

NIAEFEUP

Introdução

Projetos



Aplicação
para
estudantes



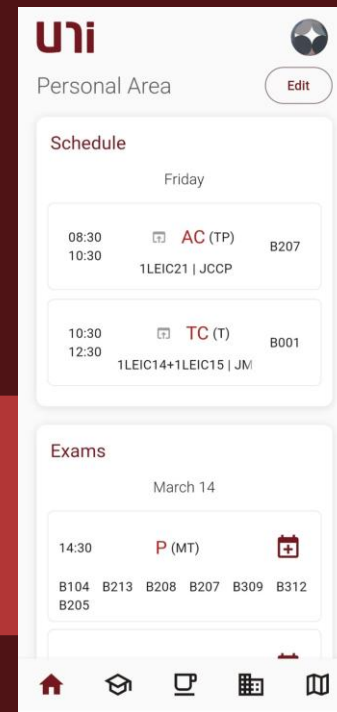
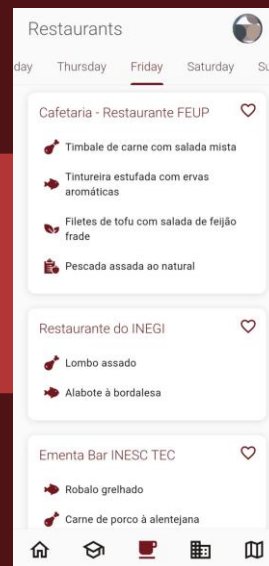
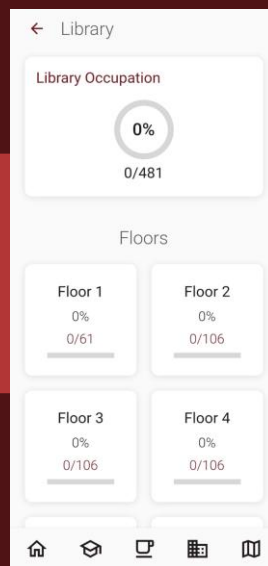
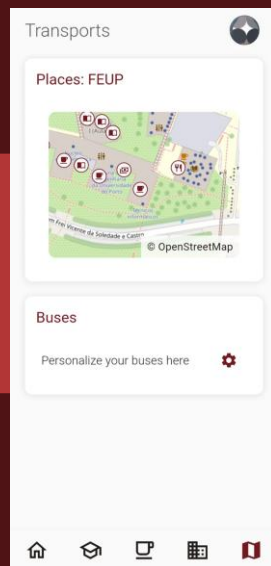
Website de
criação de
horários



Extensão browser para
melhorar o Sigarra

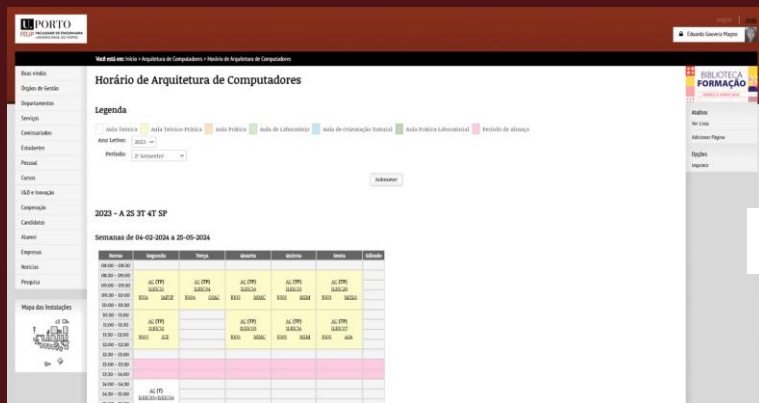
UNI

A aplicação móvel que permite aos estudantes fazer a consulta e gestão da sua vida académica. A consulta de informação como a próxima aula, o calendário de exames e a ementa semanal, torna-se um processo simples.



NitSig

Na Universidade do Porto usa-se o SIGARRA para tudo, mas a verdade é que não está à altura das expectativas dos estudantes...



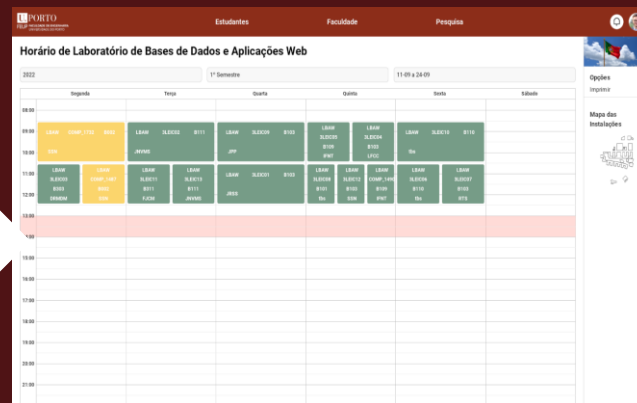
Horário de Arquitetura de Computadores

Legenda

2023 - A 23 3T 4T 5P

Semanas de 04-03-2024 a 25-05-2024

Horário	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
08:00 - 09:00						
09:00 - 10:00	AL (T)	AL (T)	AL (T)	AL (T)	AL (T)	AL (T)
10:00 - 11:00	AL (T)	AL (T)	AL (T)	AL (T)	AL (T)	AL (T)
11:00 - 12:00	AL (T)	AL (T)	AL (T)	AL (T)	AL (T)	AL (T)
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						
17:00 - 18:00						
18:00 - 19:00						
19:00 - 20:00						
20:00 - 21:00						
21:00 - 22:00						
22:00 - 23:00						
23:00 - 24:00						
24:00 - 25:00						
25:00 - 26:00						
26:00 - 27:00						
27:00 - 28:00						
28:00 - 29:00						
29:00 - 30:00						
30:00 - 31:00						
31:00 - 32:00						
32:00 - 33:00						
33:00 - 34:00						
34:00 - 35:00						
35:00 - 36:00						
36:00 - 37:00						
37:00 - 38:00						
38:00 - 39:00						
39:00 - 40:00						
40:00 - 41:00						
41:00 - 42:00						
42:00 - 43:00						
43:00 - 44:00						
44:00 - 45:00						
45:00 - 46:00						
46:00 - 47:00						
47:00 - 48:00						
48:00 - 49:00						
49:00 - 50:00						
50:00 - 51:00						
51:00 - 52:00						
52:00 - 53:00						
53:00 - 54:00						
54:00 - 55:00						
55:00 - 56:00						
56:00 - 57:00						
57:00 - 58:00						
58:00 - 59:00						
59:00 - 60:00						
60:00 - 61:00						
61:00 - 62:00						
62:00 - 63:00						
63:00 - 64:00						
64:00 - 65:00						
65:00 - 66:00						
66:00 - 67:00						
67:00 - 68:00						
68:00 - 69:00						
69:00 - 70:00						
70:00 - 71:00						
71:00 - 72:00						
72:00 - 73:00						
73:00 - 74:00						
74:00 - 75:00						
75:00 - 76:00						
76:00 - 77:00						
77:00 - 78:00						
78:00 - 79:00						
79:00 - 80:00						
80:00 - 81:00						
81:00 - 82:00						
82:00 - 83:00						
83:00 - 84:00						
84:00 - 85:00						
85:00 - 86:00						
86:00 - 87:00						
87:00 - 88:00						
88:00 - 89:00						
89:00 - 90:00						
90:00 - 91:00						
91:00 - 92:00						
92:00 - 93:00						
93:00 - 94:00						
94:00 - 95:00						
95:00 - 96:00						
96:00 - 97:00						
97:00 - 98:00						
98:00 - 99:00						
99:00 - 100:00						

Horário de Laboratório de Bases de Dados e Aplicações Web

Legenda

2023 - A 23 3T 4T 5P

Semanas de 04-03-2024 a 25-05-2024

Horário	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
08:00 - 09:00						
09:00 - 10:00	AL (T)	AL (T)	AL (T)	AL (T)	AL (T)	AL (T)
10:00 - 11:00	AL (T)	AL (T)	AL (T)	AL (T)	AL (T)	AL (T)
11:00 - 12:00	AL (T)	AL (T)	AL (T)	AL (T)	AL (T)	AL (T)
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						
17:00 - 18:00						
18:00 - 19:00						
19:00 - 20:00						
20:00 - 21:00						
21:00 - 22:00						
22:00 - 23:00						
23:00 - 24:00						
24:00 - 25:00						
25:00 - 26:00						
26:00 - 27:00						
27:00 - 28:00						
28:00 - 29:00						
29:00 - 30:00						
30:00 - 31:00						
31:00 - 32:00						
32:00 - 33:00						
33:00 - 34:00						
34:00 - 35:00						
35:00 - 36:00						
36:00 - 37:00						
37:00 - 38:00						
38:00 - 39:00						
39:00 - 40:00						
40:00 - 41:00						
41:00 - 42:00						
42:00 - 43:00						
43:00 - 44:00						
44:00 - 45:00						
45:00 - 46:00						
46:00 - 47:00						
47:00 - 48:00						
48:00 - 49:00						
49:00 - 50:00						
50:00 - 51:00						
51:00 - 52:00						
52:00 - 53:00						
53:00 - 54:00						
54:00 - 55:00						
55:00 - 56:00						
56:00 - 57:00						
57:00 - 58:00						
58:00 - 59:00						
59:00 - 60:00						
60:00 - 61:00						
61:00 - 62:00						
62:00 - 63:00						
63:00 - 64:00						
64:00 - 65:00						
65:00 - 66:00						
66:00 - 67:00						
67:00 - 68:00						
68:00 - 69:00						
69:00 - 70:00						
70:00 - 71:00						
71:00 - 72:00						
72:00 - 73:00						
73:00 - 74:00						
74:00 - 75:00						
75:00 - 76:00						
76:00 - 77:00						
77:00 - 78:00						
78:00 - 79:00						
79:00 - 80:00						
80:00 - 81:00						
81:00 - 82:00						
82:00 - 83:00						
83:00 - 84:00						
84:00 - 85:00						
85:00 - 86:00						
86:00 - 87:00						
87:00 - 88:00						
88:00 - 89:00						
89:00 - 90:00						
90:00 - 91:00						
91:00 - 92:00						
92:00 - 93:00						
93:00 - 94:00						
94:00 - 95:00						
95:00 - 96:00						
96:00 - 97:00						
97:00 - 98:00						
98:00 - 99:00						
99:00 - 100:00						

Criamos então esta extensão de *browser* que refina a experiência do estudante a utilizar o SIGARRA, tornando-o mais intuitivo e prático.

TTS

Um dos mais antigos projetos do núcleo. Usado para criar os horários perfeitos, seja com as desejáveis tardes e dias livres ou os professores preferidos.



O TTS é:

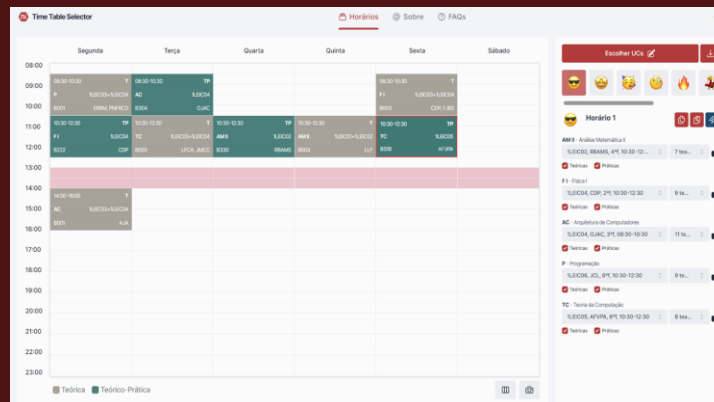
Simple

Customizável

Flexível

Compartilhável

Assistente Automático



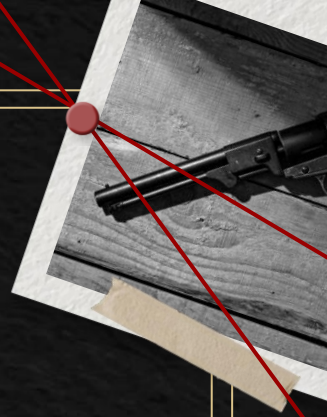
Eventos

Convívios & Workshops para toda a comunidade académica



Introdução a Python: Bug Detectives

Workshop



um abridor para o
grufo feaver buem
- go Serme
ila su aespem
em me fany um
um buem pora esuf

INDÍCE





FERRAMENTAS



VS Code



Pycharm

O PRIMEIRO PROGRAMA – “HELLO WORLD!”



1. Cria um ficheiro *main.py*
2. Copia este código para esse ficheiro
3. Vê o que é possível fazer com Python!

O PRIMEIRO PROGRAMA – “HELLO WORLD!”



```
print("Hello World!")
```

**Imprime o texto
na consola do
interpretador**

O QUE É UMA VARIÁVEL?



```
greeting = "Hello"  
animal = "panda"  
age = 23
```

"Contentor" onde
• **podemos armazenar**
qualquer tipo de dados

TIPOS DE DADOS & OPERADORES LÓGICOS



str



int



float



bool

OPERADORES ARITMÉTICOS



Operadores	Funcionalidade
+	soma
-	subtração
*	multiplicação
/	divisão
**	potência
//	divisão inteira
%	resto da divisão

EXERCÍCIO 01



- Pede ao utilizador, através da função `input()`, que digite dois números (podem ser inteiros ou decimais).
- Calcula e exhibe a soma, a diferença e o produto.

EXERCÍCIO 02



- Pede ao utilizador dois argumentos - um número e uma string
- Transforma o número em uma string
- Dá print da soma dos resultados e verifica o que acontece

Dica: Podes usar a função `type()` para verificar o tipo da variável.

OPERADORES BOOLEANOS



Operadores	Funcionalidade
==	Objetos iguais?
!=	Objetos diferentes?
>	Objeto maior?
>=	Objeto maior ou igual?
<	Objeto menor?
<=	Objeto menor ou igual?

EXERCÍCIO 03



- Escreve um programa que recebe dois números, e que avalie se o primeiro é maior que o segundo, se são iguais e se o primeiro é menor que o segundo (três prints)

CONDIÇÕES

if

```
if(2 > 6):  
    print("ola")
```

elif

```
elif(7 % 2 == 0):  
    print("aham")
```

else

```
else:  
    print("uhuu")
```

EXERCÍCIO 04



- Faz o mesmo que no exercício anterior, mas agora dá print apenas ao resultado. Se for maior escreve que é maior, se for menor escreve que é menor e se for igual escreve que é igual

CONDIÇÕES

and

or

not

```
2 > 6
```

```
10 % 2 == 0
```

EXERCÍCIO 05



- Dada uma idade, de print se a pessoa é considerada uma:
 - Criança (0 a 18)
 - Adulto (18 a 65)
 - Idoso (65+)

ESTRUTURAS DE REPETIÇÃO (LOOPS)

Loop	Execução
while <condição>:	Executa um conjunto de instruções um número indefinido de vezes - enquanto uma condição específica se verificar, o código presente no body do loop vai correr
for <variável> in <sequência>:	Executa um conjunto de instruções um número específico de vezes

ESTRUTURAS DE REPETIÇÃO (LOOPS)

Keywords	Funcionalidade
continue	Permite fazer o loop saltar para a próxima iteração do ciclo, ignorando todo o código que se lhe segue na iteração atual
break	Permite sair do ciclo a meio da sua execução (especialmente útil para interromper loops que, caso contrário, seriam infinitos)

EXERCÍCIO 06



- Inicializa uma variável com o número zero e, utilizando um *while loop*, acrescenta-lhe um valor até que chegue ao número **10**
- Faz isto novamente, mas utilizando um *loop* com o statement ***while True***

Pergunta extra: Em que casos achas que seria útil usar um `while True`?

EXERCÍCIO 07



- Dá print a todos os números **entre 0 e 9** utilizando um *for loop*
- Dá print a todos os números **entre 5 e 15 de dois em dois**, utilizando um *for loop*

EXERCÍCIO EXTRA



Agora usando todo o seu conhecimento vamos fazer o famoso exercício chamado FizzBuzz, no qual dado um número qualquer (input) de 0 até ao número dado:

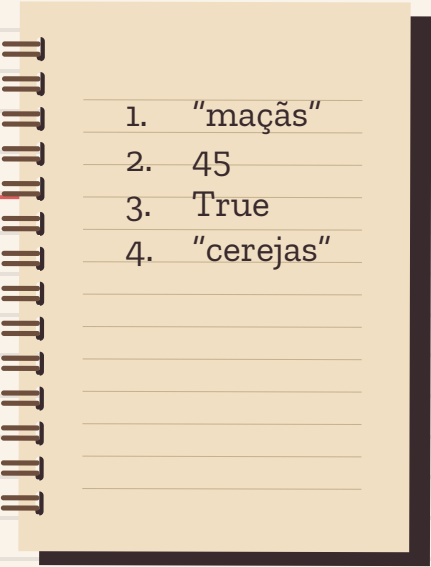
- Se o número for **divisível por três e por cinco**, dar print à string **fizzbuzz**
- Se for **apenas divisível por três**, dar print à string **fizz**; se for **apenas divisível por cinco**, dar print à string **buzz**
- Se **não for divisível nem por três nem por cinco**, dá print do **respectivo número**

LISTAS

[]

Índice

Este operador permite-nos **aceder a qualquer elemento de uma lista** (nota por que número começa a nossa lista)!

- 
1. "maçãs"
 2. 45
 3. True
 4. "cerejas"

Se quiseres começar pelo fim da lista, podes sempre utilizar números negativos — **[-1]** corresponde ao último elemento.



LISTAS

Métodos	Funcionalidade
<code>len(<lista>)</code>	Retorna o tamanho da lista
<code>max(<lista>) / min(<lista>)</code>	Retorna o maior/menor elemento da lista
<code>sort(<lista>)</code>	Ordena a lista

LISTAS



Métodos	Funcionalidade
sum(<lista>)	Retorna a soma dos elementos da lista
.append(<valor>)	Acrescenta à lista o valor dado, na última posição (fim da lista)
.pop(<índice>)	Retira da lista o elemento com o índice especificado
.count(<valor>)	Retorna o número de elementos com o valor especificado



MISTÉRIO DOS NÚMEROS ROUBADOS



Vocês são uma agência de investigação contratada para apurar o desaparecimento de alguns números preciosos de uma coleção muito rara. A lista original continha os números de 1 a 10, mas misteriosamente alguns desapareceram e outros foram colocados no seu lugar.

O vosso trabalho consiste em:

1. Identificar quais os números que estão em falta.
2. Verificar se existe algum número suspeito que se repete mais do que uma vez.
3. Determinar se a soma total dos números ainda presentes é par ou ímpar.
4. Classificar os números da seguinte forma:
 - Pequenos (1 a 3)
 - Médios (4 a 7)
 - Grandes (8 a 10)

Python ▾

```
1 colecao_encontrada = [1, 2, 3, 5, 5, 7, 9, 10] # Lista fornecida via código ou input
2 # 1. Identificar números faltando
3 # 2. Identificar duplicatas
4 # 3. Soma e par/ímpar
5 # 4. Classificar por tamanho
```



MISTÉRIO DOS NÚMEROS ROUBADOS



Números que faltam: [4, 6, 8]

Números que suspeitos: [5]

A soma 42 é par

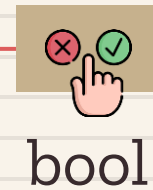
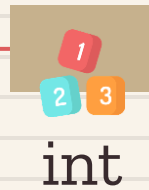
Classificação:

Pequenos [1, 2, 3]

Medios [5, 5, 7]

Grandes [9, 10]

STRINGS





STRINGS


Métodos	Funcionalidade
<code>.upper()</code>	Retorna uma string com todos os caracteres maiúsculos
<code>.lower()</code>	Retorna uma string com todos os caracteres minúsculos
<code>.find(<string>)</code>	Encontra uma string dentro de uma outra string original, retornando o índice do valor especificado (retorna -1 se não encontrar nada)





STRINGS

Métodos	Funcionalidade
<code>.replace(<string>, <string>)</code>	Substitui todas as ocorrências de uma determinada sequência por outra, na string original
<code>.count(<string>)</code>	Retorna o número de ocorrências de um determinado valor dentro da string



EXERCÍCIO 08



- Dada uma frase com letras maiúsculas e minúsculas, conte a quantidade de letras minúsculas

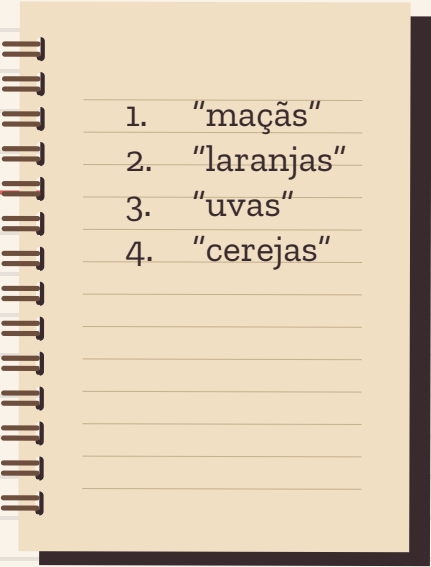
“BemvindosaoWoRksHop!”

TUPLOS

{“OLÁ”,}

Não é apenas
uma lista?

De facto, são muito
semelhantes a listas,
embora com uma grande
diferença: são estruturas
de dados **imutáveis** —
assim que são criados não
podem ser alterados de
qualquer forma.

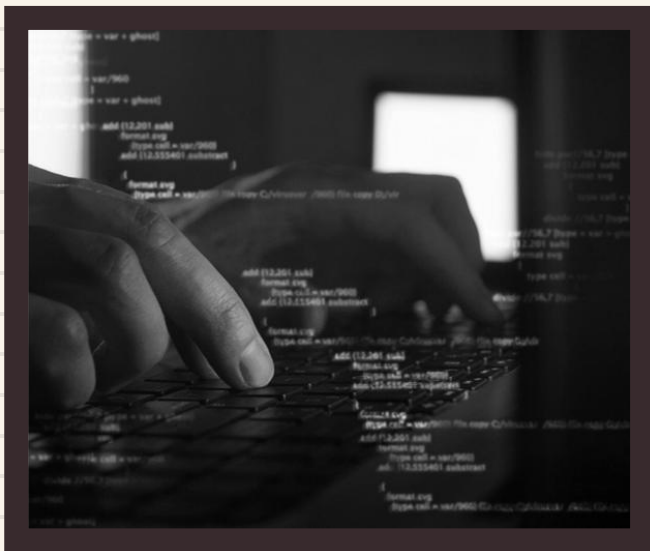
- 
1. “maçãs”
 2. “laranjas”
 3. “uvas”
 4. “cerejas”

Partilham muitos
métodos com listas;
**teoricamente é possível
modificar um tuplo se o
convertermos para uma
lista.**





BIBLIOTECAS – REUTILIZAR CÓDIGO



Milhares de funções em Python **são desenvolvidas diariamente.**

Podemos facilitar o desenvolvimento **reutilizando o código desenvolvido por outros.**

Esse código encontra-se dentro de **bibliotecas**, que são um **ficheiro** (ou conjuntos de ficheiros) **com diversas funções, métodos, classes...**


`import <nome_biblioteca>`





RANDOM

Métodos	Funcionalidade
<code>.choice(<lista>)</code>	Escolhe um elemento aleatório de uma sequência (lista, tupla, string)
<code>.randint(a, b)</code>	Retorna um número inteiro entre a e b
<code>.random()</code>	Retorna um float aleatório no intervalo 0.0 e 1.0



EXERCÍCIO 09



- Importe a biblioteca random e crie um jogo em que o utilizador deve adivinhar um número escolhido aleatoriamente entre 0 e 100. Para cada palpite, informe se o valor chutado é maior ou menor que o número secreto. Conte também o número de tentativas necessárias para acertar.

No final, exiba: “Acertou em # tentativas”

DICIONÁRIOS

```
{ "OLÁ" : 0 }
```

E agora?

Também muito **semelhantes a listas** (utilizam parêntesis diferentes), mas em vez de serem indexados somente por números, **podem ser indexados por qualquer tipo de variável** (números, strings...)

"cor" → "azul"
18 → "idade"
"fruta" → "uva"

Uma entrada de um dicionário designa-se por um par **Chave : Valor**. As chaves permitem aceder aos valores, mas o contrário não se verifica (funciona quase como uma tabela).



DICIONÁRIOS

Métodos	Funcionalidade
.keys()	Retorna um objeto com as chaves de um dicionário
.values()	Retorna um objeto com os valores de um dicionário
.items()	Retorna um objeto com os pares Chave: Valor em tuplos
.get(<chave>)	Retorna o valor de uma chave específica

EXERCÍCIO 10



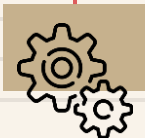
- Dada uma turma de estudantes, nos foi proposto calcular a média geral da classe.

Sala = {"lulu": 19, "juju": 17, "lolo": 18}

FUNÇÕES

def

Definir uma função



nome

Nome da função



(<argumentos>)

Objetos que a função
deve receber



FUNÇÕES

```
def media(variavel):  
    resultado = 0  
    for i in variavel:  
        resultado += i  
  
    return resultado / len(variavel)  
  
turma = [15, 17, 14]  
  
print("Turma antes:", round(media(turma)))  
  
turma.append(18)  
  
print("turma depois:", media(turma))
```

EXERCÍCIO 11



- Dada uma turma de estudantes, nos foi proposto calcular a média geral da classe.

Val não obteve a nota que desejava; por isso, realizou um segundo teste e obteve 17. Calcule a média da turma antes e depois de acrescentar Val a sala.

Sala = {"lulu": 19, "juju": 17, "lolo": 18}

Obs: Crie um função para calcular as médias

O ÚLTIMO CASO



- Você é o detetive encarregado de descobrir quem roubou a joia da família West. Há três suspeitos, cada um com um álibi e uma impressão digital encontrada na cena do crime. Seu programa deve cruzar informações e indicar o culpado.

O ÚLTIMO CASO



`filtrar_por_alibi(suspeitos, texto)`

Retorna uma lista de nomes cujos álibis **contêm** o `texto` dado (ex.: “cinema”).

`contar_pistas(evidencias)`

Retorna um dicionário `{impressao: quantidade}` contando quantas vezes cada impressão apareceu.

`identificar_suspeito(suspeitos, pistas_contadas)`

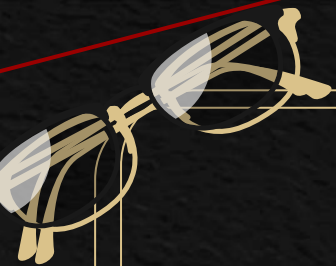
Retorna o nome do suspeito cuja impressão aparece **mais vezes** em `pistas_contadas`.

- Se nenhuma impressão bater, retorne `"Desconhecido"`.

`relatorio_final(suspeitos, evidencias)`

Função geral que:

- chama as anteriores,
- imprime um relatório amigável, ex.:



Obrigado!

Continuação de uma ótima jornada!

um abridor para o
grufo feaver buem
- go serme
ila su aespem
em me fany um
um buem porar esuf