

Aula 01

- Plano de Ensino
- Introdução a linguagem Python
- Fundamentos
- Estruturas de Condicionais e Repetições
- Tratamento de Exceções
- Funções e passagem de parâmetros
- Tuplas e Tuplas nomeadas
- Listas
- Dicionários
- Conjuntos



PLANO DE ENSINO



- Objetivo da Disciplina
 - Desenvolvimento de aplicações utilizando a linguagem de programação Python

Python

Fundamentos

Interfaces
(GUI)

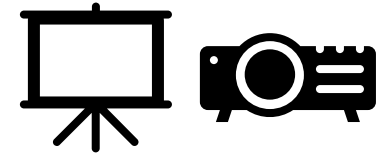
Web



- Conteúdo Programático
 - Introdução a Programação Python; Instalação e configuração do ambiente para desenvolvimento; Interpretador interativo; Entrada e Saída; Variáveis e Tipos de Dados; Estruturas Condicionais e Estruturas de Repetição;
 - Tuplas e Listas
 - Funções; Passagem de Parâmetros; Escopo; Recursão; Exceções;
 - Manipulação de Texto; Strings; Funções para Strings; Formatação; Leitura e escrita de arquivos texto;

- Conteúdo Programático (continuação)
 - Programação Orientada a Objetos; Classes; Objetos; Atributos; Métodos; Construtores; Encapsulamento;
 - Arquivos e diretórios
 - Comunicação; Programação em Rede;
 - Interfaces gráficas (GUI); Bibliotecas gráficas; Janelas; Controles;
 - Armazenamento de dados; Armazenamento persistente com banco de dados;
 - Desenvolvimento Web; Frameworks
 - Arquitetura Orientada a Serviços; Webservices

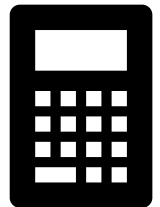
- Metodologia
 - Aulas Expositivas
 - Práticas no Laboratório de Informática



- Bibliografia Básica

- BEAZLEY, D.; JONES, B.K. Python Cookbook. Sebastopol: O'Reilly, 2013. Disponível em <http://chimera.labs.oreilly.com/books/1230000000393/index.html>, acesso em dezembro de 2017.
- LUZ; M. Learning Python, Fifth Edition. Sebastopol: O'Reilly, 2013.
- HALL, T.; STACEY, J.P. Python 3 for Absolute Beginners. New York: Apress, 2009.
- CHUN, W.I. Core Python Applications Programming. Michigan: Prentice Hall, 2012.
- Summerfield, M. Programming in Python 3 - A Complete Introduction to the Python Language, Second Edition. Indiana: Addison-Wesley, 2009.

- Avaliação
 - Prova 1 (P1)
 - Teoria = 70%
 - Prática = 30% (Atividades Práticas Avaliativas)
 - Prova 2 (P2)
 - Teoria = 70%
 - Prática = 30% (Atividades Práticas Avaliativas)
 - Média Final (MF): $(P1 * 0.40) + (P2 * 0.60)$
 - Aprovação: MF ≥ 6.0 e Frequência $\geq 75\%$



- Material de Apoio





INTRODUÇÃO





O que é ***Python***?



[...]

an interpreted, object-oriented, high-level

programming language with dynamic semantics

[...]

HETLAND, M.L. Beginning Python – From Novice to Professional. New York: Apress , 2008.

- Principais características da linguagem ***Python***
 - É uma linguagem interpretada.
 - Não há pré-declaração de variáveis, e os tipos das variáveis são determinados dinamicamente.
 - O controle de bloco é feito por meio de indentação, não delimitando com “chaves”, como na linguagem C, por exemplo.
 - Oferece tipos de dados de alto nível.
 - É orientada a objetos.

Introdução

- A linguagem ***Python*** foi criada por ***Guido Van Rossum***



Introdução

- **1982**

- Surge a ideia de criar a linguagem quando Guido Van Rossum trabalhava no **CWI** em Amsterdã na Holanda.



Centrum Wiskunde & Informatica, Centro de Matemática e Ciência da Computação

- **1987**

- Guido foi trabalhar na construção do sistema operacional **Amoeba**, percebendo a necessidade de uma linguagem de rápido desenvolvimento.

- **1991**
 - Primeira release
 - Versão 0.90
- **1994** - Versão 1.0
- **2000** - Versão 2.0
- **2001**
 - Criação da **Python Software Foundation**
 - Organização sem fins lucrativos
- **2008** – Versão 3.0

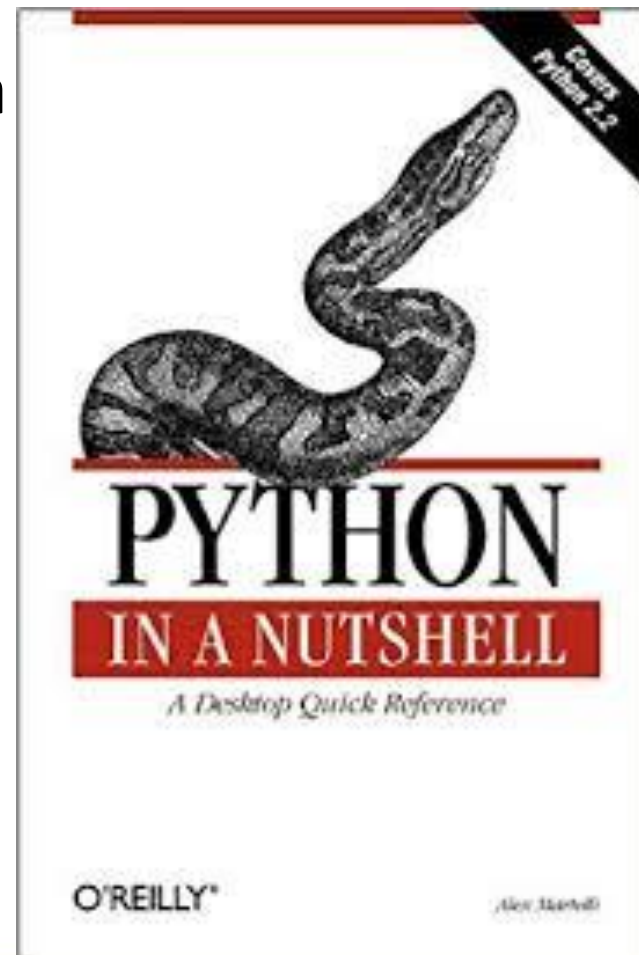
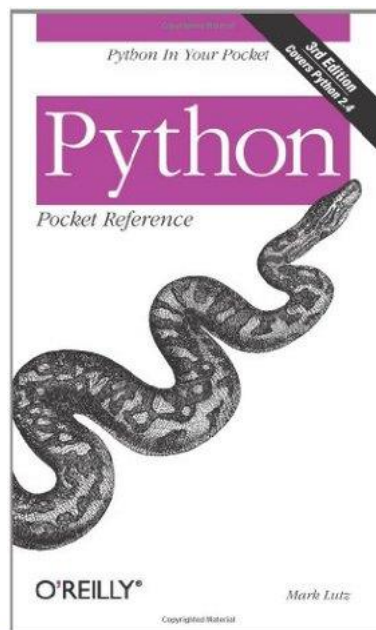


- A *origem* do nome *Python*
 - *Monty Python's Flying Circus*
 - Programa britânico de comédia que foi ao ar na BBC de 1969 até 1974.



Introdução

- A **origem** do nome **Python**
 - Livros clássicos da editora O'Reilly' estavam na capa uma *python*.



- *A filosofia Python*

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

The Zen of Python – Tim Peters

- Como participar da ***comunidade Python?***

pycon.org

- Implementações da linguagem

- ***CPython***

- Principal implementação escrita em C
 - Disponível na maioria das distros GNU e Mac OS; e também para Windows

- ***Jython***

- Escrita em Java
 - Integrante do IBM WebSphere e Oracle WebLogic

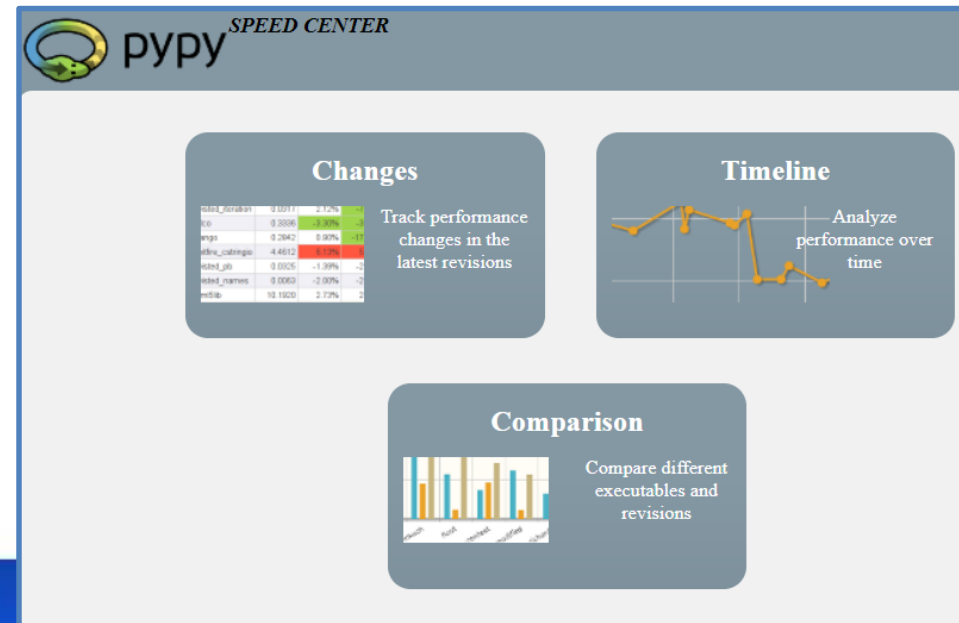
- ***IronPython***

- Desenvolvida pela Microsoft, escrita em C# e .NET CLR

Introdução

- Implementações da linguagem
 - **PyPy**
 - Python escrito em Python
 - Implementação em alto nível com desempenho 5x melhor que CPython

pypy.org



- O que eu preciso para ***programar em Python?***
 - O interpretador pode ser obtido gratuitamente em:

www.python.org/downloads

Introdução

- Qual é o *ambiente de desenvolvimento* para linguagem Python?

C:\> Prompt de Comando - python

```
G:\>python
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 18:11:49) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World!")
Hello World!
>>> _
```

Introdução

- Qual é o *ambiente de desenvolvimento* para linguagem Python?



```
hello.py - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda

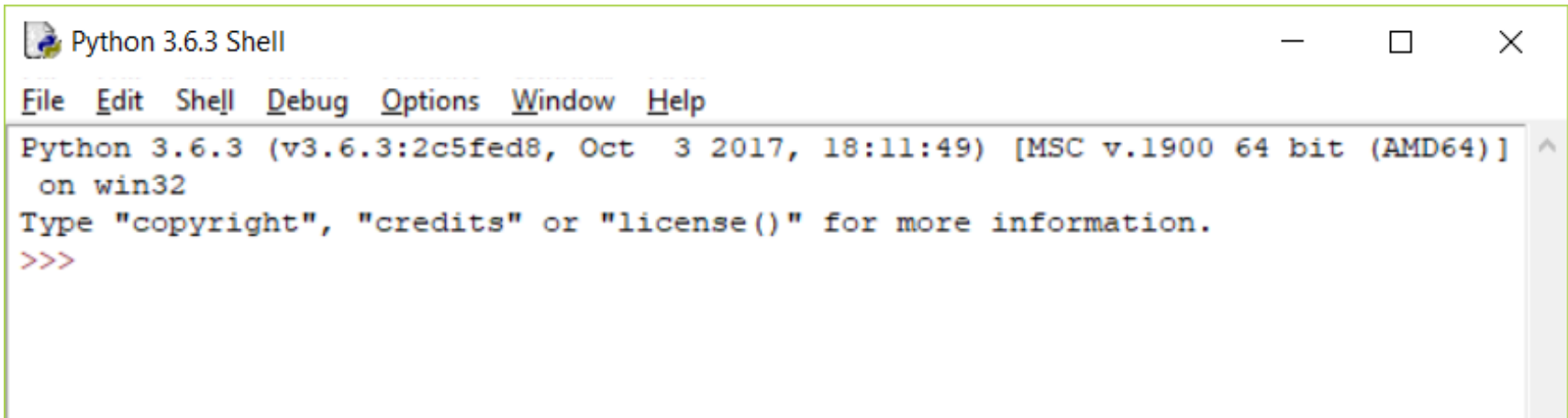
print("Hello World!")

G:\>python hello.py
Hello World!

G:\>_
```

Introdução

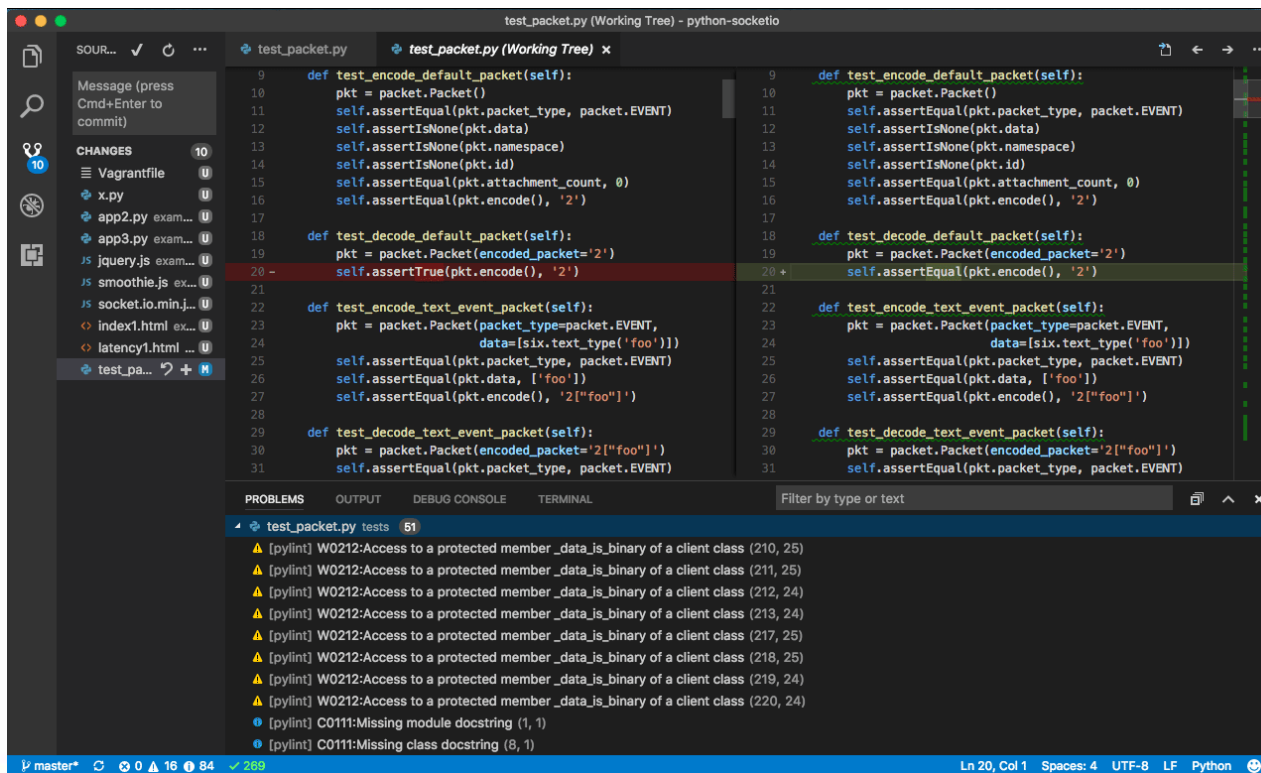
- Qual é o *ambiente de desenvolvimento* para linguagem Python?
 - *IDLE Python*

A screenshot of the Python 3.6.3 Shell window. The window has a title bar with the text "Python 3.6.3 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with the following items: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window displays the following text: "Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 18:11:49) [MSC v.1900 64 bit (AMD64)] on win32", "Type 'copyright', 'credits' or 'license()' for more information.", and the prompt ">>>".

```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 18:11:49) [MSC v.1900 64 bit (AMD64)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

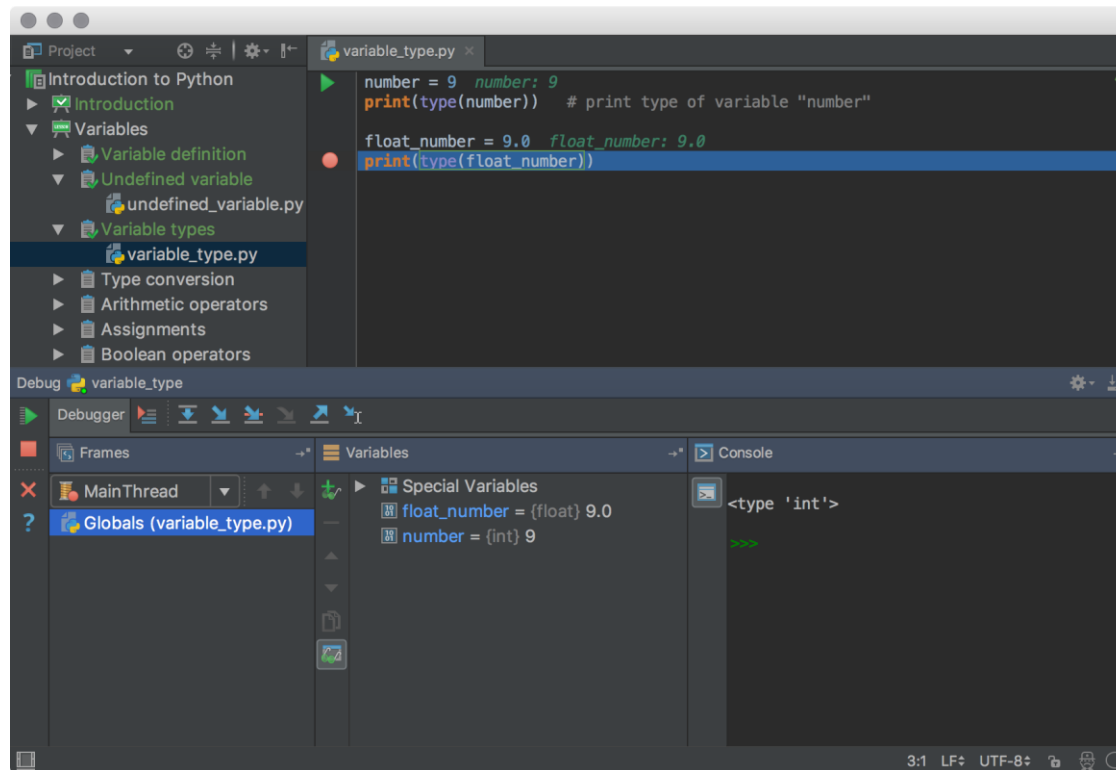
Introdução

- Qual é o *ambiente de desenvolvimento* para linguagem Python?
 - *Visual Studio Code*



Introdução

- Qual é o *ambiente de desenvolvimento* para linguagem Python?
 - *PyCharm - JetBrains*



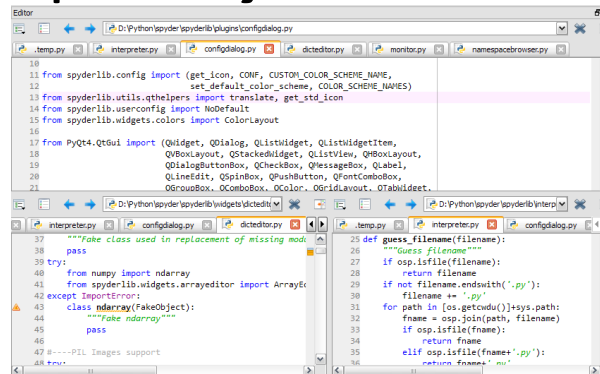
<https://www.jetbrains.com/pycharm/>

Introdução

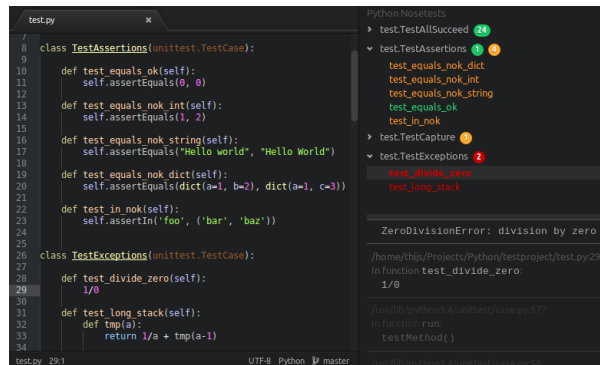
- Qual é o *ambiente de desenvolvimento* para linguagem Python?

- *Outras IDE para Python*

- *Spyder*



- *Atom*



Introdução

- Qual é o *ambiente de desenvolvimento* para linguagem Python?

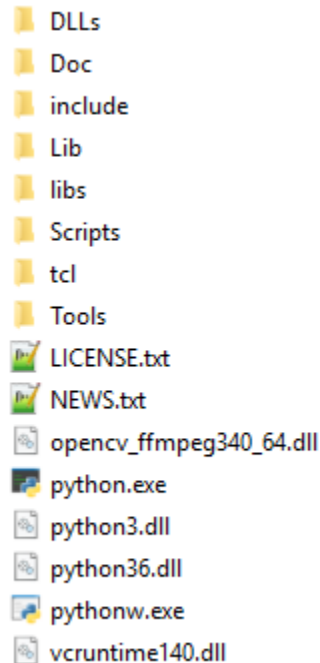
– *Outras IDE* para *Python*

Environment	Description	Web Site
IDLE	The standard Python environment	http://www.python.org/idle
Pythonwin	Windows-oriented environment	http://www.python.org/download/windows
ActivePython	Feature-packed; contains Pythonwin IDE	http://www.activestate.com
Komodo	Commercial IDE	http://www.activestate.com ³
Wingware	Commercial IDE	http://www.wingware.com
BlackAdder	Commercial IDE and (Qt) GUI builder	http://www.thekompany.com
Boa Constructor	Free IDE and GUI builder	http://boa-creator.sf.net
Anjuta	Versatile IDE for Linux/UNIX	http://anjuta.sf.net
Arachno Python	Commercial IDE	http://www.python-ide.com
Code Crusader	Commercial IDE	http://www.newplanetsoftware.com
Code Forge	Commercial IDE	http://www.codeforge.com
Eclipse	Popular, flexible, open source IDE	http://www.eclipse.org
eric	Free IDE using Qt	http://eric-ide.sf.net
KDevelop	Cross-language IDE for KDE	http://www.kdevelop.org
VisualWx	Free GUI builder	http://visualwx.altervista.org
wxDesigner	Commercial GUI builder	http://www.roebling.de
wxGlade	Free GUI builder	http://wxglade.sf.net

Introdução

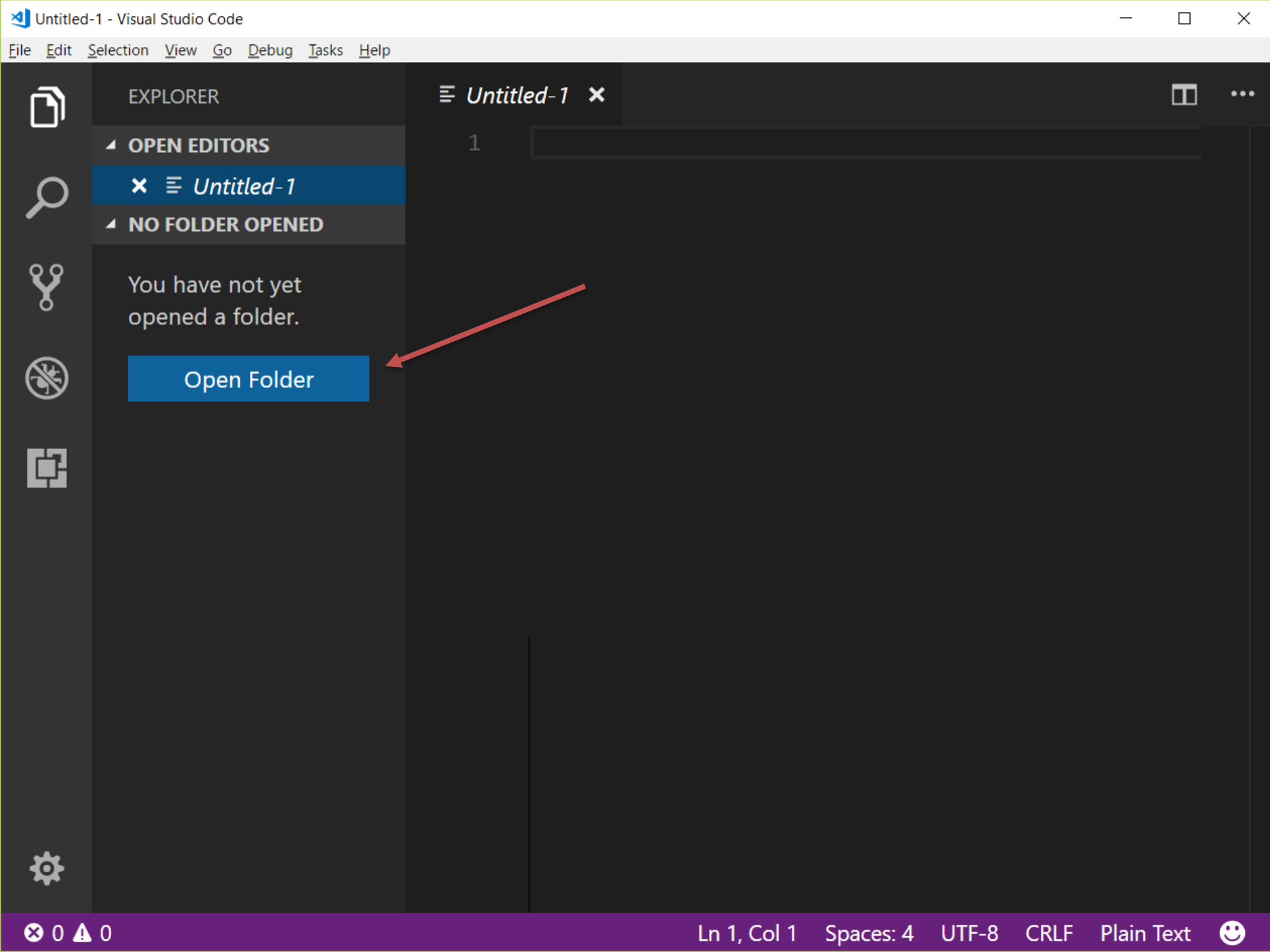
- Qual é o diretório de instalação do ***Python***?

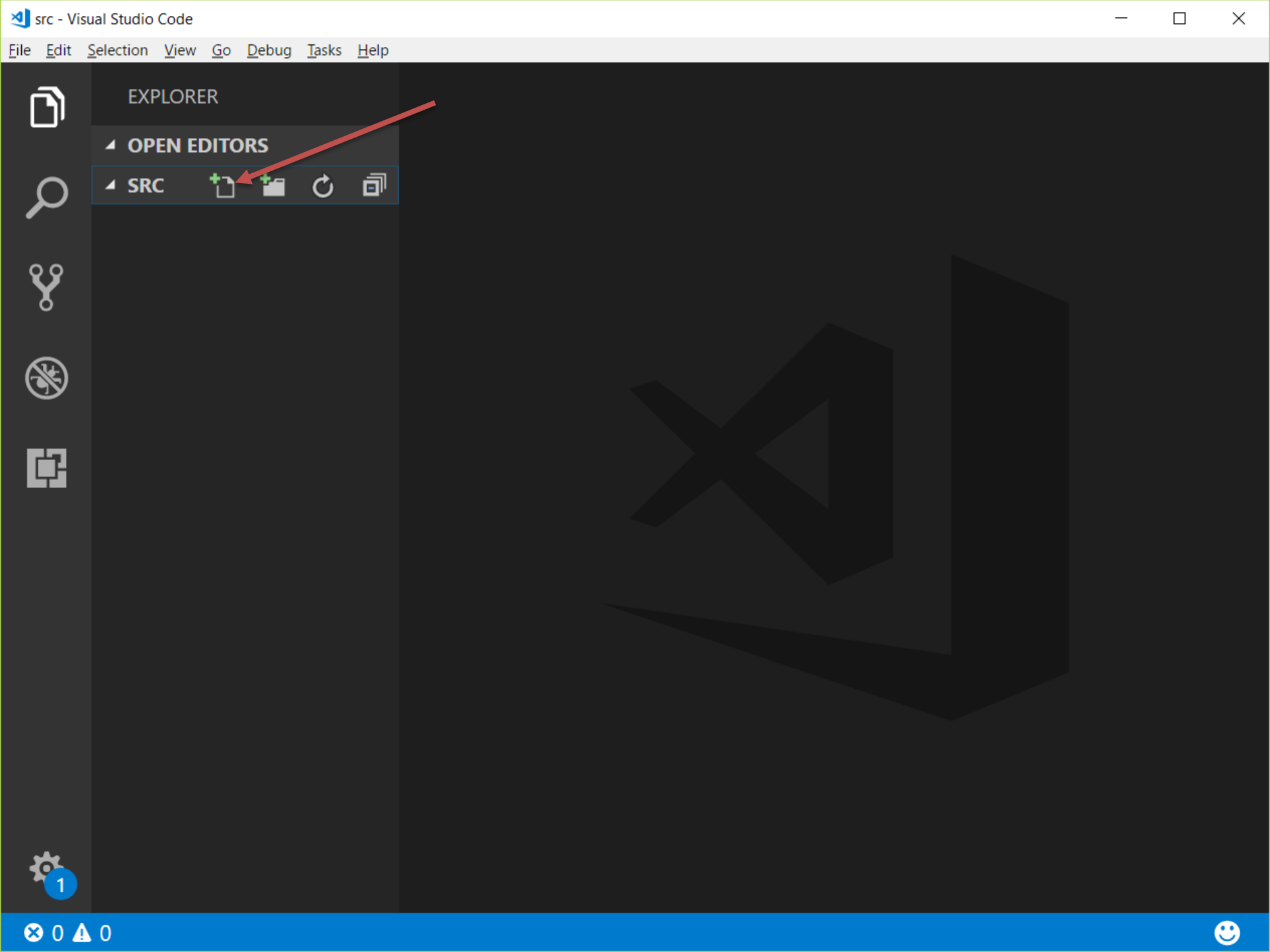
C:\Users\João da Silva\AppData\Local\Programs

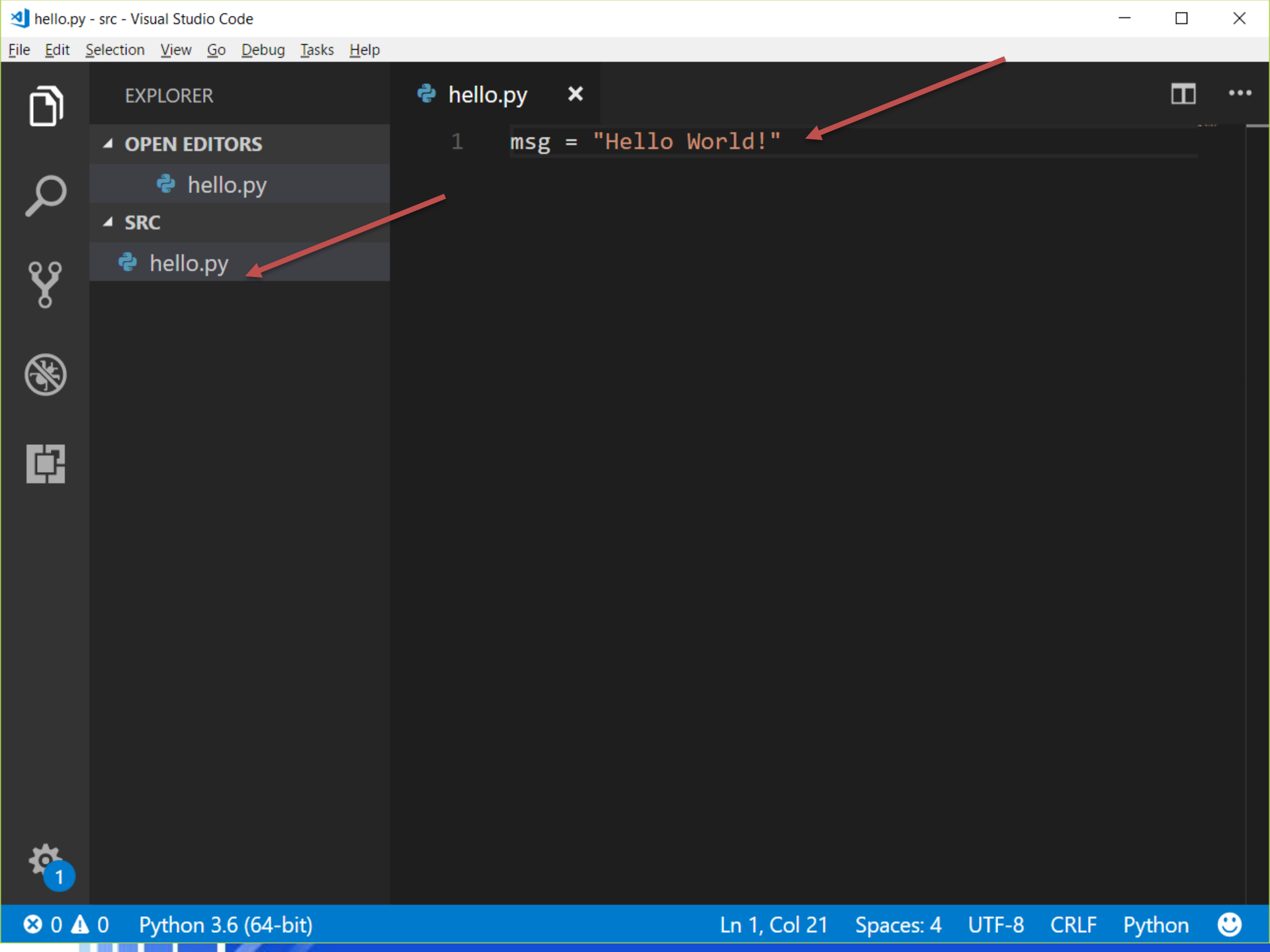


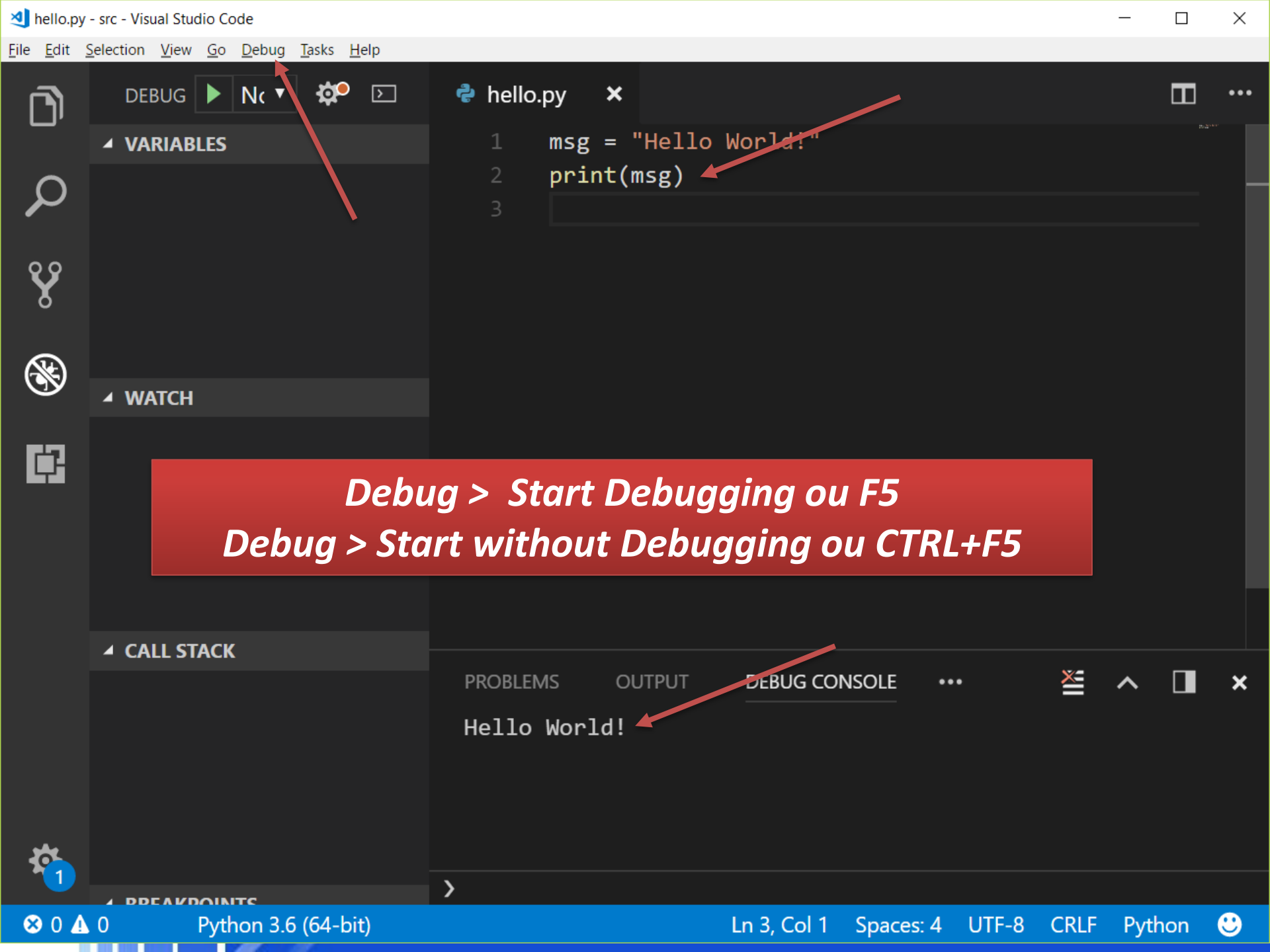


VISUAL STUDIO CODE









Debug > Start Debugging ou F5
Debug > Start without Debugging ou CTRL+F5

DEBUG Python: Termin

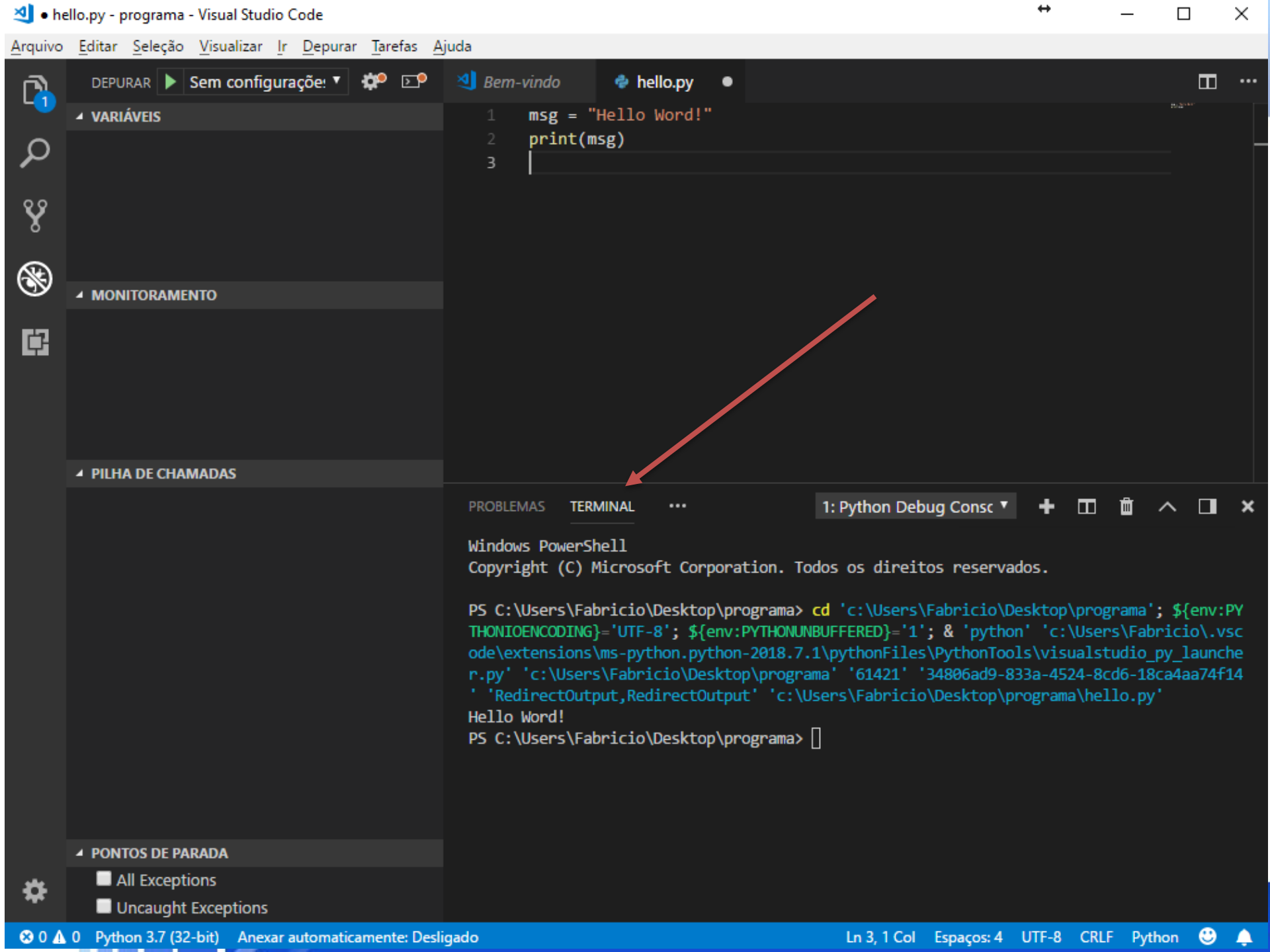
Python
Python: Attach
Python: Terminal (integrated)
Python: Terminal (external)
Python: Django
Python: Flask (0.11.x or later)
Python: Flask (0.10.x or earlier)
Python: PySpark
Python: Module
Python: Pyramid
Python: Watson
Add Configuration...

```
1 {  
2 // Use IntelliSense to learn about  
3 // Hover to view descriptions of ex  
4 // For more information, visit: https://code.visualstudio.com/docs/python/launch-configuration  
5 "version": "0.2.0",  
6 "configurations": [  
7  
8 {  
9     "name": "Python",  
10    "type": "python",  
11    "request": "launch",  
12    "stopOnEntry": true,  
13    "pythonPath": "${config:python.pythonPath}",  
14    "program": "${workspaceFolder}/hello.py",  
15    "cwd": "${workspaceFolder}"  
16 }  
17 ]  
18 }
```

View > Integrated Terminal

Hello World!

0 0 Python Python 3.6 (64-bit) Ln 2, Col 5 Spaces: 4 UTF-8 LF JSON





PYTHON



- ***Consultando a Documentação***

```
>>> help()
```

```
Welcome to Python 3.6's help utility!
```

```
>>> 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.
```

- Usando o *Interpretador interativo*

```
>>> 2+2
4
>>> 9587*365
3499255
>>> 1/2
0.5
>>> 1//2
0
>>> 5%2
1
>>> 2**3
8
```

- ***Palavras reservadas***

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

- ***Comentários***

- Múltiplas linhas `"""` ... `"""`
- Uma linha `#`

```
"""
```

```
Este é um exemplo de comentário  
de múltiplas  
linhas
```

```
"""
```

```
# comentário de 1 (uma) linha
```

- ***Variáveis***

- Uma variável é um ***nome de identificar*** que representa algum valor associado.

```
#declaração de variáveis  
x = 10  
X = 20  
  
print(x)  
print(X)
```

- ***Tipos de Dados***

- ***Tipagem dinâmica***

- O tipo ao qual a variável está associado pode variar durante a execução do programa.
 - Não quer dizer que não exista tipo específico, embora não é necessário declarar ***explicitamente***.

```
1  a = 1
2  print(type (a))
3  a = "1"
4  print(type (a))
5  a = 1.0
6  print(type (a))
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

<class 'int'>

<class 'str'>

<class 'float'>

- ***Tipos de Dados***
 - ***Numérico : Integer***
 - Inteiros de 32 bits

```
1  # tipo de dados inteiro
2  x = 2
3  y = 3
4  print(x+y)
5
6  # x elevado a y
7  print(pow(x,y))
8
9  # arredondamento
10 print(round(x/y))
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

5

8

1

Fundamentos

- ***Tipos de Dados***

- ***Numérico : Integer***

- Python promove ***int*** para ***long*** automaticamente

```
12  # conversões
13  x = 12
14  print("binário = " + bin(x))
15  print("hexa    = " + hex(x))
16  print("octal   = " + oct(x))
17
18  y = "25"
19  z = 3 + int(y)  #string para int
20  print(z)
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

binário = 0b1100

hexa = 0xc

octal = 0o14

28

- ***Tipos de Dados***
 - ***Numérico : Booleano***
 - Verdadeiro ou falso

```
27     a = True
28     b = False
29
30     print(a and b)
31     print(a or b)
32
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

False

True

- ***Tipos de Dados***
 - ***Ponto-Flutuante : float***
 - 32 bits

```
34 import math
35
36 i = 2.56
37 j = 3.66
38 k = i+j
39 print(k)
40
41 print(math.ceil(k))      #arredondamento para cima
42 print(math.floor(k))     #arredondamento para baixo
43 print(round(k,2))        #arredondamento duas casas
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

6.2200000000000001

7

6

6.22

Módulo *Math*

<code>math.acos(x)</code>	Returns the arc cosine of x in radians
<code>math.acosh(x)</code>	Returns the arc hyperbolic cosine of x in radians
<code>math.asin(x)</code>	Returns the arc sine of x in radians
<code>math.asinh(x)</code>	Returns the arc hyperbolic sine of x in radians
<code>math.atan(x)</code>	Returns the arc tangent of x in radians
<code>math.atan2(y, x)</code>	Returns the arc tangent of y / x in radians
<code>math.atanh(x)</code>	Returns the arc hyperbolic tangent of x in radians
<code>math.ceil(x)</code>	Returns $\lceil x \rceil$, i.e., the smallest integer greater than or equal to x as an int; e.g., <code>math.ceil(5.4) == 6</code>
<code>math.copysign(x, y)</code>	Returns x with y 's sign
<code>math.cos(x)</code>	Returns the cosine of x in radians
<code>math.cosh(x)</code>	Returns the hyperbolic cosine of x in radians
<code>math.degrees(r)</code>	Converts float r from radians to degrees
<code>math.e</code>	The constant e ; approximately 2.718281828 459 045 1
<code>math.exp(x)</code>	Returns e^x , i.e., <code>math.e ** x</code>
<code>math.fabs(x)</code>	Returns $ x $, i.e., the absolute value of x as a float
<code>math.factorial(x)</code>	Returns $x!$
<code>math.floor(x)</code>	Returns $\lfloor x \rfloor$, i.e., the largest integer less than or equal to x as an int; e.g., <code>math.floor(5.4) == 5</code>
<code>math.fmod(x, y)</code>	Produces the modulus (remainder) of dividing x by y ; this produces better results than <code>%</code> for floats
<code>math.frexp(x)</code>	Returns a 2-tuple with the mantissa (as a float) and the exponent (as an int) so, $x = m \times 2^e$; see <code>math.ldexp()</code>
<code>math.fsum(i)</code>	Returns the sum of the values in iterable i as a float
<code>math.hypot(x, y)</code>	Returns $\sqrt{x^2 + y^2}$
<code>math.isinf(x)</code>	Returns True if float x is $\pm \text{inf}$ ($\pm \infty$)
<code>math.isnan(x)</code>	Returns True if float x is nan (“not a number”)
<code>math.ldexp(m, e)</code>	Returns $m \times 2^e$; effectively the inverse of <code>math.frexp()</code>
<code>math.log(x, b)</code>	Returns $\log_b x$; b is optional and defaults to <code>math.e</code>
<code>math.log10(x)</code>	Returns $\log_{10} x$
<code>math.log1p(x)</code>	Returns $\log_e(1 + x)$; accurate even when x is close to 0
<code>math.modf(x)</code>	Returns x 's fractional and whole parts as two floats

Módulo *Math*

<code>math.pi</code>	The constant π ; approximately 3.141 592 653 589 793 1
<code>math.pow(x, y)</code>	Returns x^y as a float
<code>math.radians(d)</code>	Converts float d from degrees to radians
<code>math.sin(x)</code>	Returns the sine of x in radians
<code>math.sinh(x)</code>	Returns the hyperbolic sine of x in radians
<code>math.sqrt(x)</code>	Returns \sqrt{x}
<code>math.tan(x)</code>	Returns the tangent of x in radians
<code>math.tanh(x)</code>	Returns the hyperbolic tangent of x in radians
<code>math.trunc(x)</code>	Returns the whole part of x as an int; same as <code>int(x)</code>

- ***Tipos de Dados***
 - ***Ponto-Flutuante : números complexos***

```
47  z = -89.5+2+125j
48  print(z.real)
49  print(z.imag)
50
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

-87.5

125.0

- ***Tipos de Dados***

- ***Ponto-Flutuante : números decimais***

- Fornece uma maior acurácia para cálculos com números ponto-flutuante
 - As operações são mais lentas comparado ao *float*

```
51  import decimal
52  x = 23
53  y = 1.05
54  d = decimal.Decimal(23) / decimal.Decimal("1.05")
55
56  print(x/y)
57  print(d)
```

- ***Tipos de Dados***

- ***Strings***

- Sequencia de caracteres Unicode
 - Definidas por
 - Aspas simples
 - Aspas duplas
 - Aspas triplas
 - Cada caractere é um byte
 - A acentuação depende do *encoding*

- ***Tipos de Dados***
 - ***Strings***

```
1 s1 = 'Olá \"Mundo\"!!!'
2 s2 = "Linguagem de programação 'Python'"
3 s3 = 'Linguagem de programação "Python"'
4 s4 = """Lorem Ipsum is
5 simply dummy text
6 of the printing and
7 typesetting industry."""
8 print(s1)
9 print(s2)
10 print(s3)
11 print(s4)
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
Olá "Mundo"!!!
Linguagem de programação 'Python'
Linguagem de programação "Python"
Lorem Ipsum is
simply dummy text
of the printing and
typesetting industry.
```


- ***Tipos de Dados***
 - ***Strings - Códigos de barra invertida***

Sequence	Meaning
\n	Newline (LF)
\r	Carriage return (CR)
\t	Tab
\v	Vertical tab (VT)
\e	Escape character (ESC)
\f	Formfeed (FF)
\b	Backspace
\a	Bell (BEL)

- ***Tipos de Dados***
 - ***Strings*** – Indexação por colchetes

```
1  s = "João da Silva"
2  print(s[0])
3  print(s[1])
4  print(s[:4])
5  print(s[5:7])
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

J

o

João

da

- ***Tipos de Dados***
 - ***Strings*** – Operações com sequências

```
5   s = "aeiou"
6
7   #operações com sequências
8   print(s[0])           #acessar item
9   print(s[-2])          #acessar item pelo fim
10  print(s[0]*20)         #cópias concatenadas
11  print("a" in s)        #teste inclusão
12  print("x" not in s)    #teste inclusão negativo
```

- ***Tipos de Dados***
 - ***Strings*** – Fatiamento de sequências

```
15 s = "aeiou"
16 #fatiamento de sequências
17 print(s[0:2])    #s[a:b]    cópia de a(inclusive) até b(exclusive)
18 print(s[2:])     #s[a:]     cópia a partir de a(inclusive)
19 print(s[:2])     #s[:b]     cópia até b(exclusive)
20 print(s[:])      #s[:]      cópia total
21
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

2: Python Debu ▼



```
ae
iou
ae
aeiou
```

Fundamentos

- ***Tipos de Dados***
 - ***Strings*** – Funções auxiliares

```
24 s = "Abcdefg  "  
25 print(chr(65))           #retorna o caractere a partir do código  
26 print(ord(s[0]))         #retorna a código a partir do caractere  
27 print(repr(s))           #conversão de objeto para representação explícita  
28 print(len(s))            #retorna o número de bytes da string  
29 print(len(s.strip()))    #retira os brancos (espaços, tabs e newlines)  
30 print(s.upper())         #maiúsculas  
31 print(s.lower())         #minúsculas  
32 print(s.capitalize())    #primeira maiúscula por palavra  
33
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

2: Python Debu ▾



```
A  
65  
'Abcdefg  '  
10  
7  
ABCDEFGG  
abcdefg  
Abcdefg
```

- ***Conversões***

- Transforma o valor de um tipo em outro.

`int(x)` converts number `x` to an integer.

`float(x)` converts number `x` to a float.

`str(object)` converts more or less anything into a printable string.

Fundamentos

- *Operadores Aritméticos*

```
1  a = 2
2  b = 3
3  print(a+b)      #adição
4  print(a-b)      #subtração
5  print(a*b)      #multiplicação
6  print(a/b)      #divisão
7  print(a//b)     #divisão inteira
8  print(a%b)      #resto da divisão
9  print(a**b)     #potência
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
5
-1
6
0.6666666666666666
0
2
8
```

- ***Operadores Relacionais***

Função	Caractere(s) utilizado(s)	Exemplo
Igual	<code>==</code>	<code>x == y</code>
Diferente	<code>!=</code>	<code>x != y</code>
Maior que	<code>></code>	<code>x > y</code>
Maior ou igual a	<code>>=</code>	<code>x >= y</code>
Menor que	<code><</code>	<code>x < y</code>
Menor ou igual a	<code><=</code>	<code>x <= y</code>

– Operadores adicionais: ***is*** e ***is not***

Fundamentos

- *Operadores Relacionais*

```
1  a = 10
2  b = 20
3  print( a == b)
4  print(a != b)
5  print(a > b)
6  print(a >= b)
7  print(a < b)
8  print(a <= b)
9
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

False

True

False

False

True

True

- ***Operadores Lógicos***
 - *E lógico* ou **AND**
 - *Ou lógico* ou **OR**

```
1  v1 = True
2  v2 = False
3  print(v1 and v2)
4  print(v1 or v2)
5
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

False

True

Fundamentos

- ***Entrada e Saída de Dados***

- ***input***

- ***print***

```
1  # limpar o console
2  import os
3  os.system('cls')
4
5  #entrada de dados
6  num1 = float(input("Informe o primeiro valor: "))
7  num2 = float(input("Informe o segundo valor : "))
8
9  soma = (num1+num2)
10
11 #saída de dados
12 print("Soma = %.2f" %soma )
13
```

Expressão	Significado
%s	Uma string
%c	Caractere
%d	Decimal, inteiro
%f	Real (float)
%%	Um '%'

PROBLEMS

TERMINAL

...

2: Python Debu ▾



Informe o primeiro valor: 12.55

Informe o segundo valor : 31.94

Soma = 44.49

Fundamentos

- ***Entrada e Saída de Dados***
 - Exemplo

```
1  # limpar o console
2  import os
3  os.system('cls')
4
5  nome = input("Qual o seu nome? ")
6  print("Olá, ", nome, " seja bem-vindo!")
7
8  msg = "Olá, {} seja bem-vindo".format(nome)
9  print(msg)
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

Qual o seu nome? João da Silva

Olá, João da Silva seja bem-vindo!

Olá, João da Silva seja bem-vindo

- ***Entrada e Saída de Dados***
 - Exemplo

```
2  import os
3  os.system('cls')
4
5  n1 = int(input("Entre com o primeiro número: "))
6  n2 = int(input("Entre com o segundo número : "))
7  res = "{0} + {1} = {2}".format(n1,n2,(n1+n2))
8  print(res)
9
10
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

Entre com o primeiro número: 25

Entre com o segundo número : 62

25 + 62 = 87

- ***Entrada e Saída de Dados***
 - Exemplo

```
8   str1 = "João da Silva"  
9   for c in str1:  
10      print(c,end=' ')  
11
```

DEBUG CONSOLE

J o ã o d a S i l v a

ATIVIDADE PRÁTICA

- ***Exercício 1***

- Crie uma aplicação que receba o valor da base e da altura de um triângulo retângulo e apresente na tela sua área.

- ***Exercício 2***

- Crie um programa para calcular e exibir na tela o peso ideal. $IMC = (peso / (altura^2))$

- ***Exercício 3***

- Uma farmácia precisa ajustar os preços de seus produtos em 12%. Elabore uma aplicação que receba o valor do produto e aplique o percentual de acréscimo.
- O novo valor a ser calculado deve ser arredondado e apresentado com duas casas decimais.