

Fundamentos de Programação

1º Semestre
2023/2024

Aula 9



Plano

- **Preparação do ambiente de desenvolvimento de aplicações**
- **Sintaxe de Python**
 - Variáveis
 - Tipos de dados
 - Operações lógicas, aritméticas e relacionais
 - Estruturas de decisão

Python

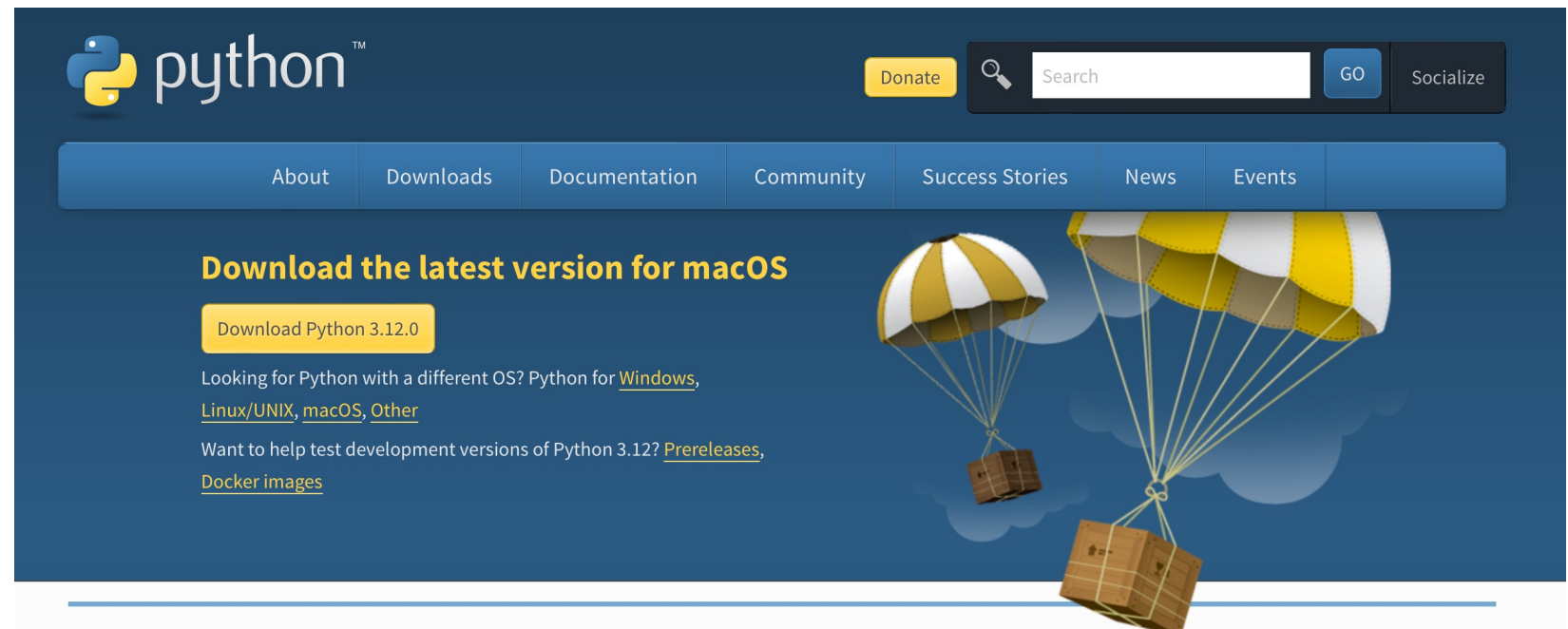
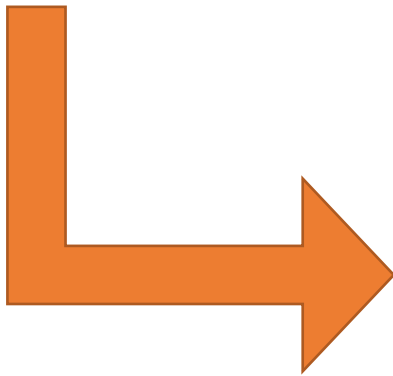


- Linguagem de programação inventada pelo cientista holandês Guido van Rossum no final da década de 1980
- Nome inspirado no grupo inglês *Monty Python*
- É a linguagem mais popular no mundo (cf. <https://www.tiobe.com/tiobe-index/>)

Oct 2023	Oct 2022	Change	Programming Language		Ratings	Change
1	1			Python	14.82%	-2.25%
2	2			C	12.08%	-3.13%
3	4	▲		C++	10.67%	+0.74%
4	3	▼		Java	8.92%	-3.92%
5	5			C#	7.71%	+3.29%

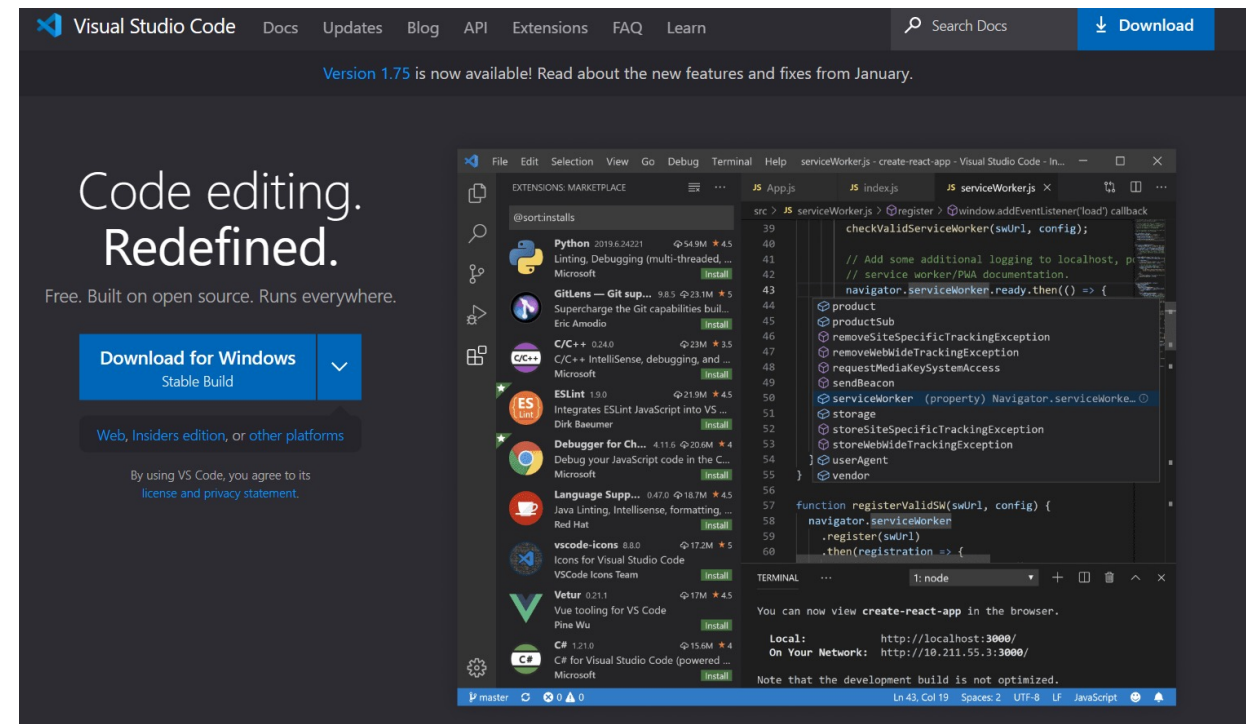
Preparação do ambiente de desenvolvimento de aplicações (1)

- Para executar código-fonte escrito em linguagem Python, é necessário o interpretador:
 - [Download Python | Python.org](https://python.org)



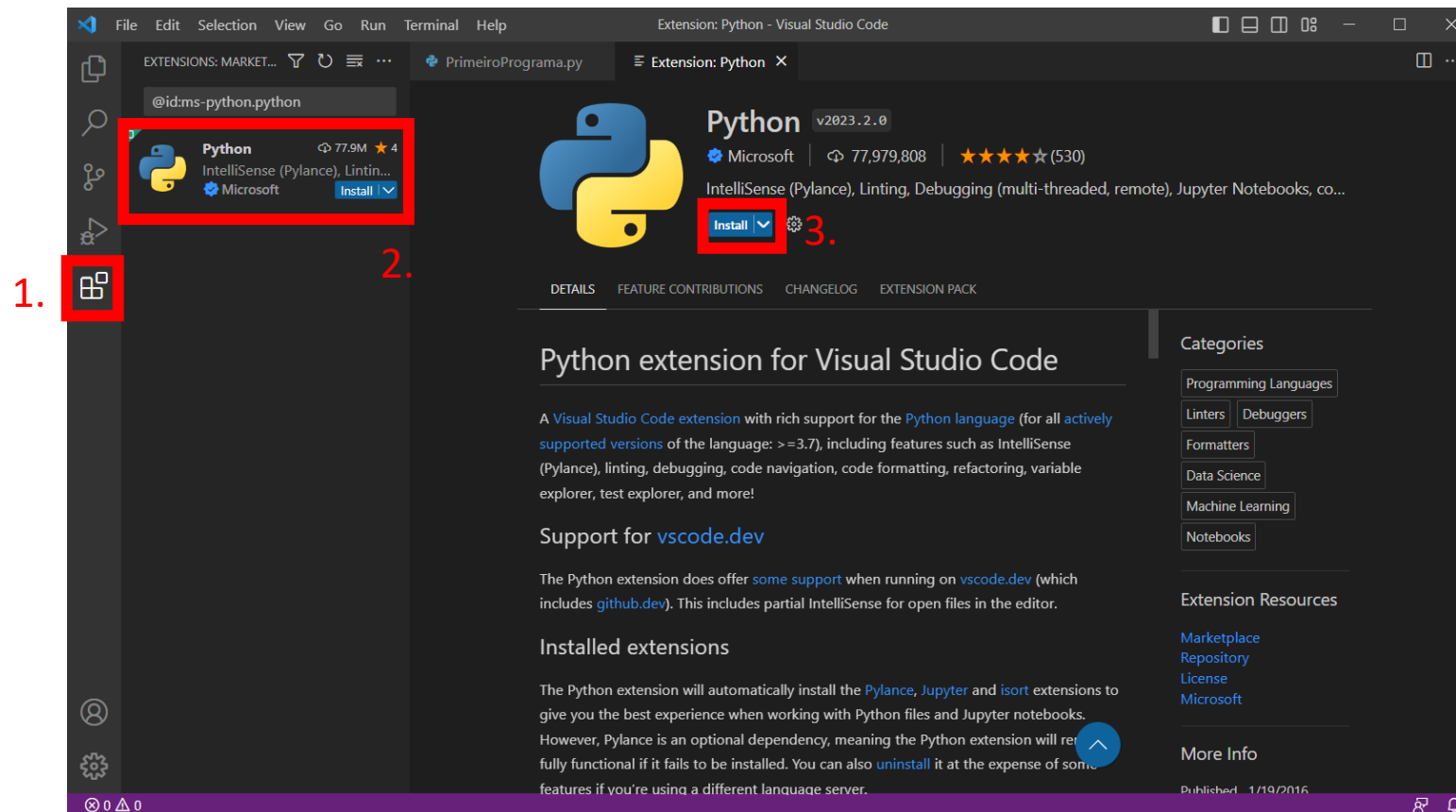
Preparação do ambiente de desenvolvimento de aplicações (2)

- É um ambiente integrado de desenvolvimento para Python
 - Na prática, é o ambiente onde vamos programar
- É gratuito e o download pode ser feito aqui:
 - [Visual Studio Code - Code Editing. Redefined](#)

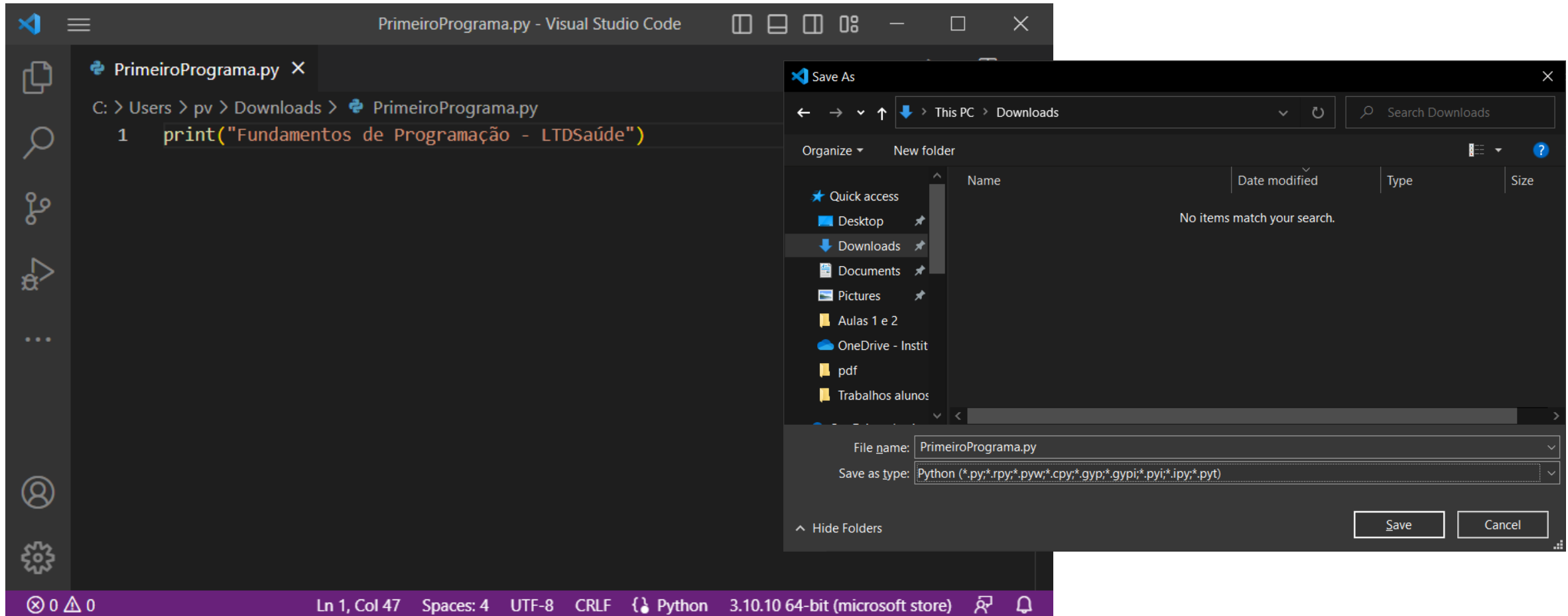


Preparação do ambiente de desenvolvimento de aplicações (3)

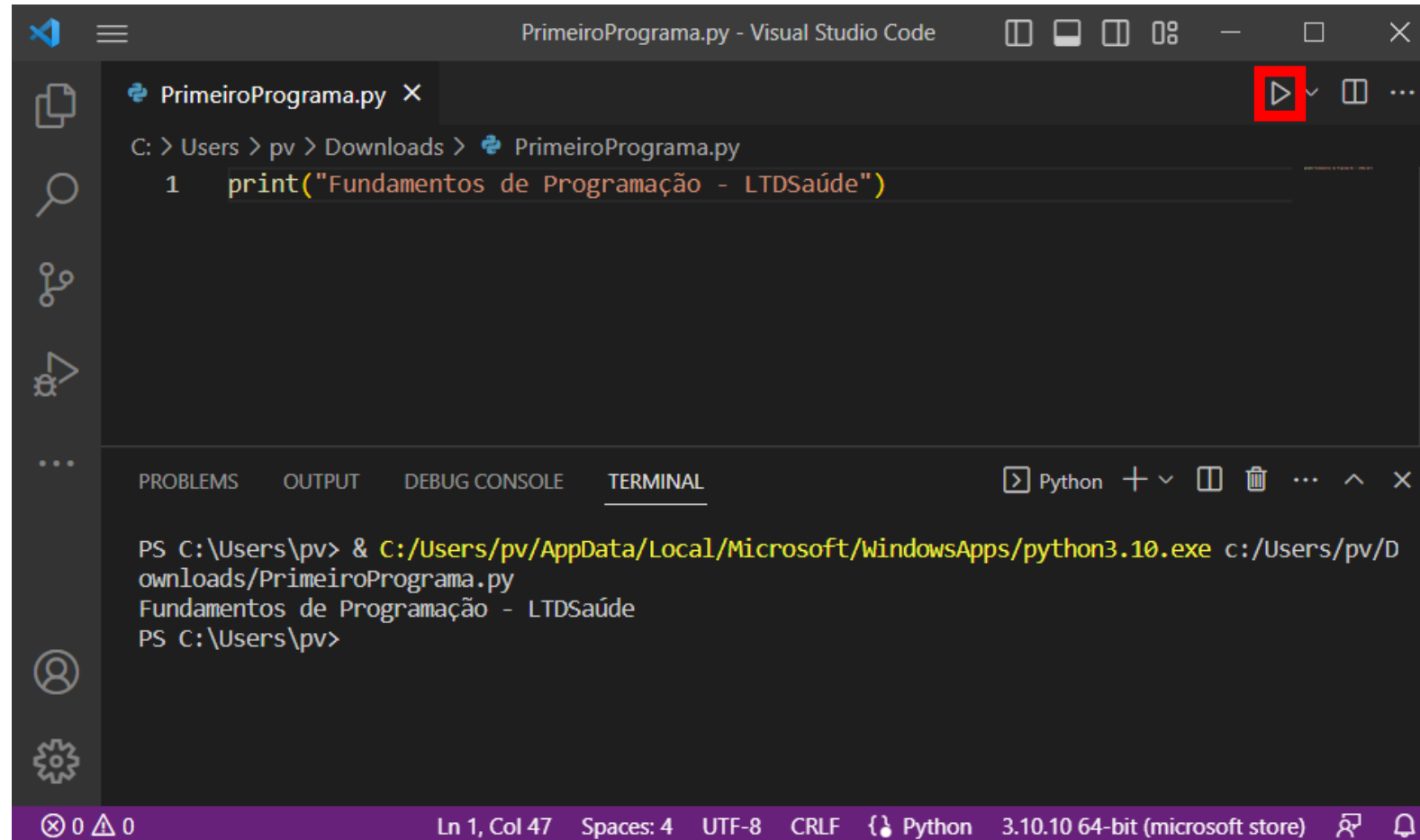
- É necessária a extensão *Python* da *Microsoft*



Criar um programa



Executar um programa



PrimeiroPrograma.py - Visual Studio Code

PrimeiroPrograma.py X

C: > Users > pv > Downloads > PrimeiroPrograma.py

```
1 print("Fundamentos de Programação - LTDSaúde")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Python + -

```
PS C:\Users\pv> & C:/Users/pv/AppData/Local/Microsoft/WindowsApps/python3.10.exe c:/Users/pv/Downloads/PrimeiroPrograma.py
Fundamentos de Programação - LTDSaúde
PS C:\Users\pv>
```

Ln 1, Col 47 Spaces: 4 UTF-8 CRLF Python 3.10.10 64-bit (microsoft store)

Comando *Help*

- Muito útil para perceber o funcionamento das funções *built-in* em caso de dúvida
- Forma expedita de consultar a documentação

```
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep:  string inserted between values, default a space.
    end:  string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

Tarefa 9

Trabalhar sobre o seguinte documento:

- **Python Tarefa 9 – Adaptação ao IDE.pdf**



A reter

- a) Para um programa executar no Visual Studio Code, é necessário:
 - 1. Criar um documento
 - 2. Colocar as instruções no documento
 - 3. Executar as instruções carregando no botão “Run”

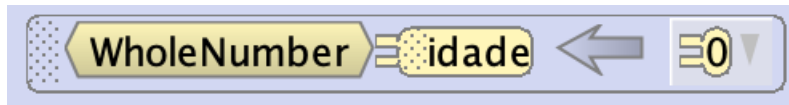
- b) Para mostrar dados na consola utiliza-se o comando *print()*:
 - Repara que *print* é um procedimento que recebe como argumento o texto a imprimir no ecrã

Criar variáveis

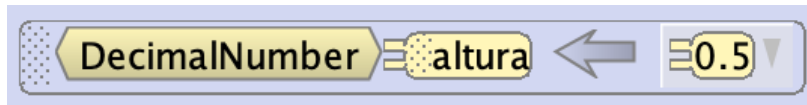
- Em *python* o programador não especifica explicitamente o tipo da variável
- Indica o seu nome e o valor que irá receber
 - Equivalente à instrução “*variable*” do Alice
 - Não esquecer da importância do nome das variáveis para a legibilidade do código

Criar variáveis com diferentes tipos de dados

- **Números inteiros**



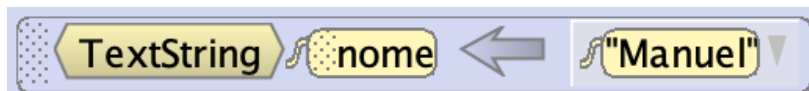
- **Números decimais**



- **Booleanos**



- **String**



```
idade = 0
altura = 0.5
ativo = True
nome = 'Manuel'
```

Tipos de dados

```
In [14]: print("Olá Mundo!")
```

Olá Mundo!

```
In [15]: i = 0  
         type(i)
```

Out[15]: int

```
In [16]: i = 0.5  
         type(i)
```

Out[16]: float

```
In [17]: i = False  
         type(i)
```

Out[17]: bool

Tarefa 10

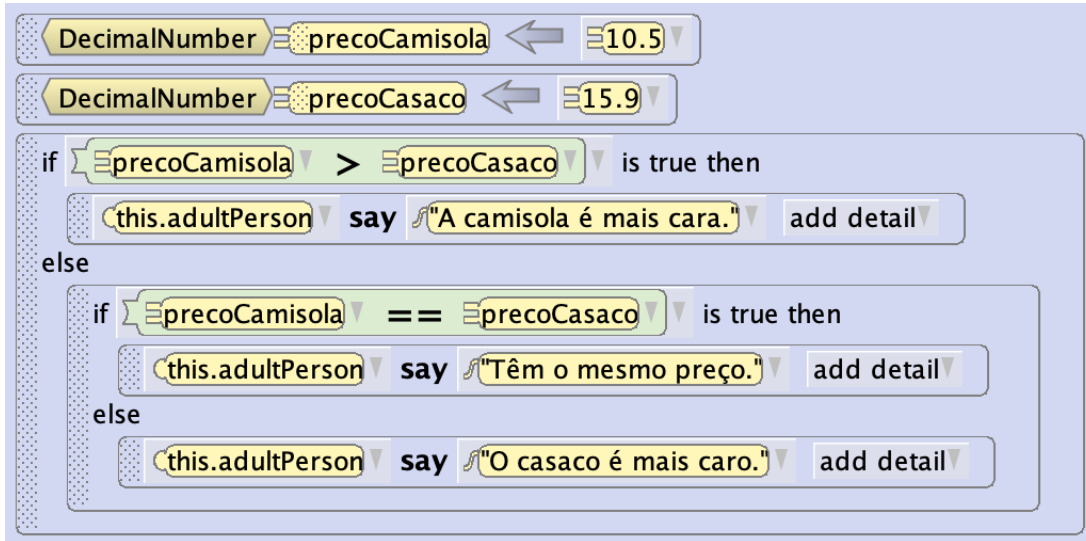
Trabalhar sobre o seguinte documento:

- **Python Tarefa 10 – Variáveis Decisões e Operações.pdf**



Estruturas de decisão com operações (1)

- Agora basta escrever o que se pretende comparar:



```
preco_camisola = 10.5
preco_casaco = 15.9
if preco_camisola > preco_casaco:
    print("A camisola é a mais cara")
elif preco_camisola == preco_casaco:
    print("Têm o mesmo preço")
else:
    print("O casaco é o mais caro")
```


Como se leem dados do teclado? (1)

1. É necessário usar a função input
 - Recebe como parâmetro a informação a mostrar ao utilizador
 - Devolve o texto introduzido em *String*

- 2. É necessário converter para o tipo pretendido e validar!
 - Usando as funções `int()`, e `float()`
 - E.g., `float(input("Introduza um número: "))`

Módulos *Python* (1)

- Ficheiro Python que contém definições e declarações
- Definem funções, classes e variáveis
- Possibilitam a estruturação e organização do código
 - para que possa evoluir em termos de complexidade
- E.g., Gerar números pseudo-aleatórios

Módulos *Python* (2)

- E.g., Números aleatórios

Importar o módulo

```
In [47]: import random
```

```
In [48]: random.randint(0, 500)
```

```
Out[48]: 63
```

Utilização da função definida no módulo
como se se tratasse de realizada por nós

`random` — Generate pseudo-random numbers

Source code: [Lib/random.py](https://docs.python.org/3/library/random.py)

This module implements pseudo-random number generators for various distributions.

For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, a function to generate a random permutation of a list in-place, and a function for random sampling without replacement.

On the real line, there are functions to compute uniform, normal (Gaussian), lognormal, negative exponential, gamma, and beta distributions. For generating distributions of angles, the von Mises distribution is available.

Almost all module functions depend on the basic function `random()`, which generates a random float uniformly in the semi-open range `[0.0, 1.0)`. Python uses the Mersenne Twister as the core generator. It produces 53-bit precision floats and has a period of $2^{19937}-1$. The underlying implementation in C is both fast and threadsafe. The Mersenne Twister is one of the most extensively tested random number generators in existence. However, being completely deterministic, it is not suitable for all purposes, and is completely unsuitable for cryptographic purposes.

The functions supplied by this module are actually bound methods of a hidden instance of the `random.Random` class. You can instantiate your own instances of `Random` to get generators that don't share state.

Class `Random` can also be subclassed if you want to use a different basic generator of your own devising: in that case, override the `random()`, `seed()`, `getstate()`, and `setstate()` methods. Optionally, a new generator can supply a `getrandbits()` method — this allows `randrange()` to produce selections over an arbitrarily large range.

The `random` module also provides the `SystemRandom` class which uses the system function `os.urandom()` to generate random numbers from sources provided by the operating system.

Warning: The pseudo-random generators of this module should not be used for security purposes. For security or cryptographic uses, see the `secrets` module.

See also: M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator", ACM Transactions on Modeling and Computer Simulation Vol. 8, No. 1, January pp.3–30 1998.

[Complementary-Multiply-with-Carry recipe](#) for a compatible alternative random number generator with a long period and comparatively simple update operations.

<https://docs.python.org/3/library/random.html>

Tarefa 11

Trabalhar sobre o seguinte documento:

- **Python Tarefa 11 – Operadores e Decisões.pdf**



Comentários?

