



Lógica de Programação



Tipos primitivos de variáveis, sintaxe e
atribuição de valores (Python)



Tipos de variáveis

- Um valor, como um número ou texto, é algo comum em um programa. Por exemplo, 'Hello, World!', 1, 2, todos são valores.
- Estes valores são de diferentes tipos: 1 e 2 são números inteiros e 'Hello World!' é um texto, também chamado de String.
- Podemos identificar strings porque são delimitadas por aspas (simples ou duplas) - e é exatamente dessa maneira que o interpretador Python também identifica uma string.

Tipos de variáveis

- A função `print()` utilizada no capítulo anterior também trabalha com inteiros:
- `>>> print(2)`
- `2`
- Veja que aqui não é necessário utilizar aspas por se tratar de um **número**.

Tipos de variáveis

- Caso você não tenha certeza qual é o tipo de um valor, pode usar a função `type()` para checar:
- `>>> type('Hello World')`
- `<class 'str'>`
- `>>> type(2)`
- `<class 'int'>`
- Strings são do tipo `str` (abreviação para string) e inteiros do tipo `int` (abreviação para integer).

Tipos de variáveis

- Outro tipo que existe no Python são os números decimais que são do tipo float (ponto flutuante):
- `>>> type(3.2)`
- `<class 'float'>`

Tipos de variáveis

- E qual será o tipo de valores como '2' e '3.2'? Eles se parecem com números mas são delimitados por aspas como strings. Utilize a função `type()` para fazer a verificação:
- ```
>>> type('2')
```
- ```
<class 'str'>
```
- ```
>>> type('3.2')
```
- ```
<class 'str'>
```
- Como estão delimitados por aspas, o interpretador vai entender esses valores como strings, ou seja, como texto.

Tipos de variáveis

- Boolean (bool)
- Tipo de dado lógico que pode assumir apenas dois valores: falso ou verdadeiro (False ou True em Python).
- Na lógica computacional, podem ser considerados como 0 ou 1.
- Exemplos:
- `fim_de_semana = True`
- `feriado = False`
- `print(type(fim_de_semana))`
- `print(type(feriado))`
- Saída:
- `<class 'bool'>`
- `<class 'bool'>`

Tipos de variáveis

- Podemos pedir para o Python lembrar de um valor que queiramos utilizar em outro momento do programa.
- O Python vai guardar este valor em uma variável. Variável é um nome que faz referência a um valor.
- É como uma etiqueta que colocamos naquele valor e quando precisarmos usar, chamamos pelo nome que foi dado na etiqueta.
- Um comando de atribuição (o sinal de igualdade =) cria uma nova variável e atribui um valor a ela:

Tipos de variáveis

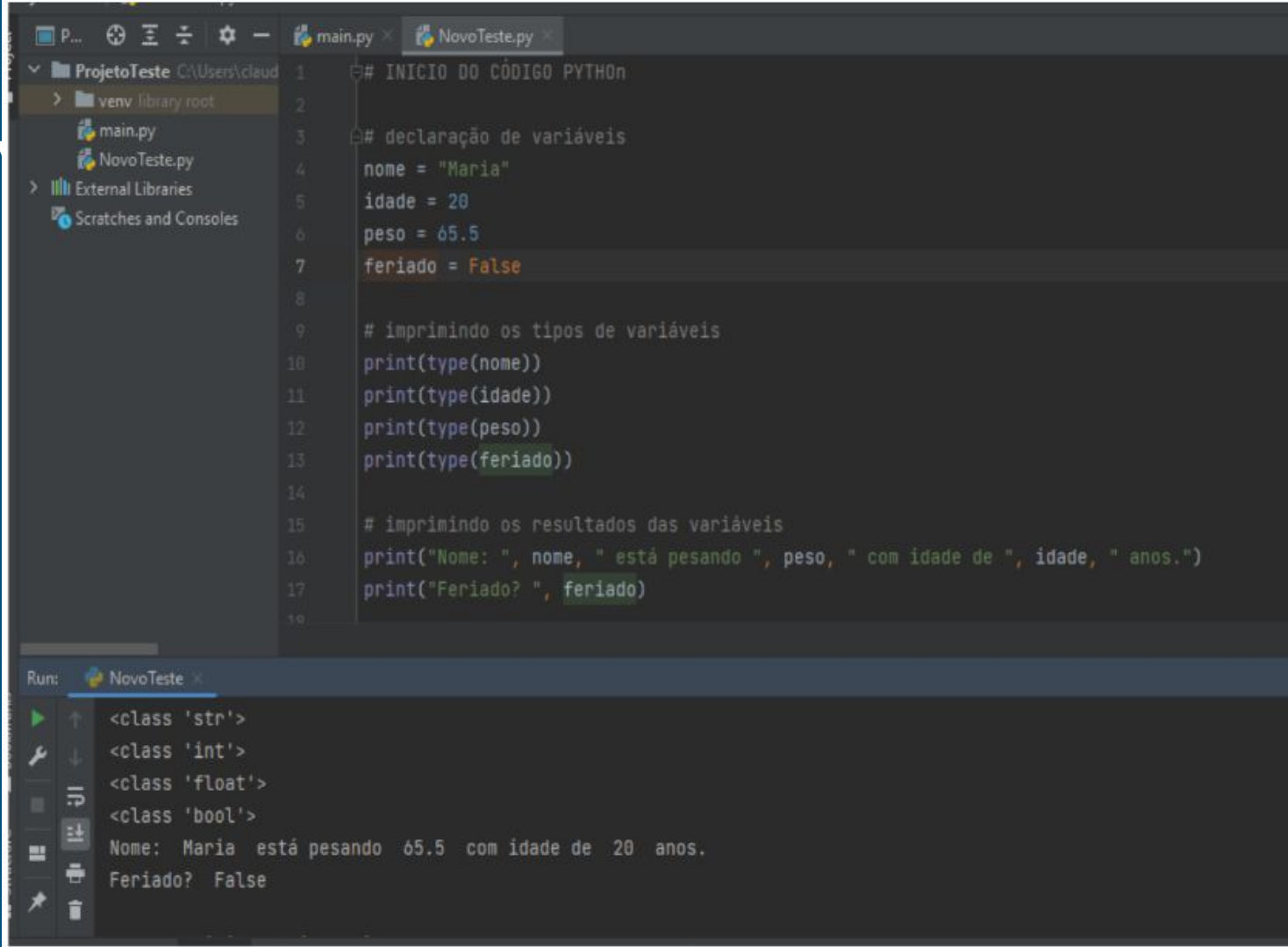
- Python é uma linguagem dinamicamente tipada, o que significa que não é necessário declarar o tipo de variável ou fazer casting (mudar o tipo de variável), pois o Interpretador se encarrega disso para nós!
- Isso significa também que o tipo da variável pode variar durante a execução do programa.

Tipos de variáveis

- `>>> mensagem = 'oi, python'`
- `'oi, python'`
- `>>> numero = 5`
- `5`
- `>>> pi = 3.14`
- `3.14`
- Três atribuições foram feitas neste código. Atribuimos a variável `mensagem` uma string; a variável `numero` um inteiro e a variável `pi` um valor aproximado do número pi. No modo interativo, o interpretador mostra o resultado após cada atribuição.

Tipos de

- Mostrar
- exemplo
- de código
- Python 01



The screenshot shows a Python IDE with two tabs: 'main.py' and 'NovoTeste.py'. The 'NovoTeste.py' tab is active, displaying the following Python code:

```
1  # INICIO DO CÓDIGO PYTHON
2
3  # declaração de variáveis
4  nome = "Maria"
5  idade = 20
6  peso = 65.5
7  feriado = False
8
9  # imprimindo os tipos de variáveis
10 print(type(nome))
11 print(type(idade))
12 print(type(peso))
13 print(type(feriado))
14
15 # imprimindo os resultados das variáveis
16 print("Nome: ", nome, " está pesando ", peso, " com idade de ", idade, " anos.")
17 print("Feriado? ", feriado)
```

Below the code editor, the 'Run' console shows the output of the program:

```
Run: NovoTeste
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
Nome: Maria está pesando 65.5 com idade de 20 anos.
Feriado? False
```

Nomes de variáveis

- Programadores escolhem nomes para variáveis que sejam semânticos e que ao mesmo tempo documentem o código.
- Esses nomes podem ser bem longos, podem conter letras e números. É uma convenção entre os programadores Python começar a variável com letras minúsculas e utilizar o underscore (_) para separar palavras como: `meu_nome`, `numero_de_cadastro`, `telefone_residencial`.
- Esse padrão é chamado de snake case.

Nomes de variáveis

- Se nomearmos nossas variáveis com um nome ilegal, o interpretador vai acusar um erro de sintaxe:
- `>>> 1nome = 'python'`
- `File "<stdin>", line 1`
- `1nome = 'python'`
- `^`
- `SyntaxError: invalid syntax`
- `>>> numero@ = 10`
- `File "<stdin>", line 1`
- `numero@ = 10`
- `^`
- `SyntaxError: invalid syntax`

Nomes de variáveis

- `>>> class = 'oi'`
- File "<stdin>", line 1
- `class = oi`
- `^`
- `SyntaxError: invalid syntax`

Nomes de variáveis

- 1nome é ilegal porque começa com um número, numero@ é ilegal porque contém um caractere especial (o @) considerado ilegal para variáveis.
- E class é ilegal porque class é uma palavra chave em Python.
- O interpretador utiliza palavras chaves como palavras reservadas da linguagem, como um vocabulário próprio.

Nomes de variáveis

- Python possui 33 palavras reservadas:

and	del	from	None	True
as	elif	global	nonlocal	try
assert	else	if	not	while
break	except	import	or	with
class	False	in	pass	yield
continue	finally	is	raise	def
for	lambda	return		

- Portanto, não podemos utilizar essas palavras para nomear nossas variáveis.

Nomes de variáveis

- Exemplos de código 02, 03 e 04

Entrada do usuário

- Agora vamos criar mais interatividade e pedir para o usuário entrar com um valor digitado do teclado.
- O Python possui uma função que captura a entrada de valores: a função `input()` .
- Quando essa função é chamada, o programa para e espera o usuário digitar alguma coisa.
- Quando o usuário aperta a tecla ENTER , o programa processa e imprime o valor digitado em forma de string:
- ```
>>> entrada = input()
```
- ```
>>> print(entrada)
```
- ```
'oi python'
```

# Entrada do usuário

---

- Mas o ideal é pedir algo específico ao usuário e dizer qual dado queremos receber. Podemos passar
- uma string para a função `input()` :
- ```
>>> nome = input("digite seu nome:\n")
```
- digite seu nome:
- Maria
- ```
>>> print(nome)
```
- caelum
- O `\n` no final representa uma nova linha e o interpretador vai quebrar uma linha após imprimir a string. Por este motivo, o valor digitado pelo usuário aparece na próxima linha.

# Convertendo uma string para inteiro

---

- A função `input()` lê o valor digitado pelo usuário como uma string.
- `chute = input('Digite um número: ')`
- Se o usuário digitar o número 42, a variável `chute` vai guardar o valor "42" , ou seja, um texto.
- Podemos checar isso através da função `type()` que retorna o tipo da variável. Vamos testar isso:
- ```
>>> chute = input('Digite um número: ')
```
- Digite um número: 42
- ```
>>> type(chute)
```
- ```
<class 'str'>
```

Convertendo uma string para inteiro

- Precisamos converter a string "42" para um número inteiro.
- `>>> numero_em_texto = '12'`
- `'12'`
- `>>> type(numero_em_texto)`
- `<class 'str'>`
- `>>> numero = int(numero_em_texto)`
- `12`
- `>>> type(numero)`
- `<class 'int'>`

Convertendo uma string para inteiro

- Mas devemos tomar cuidado, nem toda string pode ser convertida para um número inteiro:
- `>>> texto = Maria`
- `>>> numero = int(texto)`
- Traceback (most recent call last):
- File "<stdin>", line 1, in <module>
- ValueError: invalid literal for int() with base 10: Maria