudacao(nome, idade):
    return f"Olá, {nome}. Você tem {idade} anos."

print(saudacao("Carlos", 30))
```

Funções Anônimas (Lambdas)

Funções que não têm um nome, criadas usando

`lambda`

.

```
soma = lambda a, b: a + b
print(soma(5, 3))  # 8
```

Funções de Ordem Superior

Funções que recebem outras funções como parâmetros.

```
def aplicar_funcao(func, valor):
    return func(valor)

print(aplicar_funcao(lambda x: x * 2, 5))  # 10
```

Modularização e Importação de Módulos

Como dividir seu código em vários módulos e importá-los.

```
# módulo exemplo.py
def saudacao(nome):
    return f"Olá, {nome}!"

# main.py
import exemplo
print(exemplo.saudacao("Ana"))
```

Importar Módulos

Como importar módulos em seu código.

```
import math
print(math.sqrt(16))  # 4.0
```

Importar Funções Específicas de um Módulo

Como importar funções específicas de um módulo.

```
from math import sqrt
print(sqrt(25))  # 5.0
```

Importar Tudo de um Módulo

Como importar tudo de um módulo.

```
from math import *
print(sqrt(36))  # 6.0
```

Tratamento de Exceções

Como tratar exceções usando

```
try
```

e

```
except
```

.

```
try:
    print(10 / 0)
except ZeroDivisionError:
    print("Não é possível dividir por zero!")
```

Leitura de Arquivo

Como ler dados de um arquivo.

```
with open('arquivo.txt', 'r') as arquivo:
    conteudo = arquivo.read()
    print(conteudo)
```

Escrita em Arquivo

Como escrever dados em um arquivo.

```
with open('arquivo.txt', 'w') as arquivo:
    arquivo.write("Escrevendo no arquivo.")
```

Manipulação de Arquivo

Como manipular arquivos, incluindo leitura e escrita.

Leitura de Dados do Usuário

Como obter dados do usuário usando

```
input()
```

.

```
nome = input("Qual é o seu nome? ")
print(f"Olá, {nome}!")
```

Criação de Listas

Como criar e manipular listas.

```
lista = [1, 2, 3, 4, 5]
print(lista)
```

Adicionar Itens a Listas

Como adicionar itens a uma lista.

```
lista = [1, 2, 3]
lista.append(4)
print(lista)
```

Remover Itens de Listas

Como remover itens de uma lista.

```
lista = [1, 2, 3, 4]
lista.remove(2)
print(lista)
```

Acessar Itens de Listas

Como acessar itens individuais de uma lista.

```
lista = [1, 2, 3, 4]
print(lista[2])  # 3
```

Ordenar Listas

Como ordenar uma lista.

```
lista = [3, 1, 4, 2]
lista.sort()
print(lista)
```

Listas Aninhadas

Listas dentro de listas.

```
matriz = [[1, 2], [3, 4]]
print(matriz[0][1])  # 2
```

Iterar Sobre Listas

Como percorrer uma lista usando um loop.

```
lista = [1, 2, 3, 4]
for item in lista:
    print(item)
```

List Comprehensions

Como criar listas de maneira compacta.

```
quadrados = [x ** 2 for x in range(10)]
print(quadrados)
```

Criação de Dicionários

Como criar e manipular dicionários.

```
dicionario = {'chave1': 'valor1', 'chave2': 'valor2'}
print(dicionario)
```

Acessar Valores de Dicionários

Como acessar valores a partir das chaves de um dicionário.

```
dicionario = {'chave1': 'valor1', 'chave2': 'valor2'}
print(dicionario['chave1'])  # valor1
```

Adicionar Itens a Dicionários

Como adicionar novos itens a um dicionário.

```
dicionario = {'chave1': 'valor1'}
dicionario['chave2'] = 'valor2'
print(dicionario)
```

Remover Itens de Dicionários

Como remover itens de um dicionário.

```
dicionario = {'chave1': 'valor1', 'chave2': 'valor2'}
del dicionario['chave1']
print(dicionario)
```

Iterar Sobre Dicionários

Como percorrer um dicionário usando um loop.

```
dicionario = {'chave1': 'valor1', 'chave2': 'valor2'}
for chave, valor in dicionario.items():
    print(chave, valor)
```

Manipulação de Conjuntos

Como trabalhar com conjuntos.

```
conjunto = {1, 2, 3, 4}
conjunto.add(5)
print(conjunto)
```

Operações com Conjuntos

Operações como união, interseção e diferença entre conjuntos.

```
conjunto1 = {1, 2, 3}
conjunto2 = {3, 4, 5}
print(conjunto1 & conjunto2)  # Interseção: {3}
print(conjunto1 | conjunto2)  # União: {1, 2, 3, 4, 5}
print(conjunto1 - conjunto2)  # Diferença: {1, 2}
```

Manipulação de Strings

Como manipular e formatar strings.

```
texto = "Olá, mundo!"
print(texto.upper())  # OLÁ, MUNDO!
print(texto.lower())  # olá, mundo!
```

Formatar Strings

Como formatar strings usando

```
f-strings
```

.

```
nome = "João"
idade = 30
print(f"Meu nome é {nome} e tenho {idade} anos.")
```

Dividir Strings

Como dividir uma string em partes.

```
texto = "Python é ótimo"
partes = texto.split()
print(partes)  # ['Python', 'é', 'ótimo']
```

Substituir Substrings

Como substituir partes de uma string.

```
texto = "Olá, mundo!"
novo_texto = texto.replace("mundo", "Python")
print(novo_texto)  # Olá, Python!
```

Remover Espaços em Branco

Como remover espaços em branco de uma string.

```
texto = "   Python   "
print(texto.strip())  # "Python"
```

Dados em Lista

Como trabalhar com listas em Python.

```
lista = [1, 2, 3]
lista.append(4)
print(lista)  # [1, 2, 3, 4]
```

Criação de Funções

Como criar suas próprias funções em Python.

```
def saudacao(nome):
    return f"Olá, {nome}!"

print(saudacao("Ana"))  # Olá, Ana!
```

Trabalhando com Arquivos

Como ler e escrever em arquivos.

```
with open('arquivo.txt', 'w') as arquivo:
    arquivo.write("Exemplo de escrita em arquivo.")
```

Tratamento de Erros

Como usar

```
try
```

e

```
except
```

para tratamento de erros.

```
try:
    print(10 / 0)
except ZeroDivisionError:
    print("Erro: Divisão por zero!")
```