



# PROJECTO 1

## EFEITOS ESPECIAIS (Beta)

---

### INTRODUÇÃO E OBJECTIVOS

A finalidade deste projecto passa por consolidar os conhecimentos de programação em Python adquiridos até agora, através da implementação de dois programas de natureza diferente. Um objectivo secundário deste projecto consiste em introduzir a programação orientada para sistemas recorrendo, para tal, a bibliotecas desenvolvidas por terceiros.

Na primeira parte do projecto, deve implementar um conjunto de algoritmos para produzir "efeitos especiais" em texto. Na segunda parte, pretende-se que implemente um dos utilitários seguintes: programa de encriptação; listagem de conteúdo de directoria em árvore; obtenção de endereços de rede da camada MAC.

Alguns elementos do projecto são considerados "extra", sendo resumidos na secção **AValiação**. Nesta secção encontra também uma explicação sobre o que se pretende dizer com "extra" no contexto deste projecto.

### PARTE I - EFEITOS ESPECIAIS

Nesta parte deve desenvolver um programa em Python para exibir o texto introduzido na linha de comandos de acordo com determinados "efeitos especiais" (ver à frente). O seu programa deve ser invocado da seguinte forma (propositadamente, neste e noutros programas deste enunciado omitimos o interpretador; ou seja, consoante a sua plataforma poderá ter que prefixar o script com `python` ou `python3`):

```
$ efeitos.py palavra1 [palavra2] ... [palavraN]
```

Depois aplica cada um dos efeitos em baixo descritos ao texto que resulta da concatenação de `palavra1`, `palavra2`, etc. Consoante a sua plataforma, poderá ter que invocar directamente o interpretador

Os caracteres do texto (que resultam da concatenação de todas as palavras) devem ser exibidos individualmente.

Segue-se uma demonstração dos efeitos pretendidos. A não ser que exista alguma indicação noutro sentido, assumo que o programa foi invocado da seguinte forma:

```
$ efeitos.py FRASCO AZUL
```

## Efeito 1- Diagonal Esquerda

[illegible]

### Efeito 2 - Diagonal Direita, Texto Invertido

A 10x10 grid with letters placed at regular intervals along the main diagonal from bottom-left to top-right. The letters are: F (row 9, col 1), R (row 8, col 2), A (row 7, col 3), S (row 6, col 4), C (row 5, col 5), O (row 4, col 6), A (row 3, col 7), Z (row 2, col 8), U (row 1, col 9), and L (row 0, col 10).

### Efeito 3 - Diagonais Cruzadas (num. cars. ímpar)

F								F
	R							R
		A					A	
			S			S		
				C		C		
					O			
			A				A	
		Z					Z	
	U							U
L								L

**Efeito 3 - Texto:** "FRASCO VERDE" (num. cars. par)

F									F
	R								R
		A						A	
			S				S		
				C			C		
					O	O			
				V			V		
			E				E		
		R						R	
	D								D
E									E

**Efeito 4 - Em V, Texto: "ALBERTO**

A										O
	L									T
		B							R	
			E					E		
				R					B	
					T			L		
						O	A			

**Efeito 5 - Em Escada, Palavras por Ordem Inversa. Texto: "FRASCO AZUL ABERTO"**

A	B	E	R	T	O		
	A	Z	U	L			
		F	R	A	S	C	O

**Efeito 6 - Texto Deslizante em Ciclo ao Longo da linha. Texto: "ALBERTO"**

<i>t</i>	<i>Linha de texto exibida</i>							
0.0	A	L	B	E	R	T	O	
0.5		A	L	B	E	R	T	O
1.0			A	L	B	E	R	T
	...							
3.5	O					A	L	B
4.0	T	O					A	L

A coluna da esquerda representa o tempo decorrido desde o início do efeito. Estamos a assumir um intervalo de tempo entre exibição de linhas de 0.5s. Todas as quadriculas livres são preenchidas com espaços (caractere ' ') Note que o texto NÃO deve ser exibido em escada como dá a entender esta imagem estática. Cada uma das linhas fica "por cima" da anterior, dando a ilusão que o texto se está a mover para a direita. Veja também a demonstração que acompanha este enunciado.

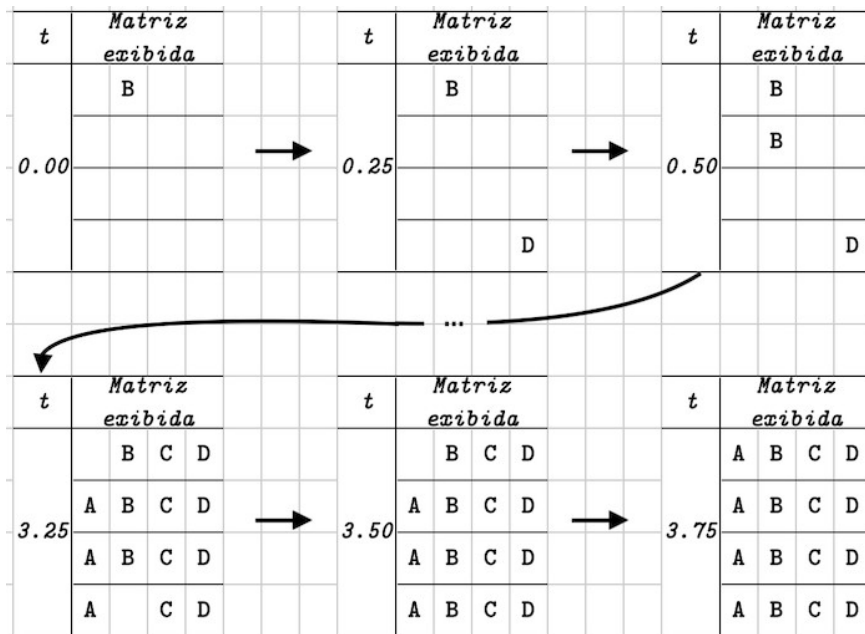
**Efeito 7 - Destapa Caracteres em Posições Aleatórias. Texto: "ALBERTINA"**

<i>t</i>	<i>Linha de texto exibida</i>							
0.00						I		
0.25		B				I		
0.50		B		R		I		
...	...							
1.50	L	B	E	R		I	N	A
1.75	A	L	B	E	R		I	N
2.00	A	L	B	E	R	T	I	N

<i>t</i>	<i>Linha de texto exibida</i>							
0.00	L							
0.25	L			T				
0.50	L			R	T			
...	...							
1.50	A	L		E	R	T		N
1.75	A	L		E	R	T	I	N
2.00	A	L	B	E	R	T	I	N

Em cima vemos duas possibilidades para o texto indicado. Estamos a assumir um intervalo de tempo entre exibição de linhas de 0.25s. Quadriculas livres preenchidas com espaços. Cada uma das linhas fica "por cima" da anterior, ou seja, em cada instante só vemos uma linha. No instante final (2.00, neste exemplo) vemos uma linha com o texto completo. Veja também a demonstração que acompanha este enunciado.

## Efeito 8 - Destapa Matriz em Posições Aleatórias. Texto: "ABCD"



Em cima, de forma resumida, tentamos ilustrar a exibição do texto. Quadriculas livres são preenchidas com espaços. Assume-se um intervalo de tempo entre exibição de matrizes de 0.25s. Veja também a demonstração que acompanha este enunciado.

## CONSIDERAÇÕES E RECOMENDAÇÕES

Para o efeito da linha deslizante, assuma que cada linha tem 40 caracteres de largura e que o intervalo entre exibição de linhas (para que seja visível o efeito deslizante) é 0,1 segundos. Quando "está sob este efeito", o programa termina apenas quando o utilizador o interrompe com CTRL+C (pergunte ao formador como detectar a inserção do CTRL+C). Como vai ter que inserir o texto na linha de 40 caracteres, sugere-se que crie uma lista com 40 caracteres (*qual o caractere com que deve inicializar a lista?*) e insira o texto nesta lista nas posições apropriadas. Depois, antes de exibir esta lista, deve transformá-la numa string com o método `str.join`. Veja o seguinte exemplo:

```
txt = 'FRASCO AZUL'
i = 5
line = ['.'] * 40
line[i:i + len(txt)] = txt
print(''.join(line))

i += 1
line = ['.'] * 40
line[i:i + len(txt)] = txt
print(''.join(line))
```

Para os restantes efeitos que dependem de um temporizador para exibir o texto, assuma também que intervalo de tempo é 0,1 segundos. Todavia, este intervalo pode ser parametrizado como indicado à frente.

Para "temporizar" a execução do código em Python pode utilizar a função `sleep` definida no módulo `time`. Esta função recebe um *float* com indicação do intervalo de tempo em segundos. Por exemplo, `sleep(1)` ou `sleep(1.0)` suspendem o processo durante 1s, ao passo que `sleep(1.5)` suspende durante um segundo e meio, e `sleep(0.001)` suspende durante um milissegundo. Para apagar o ecrã, sugere-se a invocação do comando `clear` em sistemas do tipo Unix (macOS, Linux, FreeBSD, etc.) ou o comando `cls` em sistemas Windows. Eis um exemplo que utiliza estes mecanismos:

```
import time
import subprocess

subprocess.run(['clear'])      # substituir subprocess.run(['clear'])
print("ALBERTO...")          # por subprocess.run(['cls'], shell=True) em
time.sleep(0.5)               # Windows
subprocess.run(['clear'])
print(".ALBERTO..")
time.sleep(0.5)
subprocess.run(['clear'])
print("..ALBERTO.")
time.sleep(0.5)
subprocess.run(['clear'])
```

Como alternativa a apagar o ecrã, pode simular este efeito escrevendo sempre na linha inicial. Isto é possível se chamar a função `print` com `end = '\r'` (em vez, de utilizar o valor por omissão `'\n'`). Exprimente:

```
import time
print("ALBERTO...", end='\r')
time.sleep(0.5)
print(".ALBERTO..", end='\r')
time.sleep(0.5)
print("..ALBERTO.", end='\r')
time.sleep(0.5)
print()
```

Utilize o módulo `random` para os efeitos que "destapam" posições do texto de forma aleatória. Uma função que pode ser particularmente útil é a função `random.shuffle`. Esta função "baralha" uma lista de objectos. Por exemplo, dada a string `txt`, eis como gerar uma lista com posições aleatórias que depois pode utilizar para aceder a `txt` de forma aleatória:

```
import random
txt = 'ALBERTO'
random_positions = list(range(len(txt)))
random.shuffle(random_positions)
# agora podemos aceder a txt utilizando as posições na lista random_positions
```

De um modo geral, exiba um separador apropriado entre todos os efeitos: faça uma pausa entre efeitos, seguida de um "apagar" de ecrã, para que seja visível o efeito em questão.

## INTERFACE COM O UTILIZADOR

O programa deve aceitar o parâmetro `-i` com o valor do intervalo de actualização em segundos. O valor por omissão deste parâmetro deve ser 0.1s. Eis como pode exibir ALBERTO a deslizar para a direita com um intervalo de actualização de 0.25s:

```
$ efeitos.py -i 0.25 ALBERTO
```

O programa deve começar por exibir o menu indicado à frente exactamente como demonstrado à frente.

As opções numeradas exibem o efeito correspondente em ecrã à parte, isto é, o programa deve apagar o ecrã, exibir o efeito pretendido e depois "fazer uma pausa". Após a pausa, o menu é exibido novamente.

A opção T leva a que todos os efeitos sejam exibidos. Deve colocar uma pausa seguida de um apagar de ecrã entre efeitos.

A introdução de uma opção não contemplada no menu deve ser assinalada ao utilizador como sendo uma opção inválida.

```
*****
*                                     *
*  EFEITO                           *
*                                     *
*   1 - Diagonal Esquerda           *
*   2 - Diagonal Direita, Texto Invertido *
*   3 - Diagonais Cruzadas          *
*   4 - Em V                        *
*   5 - Escada, Palavras Ordem Inversa *
*   6 - Deslizante                  *
*   7 - Destapa Posições Aleatórias  *
*   8 - Destapa Matriz              *
*   T - Todos                       *
*   E - Encerrar                    *
*                                     *
*****
```

OPÇÃO>

## PARTE II - PROGRAMA À ESCOLHA

Implemente um ou mais dos dos utilitários em baixo descritos, de acordo com as instruções do formador.

### CIFRA DE VIGENÈRE

A cifra de Vigenère é um método de encriptação equivalente a aplicar uma série de cifras de César em sequência, cada cifra com diferentes valores de deslocamento. Estes deslocamentos são estabelecidos por uma palavra-passe. O primeiro byte da palavra-passe indica o deslocamento a aplicar ao primeiro byte da sequência, o segundo byte da palavra-passe indica o deslocamento a aplicar ao segundo byte da sequência, e assim sucessivamente.

Habitualmente, esta encriptação é feita por meio de uma tabela designada de Régua ou Tábua de Vigenère (Vigenère Square) ou por Tabula Recta. O algoritmo e a utilização desta estrutura de dados são descritos em pormenor em [https://en.wikipedia.org/wiki/Vigen%C3%A8re\\_cipher#Description](https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher#Description).

O comando **vcypher** permite encriptar e desencriptar um ficheiro com a cifra de Vigenère. A sintaxe do comando deve ser a seguinte:

```
$ vcypher.py -e [-p PASSWORD] FILE
$ vcypher.py -d -p PASSWORD FILE
```

Com a opção **-e** o **vcypher** encripta o ficheiro dado por **FILE** utilizando para tal a password passada para a opção **-p**. No caso de não ser indicado nenhuma com a opção **-p**, o programa gera uma password aleatória com 6 caracteres legíveis da gama ASCII, devendo no final exibir na saída padrão qual o valor dessa palavra-passe. Com a opção **-d**, o **vcypher** desencripta o ficheiro **FILE**. Neste caso, a palavra-passe é obrigatória. Este comando deve suportar as opções longas **--encrypt**, **--decrypt** e **--password** como alternativa às opções curtas **-e**, **-d** e **-p**.

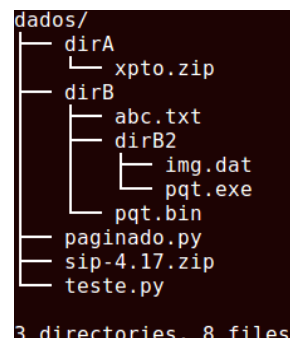
Para iludir possíveis intrusos, a extensão do ficheiro codificado deve ser **.pdf**. Utilize a biblioteca **docopt** ou **argparse** para ler os argumentos da linha de comandos.

### TREEP

Este comando, um clone simplificado do comando **tree** utilizado em Linux, lista o conteúdo de uma directoria num formato semelhante ao indicado na figura seguinte. A sintaxe de invocação é a seguinte:

```
$ treep.py [-df] [-L <level>] [-H] <caminho_dir>
```

Consulte a ajuda do comando **tree** (**\$ man tree** em Linux, também disponível na Web) para obter o significado das opções **-d**, **-f** e **-L**. Deve utilizar a biblioteca **argparse** para análise dos argumentos da linha de comandos. Como elemento extra, pode desenvolver a opção **-H** para exibir a listagem em HTML/CSS. Também aqui a saída deve similar à do **tree**.



```
dados/
├── dirA
│   └── xpto.zip
├── dirB
│   ├── abc.txt
│   ├── dirB2
│   │   ├── img.dat
│   │   └── pqt.exe
│   └── pqt.bin
├── paginado.py
├── sip-4.17.zip
└── teste.py
3 directories, 8 files
```

## L2ADDR

O objectivo deste exercício consiste em desenvolver um programa para obter endereços MAC (endereços nível 2) a partir de uma lista de endereços IPv4 passada a partir da linha de comandos. O programa deverá ser invocado da seguinte forma:

```
$ l2addr.py [-r] ipv4_addr1 [ipv4_addr2...]
```

O parâmetro opcional `-r` "refresca" a cache de IPs através do comando `ping` (ver descrição em baixo).

Em Linux, macOS e Windows, o comando `arp` permite obter a informação pretendida:

```
$ arp -n 192.168.56.1
```

Saída do comando `arp` em Linux:

<i>Address</i>	<i>HWtype</i>	<i>HWaddress</i>	<i>Flags</i>	<i>Mask</i>	<i>Iface</i>
<i>192.168.56.1</i>	<i>ether</i>	<i>0a:00:27:00:00:00</i>	<i>C</i>		<i>eth0</i>

Saída em macOS:

```
? (192.168.56.101) at a:0:27:0:0:0 on vboznet0 ifscope [ethernet]
```

De modo a garantir que o endereço IPv4 se encontra na cache ARP, é conveniente "pingar" primeiro o destino (mas limitando o número de pacotes ICMP a enviar).

Para invocar os comandos `ping` e `arp` utilize a função `popen` (Process Open) do módulo `os`. Esta função invoca um programa lançando um processo e criando uma pipe entre a saída padrão do processo invocado e o processo invocador (que é o script em Python que estamos a desenvolver). Acedemos à pipe como a um ficheiro de texto, uma vez que `os.popen` devolve um file object, isto é, um objecto que representa um ficheiro de texto.

Utilize o módulo `re` (regular expressions) para extrair os endereços de nível 2 através de uma expressão regular apropriada.

## ANTHIMIO

Este programa lê um ficheiro em formato CSV contendo dados climatéricos diários (ver tabela em baixo) e apresenta as seguintes opções ao utilizador:

OPÇÃO		DESCRIÇÃO
1	Temperatura mais elevada	Indica a data em que foi atingida a temperatura mais elevada.
2	Temperatura mais baixa	Indica a data em que foi atingida a temperatura mais baixa.
3	Temperatura média anual	Média anual das temperaturas médias diárias. A lista de anos é ordenada descendentemente por temperatura média.
4	Temperatura mediana anual	Mediana das temperaturas médias diárias por ano. A lista de anos é ordenada descendentemente por temperatura.

OPÇÃO		DESCRIÇÃO
5	Dois anos mais quentes	Indica os dois anos mais quentes em termos da temperatura média anual. Os anos são indicados por descendente de temperatura.
6	Dois anos mais frios	Indica os dois anos mais frios em termos da temperatura média anual. Os anos são indicados por ascendente de temperatura.
7	Precipitação média anual	Para cada ano, indica a precipitação média médias anuais. A lista de anos é ordenada descendente por temperatura.
8	Dois anos com maior precipitação	Indica os dois anos com maior precipitação. Os anos são indicados por descendente de precipitação.
9	Dois anos com menor precipitação	Indica os dois anos com menor precipitação. Os anos são indicados por descendente de precipitação.
C	Consultar	Devolve todos os registos entre duas datas. Se a segunda data não for indicada, devolve apenas o registo correspondente à primeira data.
G	Guardar cópia	Guarda uma cópia dos dados de entrada. Os dados são guardados no formato NOAA. Um novo nome para o ficheiro deverá ser solicitado ao utilizador. Caso este não indique novo nome, o nome da cópia será nome original + '.' + palavra 'copy' + '.csv'.
R	Relatório	Produz o ficheiro <code>relatorio.txt</code> com todas as "estatísticas" em cima apresentadas. O formato deste ficheiro fica ao seu critério.
T	Terminar programa	

Os dados podem ser lidos "manualmente" e importados para um dicionário, ou pode utilizar a classe `DictReader`. Não utilize `pandas` ou outra solução similar.

O ficheiro com os dados climatéricos encontra-se em formato CSV. Os dados neste ficheiro seguem um de dois sistemas de medidas:

1. Standard International/NOAA[1]: dados numéricos utilizam sistema métrico
2. USCS[2]: dados numéricos baseados no sistema imperial de medidas, tal como adaptado pelos Estados Unidos

Este programa aceita ambos os sistemas de medidas, até porque a obtenção das estatísticas pedidas não depende do sistema de medida utilizado. Porém, caso o utilizador invoque o programa com a opção `--uscs-to-metric` (ou `'-u'`), o programa deve interpretar os dados como estando no sistema USCS de unidades e aplicar as conversões devidas de acordo com a seguinte tabela:

CAMPO	DESIGNAÇÃO	UNIDADE/FORM. STD.NOAA/IMT.	UNIDADE/FORM. USCS	CONVERSÃO USCS → NOAA (default anthimio)
date	Date	ISO	ISO	n/a
tmin	Minimum Temperature	°C (X.Y)	°F (X.Y)	round((tmin-32) * 5/9, 1)
tmax	Maximum Temperature	°C (X.Y)	°F (X.Y)	round((tmax-32) * 5/9, 1)
tavg	Average Daily Temperature	°C (X.Y)	°F (X.Y)	round((tavg-32) * 5/9, 1)
prcp	Precipitation	mm (X.Y)	pol (X.YZ)	round(prcp * 25.4, 1)
snow	Snowfall	mm (X.Y)	pol (X.Y)	round(snow * 25.4, 1)
snwd	Snow depth	mm	pol (X.Y)	round(snow * 25.4, 0)
awnd	Average Daily Wind Speed	m/s (X.YZ)	mph (X.Y)	round(awnd * 0.44704, 2)

A opção `--metric-to-uscs (-m)` efectua as conversões inversas (não indicadas).

Por vezes, os parâmetros `prcp`, `snow`, `snwd` poderão ter o valor 'T' (sem *plicas*). Este valor vem da palavra TRACE e indica valores residuais abaixo do mínimo mensurável. Ou seja, 'T' indica uma quantidade minúscula que não foi possível medir com exactidão. Não significa zero, nem que o valor está em falta. Para efeitos de processamento numérico, o procedimento padrão consiste em normalizar o valor 'T' para 0.1 mm, no caso de `prcp`, para 1.3 mm, em se tratando de `snow`, e para 13 mm, se o campo for `snwd`. Opcionalmente, o programa deve aceitar os parâmetros `--snow-t`, `--snwd-t`, `--prcp-t`, onde o utilizador pode indicar o valor para o qual deverá ser normalizado o valor 'T' para cada um destes parâmetros. Se o programa for invocado com a opção `--zero-t`, o 'T' é normalizado para zero em todos estes parâmetros. Os parâmetros `--...-t` tomam precedência face ao parâmetro `--zero-t`.

De seguida listam-se algumas possíveis invocações do programa:

```
# erro: falta caminho para ficheiro CSV
$ anthimio.py
.
# lê dados em weather.csv, assume sistema NOAA, aplica conversão standard de valores T
$ anthimio.py weather.csv
.
# assume que dados em weather.csv estão no sistemas USCS, converte-os p/ sistema métrico
$ anthimio.py --uscs-to-metric weather.csv
.
# igual ao anterior
$ anthimio.py -u weather.csv
.
# converte valores T em snow para 0.05mm, aplica conversão standard aos restantes
$ anthimio.py --snow-t=0.05 weather.csv #
.
# converte valores T em snow para 0.05mm, converte para 0.0 restantes
$ anthimio.py --zero-t --snow-t=0.05 weather.csv
```

## AVALIAÇÃO

No contexto deste projecto, um elemento "extra" é um componente ou funcionalidade cuja cotação (ver tabela em baixo) é substancialmente inferior à de outros componentes ou funcionalidades de dificuldade semelhante. É possível ter uma boa classificação neste projecto não realizando nenhum dos elementos extras.

Elemento	Cotação Máxima (0..20)
Efeitos	15
Programa Parte II	4
Realização de outro programa Parte II	1

Como incentivo, a classificação das funcionalidades extra pode contribuir até 2 valor(es) para a nota do teste escrito e até 1 valor(es) para a nota do próximo projecto. O projecto deve ser resolvido em grupos de dois formandos. Excepcionalmente, poderá ser realizado por grupos com outras dimensões.

Deve também elaborar um relatório (ver secção **Prazo e Entregas**) que valerá no máximo 20% da cotação do projecto.

## PRAZOS E ENTREGAS

O trabalho final deve ser entregue até às 23h59m do dia DD/MM/2025. Um atraso de  $N$  dias na entrega levará a uma penalização dada pela fórmula  $0,5 \times 2^{(N-1)}$  ( $N > 0$ ).

Caso o formador não indique outro procedimento, na data final deverá entregar o seguinte:

1. Um **ZIP** contendo o projecto do VS Code. O ZIP deve possuir os nomes dos membros do grupo sem acentos e separados por \_ (underscore). Supondo que o projecto foi realizado pela Ana Alves e pelo Jose Sá, o ZIP deve ser designado de AnaAlves\_JoseSa.zip.

Este ZIP deve conter uma única directoria:

Projecto1 ou ProjectoEfeitos : directoria com o Workspace do VS Code.

- 1.1 Por seu turno a directoria Projecto1 deve conter as seguintes sub-directorias:

.vscode: directoria com as definições do VS Code

src: directoria com o código fonte

docs: documentação do projecto mais o relatório (ver em baixo)

1.2 O conteúdo da directoria /Projecto1/src deve ser o seguinte:

*Efeitos especiais*: efeitos.py

*Cifra de Vigenère*: vcypther.py

*Treep*: treep.py

*L2ADDR*: l2addr.py

*anthimio*: anthimio.py

2. Relatório em **PDF**, cuja estrutura e formatação deverão ser a do modelo fornecido em anexo.

Em termos de formatação adapte apenas a designação da acção e o nome dos módulos. Siga as recomendações relacionadas com a elaboração de um relatório dadas pelo formador Fernando Ruela. O relatório deve incluir uma capa simples com o símbolo do IEFP, referência ao Centro de Formação de Alcântara, data, indicação do curso e da acção (eg, Técnico de Informática - Sistemas 07) e dos elementos que elaboraram o trabalho.

Em termos de conteúdo o seu relatório deve possuir as seguintes secções e anexos:

2.1.1 **Introdução e Objectivos** (não plagiar a deste enunciado!).

2.1.2 **Desenho e Estrutura** que deverá conter:

- Para os três últimos efeitos, um fluxograma com o algoritmo geral desse efeito especial. Assuma que todo o texto a exibir é uma variável de entrada do fluxograma. Aqui pode ignorar a questão da concatenação. Assuma também que esse texto é uma sequência de caracteres, tal como uma string em Python.

Um dos três fluxogramas seguintes:

- **vcypther**: Um fluxograma a ilustrar o algoritmo geral de encriptação do vcypther.
- **treep**: Um fluxograma a ilustrar o algoritmo para gerar a árvore de directorias e ficheiros do treep.
- **l2addr**: Um fluxograma contendo o algoritmo geral do L2ADDR, desde a leitura dos argumentos da linha de comandos até à análise do texto do comando arp.
- **anthimio**: Um fluxograma indicando o algoritmo de cálculo de uma das seguintes estatísticas: temperatura média anual, mediana anual das temperaturas, dois anos mais quentes/frios.

2.1.3 **Implementação**. A secção Implementação deve indicar aspectos de implementação que

considere relevantes. É aqui que deve indicar a "docstring" utilizada pelo docopt (caso tenha utilizado esta biblioteca). Deve indicar para que plataforma desenvolveu e testou o seu programa, a versão do compilador e das bibliotecas utilizadas.

É também aqui que deve dar instruções sobre como "compilar" o seu programa e produzir um executável (num programa maior poderia ser pertinente criar uma secção Compilação e Instalação).

É também nesta secção que deve indicar que elementos opcionais foram implementados e como.

**2.1.4 Conclusão.** Além de seguir as recomendações sobre a elaboração de uma conclusão de relatório, deve também listar o que foi implementado e o que ficou por implementar.