Linguagens de Programação
Série de Problemas

José Jasnau Caeiro

25 de janeiro de 2021

Conteúdo

Certifique-se que existe a sub-pasta com a designação `lp2020` na pasta correspondente ao seu repositório de controlo de versões. Submeta tudo o que realiza no sistema de controlo de versões, incluindo versões que sejam alteradas. O tempo de estimado de resolução de cada problema é de 30 minutos.

1 Programação Imperativa

Esta série de problemas é relativa à aula teórica número 2.

1. Crie uma sub-pasta designada por `PA-1`. Edite nesta pasta um ficheiro designado por `pa1.py`.
   (a) Crie uma função que calcule os números de *Fibonacci*, https://en.wikipedia.org/wiki/Fibonacci_number.
   (b) Escreva uma função que calcule a *constante recíproca de Fibonacci* através duma série numérica limitada a n termos. https://en.wikipedia.org/wiki/Reciprocal_Fibonacci_constant.
   (c) Reescreva a função anterior de modo a que o resultado não ultrapasse um erro relativo estimado pré-determinado por um parâmetro passado à função, https://en.wikipedia.org/wiki/Approximation_error.

2. Crie uma sub-pasta designada por `PA-2`. Edite nesta pasta um ficheiro designado por `pa2.py`.
   (a) Escreva uma função que conte o número de vogais numa cadeia de caracteres.
   (b) Escreva uma função que tenha como argumento uma *string* em que apenas se encontram dígitos. Substitua nesta *string* todos os dígitos pares pela letra `A`.
   (c) Crie uma função que tenha como argumento uma *string* em que estejam dígitos e letras. Esta função deve devolver a soma dos dígitos.
   (d) Crie uma função que tenha como argumento uma *string* e um número inteiro positivo n. Esta função deve devolver uma *string* em que se repete n vezes a *string* inicial.

3. Crie uma sub-pasta designada por `PA-3`. Edite nesta pasta um ficheiro designado por `pa3.py`.
   (a) Represente os vetores $\vec{r_a} = \pi\hat{u}_x + \frac{\pi}{2}\hat{u}_y + 2\pi\,\hat{u}_z$ e $\vec{r_b} = \pi^2\hat{u}_x + \pi\hat{u}_y - \pi\hat{u}_z$, com túplos.
   (b) Escreva uma função que calcule o módulo dum vetor e aplique aos vetores anteriormente definidos.
   (c) Escreva uma função que calcule o produto interno de 2 vetores e use esta função para calcular este valor a partir dos vetores anteriormente definidos.
   (d) Escreva uma função que calcule o produto externo de 2 vetores e determine o seu resultado para os vetores anteriormente definidos.

4. Crie uma sub-pasta designada por `PA-4`. Edite nesta pasta um ficheiro designado por `pa4.py`.
   (a) Represente um polinómio de 3.ª ordem por uma função que tem como argumento uma lista dos coeficientes do polinómio.
   (b) Programe uma função que tendo como argumentos o polinómio $P(x)$ anteriormente definido, um intervalo de pesquisa $[a, b]$ e um erro de aproximação, calcule pelo método de *Newton* uma das raízes de P se esta existir. Se não existir deve devolver `None`. Vidé https://en.wikipedia.org/wiki/Newton%27s_method

5. Crie uma sub-pasta designada por `PA-5`. Edite nesta pasta um ficheiro designado por `pa5.py`.
   (a) Represente um sistema de equações lineares a 3 incógnitas através de listas.
   (b) Programe uma função que calcule o determinante duma matriz quadrada de dimensão 3×3 representada sob a forma de listas.
   (c) Use a regra de *Cramer* para calcular a solução do sistema de equações.

6. Crie uma sub-pasta designada por `PA-6`. Edite nesta pasta um ficheiro designado por `pa6.py`.
   (a) Represente a função matemática $f(x) = ax + sin(b\omega x + \theta) + cos(c\omega x + \theta)$ por uma função em `Python` com argumentos x, o túplo (a, b, c), e ω, θ. A função deve ser programada de forma a que se fôr chamada sem os argumentos ω e θ estes sejam por defeito $\omega = 10^3$ e $\theta = \pi$.

(b) Calcule

$$\int_0^{\pi} f(x)dx$$

pela regra dos trapézios https://en.wikipedia.org/wiki/Trapezoidal_rule. Escolha os valores das constantes.

7. Crie uma sub-pasta designada por `PA-7`. Edite nesta pasta um ficheiro designado por `pa7.py`.
   (a) Considere que pretende realizar um empréstimo a prestações constantes para adquirir uma casa. Represente as variáveis necessárias para a realização do empréstimo.
   (b) Simule as suas prestações para uma determinada taxa de *spread* e um determinado número de anos.

2 Programação Funcional

Esta série de problemas é relativa à aula teórica número 3.

1. Crie uma sub-pasta designada por `PB-1`. Edite nesta pasta um ficheiro designado por `pb1.py`.
   (a) Escreva uma função recursiva que crie uma lista de números aleatórios inteiros, com ditribuição uniforme entre 0 e 100 (inclusivé), com `N` elementos e designada por `l1`.
   (b) Escreva uma função recursiva que tem como argumento a função `l1` e produz a soma dos seus números ímpares.
   (c) Escreva uma função que subtrai o valor 1 ao seu argumento e designe esta função por `f1`.
   (d) Escreva uma função recursiva que tem como argumentos a função `f1` e a lista `l1`. Esta função deve devolver uma nova lista que resulta da aplicação de `f1` aos elementos ímpares de `l1`.

2. Crie uma sub-pasta designada por `PB-2`. Edite nesta pasta um ficheiro designado por `pb2.py`.
   (a) Escreva uma função recursiva que tem como argumento uma *string* e que devolve uma nova *string* sem vogais.
   (b) Escreva uma função recursiva que tendo como argumento uma *string* substitui todas as letras `'b'` pela letra `'a'`.
   (c) Escreva uma função recursiva que tendo como argumento uma *string* conta o número de vogais que se encontram na *string*.
   (d) Escreva uma função recursiva que gera a lista de números pares entre a e b, sendo estes valores passados em argumento.

3. Crie uma sub-pasta designada por `PB-3`. Edite nesta pasta um ficheiro designado por `pb3.py`.
   (a) Escreva uma função que cria uma lista de números aleatórios com distribuição normal de média nula e parâmetro $\sigma = 2.0$, com N elementos e designada por `l1`.
   (b) Escreva uma função que use iteradores para acesso aos elementos da lista. Esta função tem como argumento a lista `l1`. Deve criar uma nova lista `l22` e que elimina os valores negativos em `l1`.
   (c) Escreva uma função usando iteradores que calcule a soma dos valores positivos duma lista que seja passada em argumento.
   (d) Escreva uma função usando iteradores devolva a soma dos valores cujo primeiro dígito é par.

4. Crie uma sub-pasta designada por `PB-4`. Edite nesta pasta um ficheiro designado por `pb4.py`.
   (a) Escreva uma função que usando compreensões de listas devolva uma lista de números aleatórios inteiros pares, com distribuição uniforme entre 0 e 50, e de dimensão definida pelo argumento n.
   (b) Escreva uma função usando compreensões de listas que tendo como argumento uma lista de números inteiros, positivos ou negativos, gere uma nova lista de números do tipo `float` obtidos por aplicação da raiz quadrada. Sempre que não for possível determinar a raiz deve substituir por `0.0`.
   (c) Escreva uma função que tem como argumento uma lista de inteiros e que usando o operador `map()` devolva uma lista em que estes inteiros são multiplicados por 3.
   (d) Use o operador `filter()` para eliminar os números pares duma lista de inteiros.

5. Crie uma sub-pasta designada por `PB-5`. Edite nesta pasta um ficheiro designado por `pb5.py`.
   (a) Escreva uma função que usando geradores e compreensões de listas devolva uma lista de n elementos aleatórios com distribuição normal de média nula e desvio padrão $\sigma = 3.4$.

(b) Escreva uma função por meio de geradores e compreensões de listas que produza uma *string* de dimensão n com carateres minúsculos escolhidos aleatoriamente.

(c) Escreva uma função por meio de geradores e compreensões de listas que produza um túplo com n elementos inteiros aleatórios, com distribuição uniforme.

(d) Escreva um gerador de números aleatórios inteiros com distribuição uniforme sempre pares.

6. Crie uma sub-pasta designada por `PB-6`. Edite nesta pasta um ficheiro designado por `pb6.py`.

   (a) Escreva uma função que tem como argumento uma *string* e que devolve uma lista composta pelas palavras desta `string`. Designe esta lista por `l1`.

   (b) Escreva uma função que devolve uma lista de conjuntos. Cada conjunto é obtido a partir de cada item da lista `l1`.

   (c) Escreva uma função que tendo como argumento uma *string* calcula o número de vogais repetidas desta *string*. Use conjuntos.

   (d) Escreva uma função que tendo como argumento duas *string* determina o número de palavras comuns usando conjuntos.

7. Crie uma sub-pasta designada por `PB-7`. Edite nesta pasta um ficheiro designado por `pb7.py`.

   (a) Escreva uma função que tem como argumento duas *string* com o mesmo número de palavras e que que devolve uma estrutura de dados do tipo `dict` em que vai buscar à primeira `string` a chave e à segunda o valor, sucessivamente.

   (b) Escreva uma função que produz uma estrutura do tipo `dict` em que a chave é a soma dos ordinais dos carateres da duma *string* e o valor é a palavra. A *string* é composta por palavras separadas por espaços.

   (c) Escreva uma função que percorra todos os items dum dicionário com valores constituídos por *strings* e elimine as entradas com *strings* de comprimento par.

   (d) Escreva uma função que permita criar um dicionário cuja chave é um inteiro e cujo valor é uma lista de zeros da dimensão correspondente ao valor da chave.

3 Entrada e Saída de Dados e Exceções

Esta série de problemas é dedicada a:

- entrada e saída de dados
  - formatação de cadeias de carateres
  - ficheiros
  - serialização
  - ficheiros em formato CSV
  - ficheiros em formato Excel
  - bases de dados embutidas e o módulo `sqlite3`
- exceções
  - erros sintáticos
  - tratamento de exceções
  - levantamento de exceções
  - exceções definidas pelo utilizador

1. Crie uma sub-pasta designada por `PC-1`. Edite nesta pasta um ficheiro designado por `pc1.py`.
   (a) Escreva uma função que tem como argumentos uma *string* de tamanho nunca inferior a 20 carateres e um número inteiro positivo n. A função deve devolver uma *string* que é a repetição de n vezes da sub-*string* composta pelos carateres nas posições 1 a 4, e 8 a 13 da *string* passada em argumento.
   (b) Escreva uma função que recebe uma *string* e que devolve um dicionário. Este dicionário tem como chave um carater da *string* e como valor o número de vezes que este carater se encontra na *string*.
   (c) Escreva uma função que recebe como argumento uma lista de *string* e que devolve uma *string* resultante da concatenação de cada *string* presente na lista quando esta *string* tem comprimento par.
   (d) Escreva uma função que recebe um túplo composto por *strings*. A função deve devolver uma lista de túplos em que cada túplo deve ter 3 elementos associados a cada *string* do túplo de argumento da função: o comprimento ; o maior item e o menor item.

2. Crie uma sub-pasta designada por `PC-2`. Edite nesta pasta um ficheiro designado por `pc2.py`.
   (a) Escreva uma função que tem como argumentos uma *string* de tamanho nunca inferior a 20 carateres e um número inteiro positivo n. A função deve devolver uma *string* que é a concatenação da sub-*string* composta pelos carateres nas posições 1 a 4 com 8 a 13 da *string* passada em argumento. Deve adicionalmente substituir todas as vogais pelo caracter 'b' usando a função `replace()`.
   (b) Escreva uma função que recebe uma *string* e que devolve um túplo. Este túplo tem como primeiro elemento um carater da *string* e como segundo elemento o número de vezes que este carater se encontra na *string* se for uma vogal. Sendo uma consoante este valor deve ser 0.
   (c) Escreva uma função que recebe como argumento uma lista de *string* e que devolve uma *string* resultante da concatenação de cada *string* presente na lista quando esta *string* tem comprimento superior a 6 caracteres.
   (d) Escreva uma função que recebe um túplo composto por *strings*. A função deve devolver uma lista de túplos em que cada túplo deve ter 3 elementos associados a cada *string* do túplo de argumento da função: o número de vogais ; o maior item e o menor item.

3. Crie uma sub-pasta designada por `PC-3`. Edite nesta pasta um ficheiro designado por `pc3.py`.
   (a) Crie uma função que tem como argumento uma *string* contendo vírgulas, pontos finais e espaços. Esta função deve devolver uma lista contendo 3 listas. A primeira dessas listas deve resultar da separação do argumento pelas vírgulas, a segunda deve resultar da separação do argumento pelos pontos finais e finalmente a última deve resultar da separação do argumento pelos espaços.
   (b) Escreva uma função que tendo uma *string* como argumento. A *string* deve apenas ter palavras com carateres minúsculos e espaços. A função devolve uma lista com as palavras separadas por espaços. As palavras devem ser capitalizadas.
   (c) Escreva uma função que recebe uma lista de frases. A função deve devolver todas as palavras das diversas frases que comecem por vogais sob a forma duma única *string*. Esta *string* deve apresentar as palavras alternadamente em maiúsculas e minúsculas.
   (d) Crie uma função que recebe uma *string*. A função deve devolver outra *string* composta apenas das primeiras letras das palavras que compoem a *string* de entrada.
4. Crie uma sub-pasta designada por `PC-4`. Edite nesta pasta um ficheiro designado por `pc4.py`.
   (a) Escreva uma função que receba um valor do tipo `float`. A função deve devolver uma *string* gerada com a função `format()`. O número deve ser representado centrado com em 40 posições disponíveis. A precisão deve ser de 2 dígitos.
   (b) Escreva uma função que recebe dois números inteiros a e b. A função deve devolver uma *string* que representa b como hexadecimal seguido de a como octal.
   (c) Escreva uma função que recebe um número do tipo `float`. A função deve devolver uma *string* com 40 posições disponíveis e em que o número é alinhado dentro do espaço de representação disponível à direita. A precisão deve ser limitada a 2 dígitos e os espaços disponíveis devem ser preenchidos com o símbolo $. Use apenas a função `format()`.
   (d) Escreva uma função que recebe um número do tipo `float` e que devolve uma *string* com uma representação em percentagem usando a função `format()`.
5. Crie uma sub-pasta designada por `PC-5`. Edite nesta pasta um ficheiro designado por `pc5.py`.
   (a) Escreva uma função que recebe uma *string* com um nome dum ficheiro e que escreve neste 100 números aleatórios inteiros positivos, entre 0 e 100, distribuídos uniformente. A função deve separar cada número por um espaço.
   (b) Escreva uma função que recebe uma *string* com um nome dum ficheiro. Esta função deve escrever num ficheiro 100 números aleatórios do tipo *float* distribuídos uniformemente entre 0 e 1.0.
   (c) Escreva uma função que abre um ficheiro, contendo 10 linhas. Ema cada linha encontra-se uma `string` e um número inteiro positivo. O ficheiro deve ser lido com o operador `with` e deve ler linha a linha o conteudo. A função deve devolver uma lista de túplos em que o primeiro elemento corresponde à *string* e o segundo ao número inteiro.
   (d) Escreva uma função que tem como argumento uma *string* correspondente ao nome dum ficheiro. O ficheiro deve conter um texto com mais de 100 palavras. A função deve ter como argumento também um número inteiro positivo que indica onde no ficheiro se deve escrever a palavra "TESTE".
6. Crie uma sub-pasta designada por `PC-6`. Edite nesta pasta um ficheiro designado por `pc6.py`.
   (a) Escreva uma função que escreve na consola de erros e na consola de saída o que lê da consola de entrada. Deve terminar de escrever quando lê a palavra **exit**.
   (b) Escreva uma função que recebe com argumento uma *string* que corresponde ao nome dum ficheiro. A função deve ler o conteúdo do ficheiro (neste caso um texto com mais de 100 palavras) e sempre que aparece a sílaba **"te"** escreve na consola de erro a palavra **"OOPS"**.

(c) Escreva uma função que lê o terminal de entrada e que depois escreve em ficheiro apenas os dígitos.

(d) Escreva uma função que lê do terminal de entrada e depois envia para a saída em consola tudo o que se escreveu com a excepção das vogais.

7. Crie uma sub-pasta designada por `PC-7`. Edite nesta pasta um ficheiro designado por `pc7.py`.

   (a) Crie uma função que tem como argumento uma estrutura de dados do tipo `dicionário` e armazena num ficheiro. Use serialização.

   (b) Crie uma função que lê dum ficheiro de serialização uma lista de túplos de 3 elementos.

   (c) Crie uma função que lê os conteúdos dum ficheiro com 100 linhas de inteiros para uma lista e que depois armazena esta lista com serialização.

   (d) Crie uma função que armazene num ficheiro de serialização uma lista de 100 inteiros e um dicionário em que a chave é uma *string* e o valor um `float`. O dicionário tem 10 entradas.

8. Crie uma sub-pasta designada por `PC-8`. Edite nesta pasta um ficheiro designado por `pc8.py`.

   (a) Use uma folha de cálculo para criar um ficheiro com 2 colunas. Na primeira coluna deve colocar uma data e na segunda um número. Tenha 10 entradas exportado para um ficheiro do tipo CSV. Crie uma função que lê o conteúdo deste ficheiro e soma todos os números da segunda coluna.

   (b) Escreva uma função que gera 100 números aleatórios com distribuição uniforme e os guarda num ficheiro CSV.

   (c) Escreva uma função que escreve num ficheiro CSV o conteúdo duma lista de túplos. O primeiro elemento do túplo é uma *string* e o segundo elemento um inteiro.

   (d) Escreva uma função que lê o conteúdo dum ficheiro CSV e que depois cria uma lista correspondente que armazena noutro ficheiro com recurso a serialização.

9. Crie uma sub-pasta designada por `PC-9`. Edite nesta pasta um ficheiro designado por `pc9.py`.

   (a) Siga a documentação sobre o módulo `sqlite3` e crie uma base de dados com o nome `teste.db`. Faça uma conexão e crie uma tabela designada por **familia** com as seguintes entradas:

   - nome próprio
   - nome de família
   - idade

   (b) Escreva uma função que lê dum ficheiro CSV as entradas criadas na alínea anterior e as guarda na base de dados.

   (c) Escreva uma função que procura na tabela **familia** apenas as entradas de pessoas com mais de 19 anos de idade.

   (d) Escreva uma função que lê da tabela apenas as entradas das pessoas com mais de 19 anos e com um determinado nome de família.

10. Crie uma sub-pasta designada por `PC-10`. Edite nesta pasta um ficheiro designado por `pc10.py`.

    (a) Escreva uma função que produza a tabela da divisão de números inteiros entre 0 e 99 pelos números inteiros entre -5 a 5. Faça com esta função seja robusta usando excepções.

    (b) Escreva uma função que vá ler dum determinado ficheiro cujo nome é passado em argumento. Torne a função robusta à inexistência do ficheiro.

    (c) Crie uma função que usando o módulo `sqlite3` tente ler um ficheiro inexistente e que seja robustecida com exceções.

    (d) Crie uma função que produz a tabela de tangentes $\tan(x) = \sin(x)/\cos(x)$ para ângulos entre -π a π com resolução de 0.0001 rad . Torne a função à prova de erros com exceções.

4 Programação Orientada por Objetos

Esta série de problemas é dedicada ao tema da **programação orientada por objetos** usando como ilustração a linguagem de programação `Python`.

A lista de tópicos cobre:

- os mecanismos de suporte para a programação orientada por objetos
  - classes
  - instâncias de classes
  - métodos
  - construtores
  - mecanismos de herança
  - métodos especiais
  - variáveis e métodos privados
- o uso de algumas classes especiais
  - iteradores
  - geradores
  - exceções

Série de Problemas

1. Crie uma sub-pasta designada por `PD-1`. Edite nesta pasta um ficheiro designado por `pd1.py`.
   (a) Crie uma classe que representa entradas bibliográficas e designada por `Bibliografia`. São exemplos de entradas bibliográficas:
   - livros
   - artigos científicos
   - *sites*
   - *etc.*

   Uma lista bastante completa de tipos de entradas, relacionada com o sistema `BiBTeX`, encontra-se em (Paperpile, 2020). A classe deve ter um construtor que permita construir uma entrada bibliográfica geral tendo apenas os atributos comuns dos tipos referenciados no *site* acima referenciado.
   (b) Crie uma classe derivada da classe `Bibliografia` e designada por `Livro`. Crie outra classe designada por `Artigo`. Adicione os atributos correspondentes e os construtores adequados que deverão aproveitar o construtor da classe `Bibliografia`.
   (c) Cria uma função que lê referências de livros a partir dum ficheiro e que instancia diversos objetos da classe `Livro`.
   (d) Escreva um método que permite alterar todos os atributos duma instância dum livro.
2. Crie uma sub-pasta designada por `PD-2`. Edite nesta pasta um ficheiro designado por `pd2.py`.
   (a) Crie uma classe que representa instrumentos musicais, designada por `InstrumentoMusical` e com todos os atributos que considera que um instrumento musical por ter. Escreva o construtor que inicializa estes atributos.
   (b) Escreva a função `__str__()` que considera melhor apresentar a informação sobre uma instância de `InstrumentoMusical`.
   (c) Adicione uma variável de instância designada por `nota` à classe `InstrumentoMusical`. Torne esta variável privada e escreva os métodos que permitem ler e escrever valores nesta variável.

(d) Escreva um construtor que cria uma nova instância da classe `InstrumentoMusical` a partir duma instância desta mesma classe.

3. Crie uma sub-pasta designada por `PD-3`. Edite nesta pasta um ficheiro designado por `pd3.py`.

   (a) Crie uma classe que representa **obras de arte**. Dê-lhe o nome `ObraArte`. Os atributos dos objetos são:

   - o nome da obra de arte
   - a data de criação da obra de arte
   - o seu preço estimado

   Escreva o construtor adequado para que quando o objeto for instanciado possa ter os atributos correspondentes preenchidos.

   (b) Crie uma classe que representa **museus**. Dê-lhe o nome `Museu`. Os atributos dos objetos são:

   - o nome do museu
   - a morada do museu
   - a data em que foi criado

   Escreva o construtor adequado para que quando o objeto for instanciado possa ter os atributos correspondentes preenchidos.

   (c) Crie uma classe que representa os autores de obras de arte. Dê-lhe o nome `AutorObra`. Os atributos dos objetos são:

   - o nome do autor
   - a sua data de nascimento
   - a sua nacionalidade

   Escreva o construtor adequado para que quando o objeto for instanciado possa ter os atributos correspondentes preenchidos.

   (d) Crie uma classe designada por `ReferenciaObraCompleta` que é derivada das 3 classes anteriormente definidas. Escreva o construtor adequado para a criação dum objeto com todos estes atributos. Escreva a função `__str__()` que permite o seguinte trecho de código:

```
s_obj = str( obj )
s = ".. e aqui estão os atributos do objeto\n" + s_obj
print( s )
```

4. Crie uma sub-pasta designada por `PD-4`. Edite nesta pasta um ficheiro designado por `pd4.py`.

   (a) Crie uma classe designada por `Browser` e que representa *browsers* de `Internet`. Esta classe deve ter como atributos:

   - nome do *browser*
   - número de versão

   (b) Documente adequadamente a classe. Imprima a documentação da classe.

   (c) Crie uma classe derivada de `list` designada por `ListaBrowser` que representa uma lista de *browsers*.

   (d) Escrevas as funções `__str()__` das classes que criou de modo a que possa obter ao fazer `print(obj)`, em que `obj` pertence à classe `ListaBrowser` a informação sobre todos os atributos dos objetos do tipo `Browser` presentes na lista.

5. Crie uma sub-pasta designada por `PD-5`. Edite nesta pasta um ficheiro designado por `pd5.py`.

   (a) Crie uma classe que representa plantas e designada por `Planta`. Esta classe deve derivar de `object` e tem como atributos:

- nome científico da planta
- nome comum da planta

Escreva o construtor adequado para a criação adequada dum objeto deste tipo.

(b) Escreva uma classe derivada designada por `Arvore` e que representa plantas do tipo árvore. Os atributos serão os mesmos da classe `Planta`. Escreva o construtor desta classe usando o método `super()`.

(c) Escreva uma função que recebe como argumento objetos e que sempre que sempre que recebe um objeto da classe `Planta` devolve um objeto da classe `Arvore` e que caso contrário devolve o objeto que recebeu em argumento.

(d) Crie uma classe derivada de `Arvore` de `list` designada por `ArvoreEstendida` e que permite adicionar uma lista de informação avulsa a cada objeto de `ArvoreEstendida`.

6. Crie uma sub-pasta designada por `PD-6`. Edite nesta pasta um ficheiro designado por `pd6.py`.

(a) Escreva uma classe que deriva de `BaseException` e que representa o `Stock` de ferramentas duma empresa. Deve ter como atributos:

- o nome da ferramenta
- a quantidade em *stock* da ferramenta

Crie os construtores adequados.

(b) Crie um método em que sempre que tentar levantar uma ferramenta que não exista em *stock* levanta uma excepção. Demonstre o seu funcionamento.

(c) Crie uma classe derivada de `set` que representa um conjunto de objetos da classe `Stock`. Designe esta classe por `ConjuntoStock`. Crie vários objetos que representam ferramentas. Esta classe deve também derivar de `BaseException`. Programe tudo de forma a que se a soma total de ferramentas diversas for inferior a 100 se levanta uma excepção.

(d) Escreva um método da classe `ConjuntoStock` que quando se levanta esta exceção `ConjuntoStock` acrescenta de 100 ao *stock* da ferramenta com menor *stock*.

7. Crie uma sub-pasta designada por `PD-7`. Edite nesta pasta um ficheiro designado por `pd7.py`.

(a) Crie uma classe derivada de `object` designada por `IteradorRuas` que constrói um **iterador** para uma classe que representa uma coleção de ruas e designada por `Cidade`. Esta classe também deriva de `object`. Deve existir representação para as ruas em termos do seu nome, comprimento da rua e largura máxima.

(b) escreva e demonstre a utilização do operador `next` resultado do método privado `__next()__`.

(c) Crie uma classe derivada designada por `AvenidasRuas` em que também se representam séries televisivas com a indicação do número de episódios. Adeque os vários métodos de forma a iterar convenientemente.

(d) Escreva as funções `__str()__` que permitam imprimir todos os atributos das ruas e avenidas representadas.

8. Crie uma sub-pasta designada por `PD-8`. Edite nesta pasta um ficheiro designado por `pd8.py`.

(a) Crie uma classe derivada de `object` designada por `IteradorFilmes` que constrói um **iterador** para uma classe que representa uma coleção de filmes e designada por `ColecaoFilmes`. Esta classe também deriva de `object`. Deve existir representação para os filmes em termos do seu nome, realizador e data de realização.

(b) escreva e demonstre a utilização do operador `next` resultado do método privado `__next()__`.

(c) Crie uma classe derivada designada por `FilmesSeries` em que também se representam séries televisivas com a indicação do número de episódios. Adeque os vários métodos de forma a iterar convenientemente.

(d) Escreva as funções `__str()__` que permitam imprimir todos os atributos dos filmes e séries representados.

9. Crie uma sub-pasta designada por `PD-9`. Edite nesta pasta um ficheiro designado por `pd9.py`.
   (a) Crie um gerador **inesgotável** da sequência de números ímpares. Utilize as potencialidades de criação de geradores da linguagem `Python`.
   (b) Utilize geradores para utilizar no preenchimento duma lista composta por pares de valores, em que o primeiro valor é um inteiro positivo n e o segundo valor é o $y = n \times sqrt(n)$.
   (c) Crie um gerador que permita calcular as aproximações sucessivas ao número π.
   (d) Escreva um gerador que permita que de cada vez que aplique o operador `next()` haja uma *string* resultado da concatenação sucessiva duma `string` passada em argumento do gerador.

10. Crie uma sub-pasta designada por `PD-10`. Edite nesta pasta um ficheiro designado por `pd10.py`.
    (a) Crie um gerador para calcular a sequência de *Fibonacci*.
    (b) Crie agora outro gerador que aproveita o gerador que criou na alínea anterior. Este novo gerador devolve a sequência das raízes quadradas da sequência de *Fibonacci*.
    (c) Crie um gerador permita produzir a sequência de letras do alfabeto.
    (d) Crie um gerador que aproveita o gerador anterior para produzir alternadamente a sequência de letras do alfabeto mas substituindo as vogais pela letra **V**.

5 A Biblioteca Padrão de `Python`

A linguagem de programação `Python` possui uma biblioteca padrão extensa. Alguns destes problemas dão exemplo do seu uso.

A lista de tópicos cobre:

- A biblioteca padrão;
- Exemplos de módulos da biblioteca padrão;

1. Crie uma sub-pasta designada por `PE-1`. Edite nesta pasta um ficheiro designado por `pe1.py`.
   (a) Recorra à documentação sobre a biblioteca padrão sobre o módulo os. Crie uma função que coloca num ficheiro todas as variáveis do ambiente do utilizador. Dê ao ficheiro o nome `home.txt`.
   (b) Crie uma função que coloca num ficheiro com o nome `sistema.txt` todas as informações sobre o sistema operativo disponíveis através das bibliotecas os, da *string* sys.platform e do módulo platform.
   (c) Escreva uma função que processa a excepção `OSError`, a ocorrer quando tenta abrir um ficheiro inexistente para leitura. A função deve determinar qual o código numérico do erro ocorrido.
   (d) Escreva uma função que coloca em écrã a pasta de execução atual, o número de processo associado ao *script* de `Python` e o *user id*.

2. Crie uma sub-pasta designada por `PE-2`. Edite nesta pasta um ficheiro designado por `pe2.py`.
   (a) Crie uma função que cria uma pasta cujo nome é passado em argumento.
   (b) Crie uma função que cria um ficheiro cujo nome é passado em argumento usando o módulo `os`. Escreva 1000 vezes a letra **a** neste ficheiro.
   (c) Crie uma função que lê o ficheiro que criou na alínea anterior e que usando a função `lseek()` substitua a letra **a** pela letra **b** sempre que o índice posicional no ficheiro fôr múltiplo de 10.
   (d) Crie uma função que elimina todos os ficheiros duma pasta cujo conteúdo inicie pela letra **a**.

3. Crie uma sub-pasta designada por `PE-3`. Edite nesta pasta um ficheiro designado por `pe3.py`.
   (a) Crie uma função que recorre ao módulo glob. Tem como argumento o nome duma pasta e dum ficheiro. Guarda no ficheiro todos os conteúdos da pasta e das sub-pastas.
   (b) Crie uma função que usando o módulo glob coloque numa lista apenas os nomes de ficheiros contendo no seu nome algarismos. Use uma expressão regular.
   (c) Crie uma função que tem como argumento uma expressão regular e que usando o módulo glob devolve a contagem de ficheiros que obdecem à expressão regular.
   (d) Escreva uma função que recebe como argumento o nome dum ficheiro e duma pasta. Este ficheiro deverá conter 1000 caracteres **a**. Apenas deve ser criado se não se encontrar na lista de ficheiros devolvidos pela função `glob()`.

4. Crie uma sub-pasta designada por `PE-4`. Edite nesta pasta um ficheiro designado por `pe4.py`.
   (a) Use o módulo argparse para criar uma aplicação que tem argumentos opcionais na linha de comandos:
      - `--fact` para que a aplicação calcule o fatorial de `n` (o argumento passado para esta opção).
      - `--pow` para que a aplicação calcule a potência de 10 de `m` (o argumento passado para esta opção).

(b) Use o módulo argparse para criar uma aplicação que tem argumentos opcionais na linha de comandos. Crie uma opção designada por `--pasta` que tem como argumento o nome duma pasta e que devolve com o recurso à função `glob` a lista de ficheiros dessa pasta.

(c) crie uma aplicação com uma opção para devolver o produto sucessivo de vários números inteiros passados em argumento.

(d) Crie uma aplicação que tem uma opção que tem como argumento o nome dum ficheiro. Esta opção deve permitir que o identificador do processo desta aplicação seja escrito no ficheiro.

5. Crie uma sub-pasta designada por `PE-5`. Edite nesta pasta um ficheiro designado por `pe5.py`.

   (a) Use o módulo re, destinado a processar expressões regulares. Crie uma função que devolve a lista de sílabas começadas por `s` numa *string* passada em argumento.

   (b) Use expressões regulares para, nomeadamente a função `sub()` para eliminar a palavra `'não'` duma *string*. Exemplifique.

   (c) Use expressões regulares para contar num ficheiro o número de vezes que aparece a palavra `'sim'`. Exemplifique.

   (d) Use a função `split` para dividir uma *string* usando como separador uma expressão regular única para termos que includem cedilhas e vogais acentuadas.

6. Crie uma sub-pasta designada por `PE-6`. Edite nesta pasta um ficheiro designado por `pe6.py`.

   (a) Use o módulo `random` para criar duas listas. A primeira lista deve ter 1000 números aleatórios com distribuição uniforme entre -10 e 10. A segunda lista deve ter 1000 números aleatórios com distribuição gaussiana de média nula e desvio padrão $\sigma = 2.0$. Use a biblioteca `matplotlib` para comparar os gráficos das 2 distribuições.

   (b) Produza, com o recurso à biblioteca `math` a tabela dos senos, cossenos e tangentes dos números entre $-\pi$ e π com intervalo de 0.001.

   (c) Produza uma lista de 100 números aleatórios uniformente distribuídos entre -10 e 10. Crie uma função que devolva uma lista contendo o resto da divisão por π de cada um destes números da primeira lista.

   (d) Usando a biblioteca com funções matemáticas para números complexos cmath crie uma função que produz a tabela dos senos e cossenos entre $0.0 + 0.0\,j$ e $3.4 + 2.5\,j$ com intervalos de $0.1 + 0.1$.

7. Crie uma sub-pasta designada por `PE-7`. Edite nesta pasta um ficheiro designado por `pe7.py`.

   (a) Use os módulos disponíveis na biblioteca padrão para criar uma aplicação que lança um servidor `HTTP` no porto `18080`. Escreva um ficheiro `index.html` com informação a que se possa aceder. Teste com um *browser* ou com a aplicação `wget`.

   (b) Use o módulo `urllib` para aceder ao conteúdo do ficheiro `index.html`. Escreva uma função que conta o número de vogais neste ficheiro.

   (c) Abra a página http://sapo.pt com a sua aplicação e devolva quantas vezes aparece mencionado o nome de família do primeiro ministro de Portugal.

   (d) Crie uma aplicação que lança um servidor `HTTP` permita navegar num conjunto de `links` que escreveu no ficheiro `index.html`.

8. Crie uma sub-pasta designada por `PE-8`. Edite nesta pasta um ficheiro designado por `pe8.py`.

   (a) Crie uma aplicação que use o módulo `time` para medir tão precisamente quanto possível o tempo de execução duma versão recursiva do fatorial criada por si. Realize uma estatística do tempo médio de execução com uma amostra com 100 elementos. Determine o erro absoluto e o erro relativo na medida do tempo de execução para valores de n entre 20 e 30.

(b) Crie uma aplicação que lê dum ficheiro a lista de aniversários da sua família e que cria em ficheiro um calendário anual onde se encontram marcadas estas datas (use o módulo `calendar`.

(c) Crie uma aplicação que lança um servidor `HTTP` que serve os calendários dos próximos 5 anos.

(d) Escreva uma aplicação que calcula o número de dias que faltam para cada avaliação do semestre. As datas devem ser lidas dum ficheiro.

9. Crie uma sub-pasta designada por `PE-9`. Edite nesta pasta um ficheiro designado por `pe9.py`.

(a) Utilize o módulo cProfile. Escreva duas funções recursivas: a primeira realiza o fatorial e a segunda a série de *Fibonacci*. Teste as funções para valores de n que levem a que o tempo de execução quando chamadas 50 vezes, faça a sua aplicação demorar cerca de 2 minutos a chegar ao fim. Use o módulo `cProfile` para determinar onde se está a perder mais tempo.

(b) Utilize o sistema de *profiling* para determinar o perfil de execução duma aplicação. Esta aplicação deve gerar uma lista de 10000 números aleatórios do tipo `float` distribuídos uniformente entre $-\pi$ e π. A aplicação deve ter uma função que calcula a soma dos senos destes números. A aplicação deve ter uma função que calcula a soma das exponenciais destes números. A aplicação deve ter uma função que calcula a soma das tangentes destes números. Trate apropriadamente as exceções.

(c) Use o sistema de *profiling* para determinar o que é mais rápido. Uma função que perante uma lista de números inteiros de 0 a 100000, devolve a lista de números pares, usando ciclos `while` e seleções `if` ou uma função que trata cada número ímpar como uma exceção.

(d) Use o sistema de *profiling* para escolher entre uma implementação recursiva ou iterativa da função de Fibonacci. Justifique as suas opções.

6 Realização de Gráficos e Pacotes Matemáticos

Esta série de problemas é dedicada à:

- criação de conteúdos em formato PDF com o *package* ReportLab
- criação de gráficos com o *package* Matplotlib
- utilização dos pacotes matemáticos Numpy e SciPy

1. Crie uma sub-pasta designada por `PF-1`. Edite nesta pasta um ficheiro designado por `pf1.py`.
   (a) Descarregue a documentação *open source* do pacote ReportLab. Utilize a biblioteca `reportlab.pdfgen` para criar um ficheiro `PDF` com o nome `testep1.pdf`. Use um objeto do tipo `Canvas`[1] para escrever o seu nome nesse ficheiro. Escreva o seu nome na posição 50×100.
   (b) Crie um `Canvas` com a dimensão correspondente ao formato `A4` e escreva o seu nome completo, na posição $8cm \times 5cm$, num ficheiro designado por `testep1b.pdf`
   (c) Adicione ao ficheiro anterior a frase *The quick brown fox jumps over the lazy dog* escrita a vermelho na posição $5cm \times 8cm$ e ser escrita com a fonte `Helvetica, 12pt`.
   (d) Adicione ao ficheiro anterior um rectângulo preenchido a verde. Este retângulo deve ficar colocado em $12cm \times 2cm$ e com dimensões $4cm \times 3cm$.

2. Crie uma sub-pasta designada por `PF-2`. Edite nesta pasta um ficheiro designado por `pf2.py`.
   (a) Use o pacote `ReportLab` e crie um `Canvas` de dimensão `A4`. Escreva o resultado num ficheiro designado por `teste2.pdf`.
   (b) Escreva no ficheiro um círculo vermelho com o diâmetro de 5cm a meio da página.
   (c) Escreva a meio da página o seu nome completo a azul, com uma fonte `Helvetica, 16pt`.
   (d) Escreva de novo a meio da página o seu nome completo a verde, com uma fonte `Helvetica, 20pt` mas rodada de 90°.

3. Crie uma sub-pasta designada por `PF-3`. Edite nesta pasta um ficheiro designado por `pf3.py`.
   (a) Importe o objeto `pyplot` do pacote Matplotlib. Crie uma lista designada por `y` com os seguintes valores:
   `y = [1, 2, 4, 8, 16, 32, 64, 128, 256]`
   e represente num gráfico com a função `plot`.
   (b) Escreva o resultado num ficheiro PDF com a designação `grafico3.pdf`. Use a função `savefig`.
   (c) Produza o gráfico com o título `potências de 2`. Acrescente um `label` ao eixo das ordenadas: 2^n. O `label` das abscissas deve ser `n`.
   (d) Produza o gráfico acrescentando a azul cada ponto representado por uma cruzinha.

4. Crie uma sub-pasta designada por `PF-4`. Edite nesta pasta um ficheiro designado por `pf4.py`.
   (a) Utilize o objeto `pyplot` do pacote `matplotlib` e produza um gráfico da função seno, entre 0 e $4 \times \pi$, com 100 pontos. Designe a lista dos valores de x por `lx` e dos valores de y por `ly`. Use a função `plot`. Armazene o resultado num ficheiro designado por `sen.pdf`.
   (b) Escreva o título e os `label` de cada eixo.
   (c) Sobreponha no gráfico a função cosseno, a vermelho.
   (d) Sobreponha no gráfico a função $\cos(x) \times \sin(x)$.

5. Crie uma sub-pasta designada por `PF-5`. Edite nesta pasta um ficheiro designado por `pf5.py`.

[1] Consulte a documentação e os exemplos.

(a) Utilize o objeto `pyplot` para produzir um gráfico da função $f(x) = x^2 - 2x - 2$ de modo a representar as passagens por zero da função. Guarde o ficheiro em `fcc.pdf`.

(b) Escreva o título e os `labels` dos eixos.

(c) Represente uma reta que representa uma constante. Escolha uma reta que intersecta a função $f(x)$. Use a cor vermelha.

(d) Acrescente uma legenda.

6. Crie uma sub-pasta designada por `PF-6`. Edite nesta pasta um ficheiro designado por `pf6.py`.

(a) Utilize a função `subplot` para produzir um gráfico que representa em duas partes a evolução da função seno e da função cosseno entre $-2 \times \pi$ e $2 \times pi$.

(b) Armazene o gráfico em `subaba.pdf`.

(c) Dê títulos aos gráficos e escreva os `label` dos eixos das ordenadas e das abscissas.

(d) Acrescente mais uma parte gráfica que representa a função $\cos(x) \times \sin(x)$.

7. Crie uma sub-pasta designada por `PF-7`. Edite nesta pasta um ficheiro designado por `pf7.py`.

(a) Utilize o pacote `numpy` para produzir um vetor de dimensão 10 $times1$ preenchido com zeros.

(b) Utilize o mesmo pacote para produzir um vetor da mesma dimensão preenchido com o número π.

(c) Use o pacote `numpy` para somar os dois vetores.

(d) Use o pacote `numpy` para multiplicar os dois vetores. Experimente a multiplicação ponto a ponto e depois transforme o segundo vetor para dimensão 1×10 e experimente multiplicar de novo.

8. Crie uma sub-pasta designada por `PF-8`. Edite nesta pasta um ficheiro designado por `pf8.py`.

(a) Use o pacote `numpy` para criar um vetor de dimensão 10×1 preenchido com valores que vão de 0 a 9.

(b) Crie uma matriz de dimensão 10 $times10$ preenchida com valores aleatórios distribuídos uniformemente entre 0 e 10.0. Utilize a função `numpy.random.Generator.random`.

(c) Faça multiplicação do vetor pela matriz.

(d) Calcule a inversa e transposta da matriz usando o pacote `numpy`.

9. Crie uma sub-pasta designada por `PF-9`. Edite nesta pasta um ficheiro designado por `pf9.py`.

(a) Use o pacote SciPy. Use o módulo `scipy.integrate` para calcular a integral de $f(x) = (x + 2)^2$ entre -4 e 4. Use a função `quad`.

(b) Use a função `root` do pacote `scipy.optimize` para devolver as raízes da funçao $f(x)$ no intervalo acima definido.

(c) Use o pacote SciPy. Use o módulo `scipy.integrate` para calcular a integral de $f(x) = sin(x)$ entre -π e π. Use a função `quad`.

(d) Use a função `root` do pacote `scipy.optimize` para devolver as raízes da funçao $f(x)$ no intervalo acima definido.

7 Object Relational Mapping e Web Application Framework

A série de problemas é dedicada aos pacotes de *software* que:

- realizam mapeamentos entre objetos e relações em bases de dados. Este tipo de mapeamento é designado por *Object Relational Mapping* ***ORM***[2],
- permitem a realização rápida de *sites* dinâmicos e leves em termos computacionais.

O **ORM** que é utilizado nesta série de problemas é o SQLAlchemy.
A micro-plataforma *web* usada nesta série de problemas é Flask.

1. Crie uma sub-pasta designada por `PG-1`. Edite nesta pasta um ficheiro designado por `pG1.py`.
   (a) Use `SQLAlchemy` e crie uma conexão para uma base de dados `SQLite3` designada por **colegas.sqlite3**.
   (b) Crie uma tabela, usando uma *base declarativa*, designada por `Contato`. Esta tabela deverá conter os seguintes campos:

   id um número inteiro
   nome uma *string*
   telefone uma *string*
   email uma *string*

   (c) Insira nesta base de dados os contatos de 3 dos seus colegas.
   (d) Demonstre a pesquisa nesta base de dados do contato correspondente a um determinado número de telefone.

2. Crie uma sub-pasta designada por `PG-1`. Edite nesta pasta um ficheiro designado por `pG2.py`.
   (a) Use a base de dados `SQLAlchemy` criada no problema 1. Crie uma tabela, usando uma *base declarativa*, designada por `Mensagem`. Esta tabela deverá conter os seguintes campos:

   id um número inteiro
   destinatario que é uma *foreign key*[3] para uma entrada da tabela `Contato`
   texto uma *string* destinada a representar uma mensagem até 140 caracteres.

   (b) Insira nesta base de dados 3 mensagens em que duas se destinam ao mesmo contato.
   (c) Demonstre a pesquisa nesta base de dados das mensagens correspondentes a um determinado número de telefone.

3. Crie uma sub-pasta designada por `PG-3`. Edite nesta pasta um ficheiro designado por `pG3.py`.
   (a) Utilize o pacote `Flask` para criar um *site* dinâmico. Crie um ficheiro em formato `HTML` designado por `ciao.html` contendo uma mensagem de boas vindas com o seu nome e número de aluno. Utilize um túnel `ssh`, tal como se lecionou na aula prática, para visualizar o resultado do funcionamento com `Flask`.
   (b) Modifique de modo a que o ficheiro `html` que criou se encontre na pasta `templates`.
   (c) Use o pacote `numpy` para criar uma tabela da função $\sin(x)$ entre 0 e π, com intervalos iguais para 100 pontos sucessivos de x.
   (d) Use o `Flask` para gerar dinamicamente uma página `HTML` que apresenta a tabela que criou na alínea anterior.

4. Crie uma sub-pasta designada por `PG-4`. Edite nesta pasta um ficheiro designado por `pG4.py`.
   (a) Aproveite o *site* dinâmico do problema 3. Utilize o pacote FlaskSQLAlchemy no seu site dinâmico.

[2] Vidé Object Relational Mapping (wikipedia).
[3] Em português **chave estrangeira**.

(b) Crie uma base de dados designada por `aluno.db`.

(c) Crie uma tabela representando as disciplinas do seu curso de licenciatura. Nomeadamente com os seguintes campos:

id a chave primária que será um número inteiro
disciplina uma *string*
semestre um número inteiro
ects um número de vírgula flutuante
nota um número de vírgula flutuante

(d) Introduza na base de dados as disciplinas que já realizou e utilize o `Flask` para apresentar esta lista a quem visitar o seu *site*.

5. Crie uma sub-pasta designada por `PG-5`. Edite nesta pasta um ficheiro designado por `pG5.py`.

(a) Aproveite o *site* dinâmico do problema 4.Crie uma tabela designada por `Pais` para poder representa países. Nomeadamente com os seguintes campos:

id a chave primária que será um número inteiro
nome uma *string*
populacao um número inteiro

(b) Introduza na base de dados *Portugal, Espanha, França, Alemanha, Itália* e utilize o `Flask` para apresentar esta lista a quem visitar o seu *site*.

(c) Crie uma tabela designada por `Covid` e com os seguintes campos:

id um número inteiro que constitui a chave primária
pais uma chave estangeira indicando um dos países presentes na base de dados.
casos um número inteiro indicando o número de casos acumulados até aquela data.
data uma data gerada com o pacote `date` da biblioteca padrão de `Python` e que indica a data em que se observaram os casos.

(d) Preencha várias entradas de números de casos e apresente a informação numa página `HTML` gerada dinamicamente com o `Flask`.

6. Crie uma sub-pasta designada por `PG-6`. Edite nesta pasta um ficheiro designado por `pG6.py`.

(a) Coloque um conjunto de 3 imagens em formato JPEG numa pasta designada por `static`.

(b) Apresente estas imagens aleatoriamente de cada vez que refrescar a visualização do seu *site*. Pode usar o *helper* url_for para referenciar as imagens colocadas na pasta `static`.

(c) Use o pacote Flask-Bootstrap para melhorar o aspeto do seu site.

(d) Crie uma página `HTML` com um `link` para outra página `HTML` no seu *site*.

8 Ligação de Python a Outras Linguagens de Programação

A série de problemas é dedicada à integração de código realizado noutra linguagem de programação em código `Python`.

Os problemas irão ser focados em:

- integração de código programado em `C` em código em `Python`
- utilização da ferramenta `SWIG` para automatização do processo
- automatização do processo com a ferramenta de controlo da compilação `Make`

1. Crie uma sub-pasta designada por `PH-1`. Edite nesta pasta um ficheiro designado por `ph1.py`.
   (a) Crie um ficheiro designado por `fibonacci.c`. Use a linguagem de programação `C` para escrever uma versão recursiva da função que permite gerar a sequência de números de `Fibonacci`.
   Vidé uma realização em: https://www.tutorialspoint.com/data_structures_algorithms/fibonacci_recursive_program_in_c.htm.
   (b) Utilize a aplicação Simplified Wrapper and Interface Generator (SWIG) para produzir o módulo designado por `problemas` que é utilizado para chamar a função `fibonacci`.
   (c) Automatize o processo de geração do módulo `problemas` usando a ferramenta de controlo da compilação Make.
   (d) Acrescente ao módulo `problemas` a seguinte função:
   $$y(x) = x\cos(x) - 2.0$$
   usando a aplicação `Make`.
2. Crie uma sub-pasta designada por `PH-2`. Edite nesta pasta um ficheiro designado por `ph2.py`.
   (a) Consulte a documentação relativa à biblioteca de `C`, Python/C API Reference Manual, que permite a utilização de objetos de `Python` em `C`. Escreva uma função na linguagem de programação `C`, designada por `modulus()` e que devolve a soma dos valores dos elementos dum túplo de valores do tipo `float`. Use a directiva de compilação `-I`, com o argumento apropriado, para que se possa encontrar o ficheiro `Python.h`.
   (b) Use a aplicação `SWIG` e crie um módulo importável em `Python` designado por `problemas`.
   (c) Escreva um ficheiro de controlo da compilação com a ferramenta `Make` para a produção deste módulo.
   (d) Acrescente ao módulo uma função escrita em `C` que permite calcular a raiz quadrada da soma dos elementos do túplo atrá mencionado. A função `sqrt` encontra-se disponível com o ficheiro *header* `math.h` e na biblioteca `libm.so`. Use a diretiva `-lm` para que se possa ligar o código da biblioteca.
3. Crie uma sub-pasta designada por `PH-3`. Edite nesta pasta um ficheiro designado por `ph3.py`.
   (a) Escreva uma função na linguagem de programação `C`, designada por `criar_lista_senos()` que recebe como argumento uma lista de números do tipo `float` e que devolve a lista de senos correspondente. Insira esta função no módulo `problemas` anteriormente definido para utilização em `Python`.
   (b) Acrescente ao módulo uma função designada por `conta_elementos()` que recebe uma lista de listas de números inteiros e que devolve uma lista em que em cada posição se encontra o número de elementos da lista correspondente.
   (c) Acrescente ao módulo uma função em `C` designada por `criar_dicionario()` que devolve um objeto de `Python` do tipo dicionário com 5 elementos. A chave é um inteiro e o valor um número do tipo `float`. Escolha estes valores arbitrariamente.
   (d) Acrescente ao módulo uma função em `C` designada por `devolve_conjunto()` que devolve um conjunto formado a partir do argumento `lista`. Esta lista é composta por um determinado número de inteiros (repita 3 vezes um destes números).

Referências

Paperpile. (2020). Complete list of BibTeX entry types [https://www.bibtex.com/e/entry-types/].