

2 - Prática: Linguagem de Programação Python (I)

#shibang(Comentário usado em arquivos de script usado para interpretar e saber que determinado arquivo será processado pelo compilador do python, usado para quando haver mais de um runtime)

```
#!/python3
print('Olá, Mundo!')
```

▼ Pacotes e módulos

- Pacotes são pastas que são carregadas através de uma main

Para importação utilizar seguinte comando

```
import pacote.sub.arquivo #Poderá variar conforme local
```

É possível averiguar nome das variáveis(Pacotes/módulos) através do seguinte comando

Variáveis

```
print(__name__) #pacote.sub.arquivo

print(__package__) #pacote.sub
```

Para comentar o código em python existe 2 métodos, utilizando o # e o """".

Através do # você está comentando uma linha em específico

Através do """" você está gerando uma string, da qual serve para comentário, porém é também utilizada para dar valores:

```
x = """
Eu vou jogar sinuca esta noite
"""
```

```
print(x)
```

▼ Interpolar valores

De modo raso, é ler algo fora do texto e interpretar ele dentro do texto, resolvendo problemas de printar um inteiro + string

```
texto = 'Sua idade é ...'
idade = 23

# interpolar valores:
print(f'{texto}{idade}')
```

Neste caso em questão, não existe concatenação através do sinal de +, porém é possível utilizar em uma string operadores matemáticos:

```
print (3*'Bom dia! ')
```

Obs: Durante habilitação do code runner o mesmo não estava aceitando caracteres como acentuação, realizando uma análise no fórum do stackoverflow:

<https://pt.stackoverflow.com/questions/352333/erro-de-acentuação-na-saída-do-visual-studio>

Os mesmos destacavam que seria necessário realizar alguns alterações e acrescentar comandos dentro do Json de config. Realizado processo porém, código continuou sendo rodado incorretamente. Minha recomendação é indicar no code runner para o mesmo utilizar terminal, mesmo que código traga toda relação do caminho de execução, se faz necessário devido os inputs que serão realizados durante os vídeos.

Definição de tipos de variáveis:

Ao definir a entrada de valores em uma variável através do input:

```
valor1 = input('Digite quantos anos você tem:')
#valor acima será uma string
```

```
valor2 = int('Digite quantos anos você tem:')  
#valor acima será um inteiro
```

▼ Lista:

Para definir uma lista, basta realizar do seguinte modo:

```
nums = [1, 2, 3]
```

Para adicionar um valor sem índice basta realizar do seguinte modo:

```
nums.append(3)  
nums.append(4)  
nums.append(5)  
nums.append(6)
```

Para inserir com um índice, basta realizar do seguinte modo:

```
nums.insert (0, -200)
```

O print por índice é muito semelhante ao C:

```
print(nums[3])
```

The image shows a Python IDE with two tabs: `main.py` and `lista.py`. The `lista.py` tab is active, displaying the following code:

```
1
2  nums = [1, 2, 3]
3
4  print(nums)
5
6
7  nums.append(3)
8  nums.append(3)
9  nums.append(4)
10 nums.append(5)
11 nums.append(6)
12
13 nums.insert (0, -200)
14
15
16 print(len(nums))
17 print(nums[4])
18 print(nums)
19
```

Below the code editor is a terminal window with tabs for `PROBLEMAS`, `SAÍDA`, and `TERMINAL`. The `TERMINAL` tab is active, showing the execution of the script:

```
<class 'float'>
<class 'str'>
<class 'bool'>
<class 'bool'>
PS> python -u "c:\Users\leofe\Documents\Nova pasta\Curso Python\tipos\lista.py"
<class 'list'>
PS C:\Users\leofe\Documents\Nova pasta> python -u "c:\Users\leofe\Documents\Nova pasta\Curso Python\tipos\lista.py"
[1, 2, 3]
9
3
[-200, 1, 2, 3, 3, 3, 4, 5, 6]
PS C:\Users\leofe\Documents\Nova pasta>
```

On the right side of the terminal, there is a list of files or folders: `Python`, `Python`, `powershell`, `Python`, `Python`, and `Code`. The `Code` item is currently selected.

▼ Tuplas:

Para geração de lista de nomes, damos o nome de tupla ou tuple em inglês.

Para definir uma tupla, única diferença será o uso de parênteses ao invés do colchetes

```
nomes = ('Ana', 'Bia', 'Gui', 'Ana')

print(type(nomes))
print(nomes)
```

É possível também gerar um operador lógico através de listas e tuplas:

```
print('Bia' in nomes)
#True
print('bia' in nomes)
#False
```

Devido python ser case-sensitive definindo maiúsculo trará um resultado e minúsculo outro

Para printar uma sequência definida pelo índice, é possível do seguinte modo:

```
print(nomes[1:3])
```

Deste modo estará trazendo a partir do valor 2 até antes do 4.

É possível printar também do seguinte modo:

```
print(nomes[2:])
#Todos valores a partir do 3
print(nomes[:-2])
#Todos valores até ante penultimo
```

The image shows a Python IDE with three tabs: `main.py`, `tuplas.py` (active), and `lista.py`. The `tuplas.py` file contains the following code:

```
1 nomes = ('Ana', 'Bia', 'Gui', 'Ana')
2
3
4 print(type(nomes))
5 print(nomes)
6
7 print('Bia' in nomes)
8
9 print(nomes[1:3])
```

Below the editor is a terminal window with tabs for `PROBLEMAS`, `SAÍDA`, and `TERMINAL` (active). The terminal shows the command `python -u "c:\Users\leofe\Documents\Nova pasta\Curso Python\tipos\tuplas.py"` and its output:

```
<class 'tuple'>
('Ana', 'Bia', 'Gui', 'Ana')
True
('Bia', 'Gui')
```

The output `('Bia', 'Gui')` is highlighted with a red rectangle. On the right side of the terminal, a list of recent files is shown, including `Python`, `powershell`, and `Code`.

▼ Conjuntos:

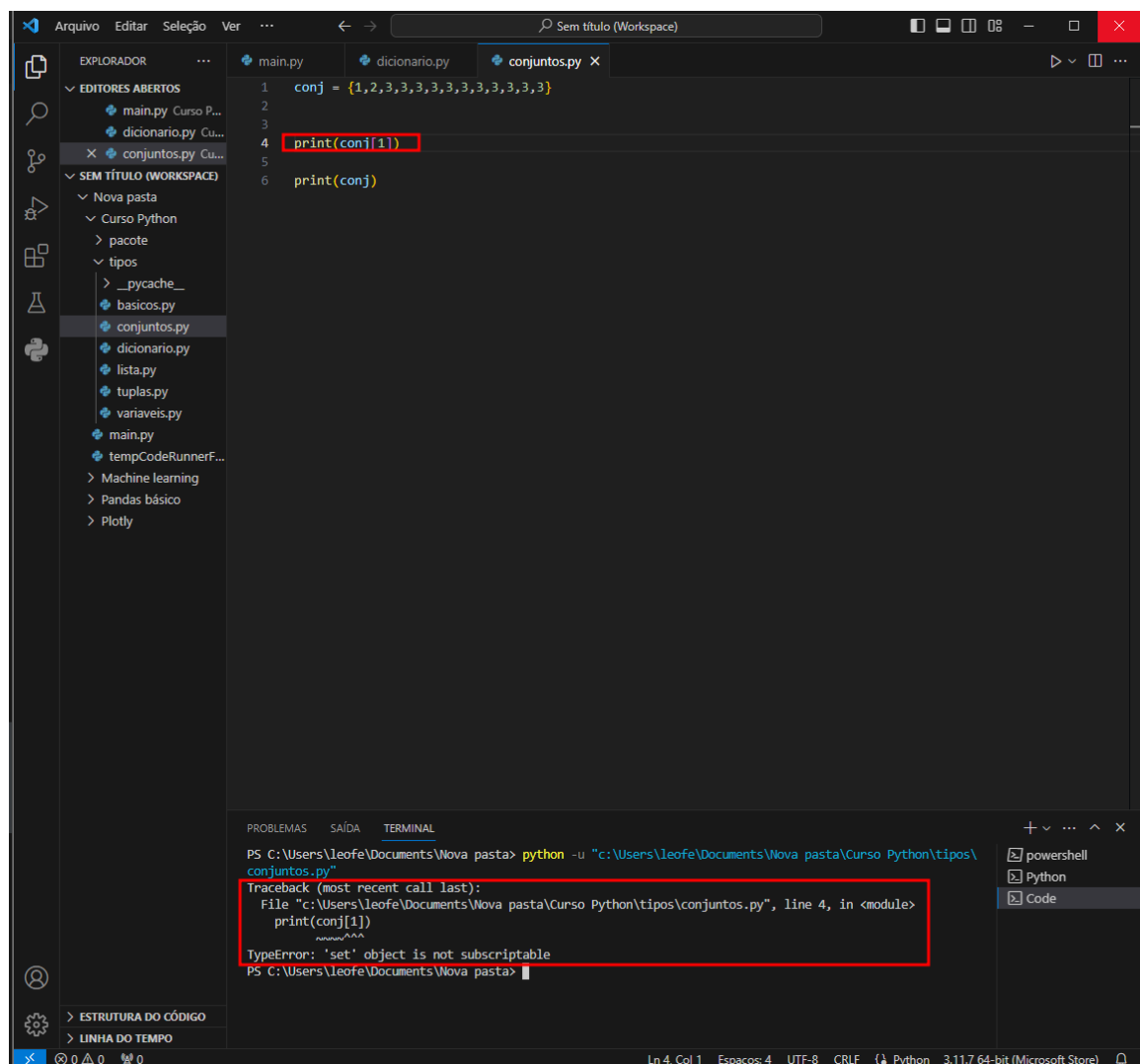
Os conjuntos em python ao contrário de C, não permitem indexação, gerando críticas para gerar comandos como o seguinte:

```
conj = {1, 2, 3, 3, 3, 3, 3}
print(conj[1])
```

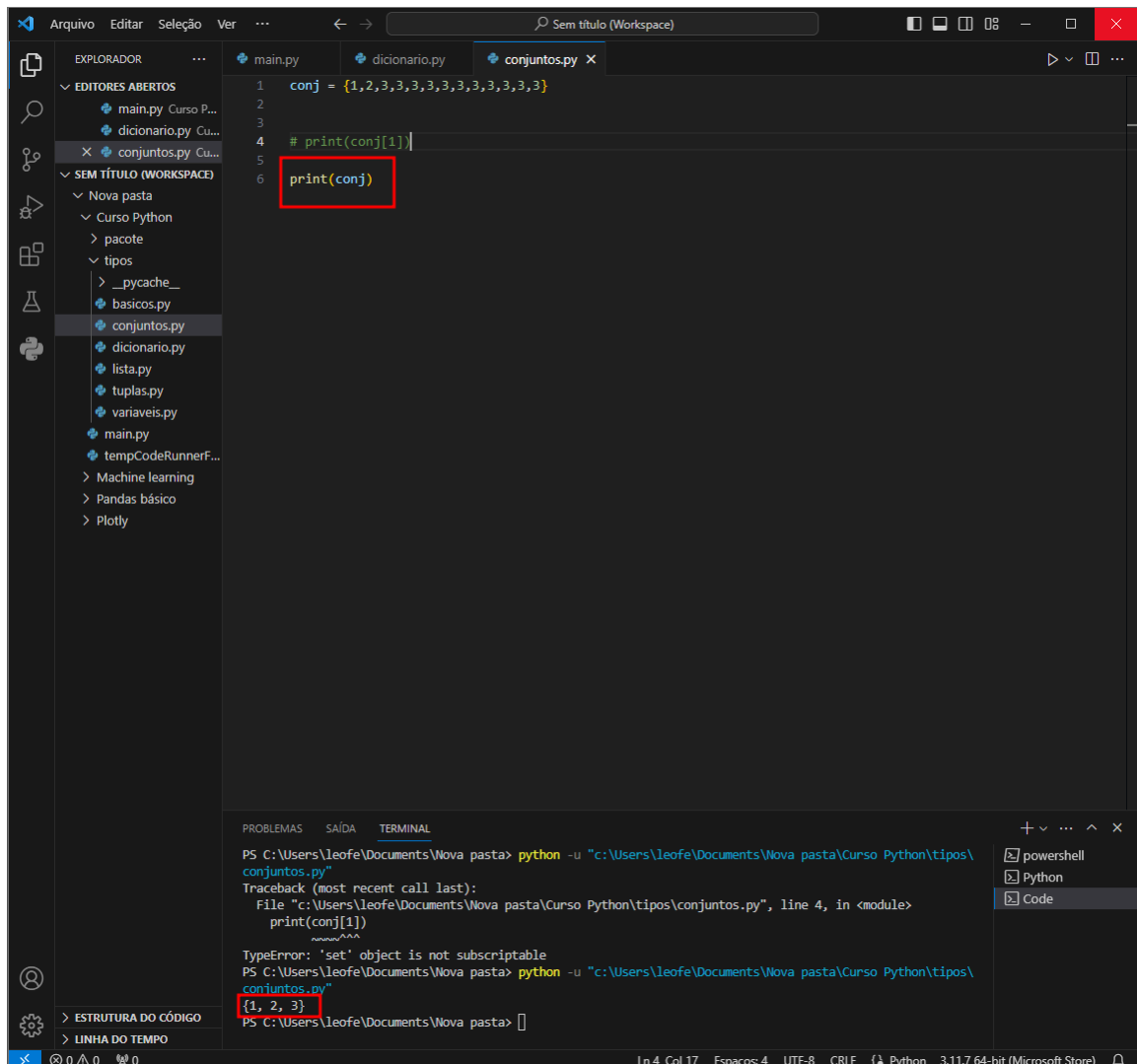
Este processo não ocorre quando for utilizado Tuplas.

Conjuntos não aceitam também registros duplicado como acima.

Erro ao tentar indicar indice no conjunto



Retorno do conjunto ao printar(valores duplicados não retornam)



▼ Dicionários:

Semelhante a estruturação de dados.

```
aluno = {
    'nome': 'Pedro Santana',
    'nota': '8.0',
    'ativo': True
}
```

```
print(type(aluno))
print(aluno['aluno'])
print(aluno['nota'])
```



```
print(aluno['ativo'])
print(len(aluno))
```

```
1
2
3
4  ▾ alunos = {
5      'nome': 'Pedro Santana',
6      'nota': '8.0',
7      'ativo': True
8  }
9
10
11  print(type(alunos))
12
13  print(alunos['nome'])
14  print(alunos['nota'])
15  print(alunos['ativo'])
16  print(len(alunos))
```

PROBLEMAS SAÍDA TERMINAL

```
True
3
PS C:\Users\leofe\Documents\Nova pasta> python -u "c:\Users\leofe\Documents\Nova pasta\Curso Python\tipos\
conjuntos.py"
<class 'dict'>
Pedro Santana
8.0
True
3
PS C:\Users\leofe\Documents\Nova pasta>
```

Ln 16, Col 19 Espaços: 4 UTF-8 CRLF Python 3.11.7 64-bit (Microsoft Store)

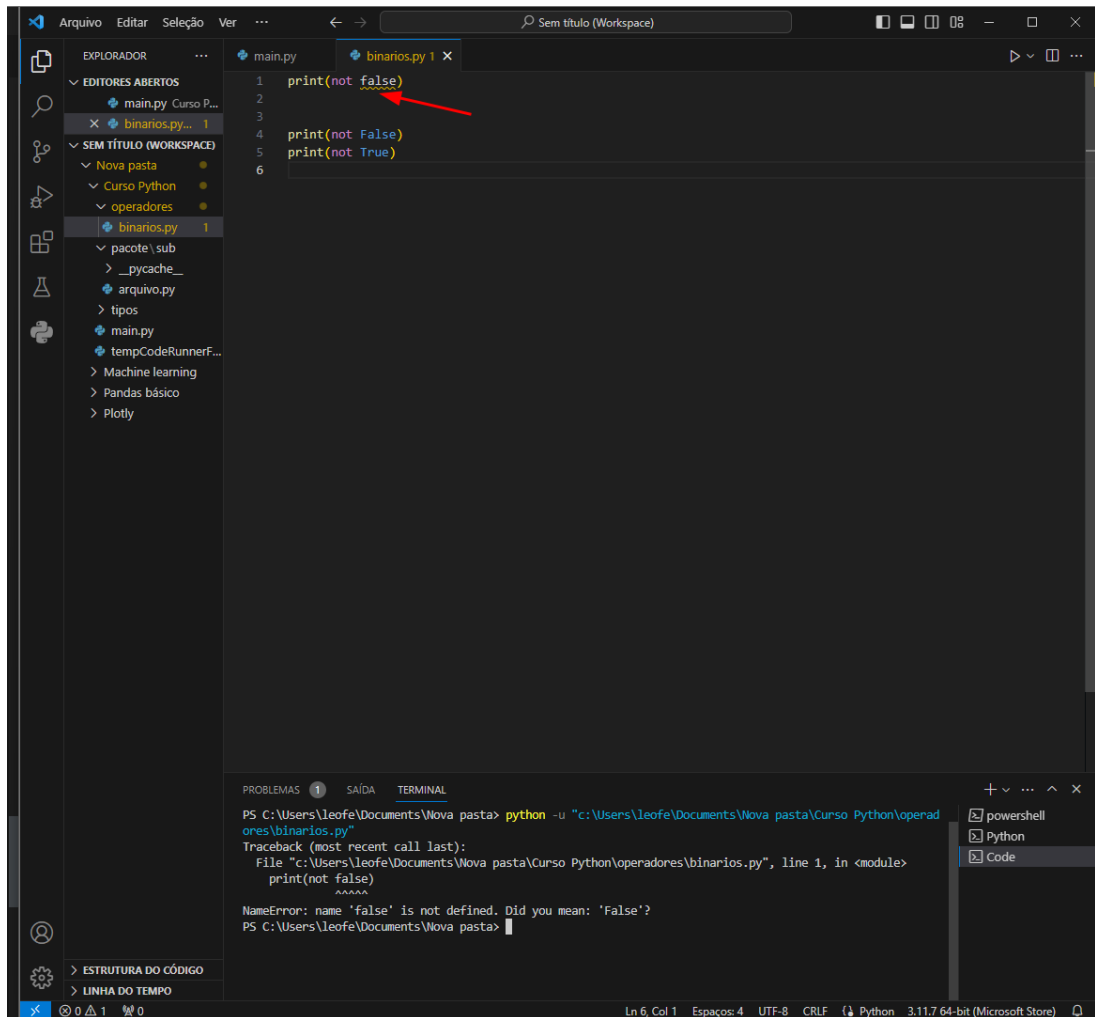
▼ Operadores:

▼ Binários:

Not:

```
print(not False)
```

```
print(not True)
```



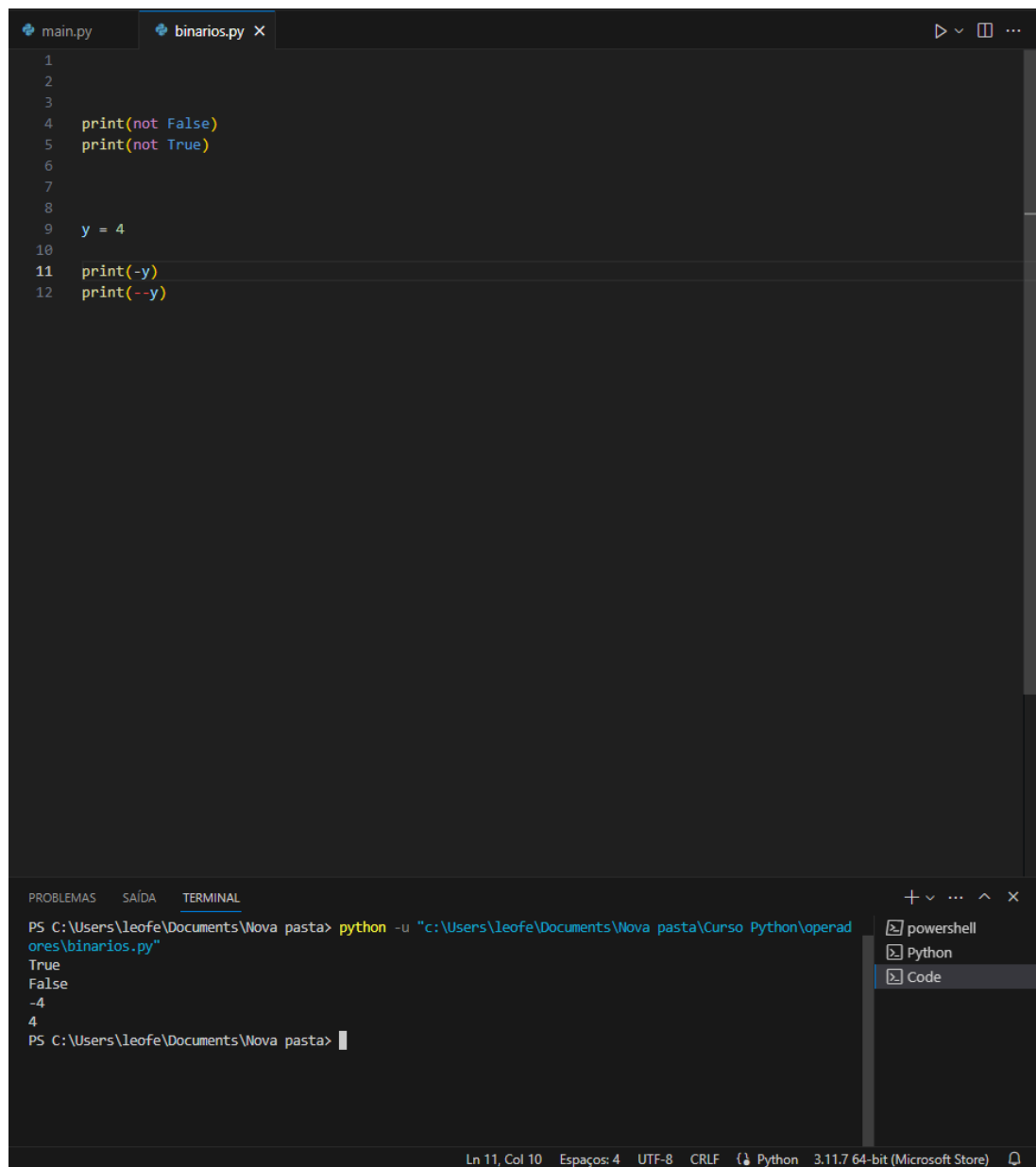
Utilizar sempre False e True com letra maiuscula, se não vai gerar critica.

-

```
y = 4
print(-y) #-4
print(--y) #4
```

Não funciona sintaxe de incremento e decremento:

```
y++
y--
```



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

```
1
2
3
4 print(not False)
5 print(not True)
6
7
8
9 y = 4
10
11 print(-y)
12 print(--y)
```

Below the code editor is a terminal window with the following output:

```
PS C:\Users\leofe\Documents\Nova pasta> python -u "c:\Users\leofe\Documents\Nova pasta\Curso Python\operadores\binarios.py"
True
False
-4
4
PS C:\Users\leofe\Documents\Nova pasta>
```

The status bar at the bottom indicates the current line and column (Ln 11, Col 10), the number of spaces (Espaços: 4), the encoding (UTF-8), the line ending (CRLF), the interpreter (Python 3.11.7 64-bit (Microsoft Store)), and a search icon.

▼ Aritméticos:

```
x = 10
y = 3
# +3 prefix
# operador vem antes do operando

# x++ postfix
# operador vem após operando

print(x + y)# sintaxe infits
print(x - y)
```

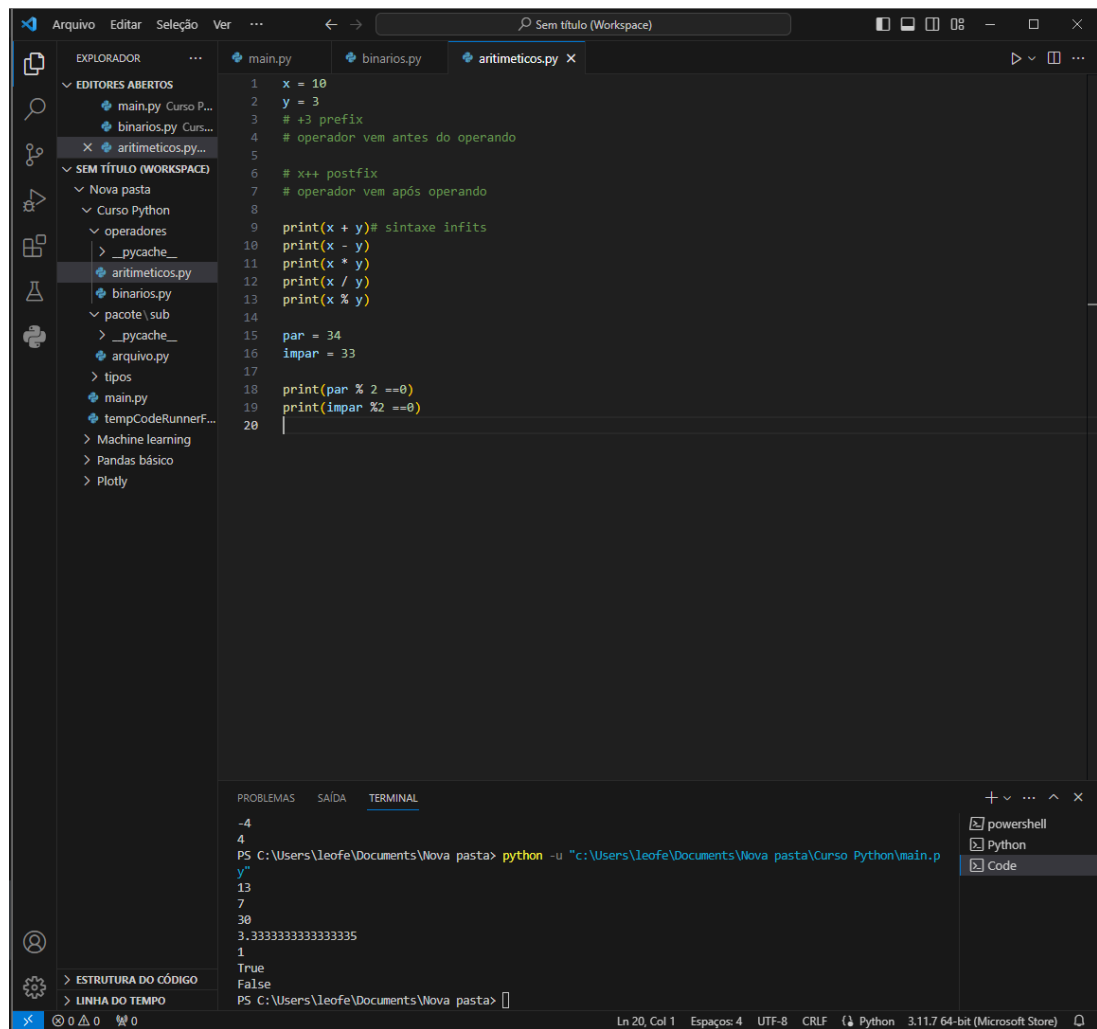
```

print(x * y)
print(x / y)
print(x % y)

par = 34
impar = 33

print(par % 2 ==0)
print(impar %2 ==0)

```



▼ Relacionais:

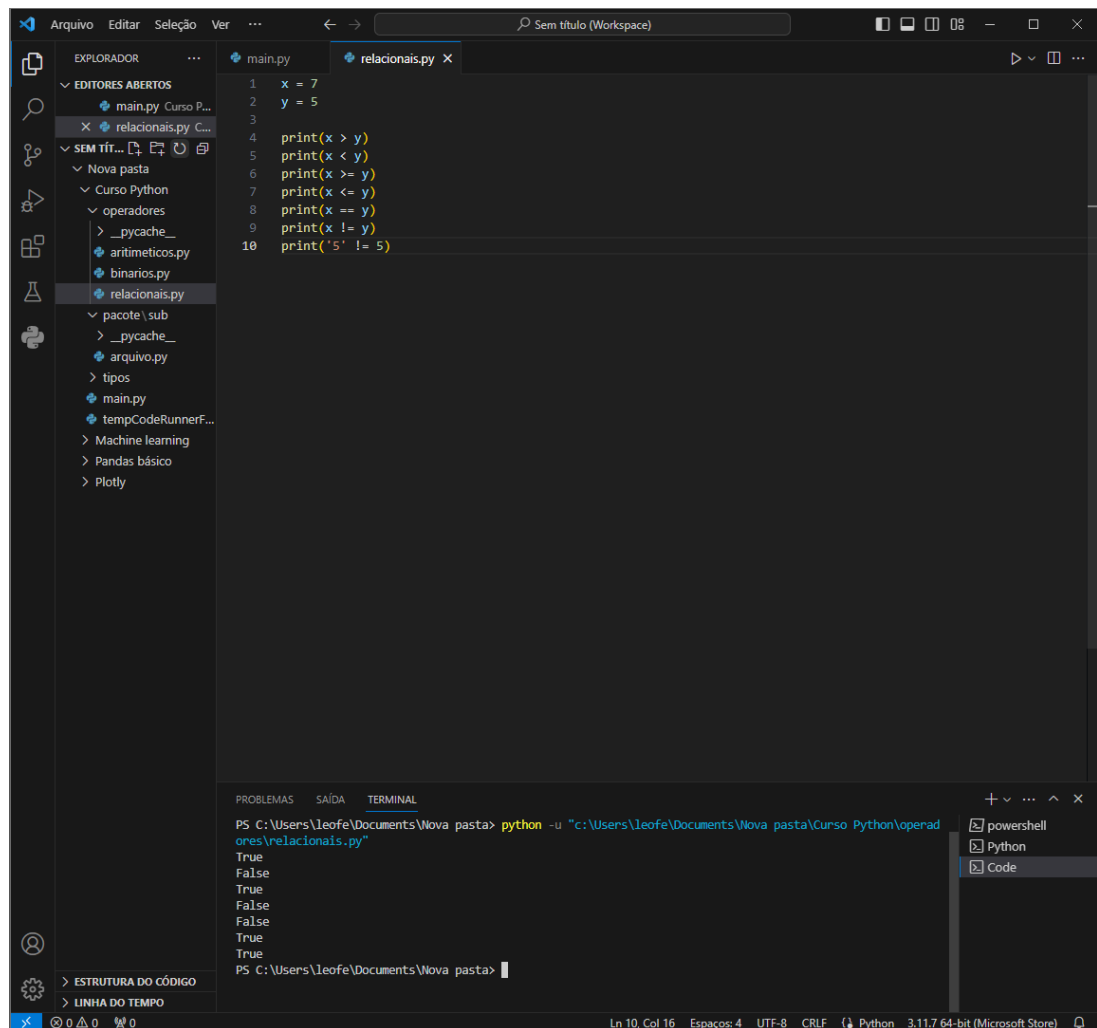
```

x = 7
y = 5

print(x > y)

```

```
print(x < y)
print(x >= y)
print(x <= y)
print(x == y)
print(x != y)
print('5' != 5)
```



▼ Atribuições:

Linguagem de tipos dinâmicos, você não precisa atribuir tipo para variáveis

```
Resultado = 2
print(resultado)

resultado = 3
print(resultado)
```

```
resultado = 'Teste'
print(resultado)
```

```
resultado = 2
resultado +=3
#resultado = 5
resultado -=1
#resultado = 4
resultado *=4
#resultado = 16
resultado /=2
#resultado = 8
resultado %=6
#resultado = 2
```

The screenshot shows the Visual Studio Code interface. The Explorer pane on the left displays the file structure of a workspace named 'Sem título (Workspace)'. The file 'atribuicoes.py' is selected. The Editor pane shows the code in 'atribuicoes.py', which includes a loop that performs arithmetic operations on a variable 'resultado' and prints the results. The Output pane at the bottom shows the execution of the script, displaying the output of the print statements: 'True', 'Soma(0)=5', 'Subtração(1)=4', 'Multiplicação(2)=16', 'Divisão(3)=8.0', and 'Resto(4)=2.0'. The status bar at the bottom indicates the file is at line 25, column 1, using UTF-8 encoding and CRLF line endings, and is running Python 3.11.7 64-bit (Microsoft Store).

```
Arquivo Editar Seleção ... Sem título (Workspace)
```

EXPLORADOR

- main.py
- atribuicoes.py

EDITORES ABERTOS

- main.py
- atribuicoes.py

SEM TÍTULO (WORKSPACE)

- Nova pasta
- Curso Python
 - operadores
 - __pycache__
 - aritimeticos.py
 - atribuicoes.py
 - binarios.py
 - relacionais.py
 - pacote\sub
 - __pycache__
 - arquivo.py
 - tipos
- main.py
- tempCodeRunnerF...
- Machine learning
- Pandas básico
- Plotly

main.py

```
1 cont=0
2 resultado = 2
3
4 resultado +=3
5 print(f'Soma({cont})={resultado}')
6 cont+=1
7 #resultado = 5
8 resultado -=1
9 print(f'Subtração({cont})={resultado}')
10 cont+=1
11 #resultado = 4
12 resultado *=4
13 print(f'Multiplicação({cont})={resultado}')
14 cont+=1
15 #resultado = 16
16 resultado /=2
17 print(f'Divisão({cont})={resultado}')
18 cont+=1
19 #resultado = 8
20 resultado %=6
21 print(f'Resto({cont})={resultado}')
22 cont+=1
23 #resultado = 2
24
25
26
27
28
29
30
```

PROBLEMAS SAÍDA TERMINAL

```
True
PS C:\Users\leofe\Documents\Nova pasta> python -u "c:\Users\leofe\Documents\Nova pasta\Curso Python\operadores\atribuicoes.py"
(0)=5
(1)=4
(2)=16
(3)=8.0
(4)=2.0
3
Teste
PS C:\Users\leofe\Documents\Nova pasta>
```

Ln 25, Col 1 Espaços: 4 UTF-8 CRLF Python 3.11.7 64-bit (Microsoft Store)

▼ Lógicos

```
b1 = True
b2 = False
b3 = True

print(b1 and b2)
#True se tudo for True

print(b1 or b2 or b3)
#True se algo é True

#Não existe XOR, para este caso utilizar o !=
print(b1 != b2)

print(not b1)
print(not b2)

print(b1 and not b2 and b3)

x = 3
y = 4

print(b1 and not b2 and x < y)
```

```
1 b1 = True
2 b2 = False
3 b3 = True
4
5 print(b1 and b2)
6 #True se tudo for True
7
8 print(b1 or b2 or b3)
9 #True se algo é True
10
11 #Não existe XOR, para este caso utilizar o !=
12 print(b1 != b2)
13
14 print(not b1)
15 print(not b2)
16
17 print(b1 and not b2 and b3)
18
19 x = 3
20 y = 4
21
22 print(b1 and not b2 and x < y)
```

PS C:\Users\leofe\Documents\Nova pasta> python -u "c:\Users\leofe\Documents\Nova pasta\Curso Python\operadores\logico.py"

False
True
True
False
True
True
True

PS C:\Users\leofe\Documents\Nova pasta>

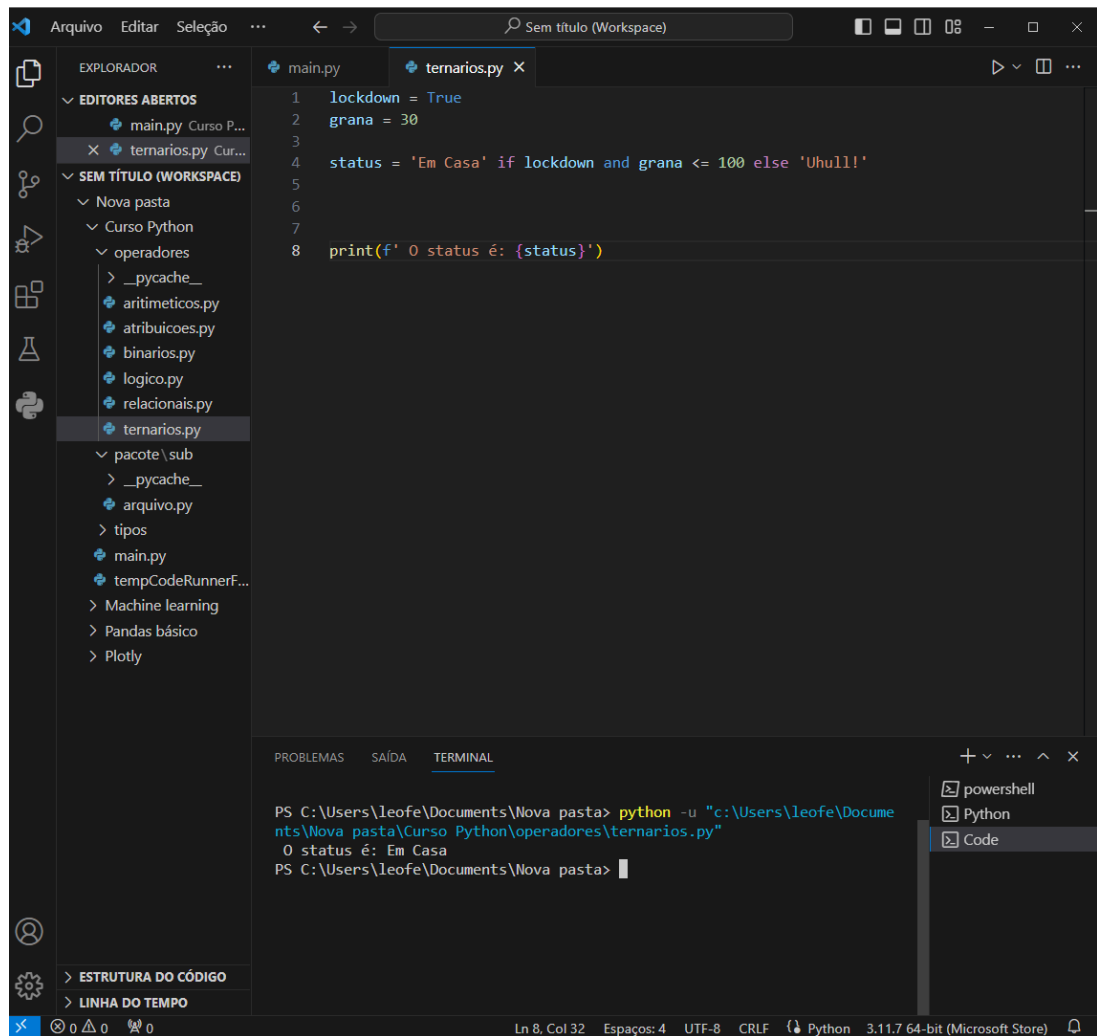
▼ Ternário:

```
lockdown = True
```

```
grana = 30
```

```
status = 'Em Casa' if lockdown and grana <= 100 else 'U
```

```
print(f' O status é: {status}')
```

▼ Controle

IF:

```
nota = float(input('Informe o valor da nota do Aluno '))

if nota >= 9:
    print(f'Sua nota: {nota} esta com status Ouro, Parabéns')
    #tabulação indicada através do TAB
    #espaçamento de aproximadamente 4 espaços

elif nota >= 7:
    print(f'Sua nota foi de: {nota}, parabéns você foi aprovado')

elif nota >= 5.5:
    print(f'Sua nota foi de {nota}, deverá realizar recuperação')
```

```
elif nota >= 3.0:
    print(f'Sua nota foi de {nota}, deverá além de realiza

else:
    print(f'Sua nota foi de {nota}, você foi Reprovado')
```

The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a workspace named 'Sem título (Workspace)' with a folder 'Curso Python' containing a file 'ifs.py'. The main editor window displays the code for 'ifs.py' with line numbers 1 through 18. The code implements a grade checking logic using if, elif, and else statements with f-strings for output. The bottom panel shows the 'TERMINAL' output, which matches the code's logic: for a grade of 9.0, it says 'status Ouro, Parabéns!'; for 5.5, it says 'deverá realizar recuperação'; and for 2.0, it says 'você foi Reprovado'.

```
1 nota = float(input('Informe o valor da nota do Aluno '))
2
3 if nota >= 9:
4     print(f'Sua nota: {nota} esta com status Ouro, Parabéns!')
5     #tabulação indicada através do TAB
6     #espaçamento de aproximadamente 4 espaços
7
8 elif nota >= 7:
9     print(f'Sua nota foi de: {nota}, parabéns você foi aprovado')
10
11 elif nota >= 5.5:
12     print(f'Sua nota foi de {nota}, deverá realizar recuperação')
13
14 elif nota >= 3.0:
15     print(f'Sua nota foi de {nota}, deverá além de realizar recuperação, realizar traba
16
17 else:
18     print(f'Sua nota foi de {nota}, você foi Reprovado')
```

PROBLEMAS SAÍDA TERMINAL

```
Informe o valor da nota do Aluno 9
Sua nota: 9.0 esta com status Ouro, Parabéns!
PS C:\Users\leofe\Documents\Nova pasta> python -u "c:\Users\leofe\Docume
nts\Nova pasta\Curso Python\controle\ifs.py"
Informe o valor da nota do Aluno 5.5
Sua nota foi de 5.5, deverá realizar recuperação
PS C:\Users\leofe\Documents\Nova pasta> python -u "c:\Users\leofe\Docume
nts\Nova pasta\Curso Python\controle\ifs.py"
Informe o valor da nota do Aluno 2
Sua nota foi de 2.0, você foi Reprovado
PS C:\Users\leofe\Documents\Nova pasta>
```

Obs: Python não utiliza a ideia de Chaves do C ao realizar funções, estruturas de repetição e afins, porém usa no lugar a indentação

▼ Verdadeiro e falso

```
a = 'valor'

if a:
    print('Existe!!!')
```

```

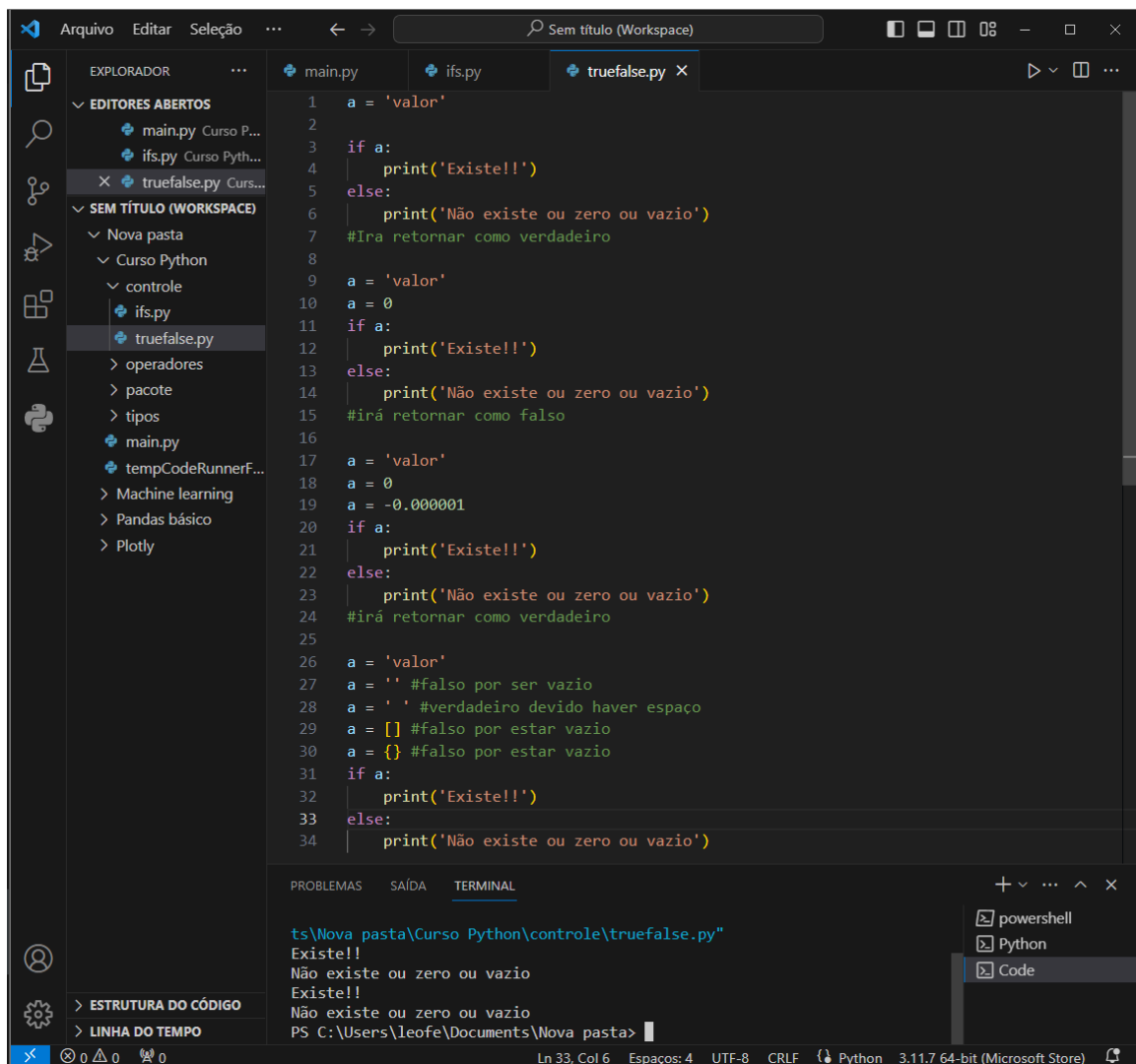
else
    print('Não existe ou zero ou vazio')
#Ira retornar como verdadeiro

a = 'valor'
a = 0
if a:
    print('Existe!!')
else
    print('Não existe ou zero ou vazio')
#irá retornar como falso

a = 'valor'
a = 0
a = -0.000001
if a:
    print('Existe!!')
else
    print('Não existe ou zero ou vazio')
#irá retornar como verdadeiro

a = 'valor'
a = '' #falso por ser vazio
a = ' ' #verdadeiro devido haver espaço
a = [] #falso por estar vazio
a = {} #falso por estar vazio
if a:
    print('Existe!!')
else
    print('Não existe ou zero ou vazio')

```



```

nota = float(input('Informe o valor da nota do Aluno '))
comportado = True if input('Comportado: (s/n)') == 's' else False
if nota >= 9 and comportado:
    print(f'Sua nota: {nota} esta com status Ouro, Parabéns!')
    #tabulação indicada através do TAB
    #espaçamento de aproximadamente 4 espaços

elif nota >= 7:
    print(f'Sua nota foi de: {nota}, parabéns você foi aprovado!')

elif nota >= 5.5:
    print(f'Sua nota foi de {nota}, deverá realizar recuperação!')

elif nota >= 3.0:

```

```

        print(f'Sua nota foi de {nota}, deverá além de realiza

else:
    print(f'Sua nota foi de {nota}, você foi Reprovado')

```

```

1  nota = float(input('Informe o valor da nota do Aluno '))
2  comportado = True if input('Comportado: (s/n)') == 's' or 'S' else False
3  if nota >= 9 and comportado:
4      print(f'Sua nota: {nota} esta com status Ouro, Parabéns!')
5      #tabulação indicada através do TAB
6      #espaçamento de aproximadamente 4 espaços
7
8  elif nota >= 7:
9      print(f'Sua nota foi de: {nota}, parabéns você foi aprovado')
10
11 elif nota >= 5.5:
12     print(f'Sua nota foi de {nota}, deverá realizar recuperação')
13
14 elif nota >= 3.0:
15     print(f'Sua nota foi de {nota}, deverá além de realizar recuperação, realizar traba
16
17 else:
18     print(f'Sua nota foi de {nota}, você foi Reprovado')
19

```

Terminal output:

```

PS C:\Users\leofe\Documents\Nova pasta> python -u "c:\Users\leofe\Documents\Nova pasta\Curso Python\controle\extra.py"
Informe o valor da nota do Aluno 9
Comportado: (s/n)S
Sua nota: 9.0 esta com status Ouro, Parabéns!
PS C:\Users\leofe\Documents\Nova pasta>

```

▼ Uso do For:

```

for i in range(10): #range quantidade de repetições
    print(i, end=',') # 0-9
print()

for i in range(1,11): #range quantidade de repetições
    print(i, end=',') #1-10
print()

```

```

for i in range(1,100,7): #range quantidade de repetições
    print(i, end=',') #1-100 de 7 em 7
    print()

for i in range(20,0,-3): #range quantidade de repetições
    print(i, end=',') #20-0 de 7 em 7
    print()

nums = [2, 4, 6, 8]

for n in nums:
    print(n, end=',') #end consegue colocar como ele vai c
    print()

texto = 'Python é muito massa!'

for letra in texto:
    print(letra, end=' ')
    print()
for n in {1, 2, 3, 4, 4, 4}
    print(n, end=' ') # irá trazer 1-4 devido set não perm
    print()
produto = {
    'nome': 'Caneta',
    'preço': 8.80,
    'desc': 0.5
}

for atrib in produto:
    print(atrib, '-->' , produto[atrib])
    print()

for atrib, valor in produto.items():
    print(atrib, '-->' , valor)
    print()

```

```
for valor in produto.values():  
    print(valor,end=' ')  
    print()  
  
for atrib in produto.values():  
    print(valor, end=' ')  
    print()
```

Retorno do console:

```
0,1,2,3,4,5,6,7,8,9,  
1,2,3,4,5,6,7,8,9,10,  
1,8,15,22,29,36,43,50,57,64,71,78,85,92,99,  
20,17,14,11,8,5,2,  
2,4,6,8,  
Python é muito massa!  
1 2 3 4  
nome --> Caneta preço --> 8.8 desc --> 0.5  
nome --> Caneta  
preço --> 8.8  
desc --> 0.5  
  
Caneta 8.8 0.5  
0.5 0.5 0.5
```

The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a project structure with a folder named 'Curso Python' containing a file 'for.py'. The main editor window displays the contents of 'for.py', which includes a list of numbers and a dictionary of product information. The terminal at the bottom shows the output of running the script, which prints the list of numbers and the product information in a formatted way.

```
0.5 0.5 0.5
PS C:\Users\leofe\Documents\Nova pasta> python -u "c:\Users\leofe\Documents\Nova pasta\Curso Python\Estrutura_Repeticao\for.py"
0,1,2,3,4,5,6,7,8,9,
1,2,3,4,5,6,7,8,9,10,
1,8,15,22,29,36,43,50,57,64,71,78,85,92,99,
20,17,14,11,8,5,2,
2,4,6,8,
P y t h o n   é   m u i t o   m a s s a !
1 2 3 4
nome --> Caneta preço --> 8.8 desc --> 0.5
nome --> Caneta
preço --> 8.8
desc --> 0.5
Caneta 8.8 0.5
```

▼ While

While não utiliza uma quantidade determinada para repetição, até a mesma ser falsa

```
x = 0

while x != -1:
    x = float(input('Informe o número ou -1 para sair: '))

print('Fim!')

nota = 0

while nota != -1:
```



```
nota = float(input('Informe o número ou -1 para sair:'))

if nota != -1:
    qtde += 1
    total += nota
print(f'A média da turma é ({total} / {qtde})')
```

▼ Extra

```
peessoas = ['Gui', 'Rebeca']
adj = ['Sapeca', 'Inteligente']
for p in pessoas:
    for a in adj:
        print(f'{p} é {a}!')

for i in [1, 2, 3]:
    pass # gerar uma classe vazia

for i in range(1, 11):
    if i % 2:
        continue # continua laço mesmo se false
    print(i)

for i in range(1,11):
    if i == 5:
        break # sai do laço de repetição
    print(i)

print('Fim! ')
```

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays the file structure of a workspace named 'Sem título (Workspace)'. It includes folders like 'EDITORES ABERTOS' and 'SEM TÍTULO (WORKSPACE)', and files such as 'main.py', 'whiles.py', and 'extra.py'. The main editor area shows the code in 'extra.py':

```
1 pessoas = ['Gui', 'Rebeca']
2 adj = ['Sapeca', 'Inteligente']
3 for p in pessoas:
4     for a in adj:
5         print(f'{p} é {a}!', end= ' ')
6     print()
7 for i in [1, 2, 3]:
8     pass # gerar uma classe vazia
9
10 for i in range(1, 11):
11     if i % 2:
12         continue # continua laço mesmo se false
13     print(i, end= ' ')
14     print()
15 for i in range(1, 11):
16     if i == 5:
17         break # sai do laço de repetição
18     print(i, end= ' ')
19 print()
20 print('Fim! ')
```

At the bottom, the TERMINAL panel shows the command prompt output:

```
PS C:\Users\leofe\Documents\Nova pasta> python -u "c:\Users\leofe\Documents\Nova pasta\Curso Python\estrutura_repeticao\extra.py"
Gui é Sapeca! Gui é Inteligente! Rebeca é Sapeca! Rebeca é Inteligente!
2 4 6 8 10
1 2 3 4
Fim!
PS C:\Users\leofe\Documents\Nova pasta>
```

▼ Problemas

Durante resolução dos códigos pude encontrar alguns problemas, desde coderunner que não estava aceitando UTF-8 até problemas com indentação:

Como estava:

```

13     nota = float(input('Informe o número ou -1 para sair:'))
14
15     if nota != -1:
16         qtde += 1
17         total += nota
18     print(f'{total} -- {qtde}')
19     print(f'A média da turma é ({total} / {qtde})')
20
21

```

Como era para ficar:

```

5     if nota != -1:
6         qtde += 1
7         total += nota
8     print(f'{total} -- {qtde}')
9     print(f'A média da turma é ({total} / {qtde})')
10

```

```

if nota != -1:
    qtde += 1
    total += nota
    print(f'{total} -- {qtde}')

```

Certo

```

if nota != -1:
    qtde += 1
    total += nota
    print(f'{total} -- {qtde}')

```

Errado

Ao jogar no bloco de notas me deparei que de fato, indentação estava puxando incorretamente. Como utilizo o Notion para documentar, escrevo os códigos através da opção de code dele. Apparently se utilizar o copiar e colar ele pode gerar divergências de indentação que ficam imperceptíveis no vscode.