

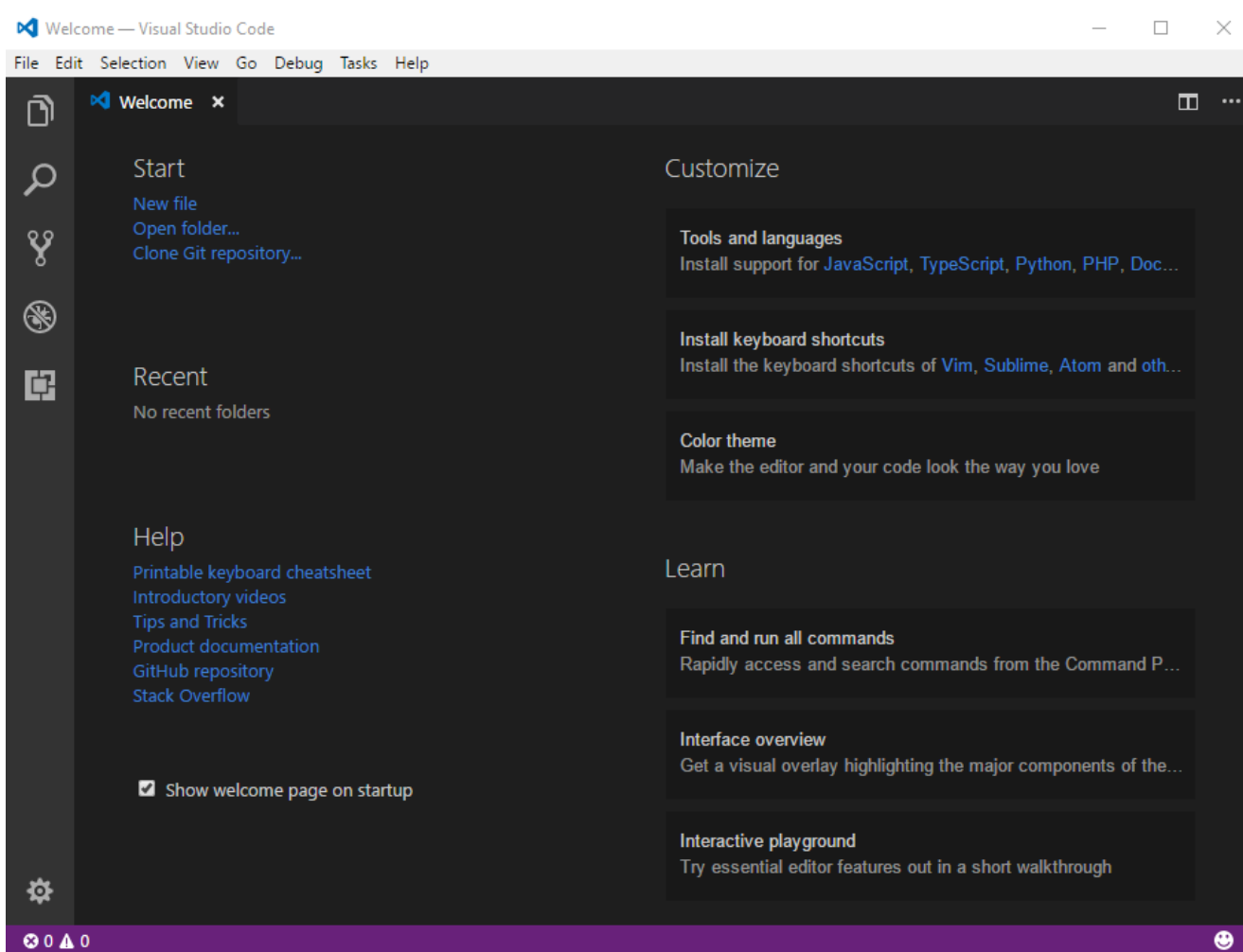


Linguagem de Programação 2

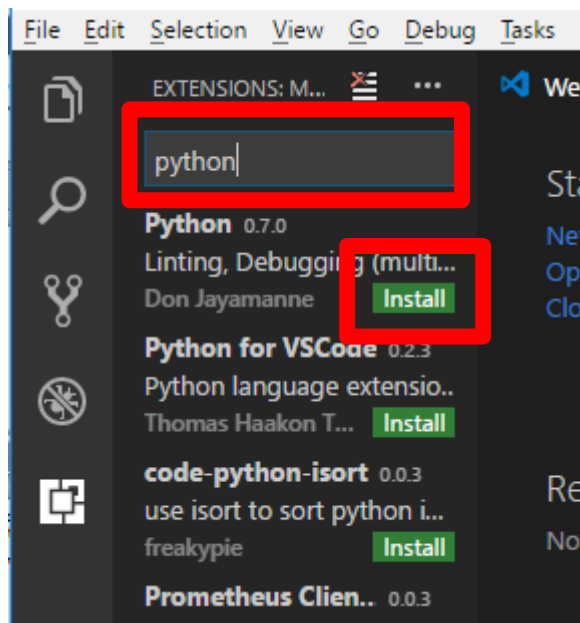
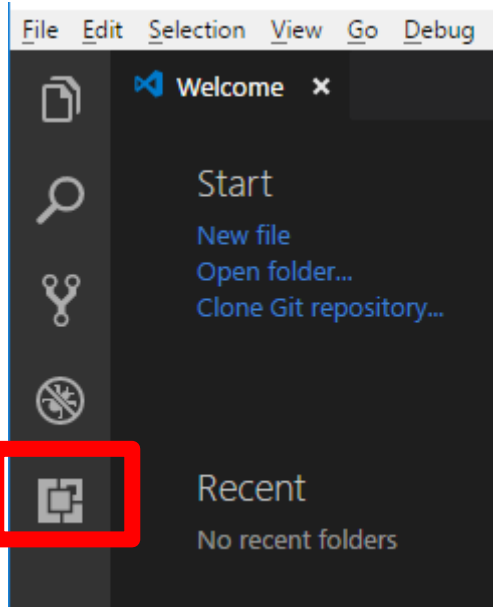
Revisão Python



- Visual Studio Code - <https://code.visualstudio.com/>



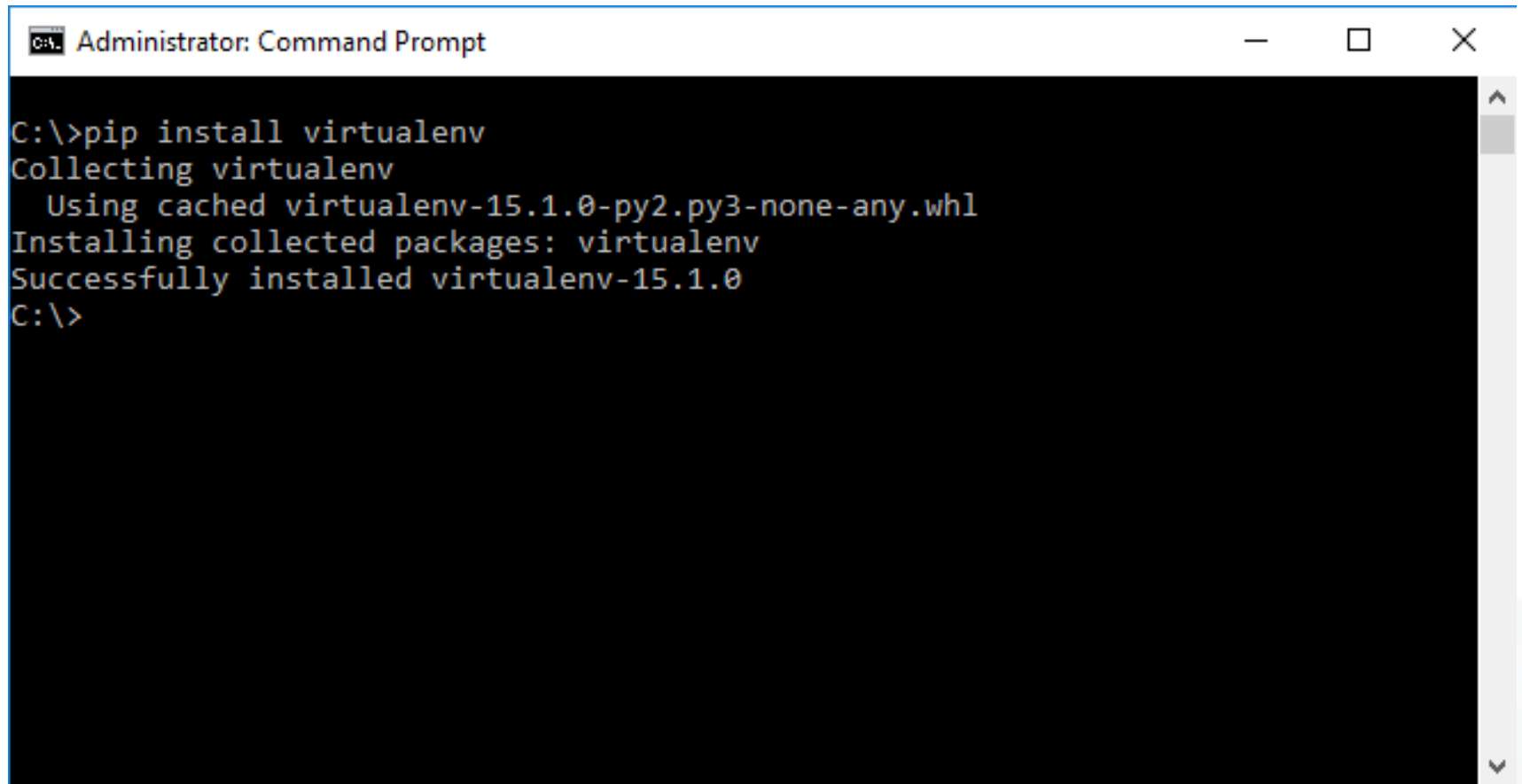
- Extensão para python



- virtualenv – módulo do python para criar ambientes python isolados
 - Ótimo para instalar extensões quando não é administrador da máquina
- Instalado com o pip (sistema de gerenciamento de pacotes do python)
- Abra o prompt de comando do Windows (cmd) e digite o seguinte comando:

```
pip install virtualenv
```

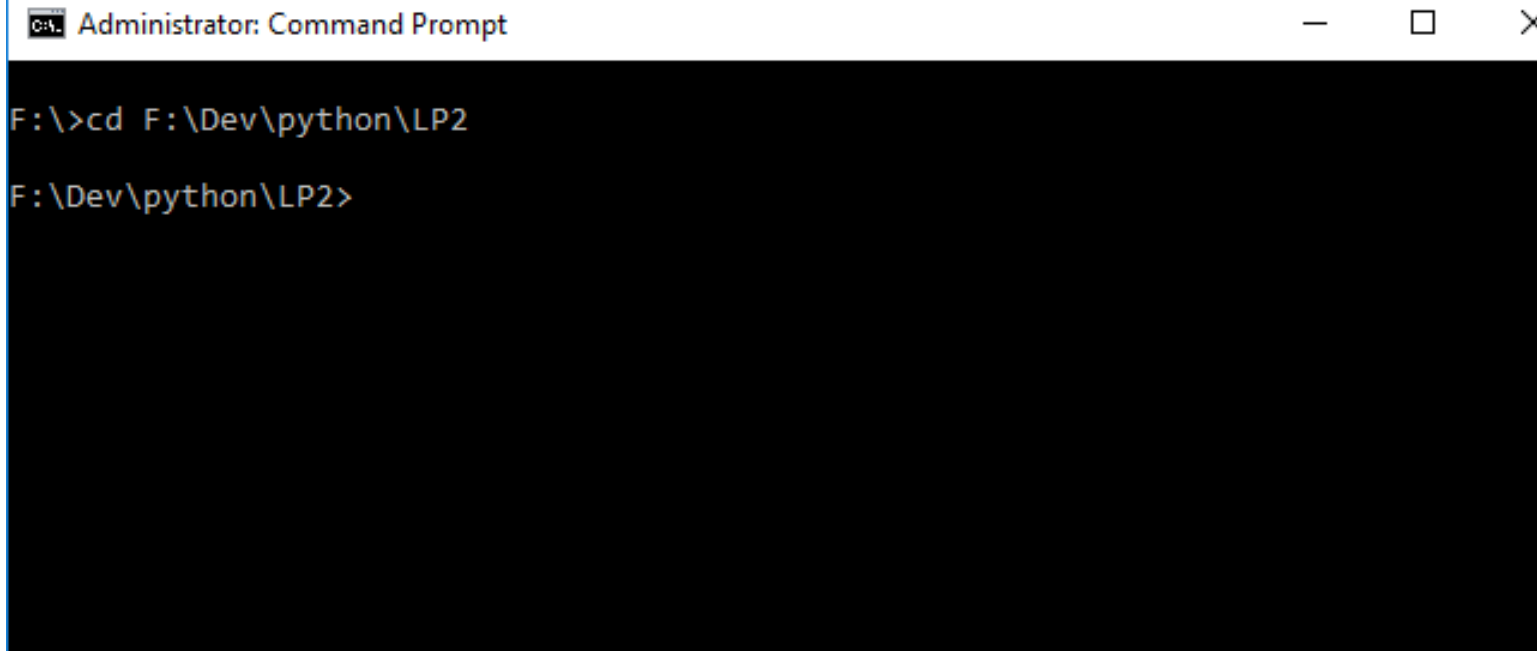




```
C:\>pip install virtualenv
Collecting virtualenv
  Using cached virtualenv-15.1.0-py2.py3-none-any.whl
Installing collected packages: virtualenv
Successfully installed virtualenv-15.1.0
C:\>
```

- Criar ambiente virtual para um projeto
- No cmd navegue até a pasta onde quer criar o ambiente virtual:

```
cd <caminho_da_pasta>
```



Administrator: Command Prompt

```
F:\>cd F:\Dev\python\LP2
```

```
F:\Dev\python\LP2>
```



- Criar ambiente virtual para um projeto
- Digitar no cmd o seguinte comando para criar o ambiente virtual

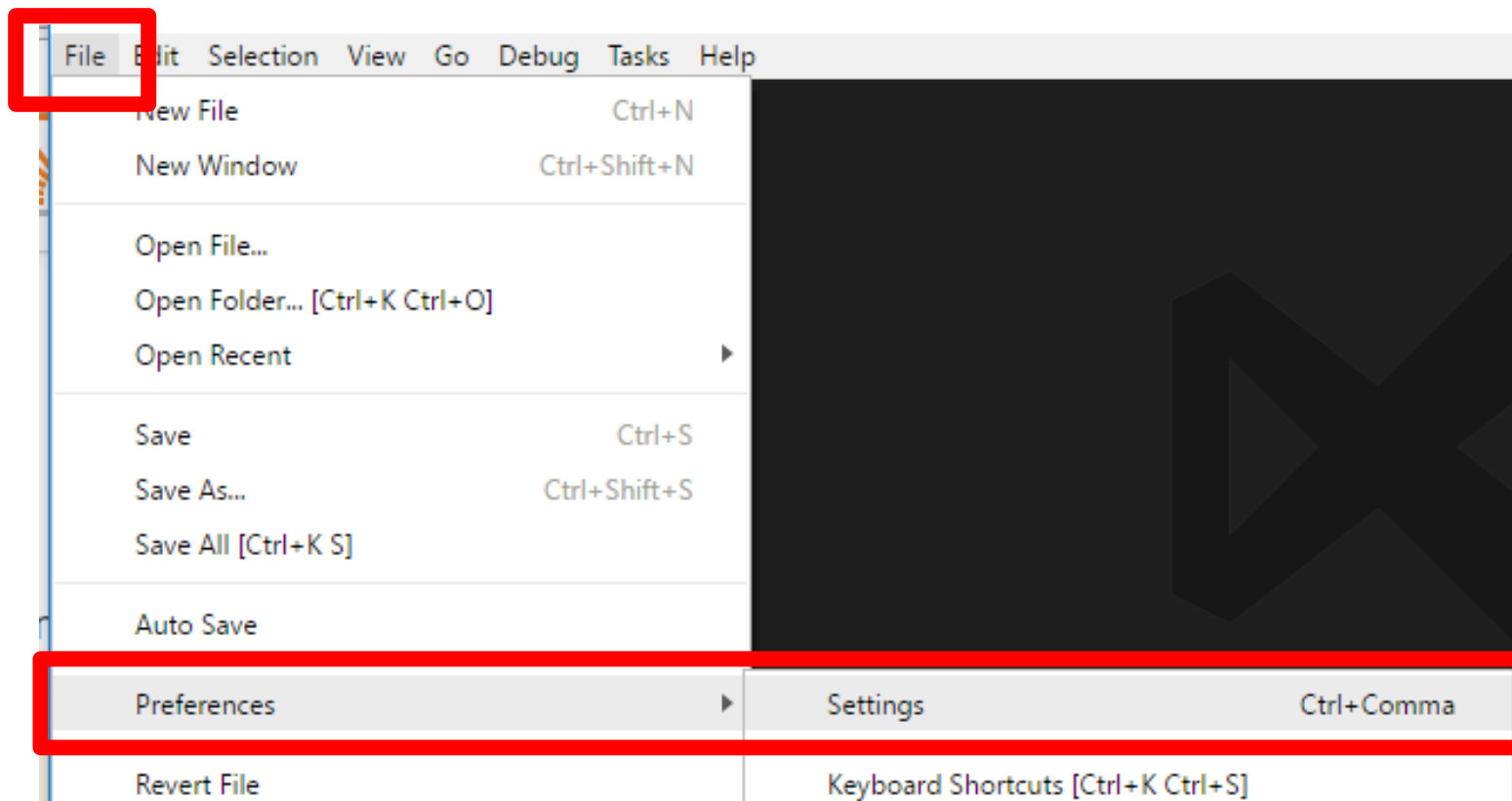
```
virtualenv <nome_da_pasta>
```

Administrator: Command Prompt

```
F:\Dev\python\LP2>virtualenv MyPython
Using base prefix 'c:\\program files (x86)\\python36-32'
New python executable in F:\Dev\python\LP2\MyPython\Scripts\python.exe
Installing setuptools, pip, wheel...done.

F:\Dev\python\LP2>
```

- Configurar virtualenv no VS Code
- Pressione o atalho Ctrl + , para abrir as configurações do VS Code, ou acesse o menu abaixo:





EXPLORER

OPEN EDITORS

{ settings.json C:\User...

NO FOLDER OPENED

You have not yet
opened a folder.

Open Folder



{ settings.json x

Search Settings

Total 429 Settings

USER SETTINGS

WORKSPACE SETTINGS

DEFAULT SETTINGS Place your settings in the file to t

Commonly Used (11)

```
// Controls auto save of dirty files.  
Accepted values: 'off', 'afterDelay',  
'onFocusChange' (editor loses focus),  
'onWindowChange' (window loses focus). If  
set to 'afterDelay', you can configure  
the delay in 'files.autoSaveDelay'.  
"files.autoSave": "off",
```

```
// Controls the font size in pixels.  
"editor.fontSize": 14,
```

```
// Controls the font family.  
"editor.fontFamily": "Consolas, 'Courier  
New', monospace",
```

```
// The number of spaces a tab is equal  
to. This setting is overridden based on  
the file contents when  
'editor.detectIndentation' is on
```

```
1 // Place your settings in this file  
2 {  
3  
4 }  
5
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

1: powershell.exe

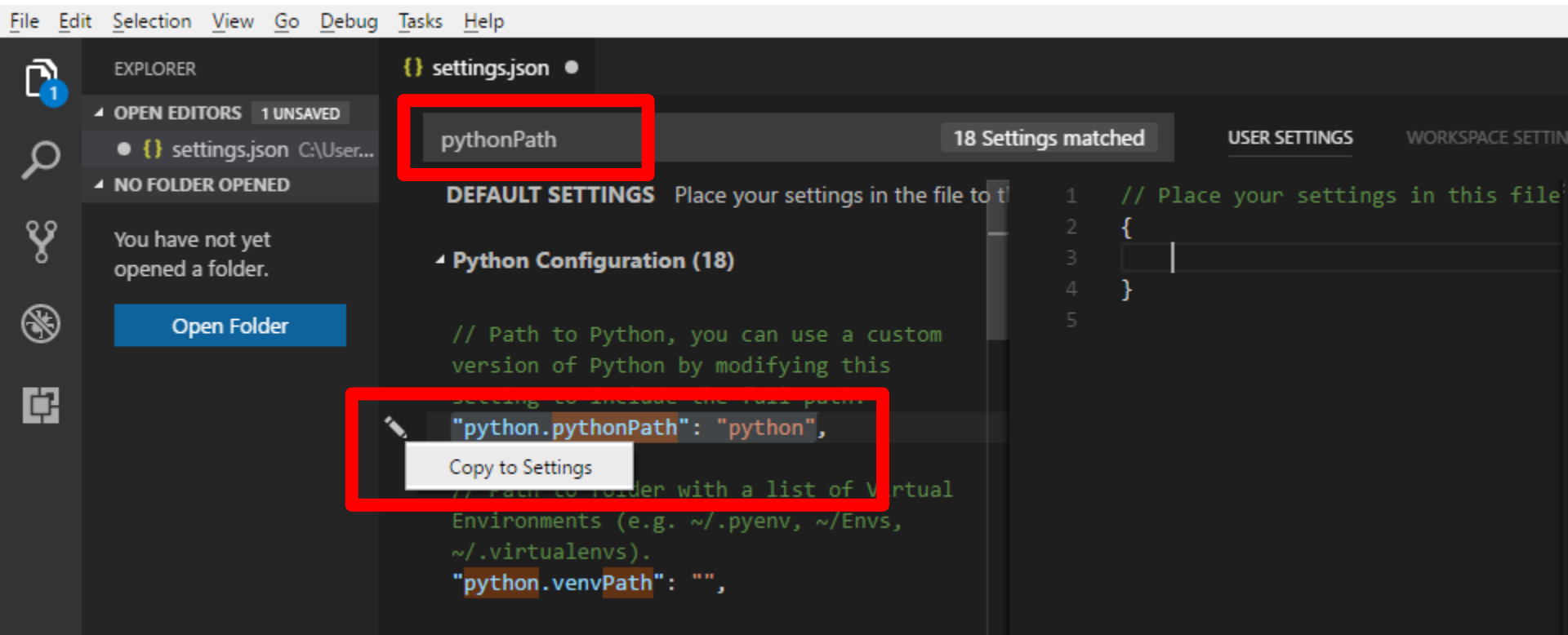


Windows PowerShell

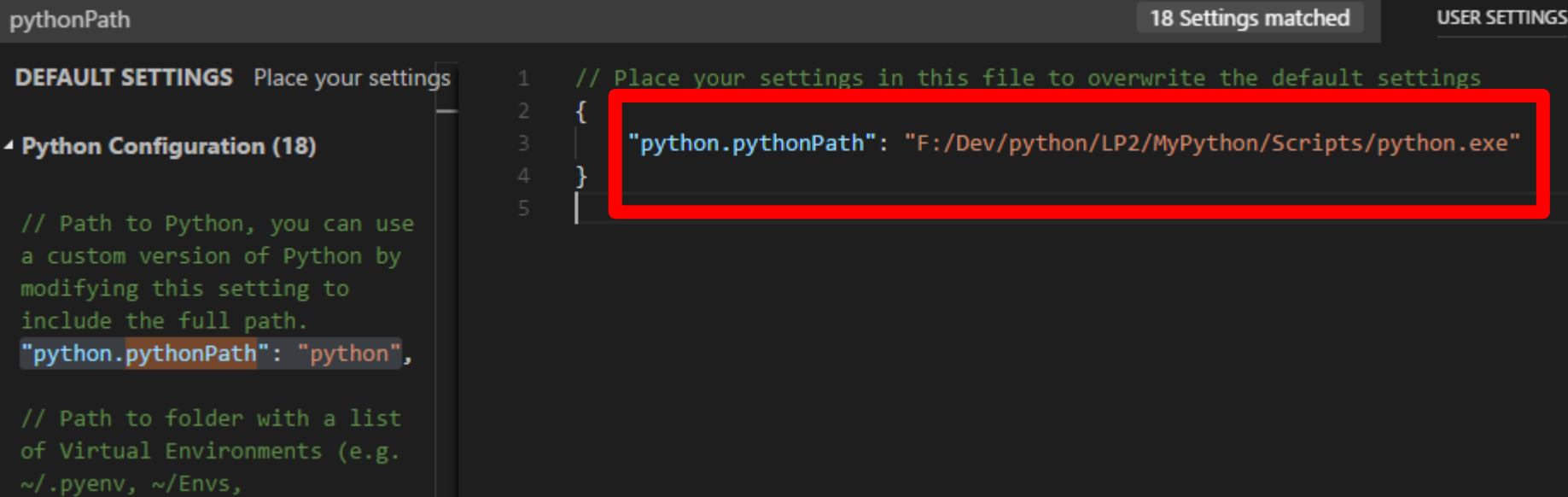
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\fsousa>

- Na caixa de pesquisa, procure por pythonPath e clique na opção de editar ao lado da propriedade



- No editor do lado direito, altere o valor da propriedade para o o caminho do seu ambiente virtual, inserindo no final /Scripts/python.exe. Depois reinicie o VS Code para aplicar as alterações



The screenshot shows the VS Code settings editor with the 'pythonPath' search filter. The left sidebar displays 'DEFAULT SETTINGS' and 'Python Configuration (18)'. The main editor shows a JSON configuration file with the following content:

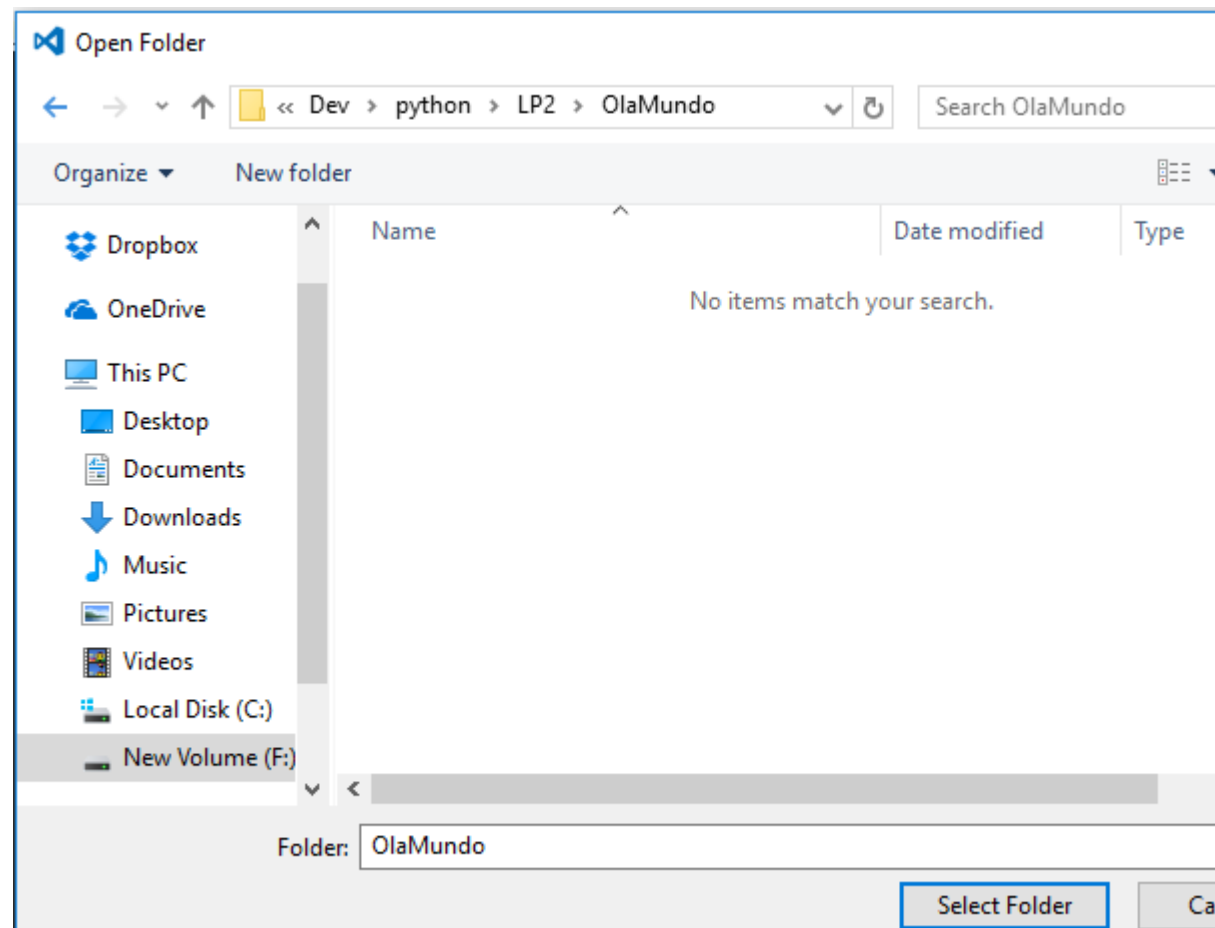
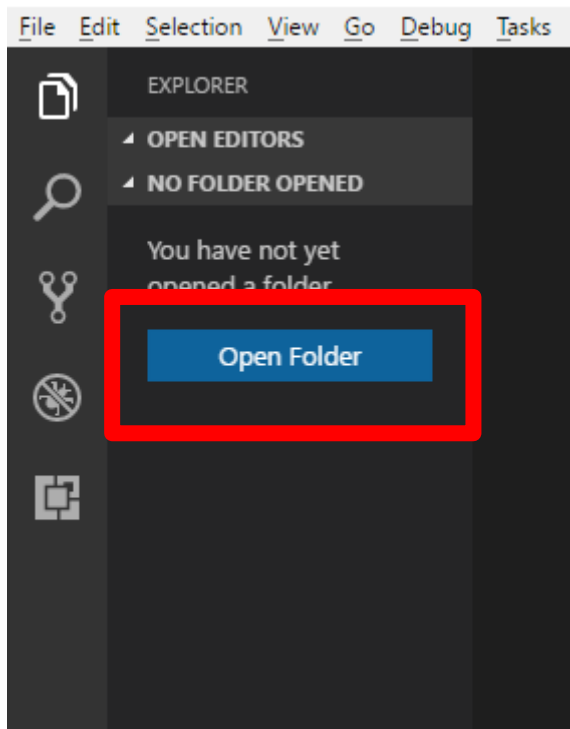
```
1 // Place your settings in this file to overwrite the default settings
2 {
3   "python.pythonPath": "F:/Dev/python/LP2/MyPython/Scripts/python.exe"
4 }
5
```

The line `"python.pythonPath": "F:/Dev/python/LP2/MyPython/Scripts/python.exe"` is highlighted with a red rectangle. The top right of the settings editor indicates '18 Settings matched' and 'USER SETTINGS'.

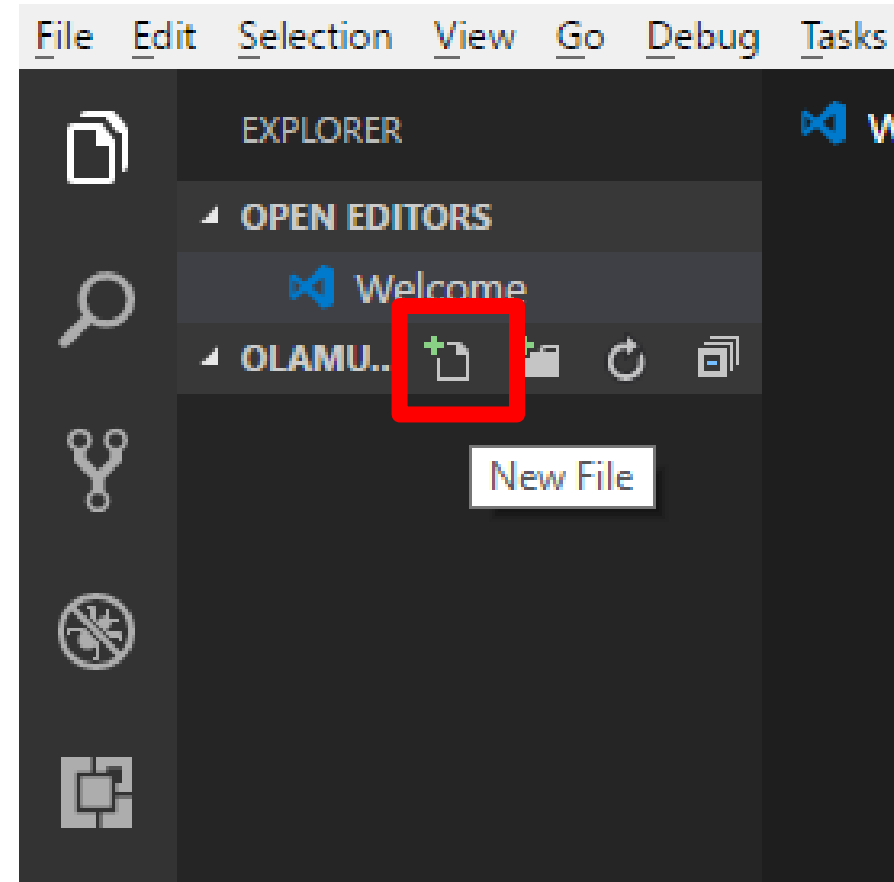


Projeto de Teste

- Clique no botão “Open Folder” para criar e selecionar uma pasta onde estará seu projeto

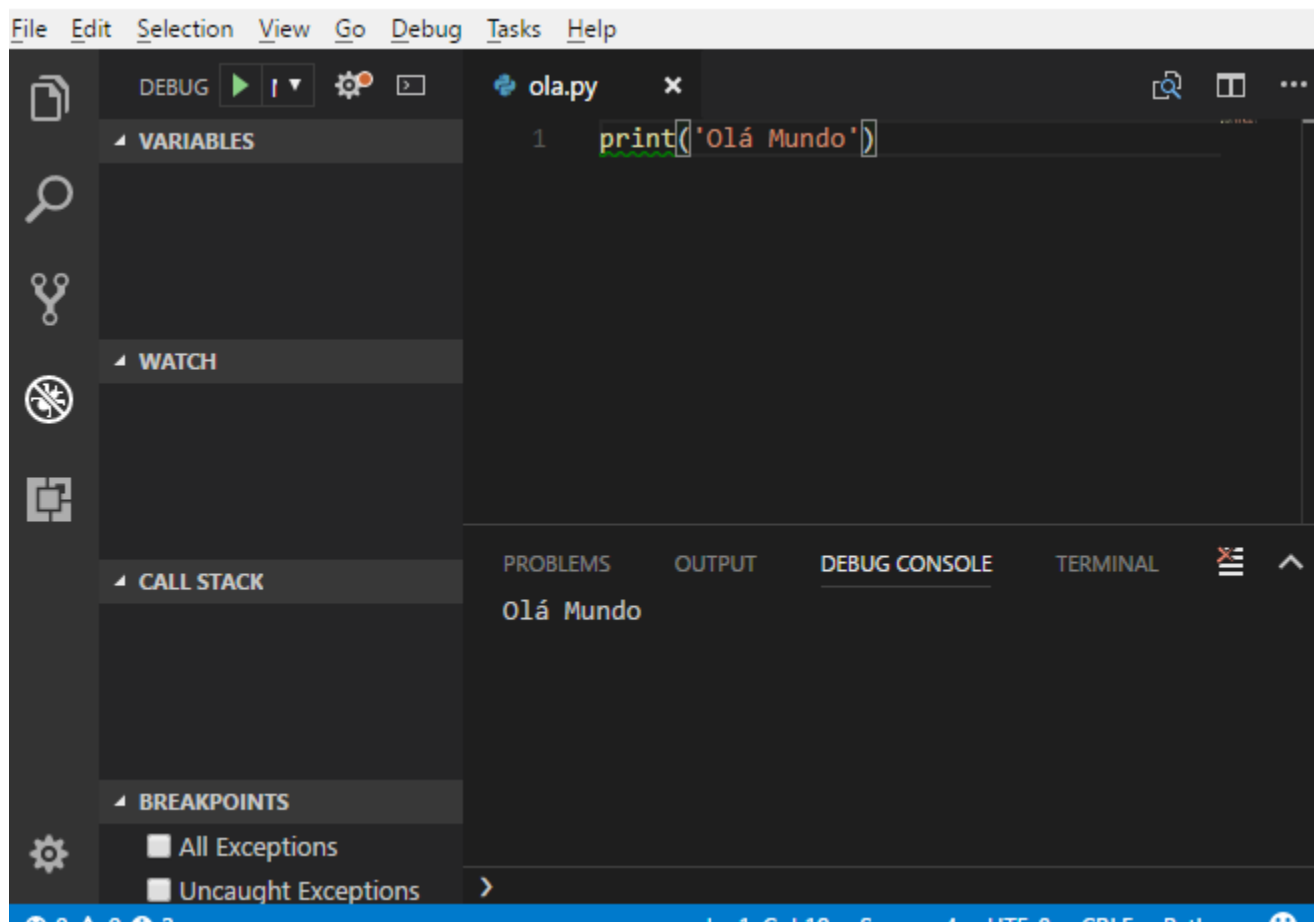


- Um novo projeto será exibido na sua IDE.
- Clique na opção abaixo para criar um novo arquivo
- Digite o nome do arquivo
 - ola.py
- Pressione Enter
- Clique para abrir no editor





- Escreva um programa em python e teste pressionando Ctrl + F5



- Python é uma linguagem de programação com as seguintes características:
 - Altíssimo nível
 - Orientada a objeto
 - Dinamicamente tipada
 - Fortemente tipada (verifica os tipos antes de fazer operações)
 - Interpretada
 - Multiplataforma



- A sintaxe do Python é clara e concisa
 - Sintaxe é a estrutura de código; define como as instruções são escritas
 - Favorece a legibilidade do código
 - Linguagem mais produtiva
 - Poucas linhas de código fazem muita coisa
 - Diferencia maiúsculas de minúsculas



- Uma variável em Python só pode ser utilizada se um valor foi atribuído a ela
 - Dizemos que a variável foi **definida** ou **iniciada**
 - Se a variável não foi definida ou iniciada, o interpretador do Python mostrará um erro na execução ao tentar utilizá-la



- Não é preciso definir um tipo de variável
- Python define o tipo da variável quando executa o comando
 - Termo técnico: define o tipo em **tempo de execução**
 - Por isso, Python é uma linguagem **dinamicamente tipada**
- Por ser dinamicamente tipada, o tipo da variável pode mudar dentro do seu programa.



- Strings - str
 - Qualquer coisa que estiver entre `""` ou ```
 - Ex: `'LP1'`, `"LP2"`, `'Python'`
- Números inteiros - int
 - Qualquer número sem aspas e sem casa decimais
 - Ex: 1, 2, 10, 1000
- Números de ponto flutuante – float
 - Qualquer número sem aspas e com casas decimais
 - Ex: 3.14, 2.3, 10.0
- Booleanos – bool
 - True e False





Revisão Python – Operadores

=	Atribuição
+	Soma de números Concatenação de string ou listas
-	Subtração
*	Multiplicação de números Repetição de Strings ou elementos de lista
/	Divisão
//	Divisão inteira
**	Potenciação
%	Resto da divisão



Revisão Python – Operadores

==	Igualdade
!=	Diferente
>	Maior que
<	Menor que
>=	Maior ou igual que
<=	Menor ou igual que
and	Operador lógico E
or	Operador lógico OU
not	Operador Lógico NÃO





Revisão Python – Funções básicas

print()	Mostrar uma mensagem na tela, seja string, número ou qualquer outro tipo de dado
input()	Comando de entrada de dados
int()	Transformar em um inteiro
float()	Transformar em um float
str()	Transforma em uma string
type()	Mostrar o tipo
len()	Tamanho da string ou da lista





Revisão Python – Operações em strings

<code>str[i]</code>	Acessar o caractere <code>i</code> da string (começando em 0)
<code>str.lower()</code>	Modifica todos os caracteres para minúsculo
<code>str.upper()</code>	Modifica todos os caracteres para maiúsculo
<code>str.format()</code>	Formatar a string
<code>str()</code>	Transforma em uma string
<code>type()</code>	Mostrar o tipo
<code>len()</code>	Tamanho da string ou da lista

- Executar instruções em um programa de forma condicional
 - if
 - if... else
 - if...elif...else



- Exemplos

```
idade = 65
if idade >= 60:
    print("Você está na melhor idade")
```

```
idade = 65
if idade >= 60:
    print("Você está na melhor idade")
if idade < 60:
    print("Você ainda não viveu nada!")
```



- Exemplos

```
idade = 65
if idade >= 60:
    print("Você está na melhor idade")
else:
    print("Você ainda não viveu nada!")
```

```
idade = 65
if idade > 60:
    print("Você está na melhor idade")
elif idade == 60:
    print("Você está no limite!")
else:
    print("Você ainda não viveu nada!")
```





- Exemplos

```
idade = int(input("Idade? "))
if idade >= 60:
    print("Idoso")
elif idade >= 18 and idade < 60:
    print("Adulto")
elif idade >= 13 and idade < 18:
    print("Adolescente")
elif idade >= 3 and idade < 13:
    print("Criança")
else:
    print("Bebê")
```





- Exemplos

```
idade = int(input("Idade? "))
if idade >= 60:
    print("Idoso")
else:
    if idade >= 18 and idade < 60:
        print("Adulto")
    else:
        if idade >= 13 and idade < 18:
            print("Adolescente")
        else:
            if idade >= 3 and idade < 13:
                print("Criança")
            else:
                print("Bebê")
```



- Identação
 - Espaços inseridos antes de comando
 - Fundamental para a sintaxe do python
 - Define onde um bloco de execução termina
 - A quantidade de espaços utilizada não importa, desde que seja mantido o mesmo padrão no mesmo bloco





- Identação

```
idade = int(input("Idade? "))
if idade >= 60:
    print("Idoso") # bloco de execução do if
else:
    # início do bloco de execução do else
    if idade >= 18 and idade < 60:
        print("Adulto")
    print('Fim do bloco if idade >= 18 and idade < 60')
    if idade >= 13 and idade < 18:
        print("Adolescente")
    else:
        if idade >= 3 and idade < 13:
            print("Criança")
        else:
            print("Bebê")
# fim do bloco de execução do else
```



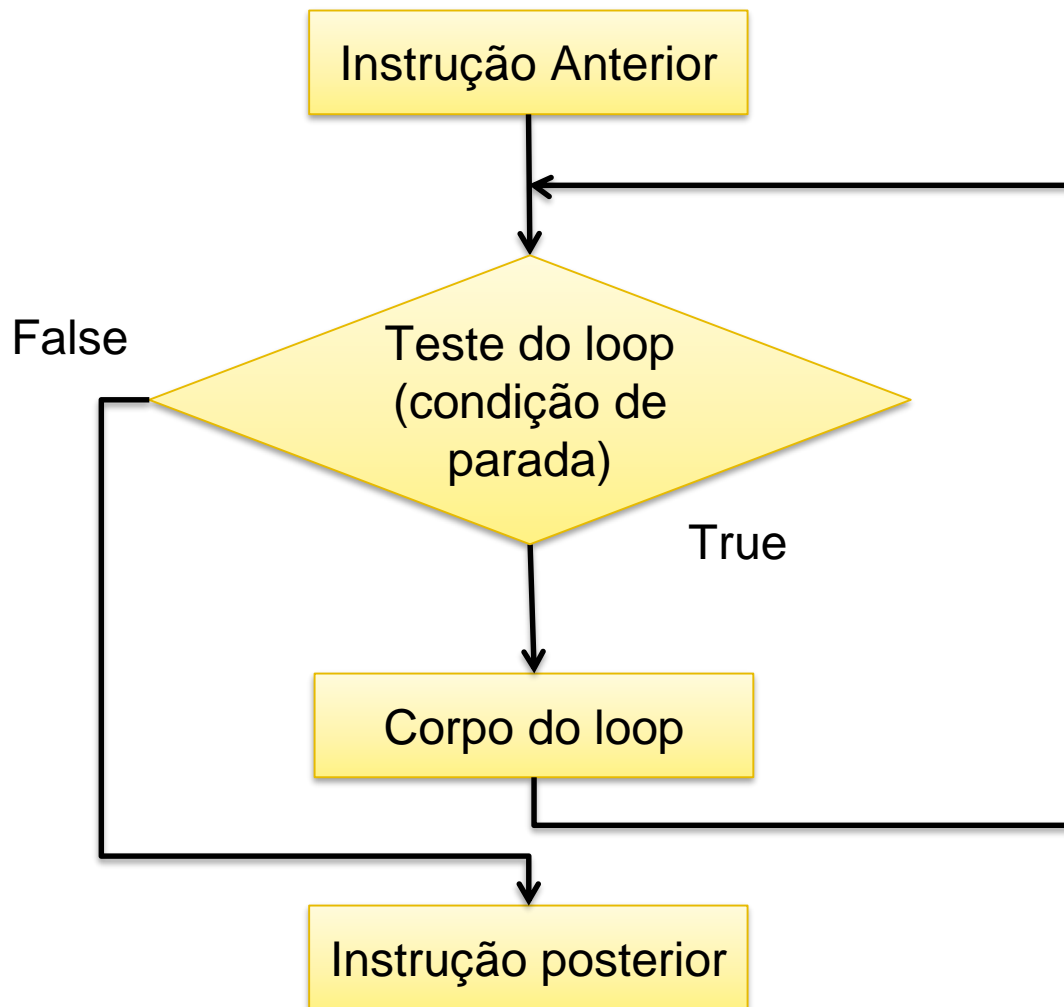
Revisão Python – Estruturas de repetição

- Estruturas que possibilitam a execução de um comando diversas vezes sem a necessidade de escrever um comando várias vezes
- Um comando é executado até que um critério de parada (teste) seja satisfeito
 - O critério de parada é um valor booleano (True ou False)





Revisão Python – Estruturas de repetição



- Dois tipos de loop:
 - while
 - Loop indeterminado
 - Não sabe quando vai parar
 - for
 - Loop determinado
 - Sabe quando vai parar



- Exemplos

```
numero = 1
while numero <= 100:
    print(numero)
    numero = numero + 1
print("finalizado")
```

- No loop while sempre deve existir uma variável de controle
 - Determina quando o loop será finalizado
 - Deve ser alterada dentro do loop
- O bloco que será repetido deve ser indentado



- Exemplos

- Executar 100 vezes (1 até 100)

```
for x in range(1,101):  
    print(x)
```

- Executar 100 vezes (0 até 99)

```
for x in range(99):  
    print(x)
```

- Mostrar os números pares de 1 a 100

```
for x in range(1,101,2):  
    print(x)
```



- Exemplos
 - Percorrer string

```
texto = "Linguagem de Programação I"  
for c in texto:  
    print(c)
```



- Construir códigos para reutilizar, evitando assim a repetição de código
- Duas partes:
 - Declaração da função
 - Chamada da função



- Declaração

- Inicia com a palavra def
- Dá um nome da função
- Coloca-se parêntesis
- Termina a declaração com :
- Escreve o bloco de execução
 - O bloco de execução deve ser indentado

```
def parabens():  
    print("Parabéns pra você")
```



- Chamada

- Apenas definir a função não faz com que ela seja executada
- As funções devem ser explicitamente chamadas (invocadas) para que sua execução ocorra

```
def parabens():  
    print("Parabéns pra você")
```

```
parabens() #chamada da função
```



- Chamada

- Uma função pode ser chamada quantas vezes forem necessárias, de qualquer região do seu programa, inclusive de outras funções

```
def parabensFernando():  
    parabens()  
    parabens()  
    print("Parabéns, Fernando")  
    parabens()
```

```
def parabens():  
    print("Parabéns pra você")
```

```
parabens()  
parabensFernando()
```



- Parâmetros / Argumentos
 - Funções podem receber parâmetros/argumento
 - Um parâmetro é uma variável inicializada quando a função é chamada
 - Ele só existe dentro da função onde foi definido, nunca fora dele
 - Seu uso e nomeação é exatamente igual a variáveis

```
def soma(n1, n2):  
    s = n1 + n2  
    print(s)  
soma(2, 3)
```



- Parâmetros / Argumentos
 - Funções podem receber parâmetros/argumento
 - Um parâmetro é uma variável inicializada quando a função é chamada
 - Ele só existe dentro da função onde foi definido, nunca fora dele
 - Seu uso e nomeação é exatamente igual a variáveis

```
def soma(n1, n2):  
    s = n1 + n2  
    print(s)  
soma(2, 3)
```



- Retorno

- Uma função pode retornar valores para o ponto em que foi chamada
- Utilize a palavra chave `return` como último comando do corpo da função

```
def soma(n1, n2):  
    s = n1 + n2  
    return s  
r = soma(2, 3)
```



- Retorno
 - Quando encontra o return, sai da função e retorna o controle para o ponto onde a função foi chamada.
 - O valor da instrução de retorno é enviado de volta para o ponto de chamada como um resultado da expressão.
 - Por isso há uma atribuição da função soma à variável r
 - O comando atribui o valor do retorno da função soma à variável r



- Retorno

- Uma função pode retornar mais de um valor.
- Basta listar mais de uma expressão na instrução de retorno.

```
def sumDiff(x, y):  
    sum = x + y  
    diff = x - y  
    return sum, diff
```



- Uma lista é um tipo de dado que organiza uma coleção de itens
- Listas em Python são **dinâmicas**.
 - Eles podem crescer e encolher sob demanda.
 - Não é preciso definir o tamanho inicial
- Listas em Python também são **heterogêneas**:
 - Isto é, uma única lista pode conter vários tipos dados.
 - Na mesma lista pode-se armazenar inteiro, float, string, objetos, booleanos...
- Em termos gerais, Listas em Python são **sequências mutáveis de objetos arbitrários**.



- Para iniciar uma variável com uma lista:

```
lista = []
```

- Para iniciar uma variável com uma lista de valores:

```
lista = [1, 2, 3, 4, 5]
```





Revisão Python – Listas

Operador	Significado
<code><lista> + <lista></code>	Concatenação
<code><lista> * <int-expr></code>	Repetição
<code><lista>[]</code>	Indexação
<code><lista>[:]</code>	Sublista
<code>for <var> in <lista>:</code>	Iteração
<code><expr> in <lista></code>	Verificação de existência
<code>del <lista>[:]</code>	Remover um ou mais elementos da lista



Revisão Python – Listas

Método	Significado
<code>len(<lista>)</code>	Comprimento
<code><lista>.append(x)</code>	Adiciona o elemento x ao fim da lista.
<code><lista>.extend(<lista2>)</code>	Adiciona múltiplos elemetos no final da lista
<code><lista>.sort()</code>	Ordena a lista.
<code><lista>.reverse()</code>	Reverte a lista.
<code><lista>.index(x)</code>	Retorna o índice da primeira ocorrência de x.
<code><lista>.insert(i, x)</code>	Insere x na posição i da lista.
<code><lista>.count(x)</code>	Retorna o número de ocorrências de x na lista.
<code><lista>.remove(x)</code>	Remove a primeira ocorrência de x na lista.
<code><lista>.pop(i)</code>	Remove o i-ésimo elemento da lista e retorna o seu valor.



Obrigado!

Prof. MSc. Fernando Sousa

