Programação com Python

Download Python 3.14.0

https://www.python.org/downloads/

Programação com Python

Python Practice Book
https://anandology.com/python-practice-book/

w3schools – Python Tutorial
https://www.w3schools.com/python/

Python Online
https://www.onlinegdb.com

Fundamentos do Python
https://www.netcad.com

Programação com Python

Linguagem de programação criada em 1991 por Guido van Rossum (https://en.wikipedia.org/wiki/Guido_van_Rossum)

Versões:

- **Python 1.x** (1994-2000) – primeiras versões.
- **Python 2.x** (2000-2020 – versões com atualizações significativas e compatíveis com as versões 1.x; a versão mais usada foi a 2.7 que terminou em 2020.
- **Python 3.x** (2008-atualidade) – grande revisão da linguagem não havendo compatibilidade com as versões anteriores; a versão mais recente é a 3.14.

Programação com Python

Documentação: https://docs.python.org/pt-br/3/index.html

Integrated Development and Learning Environment

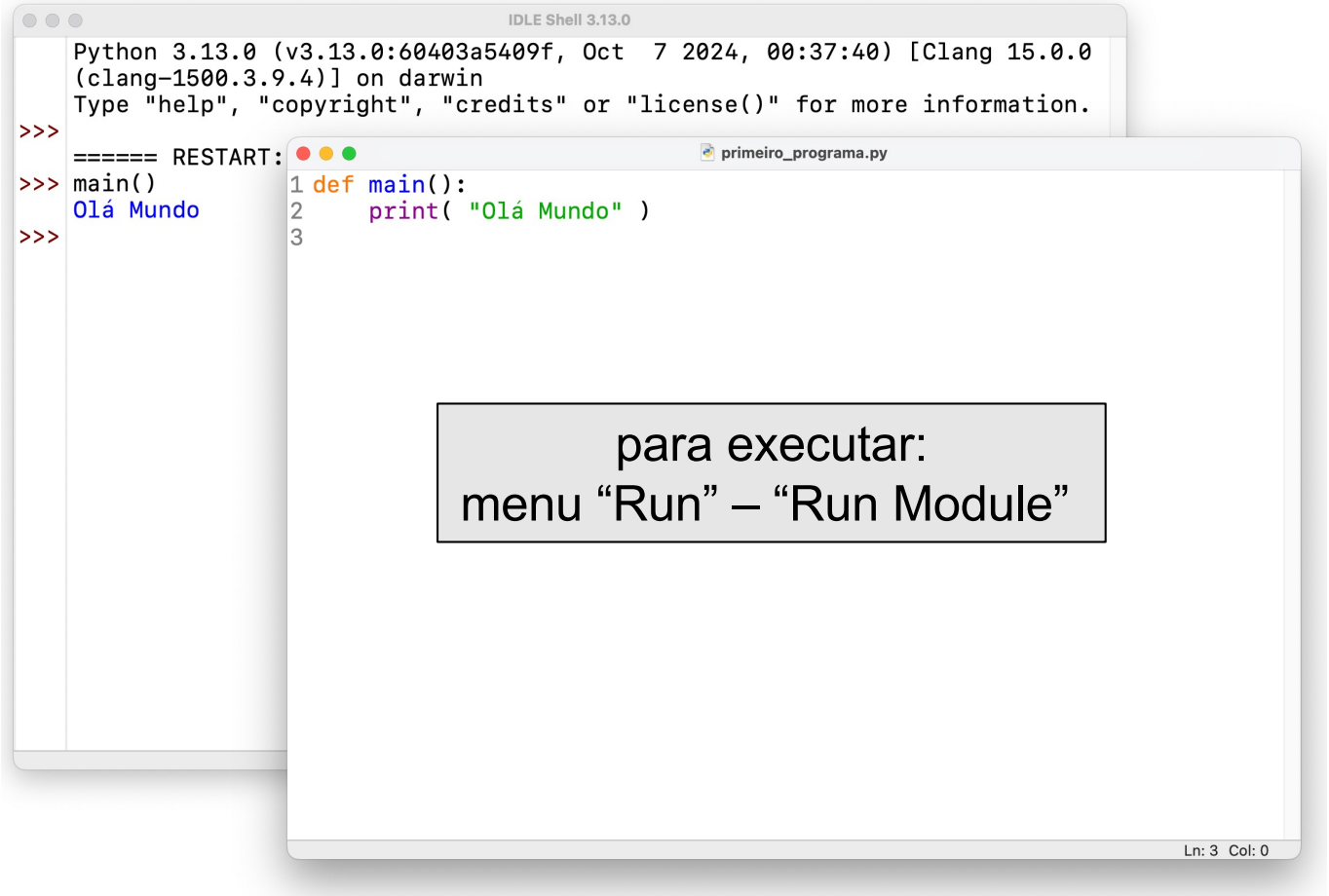
Funciona com dois tipos de janelas:

- *Editor window*
- *Shell window*

Programação com Python

https://docs.python.org/pt-br/3/library/idle.html#

shell window



editor window

Programação com Python

Aspetos a ter em conta antes de começar a programar em Python:

- A linguagem tem tipos de dados mas não é necessário declarar previamente as variáveis; as variáveis ficam com um tipo atribuído em função do contexto em que são usadas pela primeira vez.

- A linguagem não obriga a uma estruturação rígida do programa nem é obrigatória a existência de um "programa principal" como noutras linguagens; no entanto, deve-se procurar manter uma estrutura para facilitar a organização do código-fonte.

- A indentação (afastamento do texto em relação à margem) das instruções é relevante pois é a única forma de constituir instruções compostas e de manter uma hierarquia entre instruções; uma instrução composta contém instruções simples que partilham a mesma indentação.

Programação com Python

O Python não obriga a seguir esta estrutura mas é aconselhável fazê-lo

```
# estrutura de um programa em Python

'''
    colocar subprogramas
    antes do programa principal
'''
```

simula a existência de um "programa principal"

```
def main():
    '''
    colocar aqui as instruções
    do programa principal
    '''
```

testa se o módulo foi executado na linha de comando ou se foi importado por outro módulo

```
if __name__ == '__main__':
    main()
```

Programação com Python

Operadores Aritméticos

Operator	Symbol
addition:	+
subtraction:	-
multiplication:	*
exponentiation:	**
division:	/
integer division:	//
modulo (remainder):	%

Two of Python's numeric types

`int`: integer

`float`: floating point number
an approximation to a real number

prioridades

Operator precedence

**	highest precedence
- (negation)	
* / // %	
+ -	lowest precedence

Syntax: the rules that describe valid combinations of Python symbols

Semantics: the meaning of a combination of Python symbols

Programação com Python

Operadores Relacionais

Comparison Operator	Symbol
less than:	<
greater than:	>
equal to:	==
greater than or equal to:	>=
less than or equal to:	<=
not equal to:	!=

Operadores Lógicos

Logical Operator	Symbol
not:	not
and:	and
or:	or

Tabelas de Verdade

negação (not)
conjunção (and)
disjunção (or)

expr	not *expr*
True	False
False	True

expr1	*expr2*	*expr1* and *expr2*
True	True	True
True	False	False
False	True	False
False	False	False

expr1	*expr2*	*expr1* or expr2
True	True	True
True	False	True
False	True	True
False	False	False

Instruções Condicionais

```
if expressão:
    instrução1
else:
    instrução2
```

```
def main():
    n = int(input( "insira um numero: " ))

    if n%2 == 0:
        print("Par")
    else:
        print("Impar")

if __name__ == '__main__':
    main()
```

A indentação dos blocos de instruções é obrigatória! Em Python não se usam chavetas (como em Java) sendo as instruções compostas criadas através de **conjuntos de instruções com a mesma indentação**.

Indentação de Instruções

As instruções seguintes estão erradas:

```
10 if __name__ == '__main__':
11 main()
```


```
if n%2==0:
print("Par")
else:
print("Ímpar")
```

```
def main():
n = int(input("insira um número: "))
```

A indentação de instruções deve ser feita com a tecla “tab”

Instruções Condicionais – if's encadeados - elif

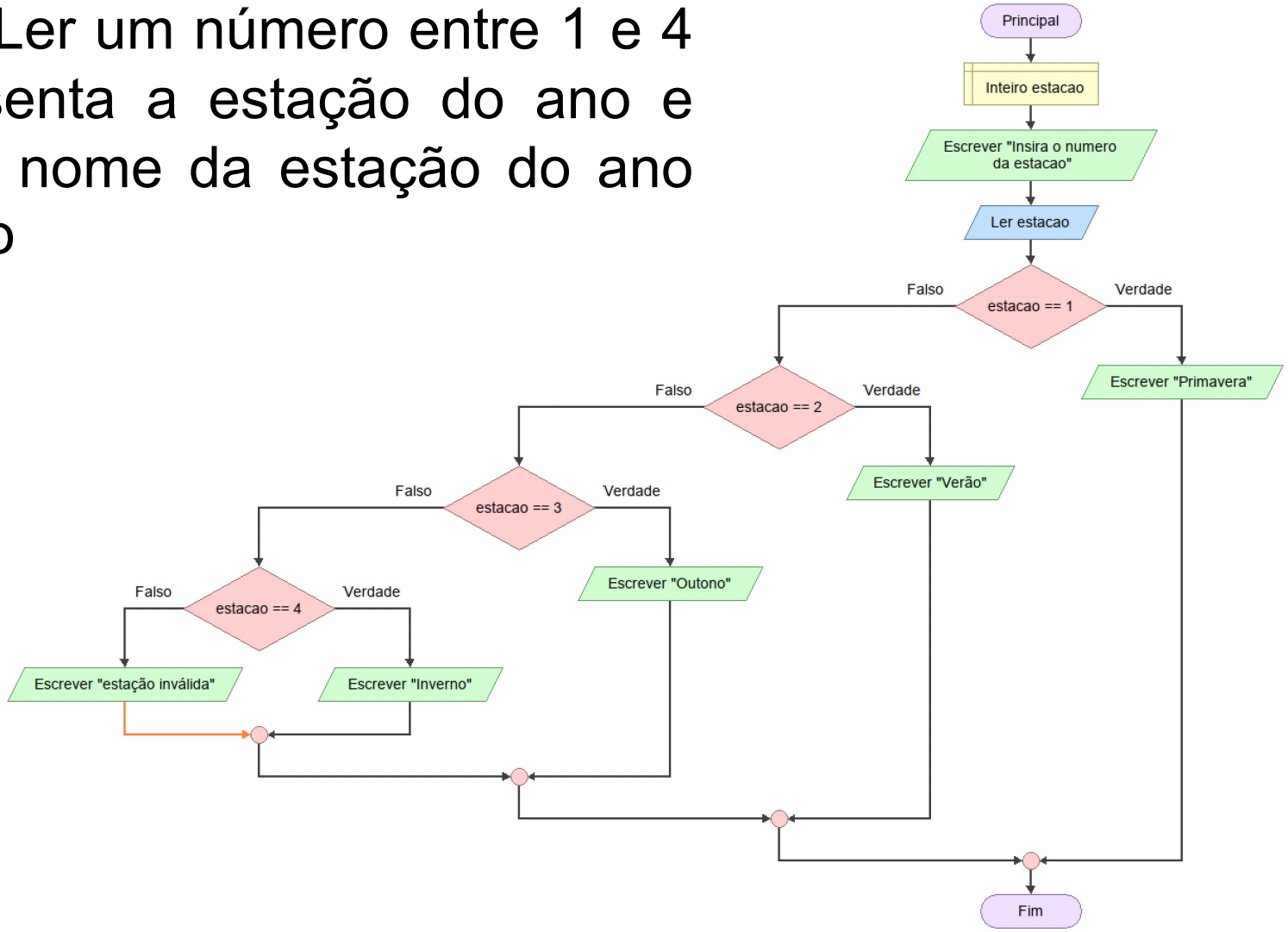
```
if expressão1:
    instrução1
elif expressão2:
    instrução2
else:
    instrução3
```

```
def main():
    n = int(input( "insira um numero: " ))

    if n == 0:
        print("Zero")
    elif n%2 == 0:
        print("Par")
    else:
        print("Impar")

if __name__ == '__main__':
    main()
```

Exemplo: Ler um número entre 1 e 4 que representa a estação do ano e escrever o nome da estação do ano por extenso



Exemplo – programa em Python

Código Python gerado pelo Flowgorithm

```
print("Insira o numero da estacao")
estacao = int(input())
if estacao == 1:
    print("Primavera")
else:
    if estacao == 2:
        print("Verão")
    else:
        if estacao == 3:
            print("Outono")
        else:
            if estacao == 4:
                print("Inverno")
            else:
                print("estação inválida")
```

Exemplo

Código Python com a estrutura recomendada

```
def main():
    print("Insira o numero da estacao")
    estacao = int(input())
    if estacao == 1:
        print("Primavera")
    else:
        if estacao == 2:
            print("Verão")
        else:
            if estacao == 3:
                print("Outono")
            else:
                if estacao == 4:
                    print("Inverno")
                else:
                    print("estação inválida")

if __name__ == '__main__':
    main()
```

Exemplo

Código Python com a estrutura recomendada e uso de `elif's`

```
def main():
    print("Insira o numero da estacao")
    estacao = int(input())
    if estacao == 1:
        print("Primavera")
    elif estacao == 2:
        print("Verão")
    elif estacao == 3:
        print("Outono")
    elif estacao == 4:
        print("Inverno")
    else:
        print("estação inválida")

if __name__ == '__main__':
    main()
```

Exemplo

escrever um programa em python para fazer o seguinte: Ler um número entre 1 e 4 que representa a estação do ano e escrever o nome da estação do ano por extenso, sabendo que 1 corresponde a "Primavera"

Na primeira resposta, o programa incluía um dicionário (estrutura de dados do Python ainda não estudada)

repita o programa anterior mas sem usar um dicionário

Claro! Aqui está o programa sem o uso de um dicionário, usando apenas estruturas condicionais `if`:

Exemplo


```python
# Verifica o número e imprime a estação correspondente
if numero == 1:
    print("A estação do ano é: Primavera")
elif numero == 2:
    print("A estação do ano é: Verão")
elif numero == 3:
    print("A estação do ano é: Outono")
elif numero == 4:
    print("A estação do ano é: Inverno")
else:
    print("Número inválido! Por favor, insira um número entre 1 e 4.")
```

Explicação do código

- O programa solicita ao usuário um número entre 1 e 4.
- Em seguida, ele usa uma série de instruções `if` para verificar qual estação corresponde ao número digitado.
- Caso o número não esteja entre 1 e 4, uma mensagem de erro é exibida.

Instruções Condicionais

seleção de múltipla escolha – *structural pattern matching*

Apenas a partir da versão 3.10 !

instrução `match...case`:

```
match subject:
    case <pattern_1>:
        <action_1>
    case <pattern_2>:
        <action_2>
    case <pattern_3>:
        <action_3>
    case _:
        <action_wildcard>
```

```
def main():
    estacao = int( input("qual a estação? "))
    match estacao:
        case 1:
            print("primavera")
        case 2:
            print("verão")
        case 3:
            print("outono")
        case 4:
            print("inverno")
        case _:
            print("estação inválida")

if __name__ == '__main__':
    main()
```

Instruções Condicionais

Apenas a partir da versão 3.10 !

seleção de múltipla escolha – *structural pattern matching*

instrução `match...case`:

Podem-se combinar vários valores no mesmo `case`, separados por `|` (“or”)

```
def main():
    n = int(input("insira um número: "))
    match n:
        case 1 | 3 | 5 | 7 | 9:
            print("ímpar")
        case 2 | 4 | 6 | 8:
            print("par")

if __name__ == '__main__':
    main()
```

Instruções Cíclicas

```
while expressão:
    instrução
```

ler uma sequência de números inteiros, até ser lido o valor 0 ou um valor negativo. Apresentar no final o número de valores lidos.

```
def main():

    n = int(input("Insira um número: " ))
    c = 1
    while n > 0:
        n = int(input("Insira um número: " ))
        c = c + 1
    print("Total: ", c )

if __name__ == '__main__':
    main()
```

Instruções Cíclicas

- Apenas existe o equivalente ao ciclo `enquanto`
- Não existe instrução equivalente a `faca...enquanto`
- É possível simular um ciclo `faca...enquanto` usando um ciclo `while` combinado com a instrução `break`.

```
def main():

    # ciclo while - equivale a "enquanto..."
    n = int(input("Insira um número: " ))
    c = 1
    while n > 0:
        n = int(input("Insira um número: " ))
        c = c + 1
    print("Total: ", c )

    # ciclo while com break - equivale a "faca...enquanto"
    c = 0
    while True:
        n = int(input("Insira um número: " ))
        c = c + 1
        if n <= 0:
            break
    print("Total: ", c )


if __name__ == '__main__':
    main()
```

Instruções Cíclicas

```
for i in range( início, fim, passo):
    instrução
```

- Pode-se omitir o `início` (por defeito, é 0)
- Pode-se omitir o `passo` (por defeito, é 1)
- Podem-se usar valores negativos

Examples: of the use of range:

- `range(10)` produces the list: [0,1,2,3,4,5,6,7,8,9]
- `range(1, 7)` produces the list: [1,2,3,4,5,6]
- `range(0, 30, 5)` produces the list: [0,5,10,15,20,25]
- `range(5, -1, -1)` produces the list: [5,4,3,2,1,0]

Instruções Cíclicas

Outros exemplos com o ciclo for

```
frase = "programando em python"
for letra in frase:
    print( letra, end='' )
```

```
programando em python
```

```
print( letra, end='' )
```

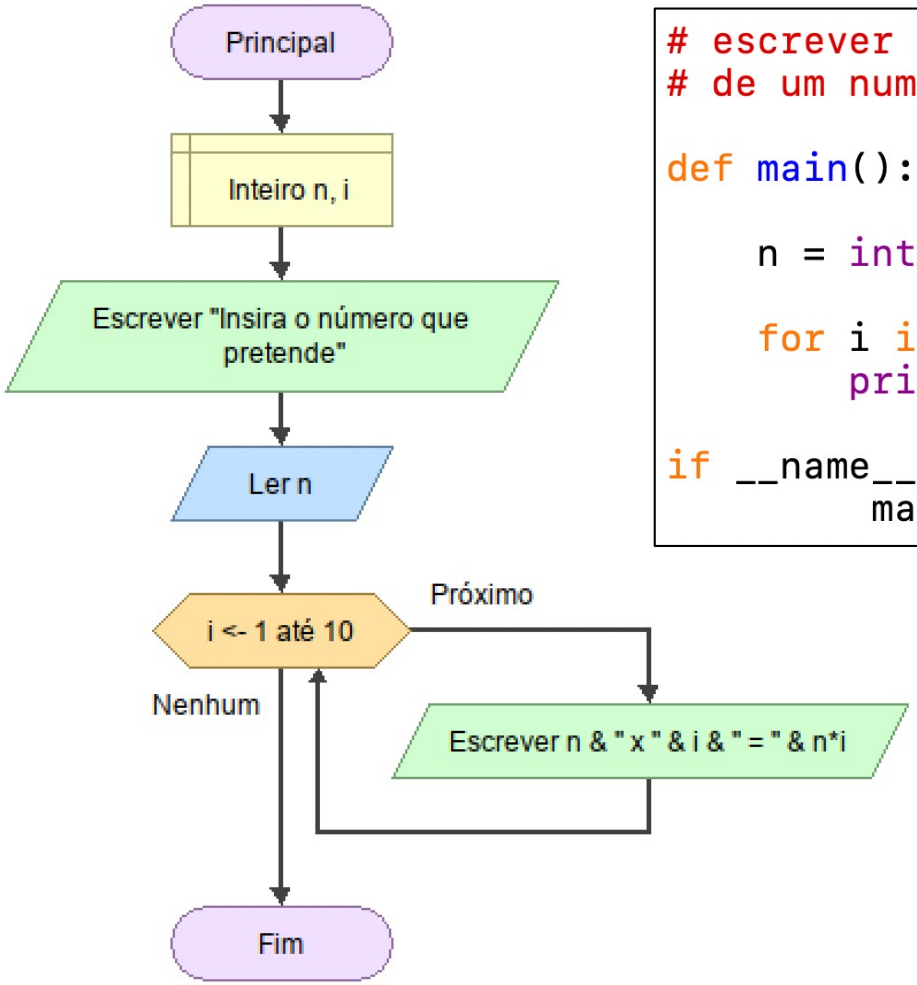
O parâmetro `end` define qual o carácter que é escrito após o `print`; por defeito, é o carácter de *newline* (`\n`)

```
for n in [1, 2, 3, 4, 5, 6, 7, 8, 9]:
    if n%2 == 0:
        print( n )
```

```
2
4
6
8
```

Programação com Python

Exemplo: Escrever a tabuada da multiplicação de um número inteiro.




```
# escrever a tabuada da multiplicacao
# de um numero escolhido pelo utilizador

def main():

    n = int(input("Insira o número que pretende: "))

    for i in range( 1, 11 ):
        print( n, " x ", i, " = ", n*i )

if __name__ == '__main__':
        main()
```

Em `range(1, 11)`, o 11 é exclusive, ou seja, a variável `i` assume valores de 1 até 10

Uso de funções (subprogramas)

```
def quadrado( n ):
    return n*n

def cubo( n ):
    return n**3

def main():
    x = int(input("Insira um número: "))
    print( "O quadrado de ", x, " é: ", quadrado(x) )
    print( "O cubo de ", x, " é: ", cubo(x) )

if __name__ == '__main__':
    main()
```

Uso de funções (subprogramas)

parâmetros formais por defeito:

```
def curso( c='"eletrónica"' ):
    print( "o nome do curso é", c )


def main():
    curso( '"informática"' )
    curso()
    curso( '"mecânica"' )

if __name__ == '__main__':
    main()
```

"eletrónica" é o valor por defeito, do parâmetro c

```
o nome do curso é "informática"
o nome do curso é "eletrónica"
o nome do curso é "mecânica"
```

Variáveis Locais e Variáveis Globais

```
x = 100         # global

def f1( x ):    # local
    print( x )

def f2( x ):    # local
    print( x )

def main():
    f1( 10 )
    f2( 50 )
    print( x )

if __name__ == '__main__':
    main()
```

```
10
50
100
```

Variáveis Locais e Variáveis Globais

```
x = 100         # global

def f1(x):      # local
    global y    # global
    y = 500
    print( x )

def f2( x ):    # local
    print( x )

def main():
    f1( 10 )
    f2( 50 )
    print( x )
    print( y )

if __name__ == '__main__':
    main()
```

A palavra reservada `global` transforma uma variável local numa variável global

```
10
50
100
500
```