



```
3     |         if response.status_code == 200:
4     |             print(f"Status: {response.status_code} - Try rerunning the code!")
5     |         else:
6     |             print(f"Status: {response.status_code}\n")
7     |
8     |         # using BeautifulSoup to parse the response object
9     |         soup = BeautifulSoup(response.content, "html.parser")
10    |         # finding Post images in the soup
11    |         find all("img", attrs={"alt": "Post Image"})
```

# PYTHON (UDEMY)

<https://www.udemy.com/course/python-3-do-zero-ao-avancado/learn/lecture/15099532#overview>

## ● Seção 01: Introdução

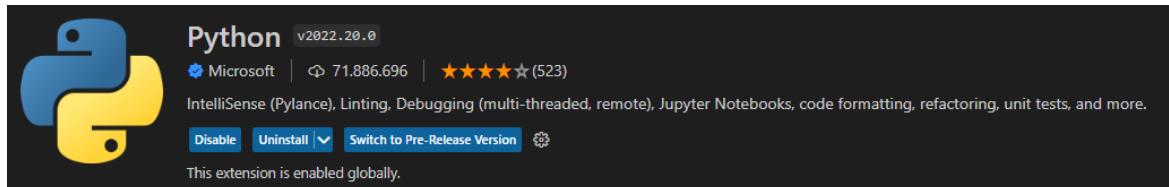


## ● Seção 02: Python e VSCode

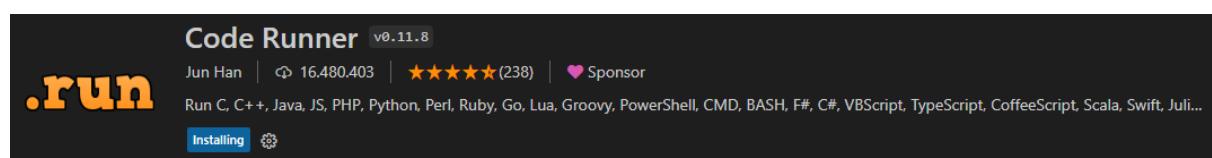


## ■ Extensões

*Python*



## *Code Runner*



```
// settings.json
{
```

```
"code-runner.runInTerminal": true,
"code-runner.clearPreviousOutput": true,
"code-runner.executorMap": {
    "python": "cls ; python -u",
},
"code-runner.ignoreSelection": true
}
```

### Settings.json (configurações do VS Code)

```
// settings.json
{
    "editor.fontSize": 18,
    "explorer.compactFolders": false,
    "workbench.iconTheme": "material-icon-theme",
    "workbench.colorTheme": "Nord",
    "code-runner.runInTerminal": true,
    "code-runner.clearPreviousOutput": true,
    "code-runner.executorMap": {
        "python": "cls ; python -u",
    },
    "code-runner.ignoreSelection": true,
    "python.defaultInterpreterPath": "python",
    "diffEditor.wordWrap": "off",
    "editor.wordWrap": "on"
}
```

## ● Seção 03: Introdução ao Python e Lógica de programação



**Extensão do arquivo:** file.py

**Comentário:** “# comentários”

**DocString:** utilizado para comentar multi linhas, lembrando que não é um comentário. O interpretador lê, mas não faz nada com essas anotações, apenas guarda na memória.

```
"""
docstrings (aspas duplas)
"""
# ou
'''
docstrings (aspas simples)
'''
```

**print():** por padrão, o print ja adiciona um separado entre os argumentos, mas ele pode ser alterado utilizando o sep.

```
print(12, 34) # 12 34 (padrão)
print(12, 34, sep = "-") # 12-34
```

Tambem é possivel alterar o que ocorre no final do comando print, com o end.

```

# \r\n -> CRLF (padrão windows)
# \n -> LF (padrão unix)

# \n (padrão)
print(56, 78, sep = "-", end = '\n')
print(78, 1011, sep = "-", end = '\n')

"""
saída
56-78
78-1011
"""

# outros caracteres
print(56, 78, sep = "-", end = '##')
print(78, 1011, sep = "-")

"""
saída
56-78##78-1011
"""

```

## ■ Tipos de Dados

O python é linguagem de programação **DINÂMICA** (a linguagem já sabe o tipo da informação que está sendo passada.) e **FORTE**.

### Strings → str

```
"rafael" # o python reconhece que isso é uma string
```

**caractere de escape:** \

```

print("Rafael \"Oliveira") # Rafael "Oliveira"
""" \ -> diz para não interpretar o próximo caractere ir
ao próximo. """

```

E se a \ precisa ser mostrada? Utiliza-se o r

```
print(r"Rafael \"Oliveira\"") # Rafael \"Oliveira\"
```

Uma maneira mais simples, é utilizar aspas duplas dentro de aspas simples, ou ao contrário.

```
print('Rafael "Oliveira"') # Rafael "Oliveira"
```

### Int e Float

**int:** números inteiros, positivos e negativos → [10](#)

**float:** números com ponto flutuante (casas decimais), positivos e negativos → [10.1](#)

Para saber o tipo de dado, utiliza-se a função [type](#)

```
print(type(10)) # <class 'int'>
print(type(10.1)) # <class 'float'>
print(type("Rafael")) # <class 'str'>
```

## Boolean → Bool

Duas respostas possíveis, [True\(sim\)](#) ou [False\(não\)](#)

```
print(10 == 10) # Sim -> True
print(10 == 11) # Não -> False
print(type(10 == 10)) # <class 'bool'>
```

## Coerção/Conversão de tipos

Converter um tipo de dado para outro, também conhecido como: type conversion, typecasting, coercion.

*Str, int, float e bool* são tipos **primitivos** e **imutáveis** (não podem ser alterados ao longo do programa)

Existem funções que convertem um tipo em outro

```
print(1 + 1) # 2
print('oi' + ' Rafael') # oi Rafael

print('1' + 1) # erro de tipos, pois não se pode concatenar str com int, somente str com str

# str -> int
int('1') # -> 1
print(int('1') + 1) # 2

# str -> float
float('1') # -> 1.0
print(float('1') + 1) # 2.0

# str -> bool
print(bool('')) # -> False -> str vazia é considerada False
print(bool(' ')) # -> True -> str com conteúdo é considerada True

# int -> str
str(11) # -> '11'
print(str(11) + 'b') # 11b
```

## Variáveis

São usadas para **salvar algo na memória do computador**. A **PEP8** (guia de estilos do python) indica que o nome das variáveis sejam com letras minúsculas, podendo utilizar números e *underline*.

O sinal de = é um **operador de atribuição**, ou seja, atribui um valor à variável.

**Ex:** nome\_variavel = expressao

```
nome_completo = "Rafael Oliveira" # str
idade = 18 # num int
```

```
maior_de_idade = idade >= 18 # expressao  
print('Nome: ', nome_completo, 'Idade: ', idade, 'Maior de Idade: ', maior_de_idade) # Rafael Oliveira 18
```

## ■ Operadores Aritméticos (matemáticos)

```
adicao = 10 + 12 # 22  
subtracao = 22 - 10 # 12  
multiplicacao = 10 * 2 # 20  
divisao = 10 / 3 # 3.33333 (o retorno sempre é float)  
divisao_inteira = 10 // 3 # 3.0  
exponenciacao = 2 ** 10 # 1024  
modulo = 55 % 2 # 1 (resto da divisão)
```

### (+) Concatenação

“Apenas strings” “devem ser concatenadas” “com strings”

```
# concatenação (+)  
concatenacao = 'A' + 'B' + 'C'  
print(concatenacao) # ABC
```

### (\*) Repetição

Um inteiro “é a string que deve ser repetida”

```
# repeticao (*)  
r_dez_vezes = 'R' * 10  
print(r_dez_vezes) # RRRRRRRRRR  
  
rafael_duas_vezes = 'Rafael' * 2  
print(rafael_duas_vezes) # RafaelRafael
```

## Precedência entre os operadores aritméticos

Cada operador é executado de acordo com sua precedência, ou seja, sua hierarquia.

```
# 1 - parenteses (n + n)  
# 2 - exponenciacao **  
# 3 - * / // % (com multiplicação e divisão na mesma conta, a ordem sera da esquerda pra direita)  
# 4 - + -  
  
conta_1 = 1 + 1 ** 5 + 5 # 1 ** 5 = 1 -> +1 = 2 -> +5 = 7  
conta_2 = (1 + 1) ** (5 + 5) # 1024  
  
print(conta_1)  
print(conta_2)
```

→ Ellipsis, usado pra indicar algum código que será escrito

```
altura = 1.85  
peso = 90  
imc = ...
```

