



CINEL

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

60h – Python

PROGRAMACAO EM PYTHON - INTERMÉDIO

julio.magalhaes@formandos.cinel.pt



Objetivos

- Desenvolver pequenos scripts.
- Conhecer tipos de armazenamentos mais complexos
- Biblioteca gráfica



Python

- 2001 criada Python Software Foundation.
- Apoiada pela Microsoft, Google, ...
- É incorporada em Sistemas Operativas 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



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 (+ , *,)



Operações sobre números inteiros

<i>Operação</i>	<i>Tipo dos argumentos</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)



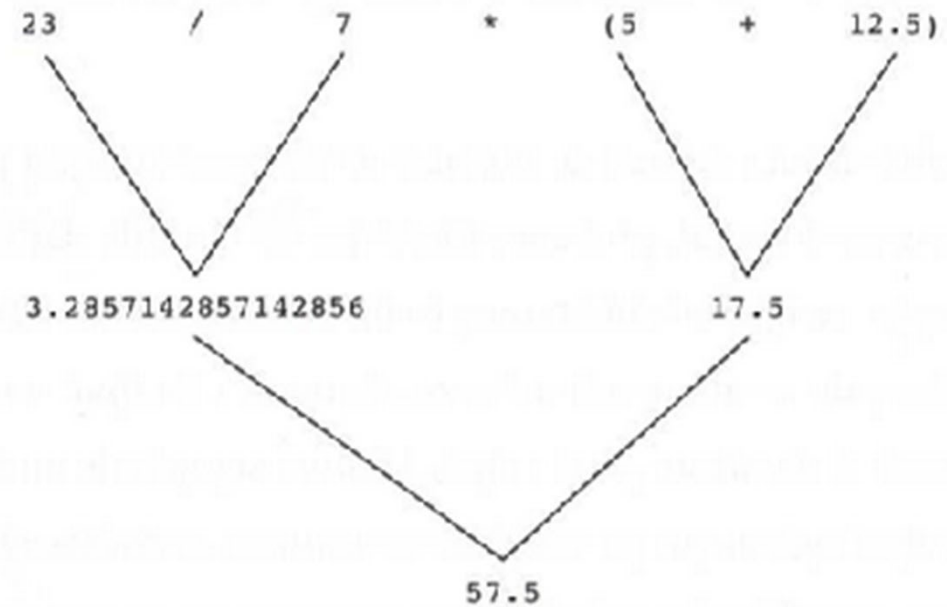
Operadores relacionais

<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 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 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'



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



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
```

```
'''
```



Carateres especiais

<i>Caráter 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



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

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



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
round, arredonda um número a quantas casas decimais pretendemos

Comando: import math

Bibliotecas disponível em <https://docs.python.org/3.10/py-modindex.html>



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, ceil**

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



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("O quadrado de {} é {}".format(valor,resposta))
```



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"
- 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 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	_					



Tuplas

- Tupla (tuple) é uma sequência de elementos
- Correspondem à noção matemática de vetor (variável indexada)
- Exemplos de tuplas em python:
 - `()`, `(1,2,3)`, `(2, True)`, `(5,)`
- Nota: `(5,)` é uma tupla com apenas 1 elemento, o nº 5.
- Atribuição de uma tupla a uma variável:
 - `notas = (13, 10, 9, 12)`



Tuplas

- notas = (13, 10, 9, 12)
- notas[0] # valor da tupla na posição 0 -> 13
- notas[-2] # 2º valor da tupla a começar do fim -> 9
- Podemos trabalhar com variáveis na indexação
 - i=1
 - print(notas[i+1]) # escreve o que está no índice 1+1 -> 9
- Erro na indexação:
 - notas[i+10] # não existe índice 11, logo obtemos um erro
#IndexError: tuple index out of range



Tuplas

- Um elemento de uma tupla, pode ser também uma tupla:
 - `a = ((1,2,3) , 4 , (5,6))` #tupla "a" tem 3 elementos
- `print(a[0])` # devolve o elemento do índice 0 -> (1,2,3)
- Como referir o elemento da posição 1 do elemento índice 0 da tupla "a"?
 - `a[0][1]` # nº 2



Tuplas

- Tuplas em Python são imutáveis, ou seja, podemos manipular/operar os elementos mas não podemos redefinir os seus elementos.
- `a = ((1,2,3) , 4 , (5,6))`
- Se fizermos `a[1] = 10`, ou seja, em vez do 4 queremos substituir por 10, obtemos um erro. `TypeError: 'tuple' object does not support item assignment`



Manipulação de tuplas

<i>Operação</i>	<i>Tipo dos argumentos</i>	<i>Valor</i>
$t_1 + t_2$	Tuplos	A concatenação dos tuplos t_1 e t_2 .
$t * i$	Tuplo e inteiro	A repetição i vezes do tuplo t .
$t[i_1:i_2]$	Tuplo e inteiros	O sub-tuplo de t entre os índices i_1 e $i_2 - 1$.
$e \text{ in } t$	Universal e tuplo	True se o elemento e pertence ao tuplo t ; False em caso contrário.
$e \text{ not in } t$	Universal e tuplo	A negação do resultado da operação $e \text{ in } t$.
<code>tuple(a)</code>	Lista ou dicionário ou cadeia de caracteres	Transforma o seu argumento num tuplo. Se não forem fornecidos argumentos, devolve o tuplo vazio.
<code>len(t)</code>	Tuplo	O número de elementos do tuplo t .



Listas

- É um novo tipo de estruturar os dados.
- É uma sequência de elementos de qualquer tipo.
- Muito semelhante aos tuplos
- Uma lista é um tipo de variável **MUTÁVEL**
- A ideia de lista é muito semelhante aos arrays e matrizes de outras linguagens.
- Exemplo de listas:
 - `[1, 2, 3]`, `[2, (1,2)]`, `['a']`, `[]`, `[1, [2], [[3]]]`
- Ao contrário dos tuplos, uma lista com 1 elemento não precisa da `,'



Listas

<i>Operação</i>	<i>Tipo dos argumentos</i>	<i>Valor</i>
$l_1 + l_2$	Listas	A concatenação das listas l_1 e l_2 .
$l * i$	Lista e inteiro	A repetição i vezes da lista l .
$l[i_1:i_2]$	Lista e inteiros	A sublista de l entre os índices i_1 e $i_2 - 1$.
<code>del(els)</code>	Lista e inteiro(s)	Em que <i>els</i> pode ser da forma $l[i]$ ou $l[i_1:i_2]$. Remove os elemento(s) especificado(s) da lista l .
<code>e in l</code>	Universal e lista	True se o elemento e pertence à lista l ; False em caso contrário.
<code>e not in l</code>	Universal e lista	A negação do resultado da operação <code>e in l</code> .
<code>list(a)</code>	Tuplo ou dicionário ou cadeia de caracteres	Transforma o seu argumento numa lista. Se não forem fornecidos argumentos, o seu valor é a lista vazia.
<code>len(l)</code>	Lista	O número de elementos da lista l .



Listas

- `lst1=[1,2,3]` e `lst2=[[4,5]]`
- `lst = lst1 + lst2` # resulta na concatenação dos elementos
- `len(lst)` # qual o tamanho da lista `lst`?
- `lst[3]` # qual o elemento do índice 3 da lista `lst`?
- Para referenciar um elemento de outro que é uma lista, utilizamos uma sintaxe idêntica aos tuplos. Por exemplo, para aceder ao 1º elemento da lista '`lst`' que se encontra na 4ª posição (índice 3) utilizamos a referência `lst[3][0]` #4



Listas

- Mutabilidade:
 - `lst=[1,2,3,[4,5]]`
 - `lst[2]='a' # resulta em [1,2,'a',[4,5]]`
 - `del(lst[1]) #apaga elemento do índice 1 -> lst=(1,'a',[4,5])`
 - `del(lst[1:]) # apaga todos os elementos do índice 1 até ao fim`



Listas como copiar

- A atribuição de listas a listas é uma atribuição simples. Por exemplo, para `lst1=[1,2,3,4]`, e pretendermos uma cópia desta lista basta fazer a atribuição `lst2=lst1`.
- Ao procedermos a este tipo de atribuição, **atenção**:
 - `lst2[1]='a'`
 - `lst2` fica com os seguintes elementos -> `lst2 = [1,'a',3,4]`
 - Mas `lst1` também fica alterada -> `lst1 = [1,'a',3,4]`

A instrução `lst2=lst1` é como criar um link para a mesma lista
(`lst1` e `lst2` são "alias"/pseudónimos)



Listas

- Para fazer uma cópia do conteúdo de uma lista para outra:
 - `l1 = [1,2,3]`
 - `l2 = l1.copy()`
 - `l1[0] = "X" # ['X,2,3']`
 - `print(l2) -> resultado seria [1,2,3]`
- O método `copy()` copia o conteúdo de uma lista para outra e não o endereço de memória para onde aponta a primeira.



Adicionar/remover

- Podemos adicionar um novo valor a uma lista recorrendo ao método "append" ou através da concatenação de listas.
- Desta forma, a inserção no final da lista será do tipo:

```
lista = ["Preto", "Magenta", "Ciano"]
```

```
lista.append("Amarelo")    ou    lista = lista + ["Amarelo"]
```

- O resultado final seria a lista com os valores:

```
["Preto", "Magenta", "Ciano", "Amarelo"]
```

- Para remover uma ocorrência de um valor numa lista poderemos utilizar o método "remove":

```
lista.remove("Preto")
```



Proposta

- Ficha de exercícios nº 1