TIPOS DE DADOS

Tipos de dados integrados

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

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

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

Tipo de texto: `str`

Tipos numéricos: `int`, `float`, `complex`

Tipos de sequência: `list`, `tuple`, `range`

Tipo de mapeamento: `dict`

Tipos de conjuntos: `set`, `frozenset`

Tipo booleano: `bool`

Tipos binários: `bytes`, `bytearray`, `memoryview`

Obtendo o tipo de dados

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

Exemplo:

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

Definindo o tipo de dados

Em Python, o tipo de dados é definido quando você atribui um valor a uma variável.

Exemplo:

```
x = "Olá, Mundo"
print(x) #exibe x:
print(type(x)) #exibe o tipo de dado de x:
```

Você pode utilizar o código acima para verificar cada tipo de dados na tabela abaixo:

Exemplo	Tipo de dados
`x = "Olá, Mundo!"`	`str`
`x = 20`	`int`
`x = 20.5`	`float`
`x = 1j`	`complex`
`x = ["maçã", "banana", "cereja"]`	`list`
`x = ("maçã", "banana", "cereja")`	`tuple`
`x = range(6)`	`range`
`x = {"nome" : "João", "idade" : 21}`	`dict`
`x = {"maçã", "banana", "cereja"}`	`set`
`x = frozenset({"maçã", "banana", "cereja"})`	`frozenset`
`x = True`	`bool`
`x = b"Olá"`	`bytes`
`x = bytearray(5)`	`bytearray`
`x = memoryview(bytes(5))`	`memoryview`

Definir o Tipo de Dado Específico

Se você deseja especificar o tipo de dados, pode usar as seguintes funções de construtor.

Exemplo:

```
x = str("Hello World") #note a sintaxe str antes do valor de x
print(x)
print(type(x))
```

Especifique um tipo de variável - CASTING

Pode haver momentos em que você deseja especificar um tipo em uma variável. Isso pode ser feito com casting. Python é uma

linguagem orientada a objetos e, como tal, usa classes para definir tipos de dados, incluindo seus tipos primitivos.

A conversão em python é, portanto, feita usando funções construtoras:

- `int ()` - constrói um número inteiro a partir de um literal inteiro, um literal flutuante (removendo todos os decimais) ou um literal de string (desde que a string represente um número inteiro)
- `float ()` - constrói um número flutuante a partir de um literal inteiro, um literal flutuante ou um literal de string (desde que a string represente um flutuante ou um inteiro)
- `str ()` - constrói uma string a partir de uma ampla variedade de tipos de dados, incluindo strings, literais inteiros e literais flutuantes

Exemplo:

```
#inteiros
x = int(1)    # x será 1
y = int(2.8)  # y será 2
z = int("3")  # z será 3
#ponto flutuante
x = float(1)     # x será 1.0
y = float(2.8)   # y será 2.8
z = float("3")   # z será 3.0
w = float("4.2") # w será 4.2
#strings
x = str("s1") # x será 's1'
y = str(2)    # y será '2'
z = str(3.0)  # z será '3.0'
```

Entrada de Dados

Python permite a entrada de dados.

O método é um pouco diferente no Python 3.6 e no Python 2.7.

Python 3.6 usa o método `input()`. Python 2.7 usa o método `raw_input()`.

O exemplo a seguir pede o nome de usuário e, quando você digita o nome de usuário, ele é impresso na tela:

Python 3.6

```
username = input("Enter username:")
 print("Username is: " + username)
```

Python 2.7

```
username = raw_input("Enter username:")
print("Username is: " + username)
```

Assim é possível solicitar a entrada de dados ao usuário para trabalhar com eles pelo código. Veremos exemplos que deixam bem claro o funcionamento dessa função.

Na versão atual do Python (3.10) segue com a sintaxe da versão 3.6.