



CINEL

Centro de Formação Profissional da Indústria Electrónica, Energia, Telecomunicações e Tecnologias da Informação

60h – Python

PROGRAMACAO EM PYTHON - INICIAÇÃO

julio.magalhaes@formandos.cinel.pt



Objetivos

- Desenvolver raciocínio lógico.
- Desenvolver algoritmos.
- Introdução à linguagem Python:
 - Lógica de programação em Python
 - Operadores aritméticos e lógicos
 - Estruturas de controlo
 - Funções e bibliotecas
 - Diferentes tipo de armazenamento de dados



Lógica de programação

- Lógica: técnica de encadear pensamentos para atingir determinado objetivo
- Sequência lógica: passos executados para atingir um objetivo
- Instruções: conjunto de regras ou normas definidas com um determinado objetivo a atingir



Algoritmo

- Constituído por um conjunto de expressões simbólicas que representam
 - ações (escolher, atribuir, ...),
 - testes de condições (estruturas condicionais) e
 - estruturas de controlo (ciclos na estrutura sequencial do algoritmo)

de modo a especificar o problema e respetiva solução

- Um programa de computador não existe sem um algoritmo associado.

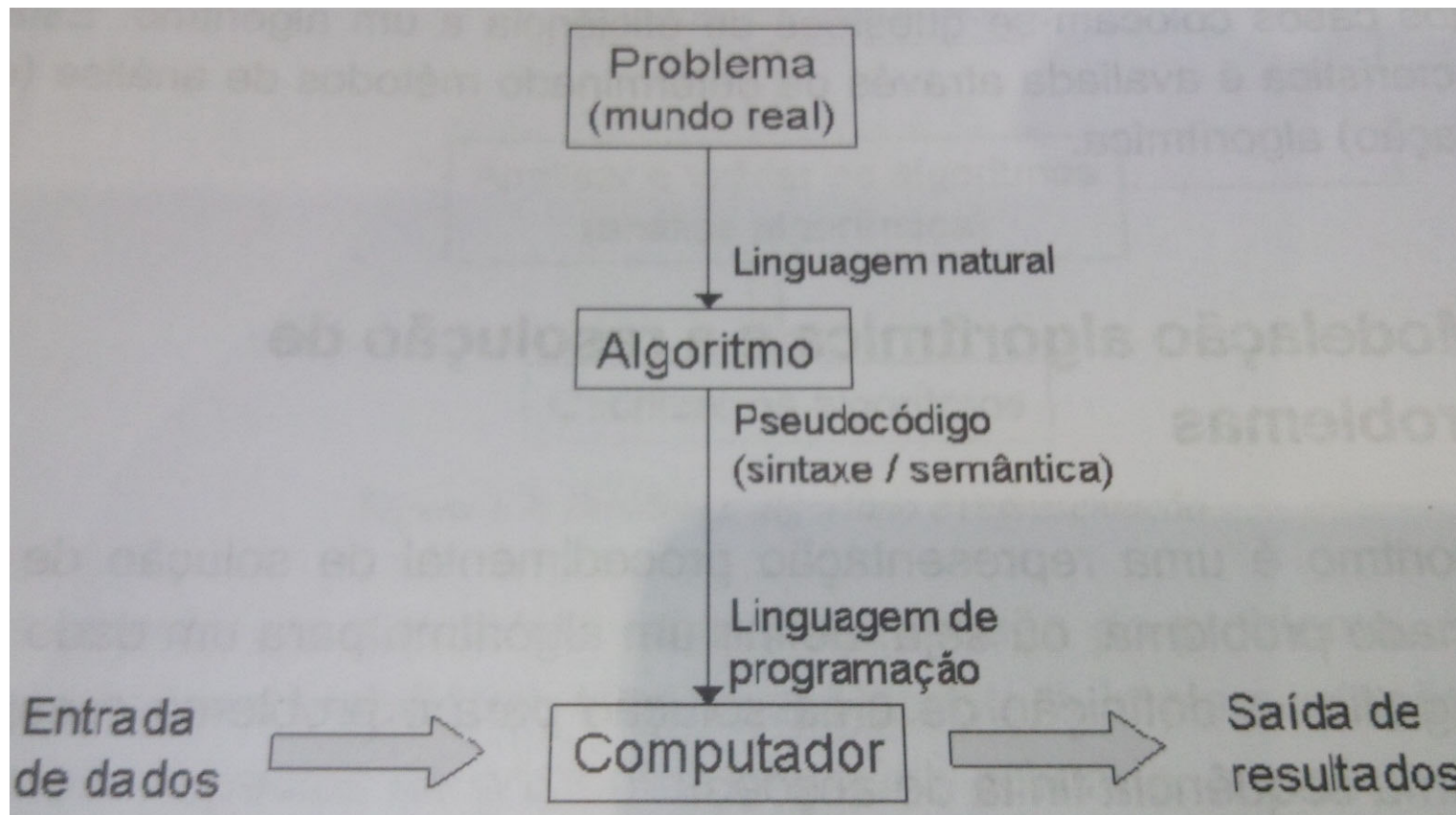


Características de um algoritmo

- Representa uma sequência finita e não ambígua de instruções.
 - Objetivo será obter a solução do problema (output) tendo por base uma entrada prévia de dados (input)
- É representado através de uma “linguagem” com sintaxe própria e semântica associada
- Deve ser eficaz -> obter o melhor desempenho (performance) possível



Características de um algoritmo

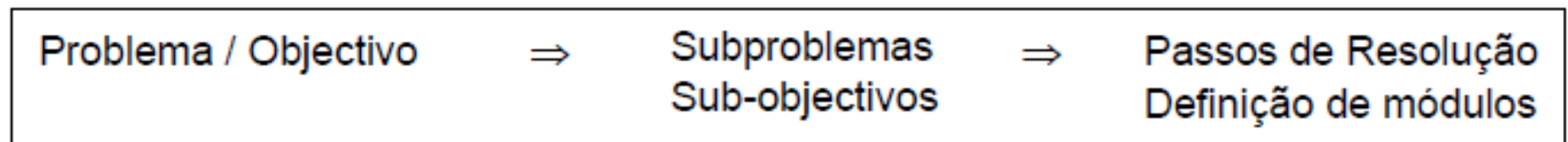




Resolução de problemas

- Abordagem descendente (Top-Down):

Esta abordagem a um dado problema, consiste em ter uma sucessiva divisão do domínio do problema em subproblemas ou tarefas com níveis de detalhe sucessivamente mais específicos.





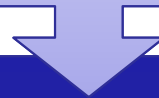
Exemplo de algoritmo

- Lavar os dentes
 1. Abrir a pasta dos dentes
 2. Pegar na escova
 3. Colocar a pasta na escova
 4. Pousar a escova
 5. Fechar a pasta
 6. Abrir a torneira
 7. Pegar no copo
 8. Encher o copo com água
 9. Fechar a torneira
 10. Pegar na escova
 11. Escovar os dentes
 12.



Algoritmos

- Para que um algoritmo seja de qualidade deve ter as seguintes características:
 - Ser corretamente definido (ordem nas ações e definidas claramente)
 - Não estar sujeito a ambiguidades (falta/duplicação de informação)
 - Ser eficaz (salvaguardar situações de exceção, ser global, não ter erros)
 - Ser eficiente (menos memória, menos tempo de execução, etc...)



BOA PROGRAMAÇÃO:

- Oferece precisão (faz o que é pedido)
- É de confiança (ao longo do tempo mantêm-se com precisão e exatidão)
- Tem potencialidade (trata todos os dados)
- Apresenta eficiência (não faz cálculos/testes desnecessários)
- É acessível (é de fácil utilização)
- Tem fácil manutenção (qualquer alteração é facilmente implementada)
- É flexível (pode ser utilizado em equipamentos distintos)



Algoritmos

Exemplo de um algoritmo...

- ir ao supermercado
- escolher artigos
- repetir
- dirigir-se à caixa
- pagar com cartão

MAS...

- Qual o supermercado?
- Quais os artigos a escolher?
- Que quantidade de cada artigo?
- Repetir o quê?



Algoritmos

Algoritmo melhorado...

- ir ao supermercado X
- dirigir-se à secção de papelaria
- escolher
 - 2 canetas
 - 1 caderno A4
- dirigir-se à secção de jardinagem
- escolher
 - 1 tesoura de poda
- dirigir-se à caixa
- pagar com cartão

MESMO ASSIM...

- Quem vai ao supermercado?
- Qual a cor das canetas?

E se fosse um robot a ir ao supermercado?

Ficaria parado após o pagamento sem saber o que fazer. As compras jamais chegariam a casa!!!

OS COMPUTADORES AGEM COMO ROBOTS. NÃO PENSAM . POR ISSO É NECESSÁRIO DAR-LHES INDICAÇÕES PRECISAS SOBRE O QUE PRETENDEMOS QUE EFETUEM.



Algoritmos

Problema...

No séc. XIX viveu na Alemanha um grande matemático: Gauss.

Aos 5 anos, Gauss andava na escola primária. Um dia, o professor deu-lhes um problema longo e aborrecido:

→ **Calculem a soma dos primeiros 100 números inteiros!**

Os alunos começaram o cálculo lento e laborioso ($1+2=3$; $3+3=6$; $6+4=10$...)

Mas poucos minutos depois, Gauss apresentou o resultado final e, perante o professor admirado, explicou o seu raciocínio:

É fácil! Escrevi os números inteiros de 1 a 100... e depois reparei que a soma de 0 com 100 dava 100... e que a soma de 1 com 99 também dava 100... e $2+98=100$... e $3+97=100$ e assim sucessivamente até $49+51=100$. Ao todo só há 50 destas somas ficando o número 50 por somar.

O resultado é, pois, $50 \times 100 + 50 = 5050!!!$

Podemos encontrar várias soluções para o mesmo problema!
Nesta situação, Gauss encontrou uma solução muito mais eficiente!



Pseudocódigo

- Linguagem intermédia que pode ser traduzida em qualquer linguagem de programação.
- É uma notação algorítmica muito mais precisa que a linguagem natural.
- Não há uma notação consensual para o pseudocódigo, daí, se utilizarem notações mais ou menos formais de modo a desenvolver os algoritmos.



Linguagem de programação

- Sistema de escrita formal para enunciar a execução de operações num computador
- Formal pois obedece a um conjunto de normas ou regras
- Alfabeto: caracteres e símbolos que constituem os elementos básicos da linguagem
- Semântica: termos, palavras e sinais que assumem determinado significado
- Sintaxe: regras que estipulam o uso correto dos termos para construir enunciações válidas



Programa

- Conjunto de frases que utilizam o alfabeto, a semântica e a sintaxe de uma determinada linguagem de programação
- Instruções de entrada/saída
- Instruções de cálculo
- Instruções lógicas e de comparação
- Instruções de pesquisa, armazenamento e movimento de dados

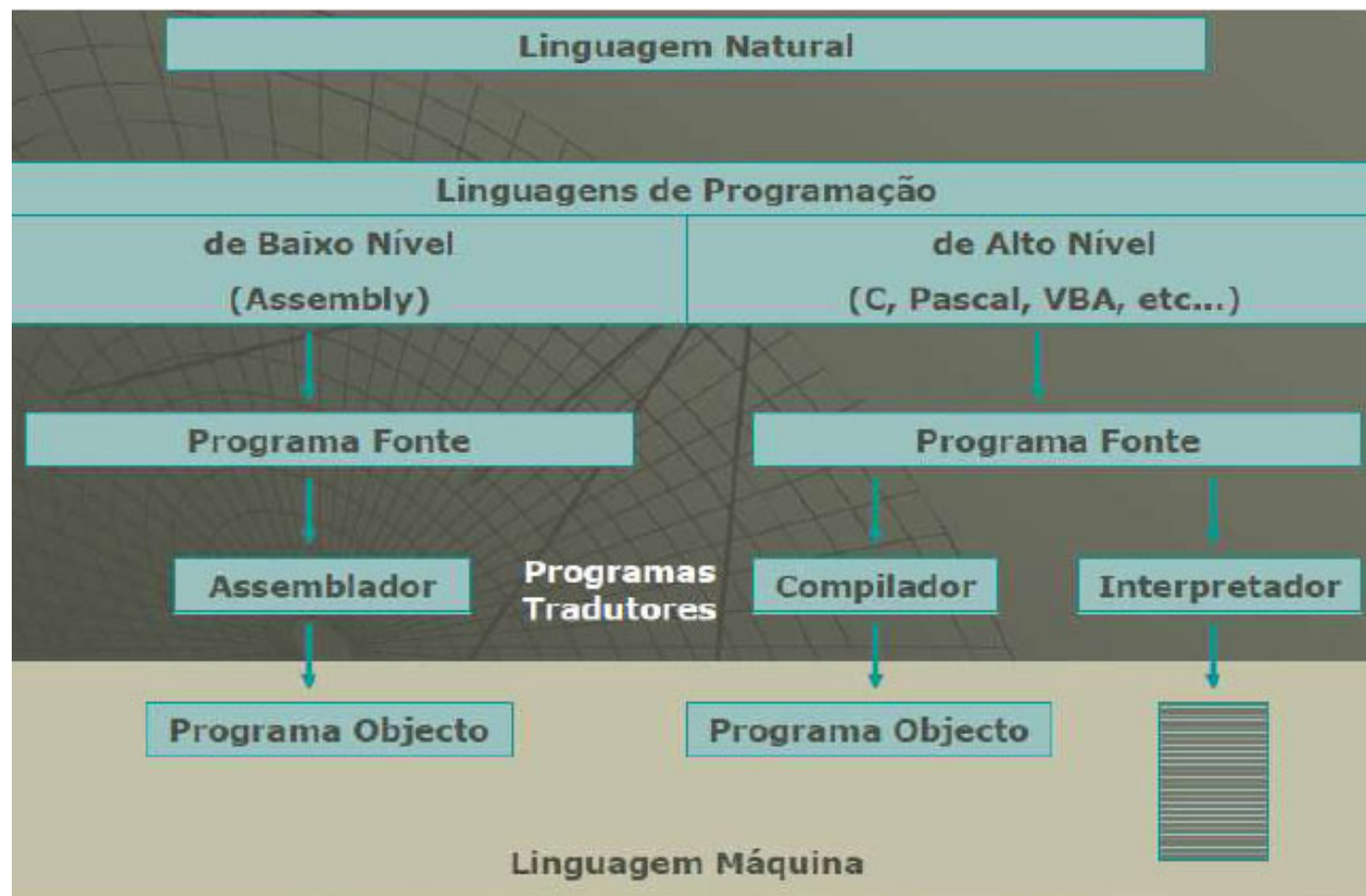


Tipos de linguagens

- Baixo nível
 - Linguagem máquina: a única que o computador reconhece e manipula; são estruturas codificadas de bits – *0011011011011010101*
 - Linguagens simbólicas de montagem (assembly): muito próximas da linguagem máquina mas baseadas em palavras mais inteligíveis, que não são mais do que mnemónicas representativas das operações em máquina: *S, ADD C, D*
- Linguagens de alto nível: estabelecem uma ponte entre a linguagem máquina e a linguagem natural; aproximam-se mais da nossa inteligibilidade: *a=2; b=3; c=a+b;*



Programas tradutores



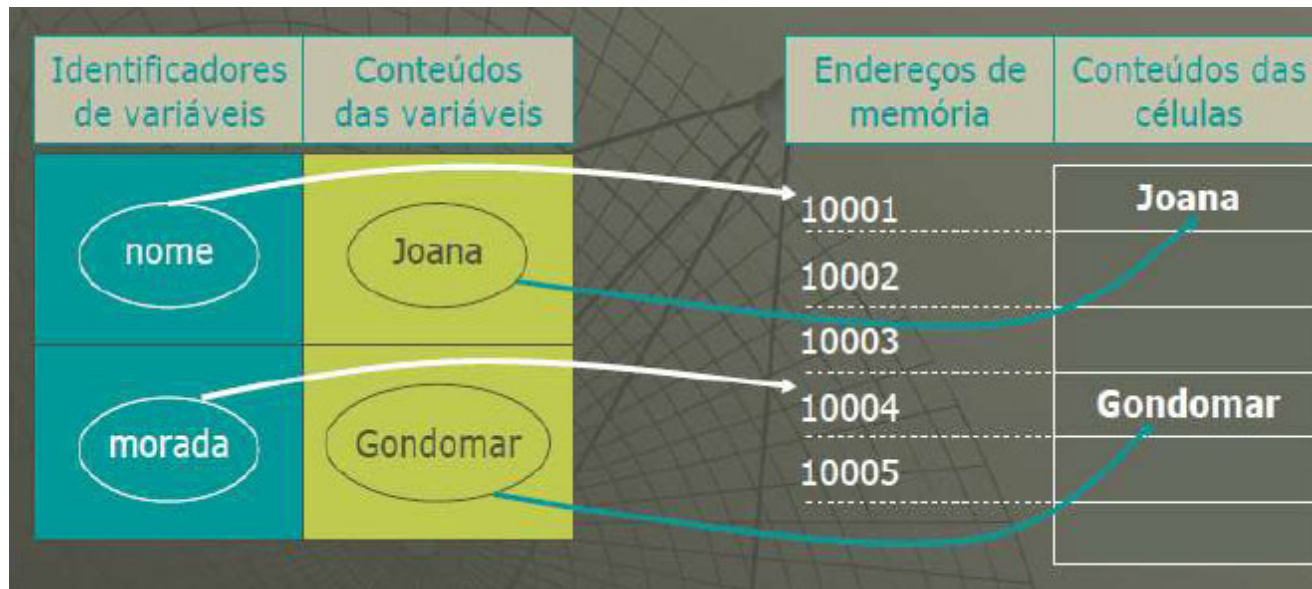


Constantes e variáveis

- Constantes: valores que se mantêm inalterados dentro de um programa
- Variáveis: entidades que podem assumir diferentes valores ao longo da execução do programa
- Exemplos:
 - 3
 - "Ola Turma!"
 - $a = 3$

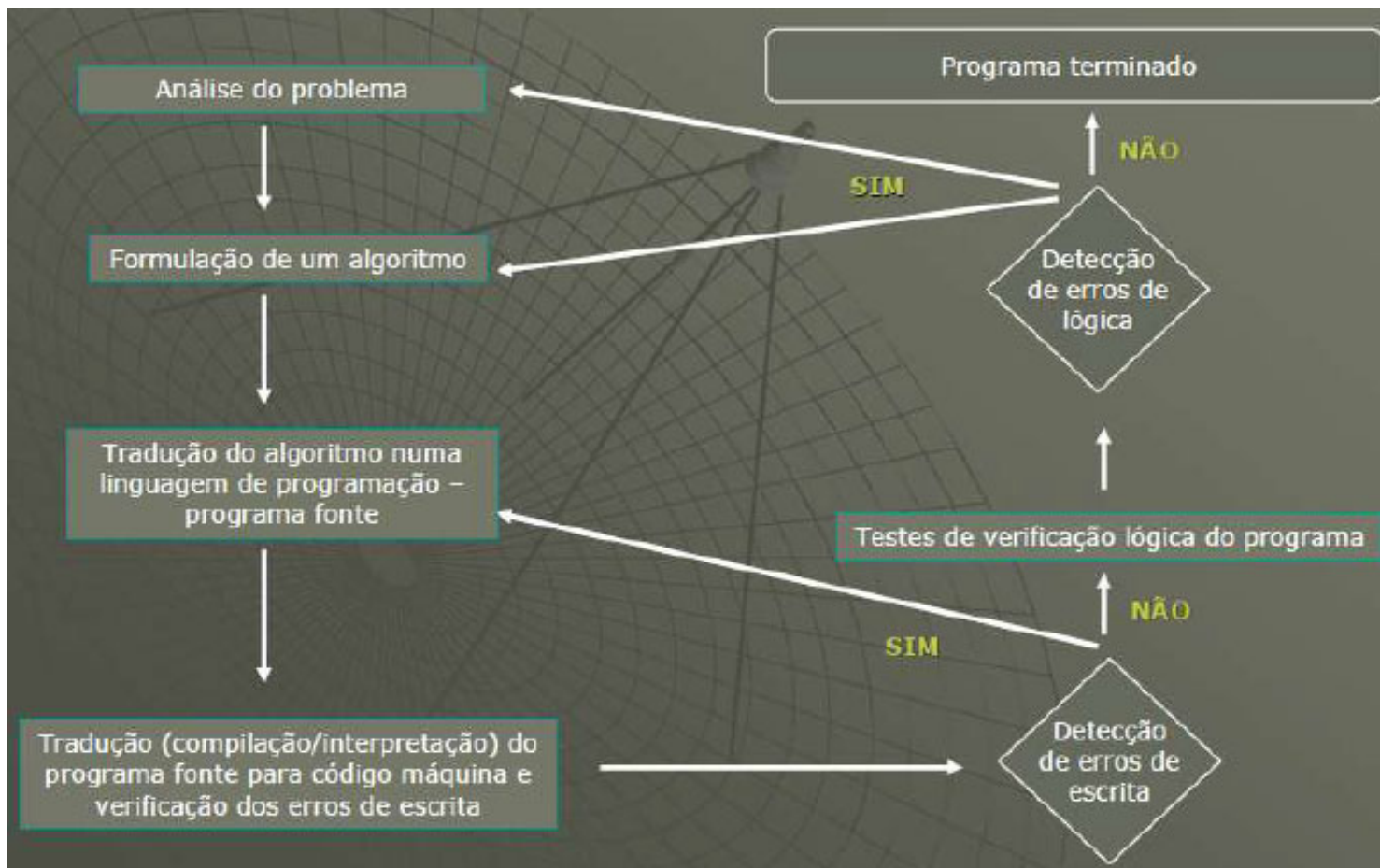


Identificadores e endereços de memória





Elaboração de um programa





Tipos de dados

- Dados simples
 - Numéricos (12, 0.123, 0, ...)
 - Alfabéticos ("escola", "ola mundo", ...)
 - Alfanuméricos ("n1", "ab43Y", ...)
 - Lógicos ou booleanos (verdadeiro – TRUE/1 ou falso –FALSE/0)
- Dados estruturados
 - Arrays (agrupamento de dados)
 - Registos (agrupamento de diferentes tipos de dados)
 - Ficheiros
 - Apontadores



Instruções

- **Atribuição**
 - $x = 3$
 - `nome = "Ana"`
 - `nota = x*y+n`
- **Leitura** (input)
 - `valor = input("Insira um valor: ")`
- **Escrita** (output)
 - `print("Olá turma")`
 - `print("O valor final é ", n)`



Expressões e operadores

- Uma expressão é um conjunto de operandos, articulados entre si por operadores
 - **Operadores** (depende das linguagens de programação)
 - Aritméticos (+, -, /, *, %)
 - Incremento e decremento (++ , --)
 - Comparação (==, !=, <, <=, >, >=)
 - Lógicos ou booleanos (not, and, or, True, ...)



Expressões

- Numéricas: utilizam apenas operadores aritméticos
 - $100 * (1 + 0.15)$
 - $qnt * custo_unidade$
 - $100 * custo_unidade + 1000$
- Booleanas: utilizam operadores lógicos e espera-se obter um resultado lógico
 - Se($preco > custo * 2.5$) então ...



Problema

- É-nos dado o preço de um terreno de forma retangular, bem como as medidas de dois lados adjacentes. Pretendemos saber se o seu preço por metro quadrado está acima ou abaixo da média dos preços praticados na zona, sendo-nos também dado esse valor.



Algoritmo vs código fonte

- alg soma:
soma=0
num=1
enquanto num <= 100 faz
 soma = soma + num
 num = num+1
print('A soma é: ', soma)
- def somar():
soma=0
num=1
while num <= 100:
 soma = soma + num
 num = num+1
print('A soma é: ', soma)

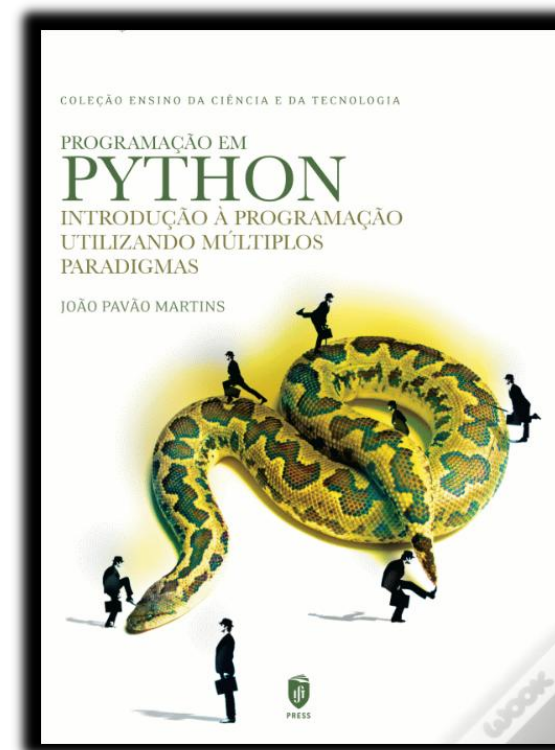
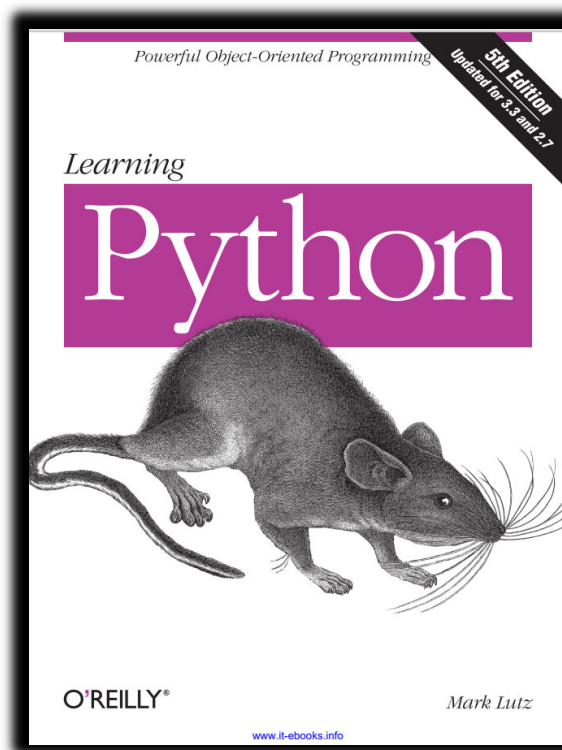
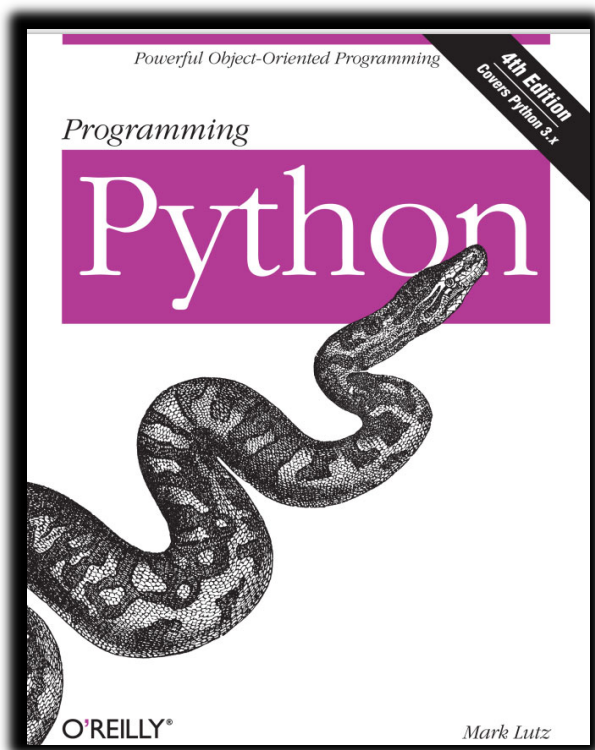


Algoritmos

- Ficha de exercícios nº. 1
- Ficha de exercícios nº. 2



Bibliografia





Python

- Linguagem de programação, ou seja, corresponde a um formalismo para escrever programas.
- Um programa em python pode ser introduzido e executado interactivamente num ambiente em que exista um interpretador do python (algo que compreende as frases escritas em linguagem python).
- Origens em 1992 com forte influência da linguagem ABC
- Também influenciada por outras linguagens: Angol, C, Haskell, JAVA, Lisp, Perl, ...
- Multiparadigma: Orientada a objetos, imperativa e funcional



Python

- Combina uma sintaxe concisa e clara com os recursos poderosos de bibliotecas standard, módulos e frameworks desenvolvidos por terceiros.
- Python é uma linguagem de alto nível. Possui tipagem dinâmica e uma das suas principais características é permitir a fácil leitura do código e exigir poucas linhas de código se comparado ao mesmo programa escrito noutras linguagens.
- É bastante versátil e aplica-se a processamento de textos, dados científicos e criação de CGIs para páginas dinâmicas para a web.
- É das linguagens mais populares do momento.



Python

- 2001 criada Python Software Foundation.
- Apoiada pela Microsoft, Google, ...
- É incorporada em Sistemas Operativos BSD, MacOs, Linux
- Código aberto (Open Source)
- Tudo é encarado como sendo um objeto (até mesmo uma simples variável)
- Inúmeras bibliotecas com fins variados - versatilidade



Python

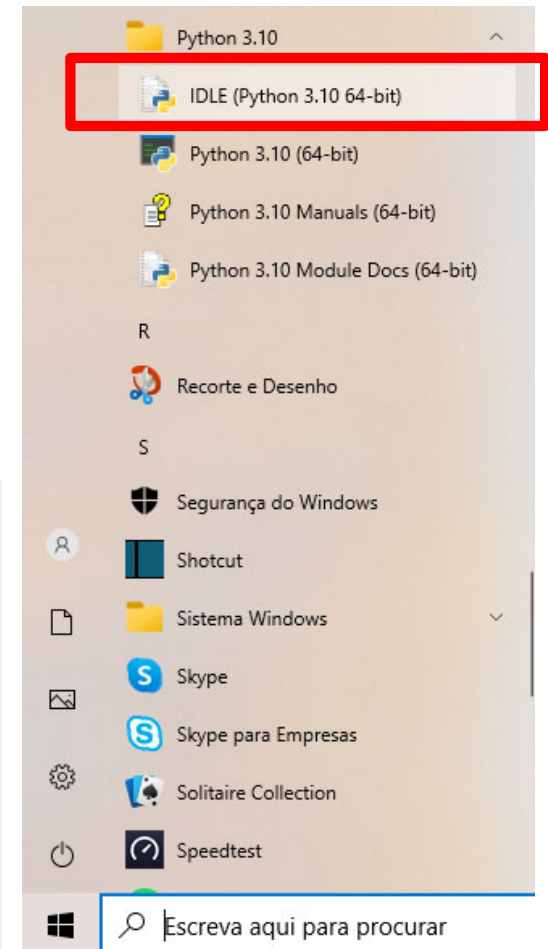
- Áreas onde se aplica o Python:
 - aprendizagem de programação;
 - Inteligência artificial;
 - Biotecnologia;
 - Computação 3D;
 - Automação;
 - Cibersegurança;
 - Partilha peer-to-peer (torrents);
 - NASA;
 -



Instalação

- Linux já tem instalado o interpretador Python
 - `sudo apt install python3.8`
- Download para Windows: aceder ao website www.python.org, descarregar e instalar

```
IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```





Constantes e Variáveis

- As constantes podem ser números, valores lógicos ou cadeia de caracteres. Sempre que é fornecida uma constante em Python, este devolve a constante como resultado da avaliação.
- A representação externa de uma entidade é a representação como vemos essa entidade. Ora, essa entidade, internamente terá outra representação. Por exemplo, a representação externa do ano 2020 é a sequência dos dígitos pela ordem que escrevemos (2020) enquanto que a sua representação interna será 11111100100 (binário).



Constantes e Variáveis

- Assim, se escrevermos a constante 2020, a resposta do Python será 2020.

- Exemplos:

```
>>> 2020
```

```
2020
```

```
>>> -1
```

```
-1
```

```
>>> +2
```

```
2
```

```
>>> 'I love python'
```

```
'I love python'
```



Tipagem em Python

- Números inteiros
 - Números sem parte decimal (com ou sem sinal)
- Números reais
 - Números com parte decimal (com ou sem sinal)
 - Normalmente representados em notação científica (muito grandes ou muito pequenos)



Tipagem em Python

- Valores lógicos
 - São representados por True e False
- Cadeia de caracteres
 - Sequência de caracteres. São representadas por plicas (') ou aspas("")
 - As plicas ou aspas não fazem parte da cadeia de caracteres. Assim, o seu comprimento diz respeito apenas aos caracteres que constituem essa sequência
 - 'Bom dia' é uma cadeia com 7 caracteres (espaço também conta)



Expressões compostas

- Para além de constantes, em Python existe um certo número de operações embutidas (ou pré-definidas), que ele conhece independentemente de qualquer indicação que seja fornecida por algum programa.
- Uma expressão composta é constituída por um operador e por um certo número de operandos.
- Operadores podem ser unários (not, - ,) ou binários (+ , *,)



Expressões compostas

>>>2020-1980

40

>>> 3 * (5 + 12) // 2

25

>>> 3.0 * (5 + 12) // 2

25.0

>>> 5 > 12

False



Operações sobre números inteiros

<i>Operação</i>	<i>Tipo dos operandos</i>	<i>Valor</i>
$e_1 + e_2$	Inteiros	O resultado de somar e_1 com e_2 .
$e_1 - e_2$	Inteiros	O resultado de subtrair e_2 a e_1 .
$-e$	Inteiro	O simétrico de e .
$e_1 * e_2$	Inteiros	O resultado de multiplicar e_1 por e_2 .
$e_1 // e_2$	Inteiros	O resultado da divisão inteira de e_1 por e_2 .
$e_1 \% e_2$	Inteiros	O resto da divisão inteira de e_1 por e_2 .
$\text{abs}(e)$	Inteiro	O valor absoluto de e .

** - potência ($n^{**}y \Rightarrow n^y$) ou utiliza função pow -> pow(n,y)



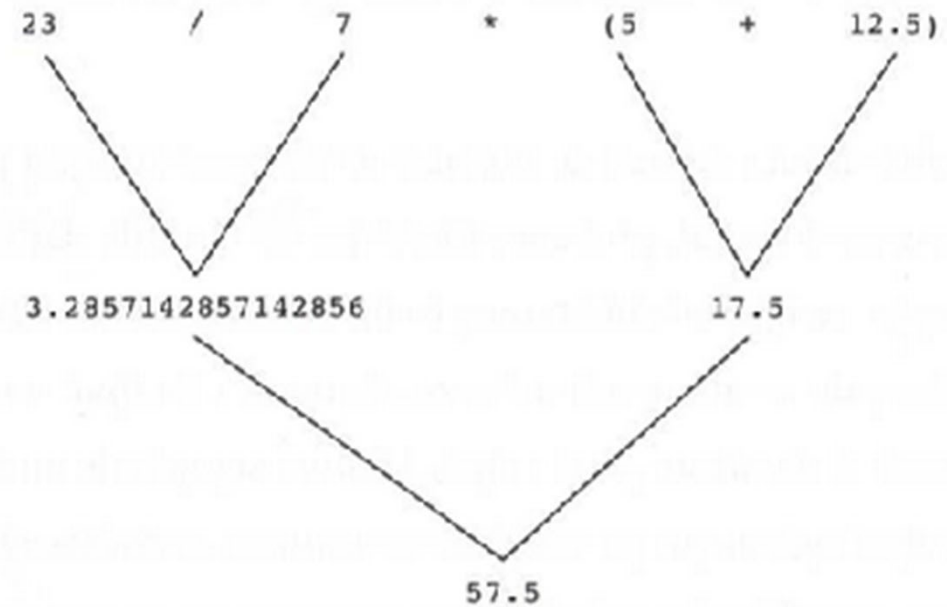
Operadores relacionais

<i>Operação</i>	<i>Tipo dos argumentos</i>	<i>Valor</i>
$e_1 == e_2$	Números	Tem o valor True se e só se os valores das expressões e_1 e e_2 são iguais.
$e_1 != e_2$	Números	Tem o valor True se e só se os valores das expressões e_1 e e_2 são diferentes.
$e_1 > e_2$	Números	Tem o valor True se e só se o valor da expressão e_1 é maior do que o valor da expressão e_2 .
$e_1 >= e_2$	Números	Tem o valor True se e só se o valor da expressão e_1 é maior ou igual ao valor da expressão e_2 .
$e_1 < e_2$	Números	Tem o valor True se e só se o valor da expressão e_1 é menor do que o valor da expressão e_2 .
$e_1 <= e_2$	Números	Tem o valor True se e só se o valor da expressão e_1 é menor ou igual ao valor da expressão e_2 .



Operadores e precedências

Prioridade	Operador
Máxima	Aplicação de funções not, - (simétrico) *, /, //, % +, - (subtração) <, >, ==, >=, <=, !=
Mínima	and or



- / -> divisão de a por b
- // -> quociente (parte inteira) da divisão de a por b
- % -> resto da divisão de a por b



Utilização de operadores aritméticos com strings

- Em Python, podemos utilizar os operadores aritméticos com strings
- Concatenação:
 - `>>>'Olá' + 'Mundo'`
 - `'OláMundo'`
- Duplicação:
 - `>>>'Olá' * 5`
 - `'OláOláOláOláOlá'`
 - `>>> print('Olá' * 5)`
 - `OláOláOláOláOlá`



Variáveis

- Toda a variável é um objecto
- Uma variável é identificada com um nome
- Altamente recomendado que se utilizem nomes sugestivos para melhor identificação do conteúdo dos dados dessa variável. Por exemplo, uma boa sugestão de um nome para uma variável conter a idade de uma pessoa seria o nome "idade"
- Recomenda-se que se inicie o nome da variável sempre com letra minúscula ou com o carácter '_'
- Nome de uma variável será composto por sequencia de outros caracteres alfabéticos incluindo o carácter '_'



Atribuição de valores

- A instrução de atribuição de valores a variáveis tem importância fundamental nas linguagens de programação imperativas, como é o caso do python.
- A instrução recorre ao operador embutido '='. Este operador recebe 2 operandos. O operando do lado esquerdo será o nome da variável e o operando do lado direito o valor a atribuir à variável

Ex: idade=30

Nome_completo='Júlio Magalhães'



Variáveis reservadas

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



Atribuição múltipla

- Python, permite fazer várias atribuições em simultâneo (cuidado...)
- $\langle \text{nome}_1 \rangle, \langle \text{nome}_2 \rangle, \dots, \langle \text{nome}_n \rangle = \langle \text{expr}_1 \rangle, \langle \text{expr}_2 \rangle, \dots, \langle \text{expr}_n \rangle$

Ex:

```
>>> nome, apelido='Julio', 'Magalhaes'
```

```
>>> print(nome)
```

Júlio

```
>>> print(apelido)
```

Magalhães

```
>>> print(nome, apelido)
```

Júlio Magalhães



Exercícios

- Testar alguns comandos de cálculo e atribuição de variáveis
- Qual o output das seguintes expressões:
 - $3+4$
 - Fazer mais
- + serve para concatenar apenas strings. Se misturar com números dá erro
- Atribua a variáveis o seu nome, apelido, altura, idade e peso.
- Determine o ano em que nasceu dando como output algo semelhante ao exemplo:

Júlio Magalhães, com 1.80m e idade 30anos, nasceu em 1990.



Output

- Função default «print»:
 - `print("Bem-vindo ao Python")`
 - `>>> Bem-vindo ao Python`
 - `print(3+5)`
 - `>>> 8`
 - `print('3+5')`
 - `>>> 3+5`
 - `print('3'+ '5')`
 - `>>> 35`
 - `print('Número',7)`
 - `>>> Número 7`



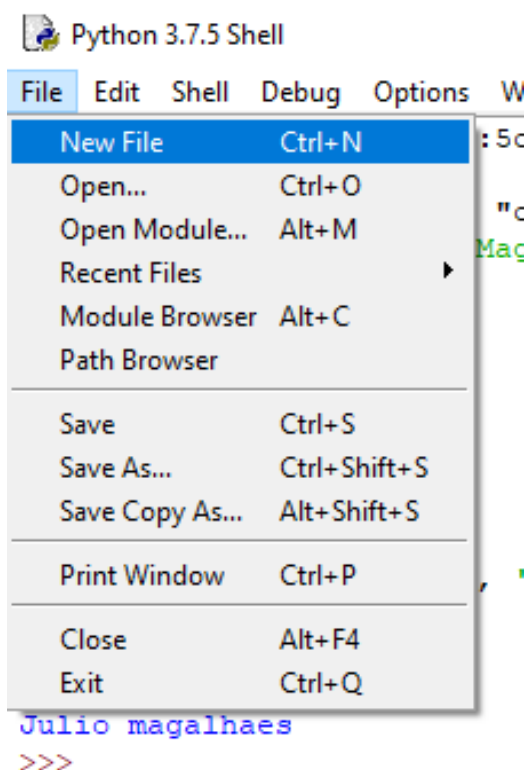
Input

- Função default «input»:
 - nome = input("Qual o seu nome? ")
 - >>> Qual o seu nome? Júlio
 - apelido = input("Qual o seu apelido? ")
 - >>> Qual o seu apelido? Magalhães
 - idade = input("Qual a sua idade? ")
 - >>> Qual a sua idade? 30
 - print(nome, apelido, idade)
 - >>> Júlio Magalhães 30

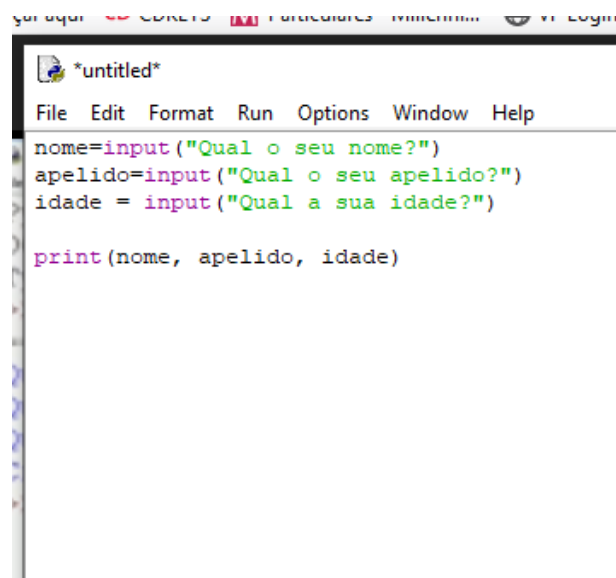


Criação de scripts

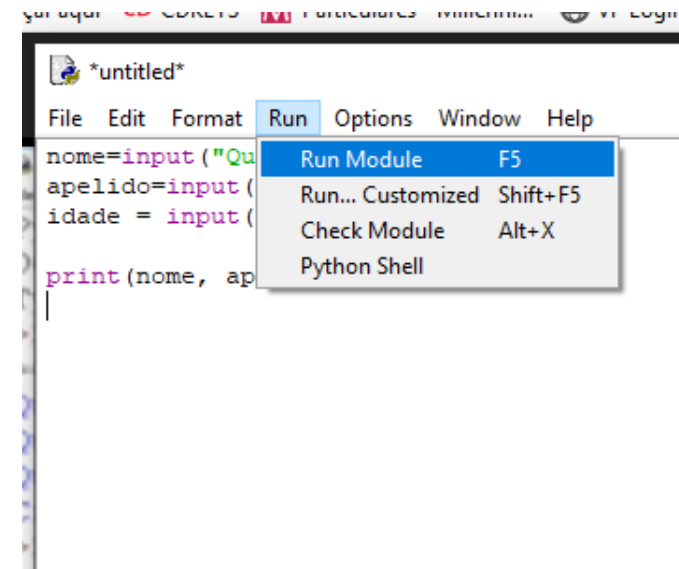
Criar ficheiro



Escrever o código-fonte



Executar





Exercícios

- Crie um script python que solicite ao utilizador o nome e que devolva como output uma mensagem de boas-vindas a esse utilizador.

Ex: Qual o seu nome? Júlio

Qual o seu apelido? Magalhães

Olá Júlio Magalhães ! Seja Bem-vindo

- Crie um script python que solicite ao utilizador o dia, mês e ano de nascimento e devolva como output algo parecido com o seguinte exemplo:

Ex: Dia? 25

Mês? dezembro

Ano? 1990

Você nasceu no dia 25 de dezembro de 1990

- Crie um script python que solicite ao utilizador 2 números inteiros e devolva a sua soma. O que aconteceu?



CINEL

IDE – PyCharm

The screenshot shows the PyCharm website homepage. At the top, there's a dark navigation bar with the JetBrains logo (20 years) and links for Tools, Languages, Solutions, Support, Company, Store, and a search icon. Below this, the 'PyCharm' section features a 'Download' button and links for 'Coming in 2020.1', 'What's New', 'Features', 'Learning Center', and 'Buy'. The main content area is divided into two columns. The left column displays the PyCharm logo (a green and yellow hexagon with 'PC' in a black square) and version information: 'Version: 2019.3.4', 'Build: 193.6911.25', and '18 March 2020'. Below this are links for 'System requirements', 'Installation Instructions', and 'Other versions'. The right column is titled 'Download PyCharm' and has tabs for 'Windows', 'Mac', and 'Linux'. It is split into two sections: 'Professional' and 'Community'. The 'Professional' section describes it as 'For both Scientific and Web Python development. With HTML, JS, and SQL support.' and offers a 'Download' button for a 'Free trial'. The 'Community' section describes it as 'For pure Python development' and offers a 'Download' button for 'Free, open-source'. At the bottom, there's a box promoting the 'Toolbox App' to download PyCharm and its future updates with ease.

PyCharm Coming in 2020.1 What's New Features Learning Center Buy [Download](#)

Download PyCharm

[Windows](#) [Mac](#) [Linux](#)

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

[Download](#)


Free trial

Community

For pure Python development

[Download](#)

Free, open-source

 **Get the Toolbox App to download PyCharm and its future updates with ease**



Comentários

- # - tudo que escrever à frente do caracter #, por linha, será ignorado pelo interpretador
- ''' texto ''' – as linhas de texto entre a plica tripla será ignorado pelo interpretador

```
# comentário por linha
```

```
'''
```

```
sequencia de  
comentários
```

```
'''
```



Tipos primitivos

- int -> 7 3 0 -3

```
a = int( input('insira 1 número: '))  
b = int( input('insira outro número'))  
s = a + b  
print('soma é ', s)
```

- float -> 4.5 0.321 -5.0 3.14
- bool -> True False
- str -> 'Olá' '7.5' '' ''



Descobrir tipo da variável

- Analise o seguinte código:

```
##descobrir o tipo da variavel  
n1 = input('Escreva 1 número: ')  
print(type(n1))  
n2 = int(input('Escreva outro número: '))  
print(type(n2))
```

- Veja o tipo das variáveis para o mesmo input '7'

```
Escreva 1 número: 7  
<class 'str'>  
Escreva outro número: 7  
<class 'int'>
```

Process finished with exit code 0



Exercício

- Faça um script leia 4 valores para variáveis, um de cada tipo e devolva o type dessas variáveis.
- No caso de o valor a ser lido for um booleano, o que acontece se ler um símbolo qualquer? E se apenas ler o carácter 'Enter'?



Formato de output em Python3

```
nome = input('Insira o seu nome: ')
print('O nome escrito foi {}'.format(nome))
print(f'O nome escrito foi {nome}.')
```

- O bloco {} será substituído pela formatação e conteúdo da variável nome.
- Abaixo, um exemplo de múltipla atribuição e output com 2 blocos de substituição. O sinal '+' significa que concatena o restante texto que se segue.

```
nome, apelido = input('Nome: '), input('Apelido: ')
print('O nome escrito foi {}'.format(nome) + ' e apelido foi {}'.format(apelido))
```



Continuando com output

- Analise os seguintes códigos:

```
a = int(input('insira 1 número: '))
b = int(input('insira outro número'))
s = a + b
print('soma entre ', a, ' e ', b, ' é de ', s)
```

```
a = int(input('insira 1 número: '))
b = int(input('insira outro número'))
s = a + b
print('A soma entre {} e {} é {}'.format(a, b, s))
#print(f'A soma entre {a} e {b} é {s}')
```



Alinhamento no output

- Ocupar 15 posições de caracteres

```
nome = input('Escreva o seu nome: ')\nprint('O seu nome é {:15}!'.format(nome))
```

Escreva o seu nome: Júlio
O seu nome é Júlio !

- Alinhar à direita

```
nome = input('Escreva o seu nome: ')\nprint('O seu nome é {:>15}!'.format(nome))
```

Escreva o seu nome: Júlio
O seu nome é Júlio!

- Alinhar à esquerda

```
nome = input('Escreva o seu nome: ')\nprint('O seu nome é {:<15}!'.format(nome))
```

Escreva o seu nome: Júlio
O seu nome é Júlio !

- Alinhar ao centro

```
nome = input('Escreva o seu nome: ')\nprint('O seu nome é {:^15}!'.format(nome))
```

Escreva o seu nome: Júlio
O seu nome é Júlio !

- Alinhar ao centro com espaços preenchidos com um símbolo `*`

```
nome = input('Escreva o seu nome: ')\nprint('O seu nome é {:*^15}!'.format(nome))
```

Escreva o seu nome: Júlio
O seu nome é *****Júlio*****!



Formatação no output

- Como fazer para escrever o resultado de um valor float apenas com 3 casas decimais?

```
n1 = int(input('Insira 1º número: '))  
n2 = int(input('Insira 2º número: '))  
print('O valor da divisão de {} por {} é {}'.format(n1,n2,n1/n2))
```

Insira 1º número: 4

Insira 2º número: 3

O valor da divisão de 4 por 3 é 1.3333333333333333

- Solução está no format do campo que pretendemos formatar:

```
n1 = int(input('Insira 1º número: '))  
n2 = int(input('Insira 2º número: '))  
print('O valor da divisão de {} por {} é {:.3f}'.format(n1,n2,n1/n2))
```



Formatação no output

- Podemos fazer vários prints no código mas indicar ao Python que a frase de output é uma só.

```
n1 = int(input('Insira 1º número: '))  
n2 = int(input('Insira 2º número: '))  
print('O valor da divisão de {:.1f} '.format(n1))  
print('por {:.1f} é {:.3f}'.format(n2,n1/n2))
```

```
Insira 1º número: 4  
Insira 2º número: 3  
O valor da divisão de 4.0  
por 3.0 é 1.333
```

- Solução é colocar a expressão `end= ''`

```
n1 = int(input('Insira 1º número: '))  
n2 = int(input('Insira 2º número: '))  
print('O valor da divisão de {:.1f} '.format(n1), end='')  
print('por {:.1f} é {:.3f}'.format(n2,n1/n2))
```



Carateres especiais

<i>Carácter escape</i>	<i>Significado</i>
\\	Barra ao contrário (\)
\'	Plica (')
\"	Aspas (")
\b	Retrocesso de um espaço
\f	Salto de página
\n	Salto de linha
\r	“Return”
\t	Tabulação horizontal
\v	Tabulação vertical

```
n = input('Escreva qualquer coisa na linha abaixo:\n')
print(n.isnumeric())
print(n.isalpha())
```

Escreva qualquer coisa na linha abaixo:
Julio77
False
False



Funções `isnumeric()`, `isalpha()`, ...

- A função `isnumeric()` devolve um booleano, ou seja, devolve 'True' caso o valor de uma variável seja passível de conversão em numérico ou devolve 'False' caso contrário. Vejamos:

```
n = input('Escreva qualquer coisa: ')\nprint(n.isnumeric())
```

Escreva qualquer coisa: Olá
False

Escreva qualquer coisa: 77
True

- A função `isalpha()` devolve booleano se todos os caracteres pertencem ao alfabeto (a-z,A-Z). Experimente...



Funções `isnumeric()`, `isalpha()`, ...

- `isalnum()` -> é alfanumérico? `'Palavra23'`
- `isspace()` -> é espaço? `' '`
- `isupper()` -> letras maiúsculas? `'PALAVRA'`
- `islower()` -> letras minúsculas? `'palavra'`
- `istitle()` -> palavra capitalizada? `'Palavra'`



Exercícios

- Elabore em python um script que dado alguma coisa no input, devolva como output:
 - O tipo primitivo;
 - É um número?
 - É alfanumérico?
 - É alfabético?
 - Está em maiúsculas?
 - Está em minúsculas?
 - Está capitalizada?



Identações - tab

- O código em python tem de estar muito bem estruturado para que o interpretador o consiga ler, caso contrário, poderá haver erros de semântica
- Conhece a expressão: “99% dos erros informáticos está entre o teclado e a cadeira”? Pois é, no desenvolvimento de código, os erros são do programador, quer por erros de sintaxe quer por erros de semântica.
- As operações que dizem respeito a determinado tipo de bloco têm de estar devidamente identados, usando as tabulações.



Comandos de seleção

- if condição :
 comandos
- else :
 comandos

-> parte do **else** pode ser opcional.



Comandos de seleção

```
n = int(input('Escreva um número: '))  
if n==10:  
    print('O valor inserido corresponde a uma dezena ({}).format(n))
```

Escreva um número: 10

O valor inserido corresponde a uma dezena (10)

Obtemos como resposta a frase a dizer que o valor equivale a uma dezena pois a condição de teste ($n==10$) deu valor Verdadeiro

Escreva um número: 12

Não obtemos qualquer resposta pois a condição de teste ($n==10$) deu valor Falso

Altere o código anterior para que devolva uma mensagem caso o valor inserido não seja 10



Comandos de seleção - encadeados

- if condição :
 comandos
- elif condição :
 comandos
- elif condição :
 comandos
-
- else :
 comandos

Escreva o seguinte código e experimente

```
n = float(input('Escreva uma nota: '))
if n <= 5 :
    print('Mau')
elif n < 10 :
    print('Negativo')
elif n < 18 :
    print('Positivo')
elif n <= 20 :
    print('Muito Bom')
else :
    print('Valor inválido')
```



Bibliotecas e bibliotecas parciais

- Funcionalidades já implementadas e que podem ser usadas.
- Comando : import biblioteca (importa todas as funcionalidades implementadas na biblioteca)

Ex: **Biblioteca** math

Funcionalidades: ceil, arredonda para cima,
floor, arredonda para baixo
trunc, trunca o número
pow, potência
sqrt, raiz quadrada
factorial, fatorial de um número

Comando: import math



Bibliotecas e bibliotecas parciais

Exemplo:

```
import math
n1 = int(input('Insira 1º número: '))
print('O valor da raiz quadrada de {} é {} '.format(n1 , math.sqrt(n1)))
```

- Bibliotecas parciais: **from biblioteca import funcionalidade** (importa apenas a funcionalidade especificada da respetiva biblioteca). Não precisa de escrever o evento biblioteca
- Comando: from **math** import **sqrt**, **pow**

```
from math import sqrt
n1 = int(input('Insira 1º número: '))
print('O valor da raiz quadrada de {} é {} '.format(n1 , sqrt(n1)))
```




Bibliotecas e bibliotecas parciais

- Bibliotecas disponível em <https://docs.python.org/3.10/py-modindex.html>
- Exercícios:
 - Importar a bib math completa e calcular a raiz quadrada, potência, truncar e arredondar.
 - Import biblioteca random e gerar 2 valores aleatórios e 2 aleatórios entre 1 e 10
- Nota: `random.random()` – valor aleatório entre $[0, 1[$
`random.randint(x, y)` – valor aleatório entre $[x, y]$



Exercícios

- Ficha nº3



Instrução for

- A instrução for é bastante útil para percorrer uma lista de valores. Por exemplo, para escrever os valores no intervalo entre 10 e 20 utilizando o for seria bastante simples:

```
for n in range(10, 20):  
    print (n)
```

- Se pretendêssemos escrever os mesmos números do exemplo mas aparecendo no ecrã, segundo a segundo, basta importar a instância sleep da biblioteca time e obrigar o interpretador a esperar 1 segundo:

```
from time import sleep  
for n in range(10, 20):  
    print (n)  
    sleep(1)
```



Funções

- Servem para melhor estruturar os programas
- Composta por um nome e possivelmente argumentos: $f(x,y)$
- Ex:

def quadrado (x):

return x * x

O nome da função é quadrado

(x) É o argumento/parâmetro da função. É o valor dado no início da função. O "x" será reconhecido dentro da função.

return x * x é uma expressão que pertence à função. Neste caso, sai da função e responde a quem a chamou com um valor que é o resultado de $x*x$.

Se não existir **return**, a função apenas termina não devolvendo qualquer resposta a quem a chamou



Funções

- Programa para calcular o quadrado de x recorrendo a funções

```
#####
```

```
#Função quadrado: recebe um valor e devolve como resposta o quadrado desse número
```

```
def quadrado (x):
```

```
    resultado = x * x
```

```
    return resultado
```

```
#####
```

```
##Código principal do programa
```

```
print("Seja bem-vindo ao cálculo do quadrado de um número.\n")
```

```
valor = int( input("Introduza um valor: ") )
```

```
resposta = quadrado(valor)
```

```
print(f"O quadrado de {valor} é {resposta}")
```



Funções recursivas

- Funções que se chamam a elas mesmas com valores intermédios até chegarem ao cálculo mais básico para depois fazerem as sucessivas substituições.

- Ex

def fatorial(n):

 if n==0:

 return 1

 else:

 return n * fatorial(n-1)



Exercícios

- Ficha nº4



Manipulação de strings

- String significa uma cadeia de caracteres.
- Por exemplo, a frase "Curso de Python" é uma string.
- Python permite de uma forma tranquila, atribuir uma cadeia de caracteres a uma variável:
 - frase="Curso de Python"
- Podemos olhar para uma string como sendo uma variável indexada, ou seja, cada character tem um índice que inicia no 0.
- Assim, o índice da letra C é o 0 e o índice da letra y é o 10 (o espaço também é um character e ocupa uma posição na string)



String

- frase="Curso de Python"
- Para nos referirmos a um determinado character, utilizamos o nome da variável "frase" com o índice pretendido. Ex: frase[9] corresponde ao "P".
- letra = frase[0] #a variável letra assume a letra "C" – índice 0 da frase



String

- frase="Curso de Python"
- No python, podemos obter parte da string de uma forma muito linear. Por exemplo, se quisermos obter a sequência "Pyth" faríamos algo do género: `sequencia=frase[9:13]`
- A variável sequencia, terá os caracteres da frase que começam no índice 9 e vai até ao 12!!! **O índice 13 não é considerado.**



String

- frase="Curso de Python"
- E se quiséssemos uma sequência mas com saltos de 2 em 2 caracteres?
- Simples: sequencia=frase[9:14:2] ; o resultado seria Pto



String

- frase="Curso de Python"
- E se quiséssemos uma sequência do início até determina posição ou de determinada posição até ao fim?
- sequencia=frase[:4] ; o resultado seria do início até à 3ª posição, ou seja **Curs**
- sequencia=frase[9:] ; o resultado seria do índice 9 até ao final da string, ou seja, **Python**



String

- frase="Curso de Python"
- E agora, se pretendêssemos uma sequência da variável frase, desde o início até ao fim mas com saltos de 3 em 3?
- sequencia=frase[::3] ; o resultado seria do início até ao final com saltos de 3 posições, ou seja, **CsdPh**



String

- frase="Curso de Python"
- Uma das ações mais utilizadas na análise de string, é saber o comprimento de uma dada frase.
- `x = len(frase)` #a variável x assume o comprimento da variável frase
- Neste caso, x ficaria com o número 15, uma vez que os espaços também são caracteres.



String

- frase="Curso de Python"
- Uma ação muito utilizada, é saber quantas ocorrências de uma dada letra ocorre na string.
- `x = frase.count('o')` #a variável x assume a quantidade de ocorrências da letra "o" na frase
- Neste caso, x ficaria com o número 2 pois a letra 'o' aparece 2 vezes na frase



String

- frase="Curso de Python"
- Desafio:
 - Quantas vezes ocorre a letra 'y' da posição 3 até à posição 6?
 - Quantas vezes ocorre a letra 'y' da posição 3 até ao final?



String

- frase="Curso de Python e outro de Python"
- Outra situação bastante recorrente, será procurar uma dada substring. Por exemplo, procurar "yth" na frase!
- frase.find("yth") irá devolver o índice 10 pois é neste índice que inicia a primeira ocorrência da sequência a procurar
- frase.find("Linguagem") irá devolver -1 pois a sequência "Linguagem" não ocorre vez nenhuma



String

- frase="Curso de Python e outro de Python"
- A instrução `in` é também muito utilizada. Permite-nos obter o resultado booleano (True ou False), caso uma sequência ocorra ou não na frase inicial.
- Ex:
 - "Curso" `in` frase -> daria resultado `True`
 - "curso" `in` frase -> daria resultado `False`



String

- frase="Curso de Python"
- As strings são imutáveis. Só através de atribuição é que podemos alterá-las.
- O que fará a instrução `frase.replace("Python","Linguagem Python")` ?
- Neste caso, a palavra a ser substituída é inferior ao que se pretende!
- O Python faz o aumento de índices automático.
- Se fizer um print ao conteúdo da variável, o que irá aparecer?



String

frase="Adoro Linguagem Python e também SQL"

- frase.upper() -> o que for maiúscula mantém, o restante transforma em maiúsculas. E se for um dígito?
- frase.lower() – mantém minúscula e altera as maiúsculas para minúsculas
- frase.capitalize() -> todos caracteres em minúsculas e só o 1º é capitalizado – maiúscula
- frase.title() -> Capitaliza todas as palavras. Só os 1º caracteres são maiúsculas



String

- frase=" Curso de Python "
- frase.strip() -> remove espaços inúteis no início e no fim da frase. Reposiciona nos índices corretos do array.
 - E se tiver vários espaços no meio? (Feito de forma diferente...)
- frase.rstrip() – o r significa right. Apaga espaços à direita (fim da string)
- frase.lstrip() – o l significa left. Apaga espaços do início.
- **lista** = frase.split() – divisão da string, considerando os espaços. Faz uma lista de arrays... onde a lista é numerada a partir da posição 0
- `'-'.join(lista)` – junta os elementos de cada palavra separados por `'-'`
- `'-'.join(frase)` – o que aconteceria?



ASCII TABLE

- Obter o número do carater
 - ord('R') -> 82
- Obter o caracter
 - chr(125) -> '}'

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	\$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	*	87	57	1010111	127	W					
40	28	101000	50	(88	58	1011000	130	X					
41	29	101001	51)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	.	93	5D	1011101	135]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					



Exercícios

- Ficha de trabalho nº5