ISEL – Instituto Superior de Engenharia de Lisboa

ADEETC – Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

LEIM

Licenciatura em Engenharia Informática e Multimédia

Unidade Curricular de Projeto

Exercícios Python de correção automática

Miguel Távora (45102)

Duarte Domingues (45140)

Orientador

Professor [Doutor] João Beleza

setembro, 2021

Resumo

Este projeto consiste no desenvolvimento de múltiplos programas, na linguagem de programação Python, que permitem gerar diferentes versões de um exercício base. Este desenvolvimento foi aplicado a 62 exercícios base originais.

O projeto surge da necessidade da criação de exercícios de trabalhos de casa para serem utilizados por uma plataforma de envio e correção automática de trabalhos de casa. Esta plataforma de trabalhos de casa é utilizada nas unidades curriculares Matemática Discreta e Programação e Matemática para Computação Gráfica, no curso LEIM do ISEL.

Neste projeto foram desenvolvidos múltiplos programas para gerar diferentes versões de exercícios baseados na linguagem de programação Python. O objetivo final do desenvolvimento de cada exercício é a produção de ficheiros em formato PDF e PNG, com as diferentes versões do exercício, geradas a partir da versão base. Os ficheiros PDF são utilizados pela plataforma para construir um PDF com as diferentes perguntas que constituem um trabalho de casa. Os ficheiros PNG são usados pela plataforma para disponibilizar o trabalho de casa numa aplicação web, acessível através de um *browser*, que permite submeter as respostas e efetuar a correção automática dos trabalhos. Cada exercício possui um único enunciado e diversas perguntas, podendo o número de perguntas variar entre exercícios.

Para a geração das diferentes versões dum exercício foi desenvolvido um sistema. Para ajudar na criação das diferentes versões foi utilizada uma biblioteca fornecida pelo docente.

O sistema desenvolvido é flexível, permitindo a criação automática de múltiplas versões, uma para cada aluno, para diferentes tipos de trabalhos de casa. Os exercícios criados por este sistema não estão somente restritos a exercícios baseados na linguagem de programação Python, permitindo a criação de uma variedade de tipos de exercícios, do tipo verdadeiro ou falso.

Usando o sistema desenvolvido, foram criados um total de 62 exercícios baseados na linguagem de programação Python. Cada trabalho de casa desenvolvido foi minuciosamente testado e analisado de forma a evitar erros, garantir coerência entre o exercício e a solução e manter o mesmo nível de dificuldade para todas as diferentes versões.

Abstract

This project consists in the development of multiple programs, written in the Python programming language, that allow the creation of different versions of a base exercise. This development was applied to 62 base exercises.

The project comes from the necessity of the creation of homework exercises destined to be used by a platform for automatic submission and correction of homework. This homework platform is currently being used on the subjects, Discrete Mathematics and Programming and Mathematics for Computer Graphics, in ISEL's LEIM course.

In this project, multiple programs were created to generate different versions of exercises based in the Python programming language. The final objective of the development of each exercise is the production of files in PDF and PNG format, with the different versions of the exercises, generated from a base version. The PDF files are used by the platform to construct a PDF with the different questions that constitute a homework. The PNG files are utilized by the platform to provide the homework in a web application, accessible through a browser, that allows the submission of the answers by the students and perform the automatic correction of the homework. Each exercise possesses a single statement and different questions, the number of questions might vary from exercise to exercise.

For the automatic generation of the different versions of an exercise, a system was developed. In order to help in the creation of the different versions a library supplied by the advisor was used.

The developed system is flexible, allowing the automatic creating of multiple versions of an exercise, one for each student, for different types of homeworks. The exercises created by this system are not restricted to only exercises based on the Python programming language, allowing the creation of a wide variety of types of exercises, in true or false format.

Stem from the developed system, in total 62 exercises were created based on the Python programming language. Each developed homework exercise was thoroughly tested and analyzed to avoid errors, guarantee coherence between the exercise and the solution and assure the same level of difficulty for every version.

Agradecimentos

Pretendemos agradecer a todos os colegas e docentes que ajudaram com o percurso académico, especialmente ao docente João Beleza que permitiu um aperfeiçoamento da implementação do sistema de geração de versões. Agradecemos também aos docentes Paulo Trigo e Hélder Bastos que forneceram os alicerces para criar um projeto mais elaborado e também pelo facto de estarem disponíveis para responder a eventuais questões.

Índice

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Este projeto tem como objetivo criar trabalhos de casa que serão posteriormente disponibilizados numa plataforma de envio e correção automática de trabalhos de casa. Esta plataforma é alimentada por ficheiros PDF e PNG, estes ficheiros são nomeadamente o produto final do projeto, onde os ficheiros correspondem aos enunciados e às perguntas dos trabalhos de casa. A plataforma funciona de tal maneira que cada aluno tem acesso ao seu trabalho de casa através de um *link* na *Web*, e será nesse mesmo *link* que irá submeter as suas respostas. O facto do trabalho de casa ser disponibilizado na *Web* é o principal factor para utilizar ficheiros no formato PDF e PNG, onde são embutidos na página como um ficheiro.

Um dos problemas envolvidos na criação dos trabalhos de casa é o facto de ser necessário gerar uma versão diferente para cada aluno, de forma a evitar que os alunos copiem. Um outro problema é a necessidade das versões dos trabalhos de casa serem diferentes, contudo mantendo a mesma dificuldade do exercício entre versões de forma a não prejudicar nenhum aluno. Uma parte interessante do projeto é conseguir que as premissas anteriores sejam cumpridas e também conseguir generalizar os trabalhos de casa a qualquer tipo de pergunta e não somente perguntas de programação na linguagem Python. Para garantir a funcionalidade referida anteriormente optou-se por realizar perguntas do tipo verdadeiro ou falso. Desta forma suportam-se tanto exercícios de programação em Python, como de qualquer outra matéria.

Para realizar um trabalho de casa é necessário construir o enunciado e as versões verdadeiras e falsas das perguntas. Tanto o enunciado como as perguntas são ficheiros LaTeX, isto é, ficheiros de texto com extensão `.tex` Os ficheiros de texto são do tipo LaTeX para ser depois possível converter para PDF.

Para uma pergunta ficar completa, além dos ficheiros LaTeX é também necessário desenvolver um programa Python que será o responsável por alterar os ficheiros do enunciado e das perguntas. Este programa tem de ser diferente para cada trabalho de casa, isto porque os exercícios são geralmente muito diferentes entre si. Na maior parte dos casos os enunciados contêm, eles próprios, programas Python. Nestes programas os nomes das variáveis e os seus valores, por exemplo, podem variar de tal forma que inviabiliza a reutilização de programas de alteração de exercícios para gerar as diferentes versões de cada exercício.

Caso a pergunta seja do tipo de programação Python é necessário também um ficheiro com o programa Python do enunciado do exercício. Este programa é diferente do programa Python que realiza as alterações nas perguntas. Caso o aluno tenha de implementar código, que não conste no enunciado, para conseguir responder à pergunta é também necessário criar um ficheiro Python, com a solução do exercício.

1.1 Motivação

Uma das motivações para desenvolver este projeto foi a grande necessidade de criação de novos exercícios para serem utilizados na plataforma de trabalhos de casa. Neste momento, a plataforma está a ser utilizada nas unidades curriculares Matemática Discreta e Programação e Matemática para Computação Gráfica.

Um outro aspeto importante na decisão da escolha do projeto, foi o facto do projeto se basear em automação de tarefas na linguagem de programação Python. A automação de tarefas é uma área em crescimento que oferece inúmeros benefícios, eliminando a necessidade da realização de ações repetiti-

vas por um humano. Desta forma é aumentada a produtividade e diminui-se a quantidade de erros, assegurando uma maior precisão e volume de produção.

O Python é a linguagem interpretada mais usada atualmente. Uma das grandes vantagens do Python é a sua flexibilidade sendo excelente para diferentes áreas tecnológicas e cientificas, como por exemplo, inteligência artificial, aprendizagem automática, análise de dados, desenvolvimento de aplicações e automação de tarefas. Este projeto foi visto também como uma oportunidade de aprofundar e solidificar os conhecimentos da linguagem Python.

Capítulo 2

Trabalho Relacionado

2.1 Plataforma de trabalhos de casa do Engenheiro Gonçalo Marques

Plataforma de trabalhos de casa do Engenheiro Gonçalo Marques é uma plataforma de envio de trabalhos de casa utilizada na unidade curricular Aprendizagem Automática dos cursos LEIM e LMATE do ISEL, da autoria do Engenheiro Gonçalo Marques.

Esta plataforma assenta no envio automático de trabalhos de casa, através de caixa de correio electrónico, para todos os alunos inscritos numa unidade curricular. Esta plataforma de trabalhos de casa é semelhante à implementada pelo Engenheiro João Sousa. Um aspeto semelhante é na geração de múltiplas versões aleatórias de um enunciado de trabalho de casa, uma para cada aluno, geradas em ficheiros no formato PDF.

A plataforma de trabalho de casa do Engenheiro Gonçalo Marques funciona à base de perguntas de escolha múltipla, pelo contrário, a plataforma utilizada no projeto fornecida pelo orientador João Sousa é apoiada por perguntas verdadeiro e falso.

O formato dos diversos enunciados de exercícios, criados em formato PDF, por a plataforma de trabalho de casa do Engenheiro Gonçalo Marques, serviu de inspiração para o nosso projeto. Desta forma foram criados os enunciados

dos diversos exercícios com um aspeto parecido no nosso projeto.

Um exemplo de um exercício gerado por esta plataforma é apresentado na seguinte figura:

1. Considere os 320 primeiros exemplos do conjunto de treino do dígito "8".

   (a) . Pretende-se visualizar os digitos deste conjunto e os vetores prórpios da matriz de covariância dos dados.

   i. A figura de cima é a imagem invertida do 234º dígito do conjunto.

   ii. A figura de baixo é a imagem (ou a imagem invertida) do 6º vetor próprio da matriz de covariância dos dados.

   iii. Todas as respostas anteriores.

   iv. Nenhuma das respostas anteriores.

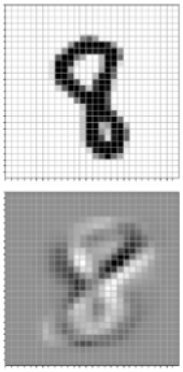


Figura 2.1: Exemplo de um exercício utilizado na plataforma de trabalhos de casa do Engenheiro Gonçalo Marques

2.2 Plataforma de exercícios do Moodle

O Moodle é um *software* livre de apoio à aprendizagem executado num ambiente virtual, fundado por Martin Dougiamas em 2002. O Moodle dispõe de uma plataforma de exercícios que permite realizar fichas, exames e trabalhos de casa no Moodle a partir do *browser*.

Esta plataforma do Moodle funciona à base de formulários na Web que permitem realizar perguntas de vários tipos. É possível realizar perguntas do tipo escolha múltipla, verdadeiro e falso, texto escrito e operações de *drag and drop* de caixas de texto. Dentro dos exercícios é possível imbutir imagens ilustrativas dos mesmos.

Neste sistema os professores têm de criar os exercícios manualmente, contudo todos os exercícios são iguais para todos os alunos. Em alternativa é pretendido realizar um sistema que consiga gerar diversas versões diferentes automaticamente.

Figura 2.2: Exemplo de exercício realizado no Moodle

Capítulo 3

Modelo Proposto

No presente capítulo iremos analisar quais são os requisitos e como serão abordados posteriormente na implementação. Neste sentido será recolhida informação para alcançar os objetivos.

3.1 Requisitos

A análise de requisitos é uma parte fundamental no desenvolvimento do sistema. Os requisitos devem ser bem analisados pois será neles que todo o projeto será posteriormente desenvolvido. Caso exista uma lacuna nesta fase, ela será propagada por todas as fases subsequentes. Nesta fase serão abordados os seguintes tópicos:

- Quais as problemáticas associadas ao trabalho.
- Quais são os objetivos que são necessários concretizar.
- Caracterização pormenorizada sobre as funções e as atributos do sistema.
- Concretização dos casos de utilização com base nos pontos referidos anteriormente.
- Agenda dos objetivos.

3.2 Análise de Requesitos

3.2.1 Problemáticas

Este projeto possui duas problemáticas fundamentais, respetivamente:

- Por detrás de cada pergunta há a necessidade um programa de computador para gerar as centenas de versões necessárias.
- Cada pergunta tem de ser individualmente programada, pois os nomes das variáveis e os seus valores variam entre cada pergunta. Por isso não é possível reutilizar um programa de uma pergunta para outra pergunta.
- Um exemplo da dificuldade da criação de cada pergunta seria fazer um exercício que envolvesse a fórmula resolvente, onde nos teríamos de escolher os valores de A, B e C. Quem está a resolver o exercício não conhece os números imaginários e por isso os valores de A, B e C tem de ser cuidadosamente escolhidos de forma a não resultar em números imaginários.

3.2.2 Síntese de Objetivos

Neste projeto será desenvolvido um sistema que permite gerar diversas versões de um exercício base. O processo de criação de versões é feito automaticamente através de um programa desenvolvido na linguagem de programação Python. Constitui o objetivo principal aplicar o sistema desenvolvido para criar diversos exercícios referentes à linguagem de programação Python

Pretende-se que os exercícios desenvolvidos venham a ser utilizados nas unidades curriculares Matemática Discreta e Programação e Matemática para Computação Gráfica do curso LEIM do Instituto Superior de Engenharia de Lisboa.

3.2.3 Clientes alvo

O público-alvo serão os docentes responsáveis das unidades curriculares Matemática Discreta e Programação e Matemática para Computação Gráfica, os alunos inscritos nestas unidades curriculares e futuros alunos.

3.2.4 Metas a alcançar

- Criar um sistema que contemple as partes comuns entre os exercícios, de forma a não ser necessário implementar sempre todos os passos para gerar as diferentes versões.
- Criar modelos de exercícios base para gerar diferentes versões das perguntas e respostas.
- Elaboração de um programa para alterar os enunciados base e gerar versões.
- Construção de perguntas baseadas em unit tests. Por exemplo existe um programa, neste caso a construção e a indexação de uma lista, e há uma bateria de testes que pretendem averiguar se a indexação foi bem executada. Neste contexto a bateria de testes são as respostas do tipo verdadeiro ou falso.
- Permitir que as versões desenvolvidas fiquem disponíveis na plataforma de trabalhos de casa utilizada em MDP e MCG.
- Permitir que diferentes docentes das unidades curriculares consigam utilizar o sistema desenvolvido facilmente, sem terem que entender como funciona internamente.

3.3 Caracterização Pormenor

3.3.1 Funções de Sistema

As funções do sistema representam aquilo que é suposto o sistema realizar. A categorização das diversas funções do sistema é importante de modo a definir as prioridades, identificar os requisitos mais importantes e custos associados.

Na seguinte tabela estão apresentados as diferentes categorias de funções do sistema.

Categoria	**Significado**
Evidente	Tem que ser realizada. O utilizador tem que ter conhecimento da sua realização.
Invisível	Tem que ser realizada. Não é visível para os utilizadores.
Adorno	Opcional. Não afecta significativamente o custo ou outras funções.

Tabela 3.1: Tipos de funções do sistema

Na figura que se segue serão apresentados todas as funções do projeto que serão posteriormente implementadas, utilizando as categorias referidas anteriormente.

Ref. #	**Função**	**Categoria**
Ref. 1.1	Escrever o programa Python de uma pergunta dum exercício base	Evidente
Ref. 1.2	Construção manual das perguntas e respostas em LaTeX dum exercício base	Evidente
Ref. 1.3	Gerar uma diretoria que contém todas as diretorias das versões do enunciado e das perguntas	Evidente
Ref. 1.4	Ler ficheiros Python e LaTeX	Invisível
Ref. 1.5	Alterar o ficheiro Python dum enunciado	Invisível
Ref. 1.6	Alterar o código Python dentro do ficheiro LaTeX do enunciado do exercício	Invisível

Ref. 1.7	Gerar um novo ficheiro Python com código alterado previamente	Evidente
Ref. 1.8	Gerar um ficheiro texto com o output da execução do código Python	Evidente
Ref. 1.9	Gera um novo ficheiro LaTeX através de alterações realizadas ao ficheiro da pergunta dum exercício	Evidente
Ref. 1.10	Alterar os ficheiros LaTeX com as diferentes perguntas verdadeiro e falso	Invisível
Ref. 1.11	Gerar ficheiros LaTeX com as perguntas do exercício alteradas	Invisível
Ref. 1.12	Gerar ficheiros PDF	Evidente
Ref. 1.13	Gerar ficheiros PNG	Evidente
Ref. 1.14	Disponibilizar um exercício completo na Plataforma de envio de trabalhos de casa	Evidente
Ref. 1.15	Criar um programa para juntar o enunciado e perguntas dum exercício em ficheiros PNG e PDF	Adorno
Ref. 1.16	Adicionar *sintax highlighting* ao código presente nos enunciados dos exercícios	Adorno

Tabela 3.2: Funções do sistema

3.3.2 Atributos do sistema

Os atributos do sistema, representam as qualidades-não funcionais do sistema, são características ou dimensões do sistema. Estes atributos podem ser divididos nas seguintes categorias:

Categoria	**Significado**
Obrigatório	Tem que ser contemplado.
Desejável	Deve estar preparado para alcançar

Tabela 3.3: Tipos de atributos do sistema

Na tabela que se segue serão apresentados os atributos do sistema que são pretendidos criar utilizando as categorias referidas na tabela anterior.

Atributos	**Atributo Detalhe/ Restrição de Fronteira**	**Categoria**
Plataforma	Linguagem programação Python	Obrigatória
Facilidade de utilização	Aprendizagem fácil do método de funcionamento da geração de perguntas do sistema	Desejável
Tolerância a falhas	Ter formas de minimizar os danos causados por eventuais erros que ocorram	Desejável
Compatibilidade	Compatibilidade com diversos sistemas operativos	Obrigatória
Rapidez do sistema	Execução do sistema automático de geração de diversas versões de cada exercício deve ser rápida	Desejável
Confiabilidade	Ter uma baixa taxa de ocorrência de falhas	Desejável
Volume de utilização	Suportar a criação de exercícios para um número elevado de diferentes alunos	Obrigatório

Tabela 3.4: Atributos do Sistema

3.3.3 Atributos e funções do sistema

É importante evidenciar todas as relações entre as funções do sistema e atributos do sistema, para isto foi criada uma tabela. Na seguinte tabela encontram-se as seguintes relações entre funções e atributos do sistema.

Ref. #	**Função**	**Atributos do sistema**
Ref. 1.1	Escrever o programa Python de uma pergunta dum exercício base	Plataforma
Ref. 1.2	Construção manual das perguntas e respostas em LaTeX dum exercício base	Confiabilidade
Ref. 1.3	Gerar uma diretoria que contém todas as diretorias das versões do enunciado e das perguntas	Plataforma, Compatibilidade
Ref. 1.4	Ler ficheiros Python e LaTeX	Plataforma, Compatibilidade
Ref. 1.5	Alterar o ficheiro Python dum enunciado	Plataforma, Compatibilidade, Tolerância a falhas, Confiabilidade, Volume de utilização
Ref. 1.6	Alterar o código Python dentro do ficheiro LaTeX do enunciado do exercício	Plataforma, Compatibilidade, Volume de utilização, Confiabilidade
Ref. 1.7	Gerar um novo ficheiro Python com código alterado previamente	Plataforma, Compatibilidade, Tolerância a falhas
Ref. 1.8	Gerar um ficheiro texto com o output da execução do código Python	Confiabilidade, Compatibilidade
Ref. 1.9	Gera um novo ficheiro LaTeX através de alterações realizadas ao ficheiro da pergunta dum exercício	Plataforma, Compatibilidade, Volume de utilização, Confiabilidade

Ref. 1.10	Alterar os ficheiros LaTeX com as diferentes perguntas verdadeiro e falso	Plataforma, Compatibilidade, Tolerância a falhas
Ref. 1.11	Gerar ficheiros LaTeX com as perguntas do exercício alteradas	Plataforma, Compatibilidade, Tolerância a falhas, Confiabilidade
Ref. 1.12	Gerar ficheiros PDF	Plataforma, Compatibilidade, Volume de utilização, Confiabilidade, Rapidez do sistema
Ref. 1.13	Gerar ficheiros PNG	Plataforma, Compatibilidade, Volume de utilização, Confiabilidade, Rapidez do sistema
Ref. 1.14	Disponibilizar um exercício completo na Plataforma de envio de trabalhos de casa	Confiabilidade, Compatibilidade, Volume de utilização
Ref. 1.15	Criar um programa para juntar o enunciado e perguntas dum exercício em ficheiros PNG e PDF	Plataforma, Facilidade de utilização, Compatibilidade
Ref. 1.16	Adicionar *sintax highlighting* ao código presente nos enunciados dos exercícios	Facilidade de utilização

Tabela 3.5: Relação entre funções e atributos do sistema

3.3.4 Casos de utilização

Os casos de utilização servem para descrever o processo, do início ao fim, de uma sequência de eventos necessárias para produzir ou completar uma ação para um ator. Um ator pode ser um humano ou uma entidade externa que interage com o sistema. Os casos de utilização podem ser identificados de duas formas, focado nos atores ou focado nos eventos. Os casos de utilização definidos neste projeto são os focados nos atores, portanto é necessário:

- Identificar os atores relacionados com o sistema
- Por ator, identificar os processos que ele inicia ou participa

Na figura seguinte estão apresentados os atores presentes do sistema e os respetivos processos em que estes iniciam ou participam.

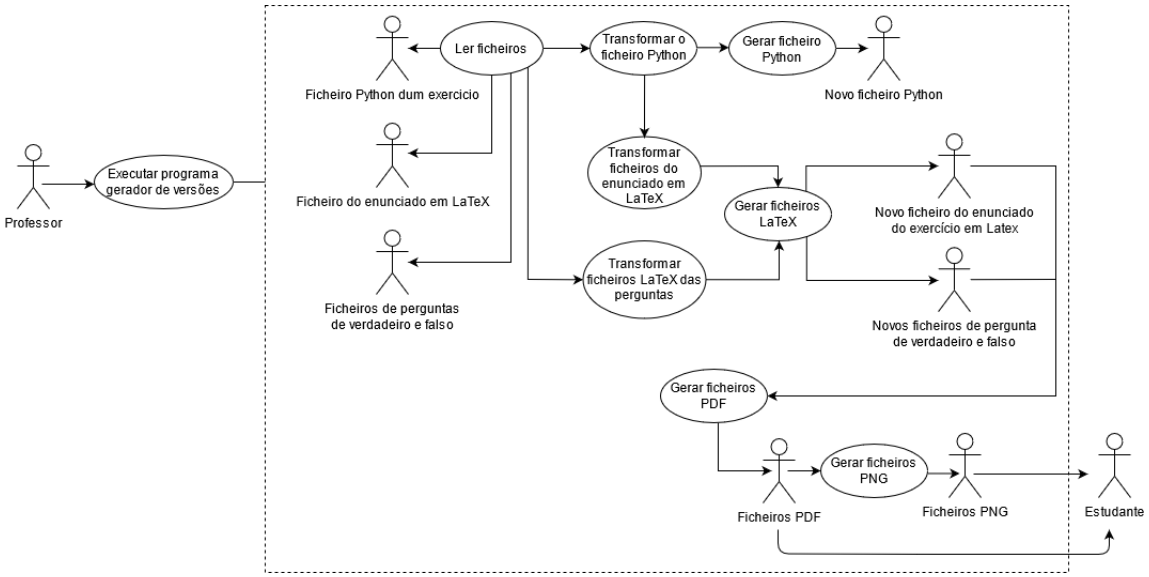


Figura 3.1: Casos de utilização

Casos de utilização - Resumido

De seguida serão apresentadas as tabelas dos casos de utilização em particular, que contêm um nome ilustrativo do caso, resumo da ação do caso e as referências às funções dos sistema.

Cabeçalho	
Nome:	Executar programa gerador de versões
Resumo:	Executar um ficheiro Python que gera diferentes versões dum exercício base, criadas em diferentes diretorias.
Referências:	R1.2, R1.3, R1.4, R1.5, R1.6, R1.7, R1.8, R1.9, R1.10, R1.11, R1.12

Tabela 3.6: Caso de utilização - Executar programa gerador de versões

Cabeçalho	
Nome:	Ler ficheiros
Resumo:	Ler os diferentes ficheiros Python e LaTeX associados a um exercício
Referências:	R1.4

Tabela 3.7: Caso de utilização - Ler ficheiros

Cabeçalho	
Nome:	Transformar o ficheiro Python
Resumo:	Transformar o ficheiro Python de modo a gerar uma versão diferente da original
Referências:	R1.5

Tabela 3.8: Caso de utilização - Transformar o ficheiro Python

Cabeçalho	
Nome:	Gerar ficheiro Python
Resumo:	Gerar um novo ficheiro Python a partir do código alterado previamente
Referências:	R1.7

Tabela 3.9: Caso de utilização - Gerar ficheiro Python

Cabeçalho	
Nome:	Transformar ficheiro do enunciado em LaTeX
Resumo:	Transformar o ficheiro LaTeX do enunciado com o código de Python anteriormente alterado
Referências:	R1.6

Tabela 3.10: Caso de utilização - Transformar ficheiros do enunciado em LaTeX

Cabeçalho	
Nome:	Transformar ficheiros LaTeX da pergunta
Resumo:	Transformar o ficheiro LaTeX com as perguntas de verdadeiro e falso
Referências:	R1.10

Tabela 3.11: Caso de utilização - Transformar ficheiros LaTeX da pergunta

Cabeçalho	
Nome:	Gerar ficheiros LaTeX
Resumo:	Gerar novos ficheiros LaTeX a partir de alterações realizadas previamente pelo código Python
Referências:	R1.9, R.11

Tabela 3.12: Caso de utilização - Gerar ficheiros LaTeX

Cabeçalho	
Nome:	Gerar ficheiros PDF
Resumo:	Gerar ficheiros PDF para os enunciados e as perguntas a partir dos ficheiros LaTeX
Referências:	R1.12

Tabela 3.13: Caso de utilização - Gerar ficheiros PDF

Cabeçalho	
Nome:	Gerar ficheiros PNG
Resumo:	Gerar ficheiros PNG para os enunciados e as perguntas a partir de ficheiros PDF
Referências:	R1.13

Tabela 3.14: Caso de utilização - Gerar ficheiros PNG

Casos de utilização - Expandido

Nos presentes casos de utilização será apresentado o fluxo de utilização do sistema.

Cenário Principal (fluxo típico de eventos)			
Ação do Ator			**Resposta do Sistema**
1	O professor(ator) executa o programa que gera as diferentes versões de um exercício		
		2	O sistema gera diversas versões do exercício em formato de texto
		3	O sistema converte os ficheiros de texto em ficheiros PDF
		4	O sistema converte os ficheiros PDF em ficheiros PNG
5	O professor disponibiliza os diversos enunciados na plataforma de trabalhos de casa		

Tabela 3.15: Cenário de utilização do sistema

3.4 Fundamentos

Nesta secção serão abordados os tópicos principais que irão tornar o projeto possível de concretizar. As ideias principais são:

- Os exercícios realizados durante todo o projeto são exercícios simples sobre perguntas de programação em Python. Um outro interesse é o facto de melhorar a aprendizagem dos alunos no que toca á programação em Python. Um foco importante a ter em conta é os alunos serem obrigados a realizar código de forma a conseguirem resolver corretamente os exercícios. Isto pois programar aprende-se praticando e não somente a ouvir ou ler.

- O facto da linguagem de programação utilizada ser o Python. Como o Python é uma linguagem dinâmica o programador não tem que definir os tipos das variáveis, aumentado a velocidade de produção de código. Uma outra vantagem é o facto do Python ser portável entre diferentes sistemas operativos. No desenvolvimento do sistema vai ser utilizada uma biblioteca Python da autoria do orientador, que interpreta, como *string*, código Python, sendo assim necessário utilizar Python para o desenvolvimento do sistema. Por fim, o Python possui também uma grande quantidade de bibliotecas que permitem alteração e geração de ficheiros.

- Utilizar automatização de tarefas é um detalhe importante de forma de ser possível gerar versões automaticamente, não sendo necessário criar manualmente pelo professor. Se for necessário, por exemplo, criar 1500 versões de uma pergunta, isso simplesmente não é possível de fazer manualmente.

- Utilidade do LaTeX para criar uma formatação automática dos ficheiros, permitindo através de comandos simples exibir código e texto. Através da utilização do LaTeX irá ser possível criar ficheiros com um formato predefinido literáriamente correto, permitindo que todos os exercícios possuam uma estrutura constante.

- Para melhorar a gestão de tempo foi criado uma agenda com os objetivos principais do projeto e as datas pretendidas para a sua realização.

3.5 Abordagem

Nesta fase é pretendido mostrar a abordagem que será tomada para construir todo o projeto. Para isso será utilizada a agenda que se segue para melhor obter uma gestão de tempo.

Semana	Objetivo	Funcionalidade
22 março - 25 março	Correr o exemplo fornecido pelo orientador	Instalação dos programas de suporte
26 março - 2 abril	Gerar diversas versões do enunciado e das perguntas em ficheiros LaTeX	Criação do sistema
3 abril - 10 abril	Gerar versões PDF e PNG a partir das versões geradas em LaTeX	Criação do sistema
11 abril - 2 maio	Criar exercícios para serem utilizados pela plataforma	Criação de exercícios
3 maio - 9 maio	Validação do correto funcionamento dos exercícios	Validação do sistema
10 maio - 30 maio	Criar mais exercícios para serem utilizados pela plataforma	Criação de exercícios
31 maio - 7 junho	Criar o programa que permita juntar ficheiros LaTeX, tudo no mesmo ficheiro com o enunciado e as perguntas	Criação do sistema
8 junho - 15 junho	Criar mais exercícios para serem utilizados pela plataforma	Criação de exercícios
16 junho - 23 junho	Validar a correta criação das diversas versões	Validação do sistema
24 junho - 31 julho	Melhoramento do sistema de geração de versões	Validação do sistema
1 agosto - entrega	Criar mais exercícios e realizar o relatório	Criação de exercícios

Tabela 3.16: Agenda dos objetivos

Pela agenda é possível denotar que primeiramente será necessário instalar e estudar o funcionamento das bibliotecas úteis para o projeto, nomeadamente a biblioteca para gerar ficheiros PDF a partir de ficheiros LaTeX (pdflatex), uma biblioteca para gerar ficheiros PNG a partir de ficheiros PDF(ghostscript) e uma biblioteca fornecida pelo orientador para alterar *strings*(`question_transformer`).

- Após ter as bibliotecas todas a funcionar, é pretendido criar a nossa implementação de todo o sistema de geração de versões. Será a partir deste sistema que serão geradas todas as versões dos exercícios.

- Quando o sistema estiver a funcionar, serão então criados os exercícios, para serem utilizados pela plataforma de trabalhos de casa.

- Como por vezes existem falhas de implementação no código será necessário despender algum tempo a validar a correta obtenção dos resultados.

- De seguida, para melhorar a visualização dos resultados, irá ser criado um sistema que permite juntar ficheiros LaTeX num único ficheiro.
- Um ponto importante no melhoramento do sistema irá ser conseguir escrever num ficheiro de texto qual é o *output* produzido pelo programa Python com a solução do exercício.
- Por fim, será melhorado o sistema, de maneira a permitir replicar a geração de uma versão. A partir do valor do gerador de números aleatórios utilizada na criação de uma versão será possível recriar o exercício. Desta forma o sistema consegue resistir melhor a possíveis erros numa determinada versão. Por exemplo, caso surja alguma dúvida por parte de um aluno numa versão especifica de um exercício é possível recriar rapidamente esta versão e verificar se a geração do exercício foi efetuada corretamente.

Capítulo 4

Implementação do Modelo

Neste capítulo, será explicada em pormenor, a implementação do modelo projetado no capítulo 3. Desta forma, irão ser esclarecidos com rigor, todos os aspetos e detalhes relevantes do projeto. Irão primeiramente ser referido quais foram as diversas tecnologias utilizadas e a forma como foram aplicadas no projeto.

4.1 Tecnologias

A linguagem de programação usada para a implementação do sistema foi o Python. Foi necessário numa fase inicial escolher e estudar os programas e bibliotecas mais adequadas para a realização de certas partes do trabalho.

4.1.1 Python

O Python é uma linguagem de programação de alto nível, interpretado e dinâmico. A sua *syntax* ajuda os programadores a produzir código em menos passos quando comparado com o C++ ou Java. A grande flexibilidade de ser possível trabalhar com ou sem paradigmas orientados a objetos constitui uma grande vantagem comparado a outras linguagens.

- Em informática uma linguagem de programação de alto nível é uma linguagem de programação com forte abstração dos detalhes do computador. Ao contrário das linguagens de baixo nível torna o processo de desenvolvimento de um programa mais simples e compreensível. A

quantidade de abstração que a linguagem de programação fornece é que define quão alto é o seu nível.

- Uma linguagem interpretada é uma linguagem que lê instruções, sem as compilar, e as traduz para código máquina.

- Uma linguagem dinâmica é uma linguagem que não é necessário definir o tipo das variáveis e é possível alterar o tipo da variável durante a execução.

4.1.2 LaTeX

O LaTeX é um sistema de preparação de documentos. Em LaTeX, a estrutura geral do documento é definida através dum ficheiro de texto que contém conteúdo definido por uma linguagem de marcação que utiliza códigos LaTeX especiais. Os documentos são guardados em ficheiros com a extensão `.tex`, podendo ser compilados a partir duma distribuição TeX como o MikTeX e o TeX Live. Os ficheiros compilados produzem um documento de saída, que pode ser visualizado, em DVI ou PDF.

O objetivo final do programa é obter diversas versões feito através de substituição de *strings* de forma que o modelo base do exercício não seja misturado com o programa Python de geração de versões. A melhor escolha para realizar este tipo de abordagem é feito através da utilização do LaTeX. Isto porque é possível manter os ficheiros LaTeX do exercício base separados do programa de geração de versões. Os ficheiros do exercício são ficheiros com extensão `.tex` e o programa de geração de versões possui extensão `.py`.

Os modelos base dos exercícios foram criados em ficheiros de texto LaTeX. Os enunciados e as perguntas estão em ficheiros LaTeX separados, de forma a gerar ficheiros PDF para cada enunciado e as perguntas verdadeiro e falso. Isto permite a reordenação automática do enunciado e das perguntas na altura da disponibilização das perguntas aos alunos.

Existe uma grande lista de vantagens, associadas à utilização de LaTeX neste projeto, as vantagens principais são as seguintes:

- *Software* livre – O LaTeX é totalmente gratuito e livre para uso.
- Compatibilidade – O LaTeX funciona em todos os sistemas operativos principais, o que permite remover problemas de incompatibilidade no sistema implementado.
- Criação de ficheiros com um padrão específico – LaTeX permite criar ficheiros de texto com um formato específico, permitindo que todos os ficheiros dos exercícios sigam uma estrutura predefinida, preparada para ser utilizada na plataforma de trabalhos de casa.
- Flexibilidade – Os ficheiros LaTeX oferecem diversas funcionalidades, não só de escrita de texto, como permite incluir ficheiros PDF e noutras extensões no seu interior e também permite incluir código com *syntax highligthing*.
- Facilidade de geração de ficheiros PDF – A partir de extensões de TeX, como o pdflatex, é fácil e rápido produzir ficheiros PDF, a partir de ficheiros de texto.
- Ficheiros leves – Os documentos TeX são leves e de fácil exportação, sendo assim ideais para o projeto, visto que para cada exercício é gerado um número muito elevado de documentos.
- Elevada gama de funcionalidades - Devido à elevada gama de funcionalidades do LaTeX, e ao facto de se poder criar funções, é possível simplificar e facilitar procedimentos, como por exemplo, adicionar *Sintex Highlighting* a código, permitindo criar documentos esteticamente mais apelativos.

Uma possível desvantagem associada ao LaTeX é a sua íngreme curva de aprendizagem, porém isto não foi um problema grave devido à existência previa de conhecimento deste software.

4.1.3 `pdflatex`

O `pdflatex` é uma ferramenta livre para uso, que permite gerar ficheiros PDF a partir de ficheiros de texto LaTeX. O `pdflatex` é construído sobre `pdfTeX`, uma extensão de TeX, para produção de ficheiros PDF a partir de ficheiros de fonte TeX.

De forma a poder utilizar o `pdflatex`, foi necessário instalar uma distribuição TeX, nomeadamente o MiKTeX, que consiste numa implementação do sistema TeX e um conjunto de programas relacionados, inclusive, `pdflatex`. Poderia ter sido utilizada outra distribuição TeX, como por exemplo, TeX Live, porém, devido à facilidade de instalação e capacidade de atualização automática, optou-se pelo uso do MiKTeX. A utilização do `pdflatex` é simples e funciona nos diversos sistemas operativos principais, Windows, Linux e macOS. Para realizar a conversão de um ficheiro de texto LaTeX para PDF é apenas necessário utilizar o seguinte comando:

```
pdflatex "nome_do_ficheiro.tex"
```

De seguida, encontra-se um exemplo da transformação dum ficheiro LaTeX para um ficheiro PDF:

```
\documentclass[12pt,varwidth=16cm,border=1pt]{standalone}

\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[portuguese]{babel}

\begin{document}

A classe \verb+F+ não tem atributos.

\end{document}
```

Figura 4.1: Conteúdo de um ficheiro LaTeX original

A classe `F` não tem atributos.

Figura 4.2: Documento PDF gerado a partir de um ficheiro LaTeX

4.1.4 Ghostscript

O Ghostscript é um interpretador para ficheiros PDF e funciona como conversor de formatos de arquivo. No caso do projeto será utilizado o conversor de arquivos nomeadamente para converter ficheiros PDF para PNG. Esta biblioteca permite realizar operações em linha de comandos e uma interface gráfica para facilitar a interação com o utilizador, a interface gráfica não será utilizada na implementação do sistema. No projeto será utilizado o interpretador, visto que tudo será executado em linha de comandos tal como o `pdflatex`. Esta biblioteca foi toda construída em C e tem compatibilidade com diversos sistemas operativos como por exemplo Windows, macOS e Linux.

Em alternativa ao Ghostscript poderia ser utilizado a biblioteca do Python `pdf2image`. Esta alternativa é melhor no que toca a instalação de software, visto que basta utilizar o comando `pip` da linha de comandos. Por outro lado na utilização do Ghostscript é necessário instalar e definir nas variáveis de ambiente do Windows o `PATH` para o interpretador.

Contudo foram comparados os tempos de execução para ambos os conversores obtidos através dos seguintes *scripts*:

```
import os
import time
start_time = time.time()

for i in range(2,8):
    ghostscript_command = 'gs -r150 -dTextAlphaBits=4 -dGraphicsAlphaBits=4\
     -sDEVICE=png16m -o '+'teste'+str(i)+'.png'+' '+ 'teste'+str(i)+'.pdf'

    os.system(ghostscript_command)

print("--- %s seconds ---" % (time.time() - start_time))
```

Figura 4.3: Código teste do Ghostcript

```python
from pdf2image import convert_from_path
import time
start_time = time.time()

images = []
# Store Pdf with convert_from_path function
for i in range(2,8):
    images.append(convert_from_path("teste"+str(i)+".pdf"))

for i in range(len(images)):
      # Save pages as images in the pdf
    images[i][0].save('page'+ str(i) +'.png', 'PNG')

print("--- %s seconds ---" % (time.time() - start_time))
```

Figura 4.4: Código teste da biblioteca `pdf2image`

Perante os dois *scripts*, onde ambos foram testados nas mesmas condições e com os mesmos ficheiros, onde no total foram utilizados 6 ficheiros, os resultados dos tempos de execução foram respetivamente:

```
--- 1.412022352218628 seconds ---
```

Figura 4.5: Tempo execução do *script* do Ghostscript

```
--- 3.7302820682525635 seconds ---
```

Figura 4.6: Tempo execução do *script* do `pdf2image`

Devido ao tempo de execução da geração das versões do `pdflatex` e do conversor de ficheiros PDF para PNG ser um processo bastante demoroso optou-se pela biblioteca que produzisse resultados mais rapidamente. Como o Ghostscript demorou cerca de metade do tempo de execução foi então optado pela utilização do Ghostscript.

4.1.5 Biblioteca `questions_transformer`

A biblioteca `questions_transformer`, é uma biblioteca escrita na linguagem Python, da autoria do orientador João Beleza, que oferece um mecanismo sofisticado de substituição de *strings*. Utilizámos esta biblioteca para realizar alterações, ao conteúdo em *string*, dos ficheiros Python e LaTeX dos

exercícios. Esta biblioteca foi útil para criar as diferentes versões de um exercício.

Processos de alterações de *strings*, em código, são sempre difíceis devido ao facto de existir uma gama elevada de diferentes tipos de erros que podem surgir. A biblioteca `questions_transformer`, não só permite realizar alterações de *strings*, mas também, fornece mecanismos que permitem lidar com possíveis complicações e pontos de falha. No processo de transformação de ficheiros Python e LaTeX foram utilizadas as seguintes funções da biblioteca `questions_transformer`:

- `add_changeable` – Regista uma *string* para ser alterada. A partir desta função, é possível evitar que realizando uma alteração a uma determinada *string*, *strings* maiores a serem também alteradas que contenham esta mesma *string* não sejam alteradas.
- `change_all_occurences` – Altera todas as ocorrências de uma dada *string*, por outra *string*.
- `change_token_all_occurences` – Altera todas as ocorrências encontradas de *tokens* Python 3. Um exemplo dum *token* Python 3, pode ser, o nome de uma função ou o nome de uma variável. Esta função é muito útil, sendo que permite, por exemplo, substituir uma variável denominada "a" sem alterar o "a" presente no nome da função "range".

Os mecanismos de substituição de *strings*, fornecidos por esta biblioteca, facilitaram bastante o processo de criação de exercícios, aumentado a produtividade e assegurando uma baixa taxa de ocorrência de falhas.

4.1.6 Visual Studio Code

O Visual Studio Code é um editor de código desenvolvido pela Microsoft. O Visual Studio Code é *open source*, tem a vantagem de ser altamente adaptável, suportando um elevado número de diferentes linguagens de programação, como por exemplo, o Python, Java e C++. Visual Studio Code foi o editor de código utilizado ao longo do desenvolvimento do projeto.

Uma mais-valia do Visual Studio Code, é o facto de ser código aberto, oferecendo uma grande quantidade de extensões, que foram úteis para o projeto. Um exemplo duma destas extensões, é a extensão LaTeX Workshop, que fornece um conjunto de ferramentas que auxiliam o processo de escrita de documentos LaTeX, como *autocomplete* e colorização. Uma outra vantagem da utilização do Visual Studio Code, é a inclusão de controlo de versões Git incorporado, que permitiu acelerar e simplificar o processo de trabalho em equipa.

4.2 Implementação do sistema base de geração de versões

O sistema para desenvolver múltiplas versões dum exercício, é constituído por duas partes fundamentais, uma parte genérica que engloba tudo o que é semelhante entre diferentes exercícios e uma parte específica a cada exercício, nomeadamente as variáveis do programa, valor das variáveis, nomes de classes, entre outros. Isto aplica-se caso o exercício seja de programação em Python.

Nesta etapa iremos explicar as partes comuns de cada exercício e como foram implementadas no sistema de geração de versões.

4.2.1 Terminologia utilizada nos exercícios

A terminologia utilizada para o conjunto de todos os ficheiros nomeadamente os de código Python, o ficheiro LaTeX do enunciado e as perguntas de verdadeiro e falso também em LaTeX é designado por exercício. Este exercício é a base utilizada para gerar todos as versões dos exercícios. Uma versão é nomeadamente um exercício similar ao exercício base, mas com alterações nomeadamente no código Python, caso aplicável, e nas perguntas de verdadeiro e falso.

Apesar das perguntas serem todos afirmações e o aluno na plataforma responder se a afirmação é verdadeira ou falsa, estas afirmações serão referidas como perguntas durante todo o decorrer do relatório, isto para facilitar a compreensão do leitor e por ser a parte onde efetivamente o aluno terá de responder.

Além do enunciado, é também necessário criar um ficheiro LaTeX distinto, um para a versão verdadeira e outro para a versão falsa, isto para cada pergunta distinta. Quer isto dizer que se um exercício possuir 5 perguntas terão de ser criados no total 10 ficheiros com perguntas, onde cinco delas serão verdadeiras e cinco delas falsas. Isto deve-se ao facto de que a escolha entre a versão verdadeira ou falsa das questões é feita pela plataforma de trabalhos de casa.

4.2.2 Criação dos ficheiros base dum exercício

Inicialmente começa-se por criar um ficheiro Python com o código base do enunciado dum exercício. Caso no exercício seja necessário o aluno programar alguma parte adicional, que não conste no enunciado base, para obter os resultados é necessário criar um ficheiro Python com o programa com a resolução do exercício. Na seguinte figura encontra-se um exemplo dum enunciado dum exercício em Python, designado por `program.py`:

```
class A:

    def __init__(self):

        self.a = None

class B:

    def __init__(self):

        self.b = None

x=10
x=[]
print(type(x))
x = A()
x = B()
print(type(x))
```

Figura 4.7: Exemplo dum ficheiro Python dum enunciado de um exercício

De seguida é necessário um ficheiro LaTeX com o enunciado do exercício, onde internamente contém também o código do ficheiro Python referido anteriormente. O código inserido no ficheiro LaTeX do enunciado tem de ser exatamente igual ao código do ficheiro Python do exercício base, visto que a geração de ficheiros LaTeX para diferentes versões é feita através de alteração de *strings*.

Os ficheiros das perguntas dos exercícios são ficheiros do género:

```
\documentclass[12pt,varwidth=16cm,border=1pt]{standalone}

\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[portuguese]{babel}

\begin{document}

Na linha 32 e 33 o print imprime na consola o seguinte output:
\newline
x
\newline
12462

\end{document}
```

Figura 4.8: Ficheiro LaTeX deu origem ao ficheiro anterior

O ficheiro LaTeX anterior gera o ficheiro com a afirmação que se segue:

Na linha 32 e 33 o print imprime na consola o seguinte output:
x
12462

Figura 4.9: Exemplo dum ficheiro PNG duma pergunta

4.2.3 Classe GenerateRandomVersion

A classe responsável por realizar as diferentes versões da parte comum entre exercícios é designada por GenerateRandomVersion. Esta classe possui um construtor que recebe os seguintes argumentos:

- Uma lista com o nome dos ficheiros Python utilizados no exercício.
- Uma lista com todos os nomes do enunciado e das perguntas verdadeiro e falso dos ficheiro LaTeX mas sem a extensão do ficheiro "`.tex`". Desta forma é possível criar os ficheiros LaTeX das versões, os ficheiros PDF e PNG obtidos a partir das versões todos como o mesmo nome.
- A diretoria onde se encontram os ficheiros LaTeX e Python do exercício base.
- O número de versões que é pretendido gerar.
- O índice da lista com os nomes dos ficheiros Python onde o *output* se encontra, caso no exercício não seja necessário o aluno programar será o valor 0 caso contrário será 1.
- A diretoria para onde serão geradas as diversas versões.
- Um argumento opcional caso seja pretendido repetir a construção de uma versão de um exercício com uma determinada *seed*(número inteiro), passa-se o número da *seed* e todas as versões serão geradas com essa *seed*.

Esta classe é abstrata, para permitir que a classe que estender desta só necessite de implementar as partes distintas entre cada exercício. Apesar de o Python não possuir a definição explicita de classes abstratas, isto foi feito através da utilização do *raise NotImplementedError* no interior dos métodos que é pretendido serem abstratos.

Esta classe possui métodos para as seguintes funcionalidades:

- Criar a diretoria geral onde irão ficar guardadas todas as versões do exercício e dentro desta uma diretoria específica para cada versão do exercício.

- Obter uma lista com os ficheiros LaTeX após estes serem carregados pela biblioteca `questions_transformer`, através duma função designada por `load_text`.
- Alterar e aplicar as transformações feitas pela biblioteca `questions_transformer` no enunciado e gerar a nova versão do enunciado.
- Obter o resultado da alteração do exercício em formato de *string*.
- Gerar os ficheiros LaTeX das perguntas das diferentes versões.
- Obter o resultado da execução do programa e escrever num ficheiro de texto.
- Realizar todo o sistema de alteração e geração dos ficheiros, esta comportamento é feito pelo método designado por `make_versions`.

A classe possui o seguinte diagrama de classes:

Figura 4.10: Diagrama de classes do GenerateRandomVersion

Perante o seguinte UML, é possível denotar que a classe possui apenas métodos úteis para a classe que estender dela, possui também métodos para a classe que será responsável por gerar os ficheiros PDF e métodos que são utilizados no método `make_versions`. Esta classe utiliza alguns métodos estáticos da classe Helper, nomeadamente para ler e escrever ficheiros. Para

criar métodos estáticos em Python é utilizado o *decorator* @classmethod. Quando o *decorator* é utilizado antes da definição do método já não é necessário criar uma instância da classe para utilizar o método, visto que este passa a ser estático.

4.3 Implementação da parte específica de cada exercício

Nesta etapa iremos explicar em pormenor, como é construída a parte específica de cada exercício e como esta se enquadra na parte da implementação base de todos os exercícios.

4.3.1 Classe QuestionTransformer

A classe responsável por implementar a parte específica de cada exercício é designada QuestionTransformer. Nesta classe é que vão ser definidas as transformações que serão efetuadas para criar as diversas versões dum exercício, sendo necessário então implementar uma versão única desta classe para cada exercício.

Apesar do nome da biblioteca fornecida pelo orientador, ser chamada `questions_transformer` e este nome ser semelhante ao nome dado pelo grupo ás classes, as classes QuestionTransformer foram inteiramente criadas pelo grupo.

A classe QuestionTransformer estende da classe GenerateRandomVersion, desta forma, a classe QuestionTransformer só necessita de implementar as partes diferentes entre os exercícios.

A classe QuestionTransformer chama o construtor da super classe e passa os argumentos da classe iguais ao construtor da classe GenerateRandomVersion. Esta classe por sua vez utiliza a biblioteca `questions_transformer` para realizar alteração de ficheiros LaTeX e ficheiros de código Python.

A classe GenerateRandomVersion possui dois métodos abstratos designados por: `create_python_program` e `create_latex_files`, ambos têm de ser definidos pela classe QuestionTransformer.

Método `create_python_program`

O método `create_python_program` é responsável por alterar o ficheiro Python com o código do enunciado e por alterar também a resolução do exercício. Para realizar esta alteração, é necessário definir quais são os valores (*strings*) que é pretendido alterar e quais são os valores que irão tomar o seu lugar. Para definir quais são as *strings* do código Python que serão alteradas, primeiro define-se uma lista com essas *strings* e de seguida chama-se o método `add_changable_strings`, definido na classe GenerateRandomVersion, que utiliza a lista passada como argumento, para registar as *strings* que se pretende alterar.

O método `add_changable_strings` utiliza internamente um ciclo para adicionar todos os índices da lista. Este método utiliza a biblioteca denominada `questions_transformer` para definir quais as *strings* que é pretendido alterar.

Dentro do método `create_python_program` os valores que irão tomar o lugar dos valores definidos no exercício base, irão ser gerados aleatoriamente, de forma a cada versão ser diferente das restantes. Os valores aleatórios podem ser valores do tipo: números inteiros, números decimais, *strings* ou outros, no final sendo sempre convertidos para *string*. A alteração das *strings* é realizada a partir dos valores aleatórios gerados, recorrendo à utilização do módulo *random* do Python. Foram utilizados por exemplo, os seguintes métodos do módulo:

- `seed` - método utilizado para inicializar o gerador de números aleatórios
- `choice` - método que retorna um elemento aleatório de uma lista
- `randint` - método que retorna um número inteiro aleatório de um intervalo especifico
- `sample` - método que retorna uma lista de tamanho defendido pelo programador, de índices não repetidos de uma lista, aleatoriamente

Finalmente a partir das *strings* definidas para serem substituídas e dos valores gerados aleatoriamente, utiliza-se os métodos `change_all_occurrences` e `change_token_all_occurrences`, para realizar as alterações.
Nestes métodos passa-se no primeiro argumento a *string* que é pretendido alterar e no segundo argumento o valor gerado aleatoriamente. Por fim é retornada uma lista com os ficheiros Python alterados e o valor da semente do gerador de números aleatórios da versão gerada. Para ser possível repetir a geração de um exercício com uma determinada *seed* é feita uma condição sobre o atributo `repeated_seed`, caso o seu valor seja diferente de *None* a *seed* utilizada é o valor da `repeated_seed`.

Caso o exercício não seja de programação em Python não é necessário realizar nenhuma alteração ao mesmo.

Método `create_latex_files`

O método `create_latex_files` é responsável por alterar os ficheiros LaTeX das perguntas de verdadeiro e falso.

Inicialmente para conseguir alterar os ficheiros LaTeX das perguntas, é necessário utilizar o método `get_text_files` da classe GenerateRandomVersion. Este método lê o conteúdo dos ficheiros LaTeX das perguntas, através de uma função chamada `load_text` da biblioteca `questions_transformer`, e de seguida adiciona a uma lista os textos que são pretendidos alterar.

Seguidamente é necessário definir as *strings* que se pretende alterar, utilizando o método `add_changable_strings`. De forma semelhante ao método anterior, define-se quais são os novos valores para substituição, e é chamada a função `change_all_occurrences` para realizar a alteração dos valores. Por fim, são retornados os textos alterados.

Caso seja necessário utilizar atributos extra durante a execução do programa, basta defini-los no construtor da classe QuestionTransformer. Os atributos são normalmente utilizados para passar o valor de variáveis do método `create_python_program` para o método `create_latex_files`.

O esquema de execução do método `make_versions` é o que se segue:

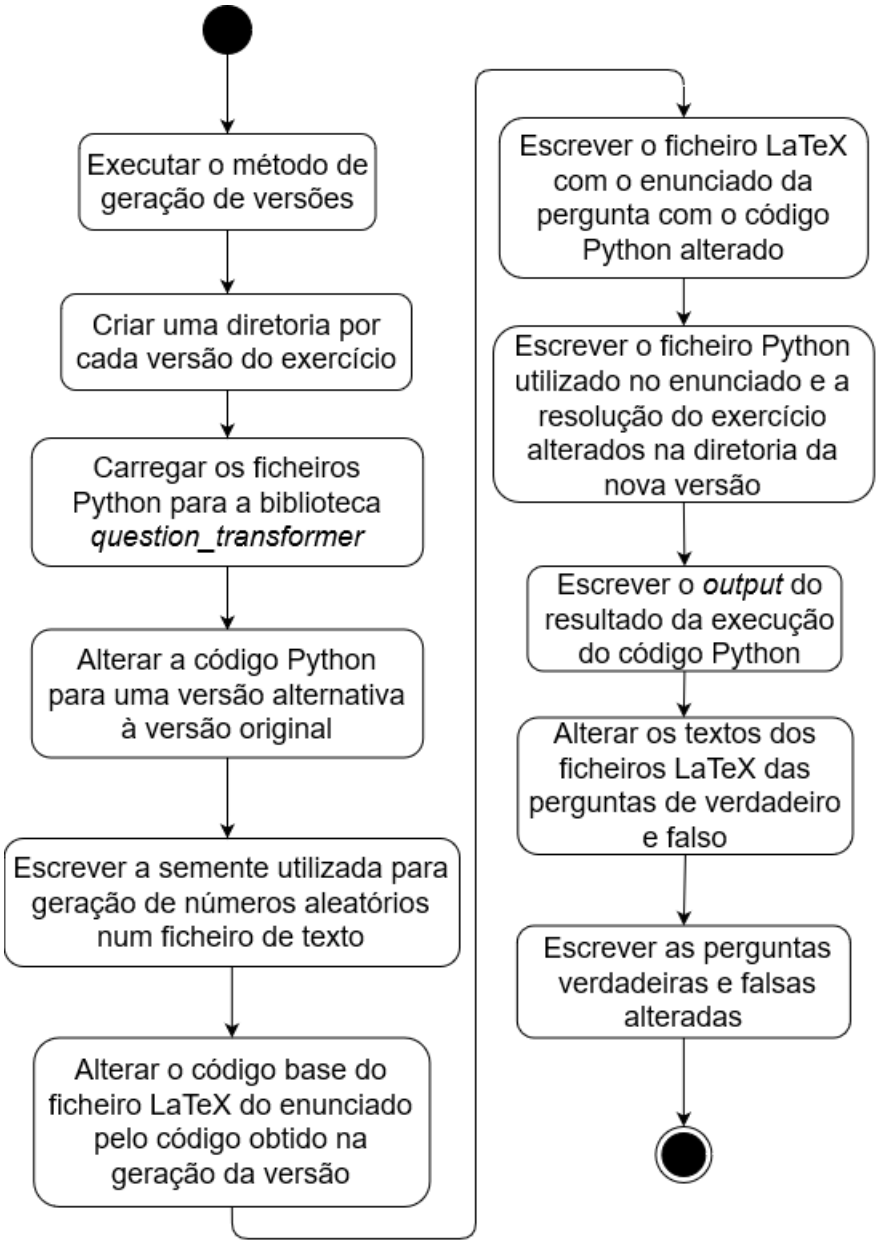


Figura 4.11: Diagrama de atividades do método `make_versions`

4.4 Ficheiros change_files

Os ficheiros Python com o nome começado por `change_files` são os ficheiros de controlo utilizados para executar o programa de geração de ficheiros. Isto é, estes são os ficheiros que devem ser executados para realizar a geração de versões. Por cada exercício criado existe um ficheiro `change_files` associado, desta forma, quando é necessário executar um determinado exercício basta aceder ao ficheiro `change_files` associado e correr o programa Python. Estes programas são simples, compostos somente pelos argumentos necessários na classe QuestionTransformer associado ao exercício. Os argumentos são:

- A localização em formato de *string* dos ficheiros LaTeX do enunciado e das perguntas designado por `directory_latex`.
- Uma lista com quais os ficheiros Python que é pretendido alterar, designado por `python_files_list`.
- Uma lista com o nome dos ficheiros LaTeX que é pretendido alterar, onde a lista deve ter sempre no primeiro índice a pergunta e de seguida as perguntas, designado por `general_files_to_change_list`.
- O número de versões que é pretendido gerar, designado por `num_versions`.
- O nome da diretoria para onde serão geradas as versões, designado por `version_path`.

Após definidos todos estes atributos, estes são passados á classe QuestionTransformer associada ao exercício e é chamado o método `make_versions`, desta forma são geradas as diferentes versões do exercício na diretoria definida.

O diagrama de classes da classe QuestionTransformer e dos ficheiros `change_files` é o que se segue:

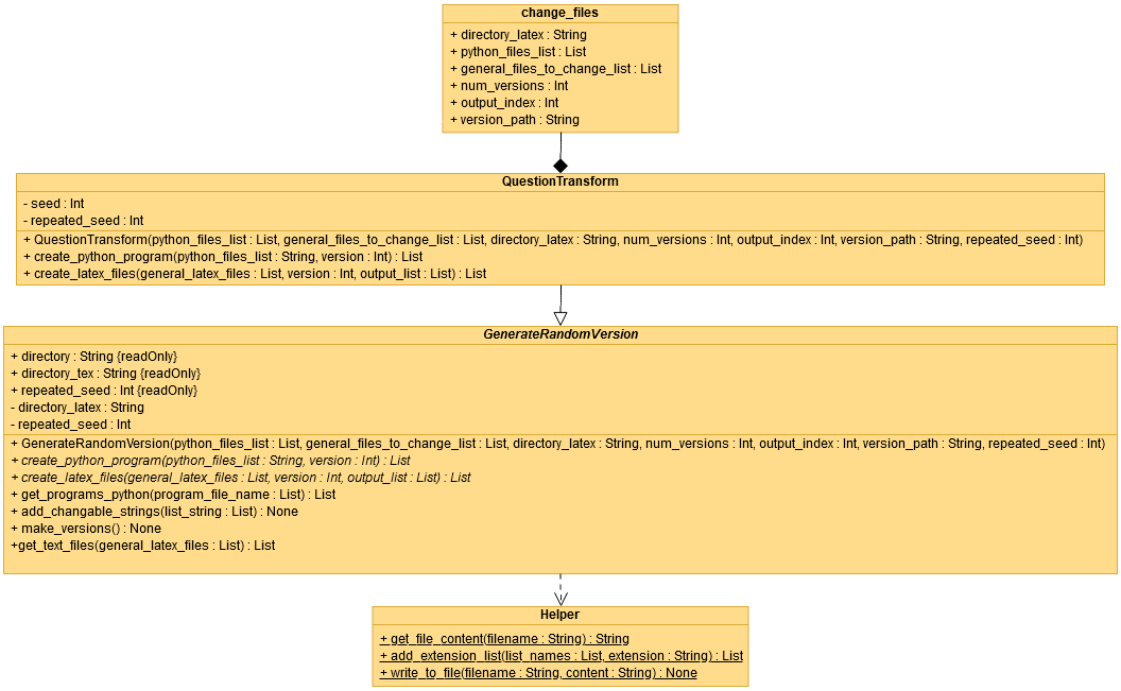


Figura 4.12: Diagrama de classes da classe QuestionTransform e ficheiro `change_files`

4.5 Geração dos ficheiros PDF e PNG

Para ser possível utilizar o sistema de trabalhos de casa é necessário ter o resultado das diferentes versões em ficheiros no formato PDF e PNG, para ser possível disponibilizar posteriormente para os alunos. Nesta seção será mostrado todo o processo necessário para a geração dos ficheiros PDF e PNG a partir das diferentes versões dos ficheiros LaTeX.

Definição de variáveis de ambiente

Como mencionado anteriormente, de forma a gerar os ficheiros PDF e ficheiros PNG, para os diferentes exercícios, foi necessário utilizar as bibliotecas `pdflatex` e Ghostscript.

Ambas as bibliotecas foram utilizadas a partir de comandos executados na linha de comandos. Isto foi possível devido ao módulo `os` do Python, que oferece funcionalidades para interação com o sistema operativo. Para ser possível utilizar comandos do `pdflatex` e do Ghostscript, é necessário primeiramente adicionar ao PATH o executável dos programas. Este procedimento permite compatibilidade entre todos os sistemas operativos principais.

4.5.1 Classe FilesGenerator

A classe responsável por converter os ficheiros LaTeX, das diferentes versões dos exercícios, para ficheiros PDF e PNG é a classe FilesGenerator. Esta classe possui um construtor que recebe os seguintes argumentos:

- Uma lista com os nomes dos ficheiros LaTeX do exercício.
- A diretoria utilizada na classe QuestionTransformer, para assegurar. coerência entre as diretorias das versões geradas.
- O número de versões geradas.
- Um argumento opcional designado `maintain_files` para decidir se mantém ou não os ficheiros temporários, nomeadamente ficheiros `.log` e `.aux`, por omissão estes ficheiros são removidos.

Esta classe dispõem dum método para converter os ficheiros LaTeX para PDF, `convert_latex_to_pdf`, e outro método para converter ficheiros PDF para PNG, `convert_pdf_to_png`.

O método `convert_latex_to_pdf` tem o seguinte funcionamento:

1. Percorrer todos os ficheiros LaTeX guardados numa diretoria duma versão.
2. Utilizando o seguinte comando: `pdflatex "nome_do_ficheiro.tex"`, do pdflatex, gerar ficheiros PDF a partir dos ficheiros LaTeX.
3. Adicionar os ficheiros PDF criados à diretoria da respetiva versão, este passo é realizado automaticamente pelo comando referido anteriormente.

O método `convert_pdf_to_png` funciona da seguinte maneira:

1. Percorrer todos os ficheiros PDF guardados numa diretoria duma versão.
2. Utilizando o seguinte comando: `'gs -r150 -dTextAlphaBits=4 -dGraphicsAlphaBits=4 -sDEVICE=png16m -o '+nome_resultado.png+' '+ nome_ficheiro.pdf`, do Ghostscript, para gerar ficheiros PNG a partir dos ficheiros PDF.
3. Adicionar os ficheiros PNG criados à diretoria da respetiva versão, este passo é realizado automaticamente pelo comando referido anteriormente.

Finalmente, esta classe tem também um método chamado `execute`. O método executa todo o processo de geração de ficheiros PDF e PNG.

O método tem o seguinte funcionamento:

1. Iterar sobre todas as versões dum exercício.

2. Chamar o método `convert_latex_to_pdf`, convertendo os ficheiros LaTeX de cada versão em PDF.

3. Chamar o método `convert_pdf_to_png` para gerar os ficheiros PNG a partir dos ficheiros PDF criados no ponto anterior.

O diagrama de classes da classe FilesGenerator é:

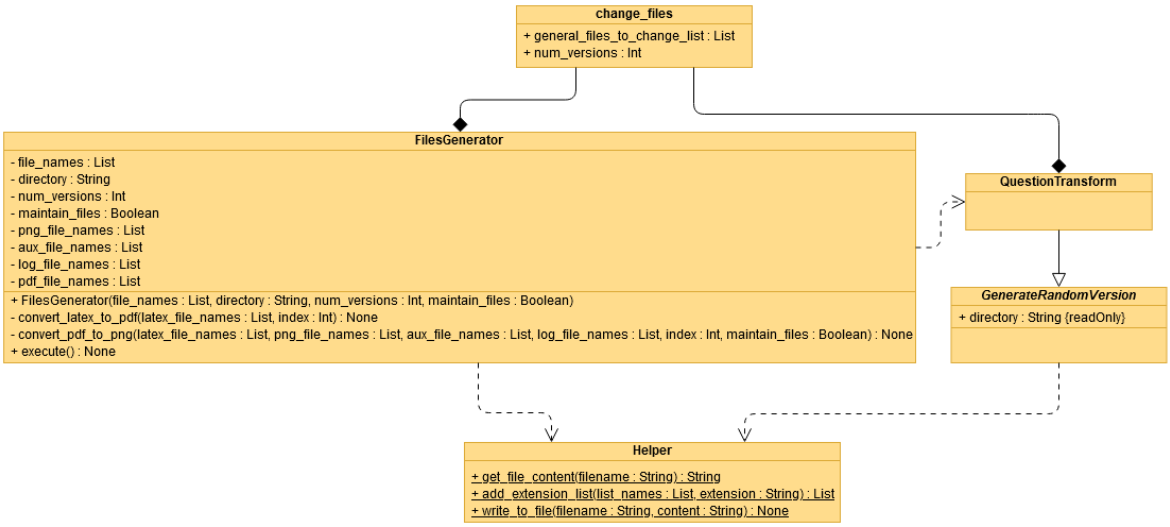


Figura 4.13: Diagrama de classes da classe FilesGenerator

4.6 Geração da união de versões

Visto que é pretendido criar diversos exercícios, por vezes torna-se complicado de conseguir verificar se a geração dos exercícios foi feita corretamente ou não. Por isso foi necessário criar um programa que exibe automaticamente todos as perguntas verdadeiro e falso e o enunciado da versão de um exercício, juntos num único ficheiro, neste caso PDF, designado por `result.pdf`.

Nesta etapa será então explicado todo o funcionamento do sistema da geração do ficheiro PDF com o enunciado e as diferentes perguntas de verdadeiro ou falso.

Classe MergeFiles

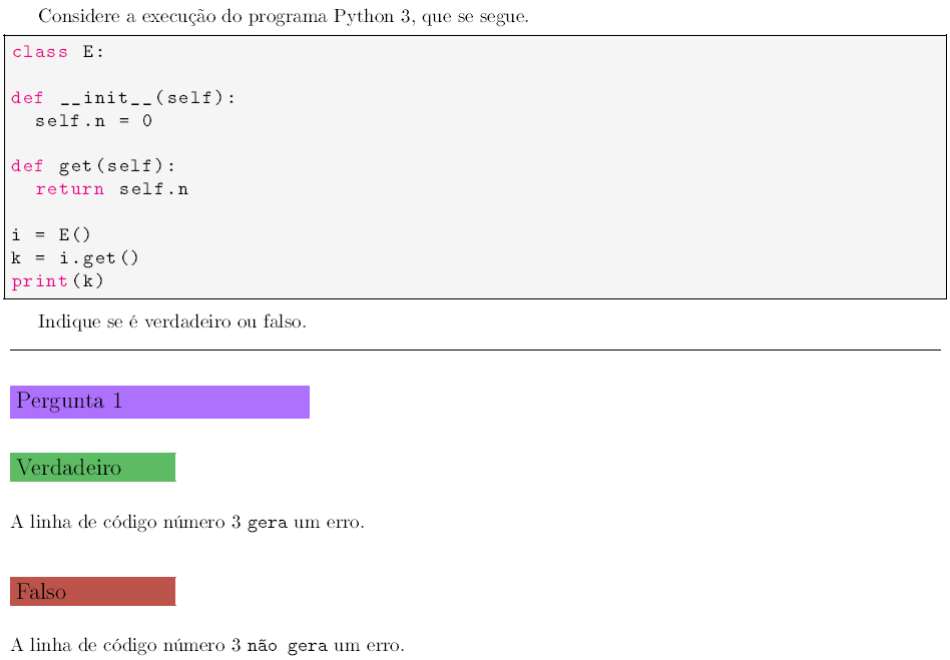
A classe MergeFiles é responsável por juntar os ficheiros todos num só. Esta classe possui um construtor com os seguintes argumentos:

- Uma lista com o nome dos ficheiros do enunciado e das perguntas de verdadeiro e falso.
- Qual a diretoria onde foram gerados os ficheiros PDF e PNG com o enunciado e as perguntas.
- O número de versões que foram geradas.
- Um argumento booleano, designado por `remove_tex` por omissão, que remove o ficheiro LaTeX que deu origem ao ficheiro PDF final com os ficheiros todos juntos, para manter o ficheiro basta definir o argumento como False.
- Um argumento booleano, designado por `add_new_line`, que adiciona um `\newline` no fim de cada pergunta para facilitar a visualização das perguntas. Por vezes as perguntas contêm troços de código isto gera um erro no LaTeX e por isso tem de se definir o argumento como False.

Esta classe possui dois métodos para fazer a união entre todos os ficheiros sendo eles: `get_indexes_of_small_string` e `execute`.

O método `get_indexes_of_small_string` é o método responsável por obter os índices de início e fim duma *string* pequena dentro duma *string* maior. Normalmente este método é utilizado para obter os comandos de `\begin` e de `\end` de um documento LaTeX.

O método `execute` é o método responsável por fazer a geração do ficheiro LaTeX com o enunciado e todas as perguntas de verdadeiro e falso juntas num único ficheiro PDF. O método une os diversos ficheiros LaTeX de texto a partir de comandos como o `\begin{document}` ou o `\end{document}`. A partir destes comandos une todo o conteúdo dos ficheiros numa única *string*. Adicionalmente este método também põe marcadores com o número da pergunta e se a pergunta é a versão verdadeira ou falsa da pergunta. Como se pode observar na figura que se segue:



Considere a execução do programa Python 3, que se segue.

```
1  class E:
2
3  def __init__(self):
4    self.n = 0
5
6  def get(self):
7    return self.n
8
9  i = E()
10 k = i.get()
11 print(k)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

A linha de código número 3 `gera` um erro.

Falso

A linha de código número 3 `não gera` um erro.

Figura 4.14: Exemplo de um ficheiro gerado pela classe MergeFiles

A execução do método `execute` é a seguinte:

1. Troca o cabeçalho do ficheiro LaTeX, substituindo o comando LaTeX `documentclass`, para este ocupar o espaço de uma folha A4.
2. É construída uma *string* que contem o conteúdo do enunciado antes de ser terminado o ficheiro LaTeX.
3. Por cada pergunta é adicionado o seu conteúdo à *string* já construída.
4. O conteúdo do ficheiro é fechado.
5. É criado um ficheiro LaTeX com o conteúdo final.
6. É gerado o ficheiro PDF a partir do ficheiro LaTeX obtido.

O diagrama de classes da classe MergeFiles é:

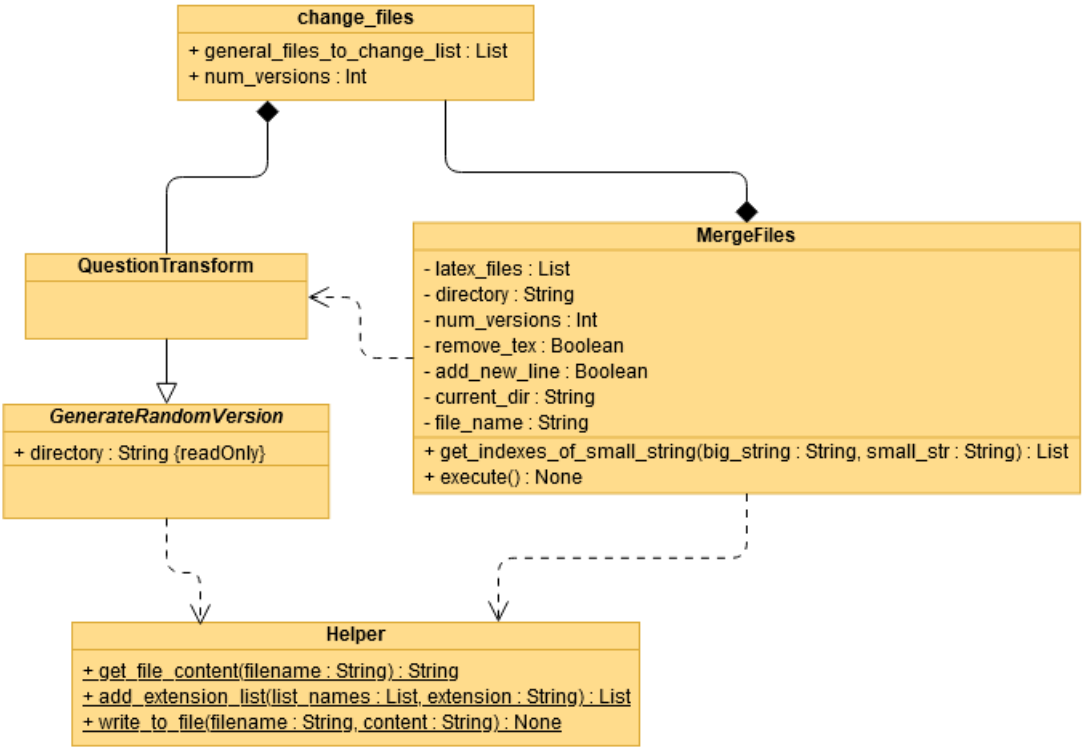


Figura 4.15: Diagrama de classes da classe MergeFiles

4.7 Etapas completas da criação e execução de um exercício

Nesta etapa será explicado, o funcionamento de todo o sistema de geração de diferentes versões de um exercício, desde ficheiros necessários a todo o processo de execução.

1. Criar e escrever o ficheiro Python com o código base do exercício e também, caso seja necessário, completar o código base com um ficheiro Python com a solução do exercício.

2. Criar e escrever os ficheiros LaTeX com o enunciado e as perguntas tanto a versão verdadeira como a versão falsa.

3. Criar a classe QuestionTransformer e programar as partes diferentes de cada exercício.

4. Criar o ficheiro `change_files` com os parâmetros do correspondente exercício.

5. Executar o ficheiro `change_files`.

6. O programa inicialmente vai ler o conteúdo do ficheiro Python com a pergunta base e a solução.

7. A partir do que foi programado na classe QuestionTransformer, é alterado o conteúdo dos ficheiros Python e criado uma *string* com o resultado.

8. É alterado o conteúdo do ficheiro Python no enunciado da pergunta.

9. É escrito os ficheiros Python resultantes das alterações na nova diretoria.

10. São alteradas as perguntas a partir do que foi programado na classe QuestionTransformer.

11. É escrito na nova diretoria o enunciado com o programa alterado e as diferentes perguntas de verdadeiro e falso alteradas pelo programa.

12. São removidos os ficheiros desnecessários, nomeadamente ficheiros `aux` e ficheiros `log`.

13. São criados os ficheiros PDF a partir dos ficheiros LaTeX alterados.

14. São gerados os ficheiros PNG a partir dos ficheiros PDF.

15. É gerado um ficheiro LaTeX com todo o conteúdo dos ficheiros tanto do enunciado como das perguntas.

16. É gerado um ficheiro PDF a partir do ficheiro criado anteriormente.

17. É removido o ficheiro LaTeX que originou o ficheiro PDF com o conteúdo de todos os ficheiros LaTeX, caso o argumento `remove_tex` estiver com o valor *true*.

4.8 Organização dos *packages*

Os diferentes ficheiros e classes foram organizados por *packages* de forma a reduzir a complexidade, juntado funcionalidades parecidas no mesmo *package*. Esta metodologia facilitou a criação de exercícios, pelo facto, de juntar todos os ficheiros correspondentes a um exercício dentro de um *package* único.

A organização dos *packages* no sistema desenvolvido é a seguinte:

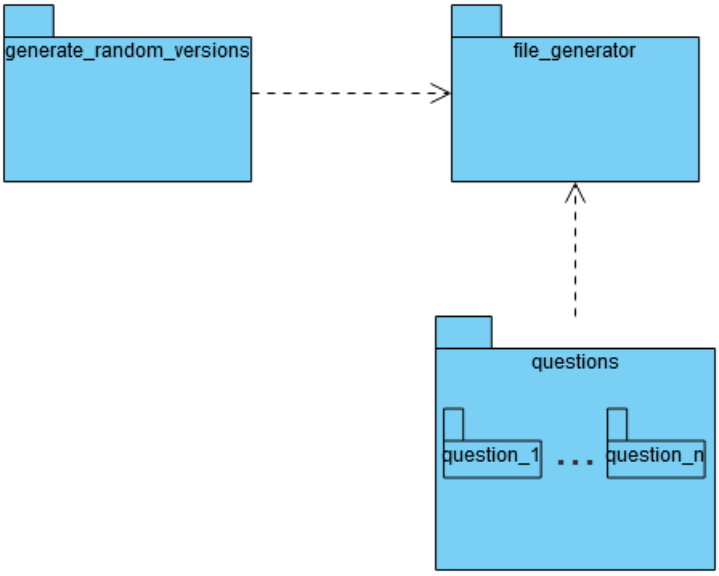


Figura 4.16: Diagrama de *packages* do sistema

O *package* `file_generator` é responsável por armazenar classes, que são responsáveis por manipular ficheiros, nomeadamente:

- FilesGenerator
- Helper
- MergeFiles

O *package* `generate_random_versions` armazena somente a classe GenerateRandomVersion, visto que esta classe utiliza funcionalidades da classe Helper, este *package* interage com o *package* `file_generator`.

O *package* `questions` é responsável por armazenar as partes individuais de cada exercício, estando cada exercício guardado numa pasta diferente. Dentro de cada pasta de um exercício, estão armazenados os seguintes ficheiros e classes:

- Programa Python com a resolução total do exercicio, caso exista solução.
- Programa Python do enunciado do exercicio.
- Classe QuestionTransformer.
- Ficheiros LaTeX das perguntas verdadeiro e falso, e do enunciado do exercicio.

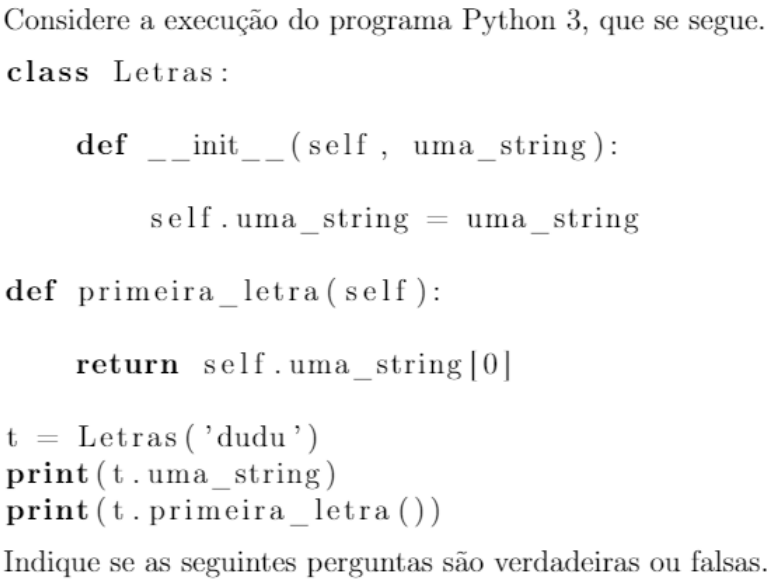
Os diferentes ficheiros de *output* da geração dum exercício, são guardados num *package*. O *package* é composto por um conjunto de *sub-packages*, um para cada versão, contendo cada *sub-package* todos os ficheiros de *output* para uma versão única do exercício.

4.9 Estrutura dos ficheiros LaTeX

Um detalhe que foi tido em atenção no processo de criação dos ficheiros LaTeX, foi evitar que o conteúdo LaTeX fosse apresentado num formato definido de página, como por exemplo, uma página A4. De modo a conseguir implementar este comportamento, utilizou-se o módulo LaTeX *standalone*.

De modo a tornar o código apresentado nos ficheiros LaTeX, mais perceptível, foi adicionado *Sintax Highlighting*. Para este propósito, foi inicialmente definido um conjunto de cores apelativas. De seguida, foi utilizado um esquema que permite detetar certas *keywords* e propriedades do Python, apresentando-as com cores sugestivas.

Nas seguintes figuras, está apresentado um troço de código dum exercício, com e sem *Sintax Highlighting*.


Considere a execução do programa Python 3, que se segue.

```
class Letras:

    def __init__(self, uma_string):

        self.uma_string = uma_string

def primeira_letra(self):

    return self.uma_string[0]

t = Letras('dudu')
print(t.uma_string)
print(t.primeira_letra())
```

Indique se as seguintes perguntas são verdadeiras ou falsas.

Figura 4.17: Código Python em exercício, sem *Sintax Highlighting*

Considere a execução do programa Python 3, que se segue.

```
class Letras:

    def __init__(self, uma_string):

        self.uma_string = uma_string

def primeira_letra(self):

    return self.uma_string[0]

t = Letras('dudu')
print(t.uma_string)
print(t.primeira_letra())
```

Indique se as seguintes perguntas são verdadeiras ou falsas.

Figura 4.18: Código Python em exercício, com *Sintax Highlighting*

Capítulo 5

Validação e Testes

Neste capítulo será explicado em pormenor como foi feita a validação da geração dos exercícios e os testes necessários para confirmar o bom funcionamento dos mesmos. Inicialmente será explicado como é feita a validação e de seguida quais os testes que foram realizados.

5.1 Validação

Um ponto importante a notar é que o processo de criação de diferentes versões de um exercício é um processo delicado. Existem erros que surgem apenas uma vez em 100 diferentes versões, portanto a correta utilização das diferentes funções da biblioteca, em conjunto com um pensamento crítico e cauteloso é crucial para evitar possíveis falhas.

Para realizar a validação do programa basta executar o mesmo. Desta forma se a plataforma conseguir gerar todos os ficheiros correspondentes a um exercício sem gerar um erro ou exceção é porque o exercício pode ser testado. Caso os argumentos passados em algum dos ficheiros utilizados estiverem incorretos o programa irá gerar um erro e não será possível passar á fase de testes até este problema estar resolvido.

5.2 Testes

Para testar o programa inicialmente é necessário executar o programa designado por `change_files` do exercício específico, após isso os ficheiros gerados vão para uma determinada diretoria com o nome do determinado exercício, onde internamente existem sub-diretorias com os ficheiros gerados para as diversas versões. Por observação e utilizando uma classe para verificar certos pormenores, nomeadamente se existem programas iguais entre as diferentes versões, verifica-se se assim que não existem versões repetidas entre os exercícios.

Para testar o programa são realizados os seguintes testes:

- Comparar os diversos ficheiros Python, utilizando a classe `Validator`, desta forma é possível verificar se existem versões iguais entre alunos e também se os exercícios criados com uma determinada *seed* estão a ser gerados corretamente.
- Visualizar a correta geração dos ficheiros criados, manualmente, através da visualização do ficheiro `result.pdf` (ficheiro com o enunciado e as perguntas).
- Comparar duas versões diferentes do mesmo exercício.
- Comparar o ficheiro de texto do *output* com o a execução do ficheiro Python da resolução do exercício.
- Comparar os ficheiros gerados com *seed* igual.

5.2.1 Comparação de ficheiros Python

Para verificar a correta geração de versões por vezes é necessário validar se as diversas diversões são de facto diferentes entre elas ou não. Para isso foi criada uma classe designada `Validator`.

Esta classe recebe como argumentos:

- Os ficheiros Python utilizados.

- A diretoria das versões.
- O número de versões.

Esta classe tem um método designado por `execute`, este método imprime na consola se existem ficheiros Python iguais entre as diferentes versões ou não. Desta maneira é possível comparar os diversos ficheiros Python e também permite saber a geração de versões com uma *seed* específica está ou não a ser gerada corretamente. Como se pode observar o programa produz os seguintes *outputs*:

```
Não existe programas iguais
Não existe programas completos iguais
```

Figura 5.1: *Ouput* produzido pela classe `Validator` quando não há ficheiros Python iguais

5.2.2 Verificação da geração dos resultados

Para verificar a correta geração dos ficheiros PDF utiliza-se o ficheiro `result.pdf`, onde contém o enunciado e as perguntas todas no mesmo ficheiro o que permite mais facilmente perceber se a geração está a ser feita corretamente.

```
1  class G:
2
3      def __init__(self):
4
5          self.g = None
6
7  class Z:
8
9      def __init__(self):
10
11         self.z = None
12
13 i=39
14 i=[]
15 print(type(i))
16 i = G()
17 i = Z()
18 print(type(i))
```

Figura 5.2: Exemplo de verificação da geração do enunciado de uma versão de um exercício

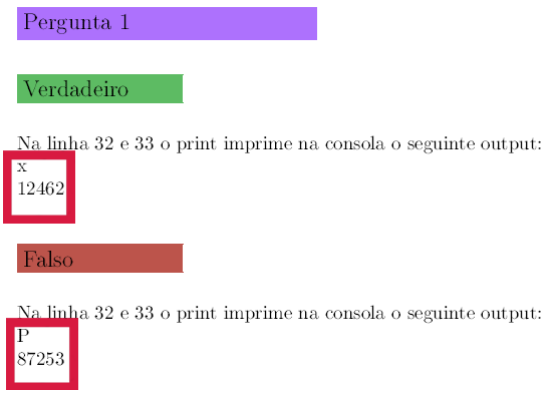


Figura 5.3: Exemplo de verificação da geração das perguntas de uma versão de um exercício

5.2.3 Comparação de versões diferentes

Um dos cuidados a ter, é verificar que todas as versões do mesmo exercício estão a ser alteradas como pretendido, não existindo versões iguais entre exercícios. Outro ponto a ter em atenção é verificar que o nível de dificuldade é semelhante entre todas as versões.

```
class G:

    def __init__(self):

        self.g = None

class Z:

    def __init__(self):

        self.z = None

i=33
i=[]
print(type(i))
i = G()
i = Z()
print(type(i))
```

Indique se é verdadeiro ou falso.

Figura 5.4: Primeiro exemplo da geração de exercícios

```
class C:

    def __init__(self):

        self.c = None

class H:

    def __init__(self):

        self.h = None

k=49
k=[]
print(type(k))
k = C()
k = H()
print(type(k))
```

Indique se é verdadeiro ou falso.

Figura 5.5: Segundo exemplo da geração de exercícios

5.2.4 Comparação do ficheiro de texto do *output* com a execução do ficheiro

Um passo importante no processo de realização de testes é a comparação do ficheiro de texto com a execução do programa Python em relação ao *output* do programa Python do exercício. Desta maneira é possível verificar que o ficheiro de texto do *output* foi criado com sucesso.

A vantagem de ter o *output* do programa Python do exercício num ficheiro de texto é que, desta maneira é mais fácil visualizar os resultados duma versão especifica e oferece um reforço do correto funcionamento da versão dum exercício.

```
x
12462
[12462, 92019, 87253, 77285, 32341, 29773]
xmmPIz
<class '__main__.UmaClasse'>
<class 'str'>
```

Figura 5.6: Exemplo do *output* do programa `full_program.py`

```
x
12462
[12462, 92019, 87253, 77285, 32341, 29773]
xmmPIz
<class 'UmaClasse'>
<class 'str'>
```

Figura 5.7: Exemplo do conteúdo do ficheiro de texto com o *output* do programa

5.2.5 Comparação da geração de exercícios com uma *seed* especifica

Por vezes, um aluno pode queixar-se que o programa do exercício não está a funcionar corretamente. De modo a confirmar que a geração do exercício foi realizada sem erros, é gerado novamente o exercício com a *seed* utilizada na versão do aluno com queixas.

A *seed* utilizada na criação da versão de um exercício encontra-se num ficheiro de texto designado por `seed.txt`, na diretoria com os ficheiros de *output* da versão específica.

A partir da leitura do ficheiro `seed.txt` é passado o valor da *seed* no construtor da classe QuestionTransformer no argumento opcional `repeated_seed`. Desta forma todos os exercícios serão gerados a partir do valor da *seed*, por isso deverão ser todos iguais. A partir de uma versão gerada a partir da *seed* e uma versão do exercício utilizado na plataforma de trabalhos de casa são comparadas as duas manualmente, desta forma é possível saber se o exercício foi gerado corretamente.

A classe Validator permite saber se as versões estão a ser todas geradas com a mesma *seed*, visto que os ficheiros Python terão que ser todos iguais entre eles.

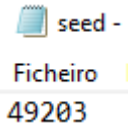


Figura 5.8: Exemplo dum ficheiro `seed.txt`

Capítulo 6

Conclusões e Trabalho Futuro

No projeto foi possível aprofundar o conhecimento nas áreas de geração de automatização de tarefas, programação em Python, manipulação e geração de ficheiros.

A partir da implementação dos trabalhos de casa foi possível também rever certos detalhes esquecidos sobre a programação Python lecionado nas unidades curriculares de Matemática Discreta e Programção(MDP) e Matemática para Computação Gráfica(MCG).

A utilização de programas pela linha de comandos do Python foi uma novidade para os membros do grupo. Esta funcionalidade permite facilmente utilizar programas externos ao Python para realizar certas funcionalidade de um programa.

O FEIM(Fórum de Engenharia Informática e Multimédia) realizado em 2021 permitiu obter uma maior noção do que é o mercado de trabalho e as necessidades efetivas no mesmo. Uma outra vantagem foi a possibilidade de mostrar o projeto a um maior número de pessoas e desta forma obter algum *feedback* sobre todo o projeto.

Visto que o projeto foi feito em grupo, uma ferramenta importante para o controlo de versões foi o Git. Desta forma sempre que eram feitas alterações estas ficavam disponíveis para ambos os membros do grupo. A partir desta funcionalidade foi possível aumentar a produtividade e eficiência.

Em suma, o projeto consiste na criação de versões diferentes de um exercício base inicialmente em ficheiros LaTeX e posteriormente em ficheiro PDF. Os ficheiros PDF serão posteriormente disponibilizados na plataforma de trabalhos de casa nas unidades curriculares de MDP e MCG do curso de LEIM do ISEL. No total foram criados 62 exercícios.

Em relação ao trabalho futuro, o grupo pretendia também realizar uma ferramenta de geração de trabalhos de casa automaticamente. Esta geração seria feito por meio de uma interface gráfica desenvolvida em Jupyter Notebook. Através da interação do utilizador com a interface, seria possível gerar o código da classe QuestionTransformer automaticamente onde implementaria as transformações requeridas pelo utilizador. Contudo esta ideia não foi possível concretizar pela falta de tempo, visto que requeria muito tempo para ser implementada e testada devidamente.

Apêndice A

Exemplos de trabalhos de casa

No apêndice serão apresentados os diversos exercícios de trabalhos de casa realizados ao longo do projeto. Para cada exercício é demonstrado um exemplo de uma versão, com o enunciado e as respetivas perguntas verdadeiro e falso.

Considere o programa Python 3, que se segue. Ignore a variável `seed`, e a função `random_int`. Elas destinam-se apenas à geração de números pseudo-aleatórios.

```
seed = 30785

def random_int(min, max):
    global seed
    seed = (16807*seed) % 2147483647
    return int(min + (max - min)*(seed / 2147483646))

class X:

    def __init__(self, x):

        self.j = x

    def h(self):

        return self.j

p = []
for b in range(1714):
    p.append(X(random_int(100, 327)))
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O valor do atributo j, do objeto no índice 516, da lista p, é 168.

Falso

O valor do atributo j, do objeto no índice 516, da lista p, é 167.

Pergunta 2

Verdadeiro

`X` é uma classe.

Falso

`X` é um objeto.

Pergunta 3

Verdadeiro

`__init__` é o construtor da classe `X`.

Falso

`__init__` é uma classe.

Pergunta 4

Verdadeiro

`self` é um objeto.

Falso

`self` é um atributo.

Pergunta 5

Verdadeiro

`z` é um método.

Falso

`z` é o construtor.

Em Pyhton 3, complete a classe UmaClasse, no ficheiro `uma_class.py`:

- para além de self, o construtor da classe UmaClasse, tem o argumento `um_argumento`;
- os objetos do tipo UmaClasse têm o atributo `um_atributo`;
- o atributo `um_atributo` é inicializado, no construtor, com o valor do argumento `um_argumento`;
- os objetos do tipo UmaClasse têm um método `um_metodo`. O método `um_metodo` retorna o atributo `um_atributo`, com tamanho 6 desde o índice `0` até ao `5`, do objeto self.

Considere a execução do seguinte código Python 3.

```
import string

seed = 59581

def pseudo_random_integer(min_int, max_int):
    global seed
    seed = (16807*seed) % 2147483647
    return int(min_int + (max_int - min_int) * seed / 2147483646)



class UmaClasse():

  def gerar_lista_aleatoria(self):

    strings_aleatorias = []

    for i in range(2301):

      strings_aleatorias.append(self.um_metodo()[pseudo_random_integer(0, 5)
   ])

    return strings_aleatorias


string_completa = string.ascii_lowercase+string.ascii_uppercase
string_aleatoria = ''.join([string_completa[pseudo_random_integer(10,36)] for
    i in range(2301)])
lista_numeros_aleatorio = [pseudo_random_integer(10000,99999) for i in range
   (2301)]

objeto1 = UmaClasse(string_aleatoria)
objeto2 = UmaClasse(lista_numeros_aleatorio)

print(objeto1.gerar_lista_aleatoria()[35])
print(objeto2.gerar_lista_aleatoria()[42])

y = objeto2
t = y.um_metodo()
print(t)

y = objeto1
t= y.um_metodo()
print(t)
print(type(y))
print(type(t))
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

Na linha 32 e 33 o print imprime na consola o seguinte output:
w
86695

Falso

Na linha 32 e 33 o print imprime na consola o seguinte output:
C
54748

Pergunta 2

Verdadeiro

O print na linha 37 imprime na consola:
`[20634, 92579, 54748, 62101, 86695, 58264]`

Falso

O print na linha 37 imprime na consola:
`wnCFEE`

Pergunta 3

Verdadeiro

O print na linha 41 imprime na consola:
`wnCFEE`

Falso

O print na linha 41 imprime na consola:
`[20634, 92579, 54748, 62101, 86695, 58264]`

Pergunta 4

Verdadeiro

Na linha 42 o print do tipo da variável `y` é <class '`uma_classe.UmaClasse`'>.

Falso

Na linha 42 o print do tipo da variável `y` é <class '`string`'>.

Pergunta 5

Verdadeiro

Na linha 43 o print do tipo da variável `t` é <class '`string`'>.

Falso

Na linha 43 o print do tipo da variável `t` é <class '`uma_classe.UmaClasse`'>.

Considere a execução do programa Python 3, que se segue.

```
class M:

    def __init__(self, m):

        print('construtor: INÍCIO')
        print('construtor: criação de um objeto da classe M')
        print('construtor: definição do atributo m')
        print('construtor: inicialização do atributo m com o valor: ' + str(m
    ))
        self.m = m
        print('construtor: FIM')

    def x(self):

        print('método y: INÍCIO')
        print('método y: execução do método y')
        print('método y: no objeto do tipo m com atributo m: ' + str(self.m))
        print('método y: FIM')
        return self.m

m = M(47)
x = M(27)
m.x()
x.x()
k = M(4)
print(type(x))
```

Indique se e verdadeiro ou falso.

Pergunta 1

Verdadeiro

Considere a linha de código 21:

```
x = M(27)
```

A linha de código origina o seguinte output:

```
construtor: INÍCIO
construtor: criação de um objeto da classe M
construtor: definição do atributo m
construtor: inicialização do atributo m com o valor: 27
construtor: FIM
```

Falso

Considere a linha de código 21:

```
x = M(27)
```

A linha de código origina o seguinte output:

```
construtor: INÍCIO
construtor: criação de um objeto da classe M
construtor: definição do atributo x
```

```
construtor: inicialização do atributo x com o valor: 27
construtor: FIM
```

Pergunta 2

Verdadeiro

Considere a linha de código 22:

```
m.h()
```

A linha de código origina o seguinte output:

```
método h: INÍCIO
método h: execução do método h
método h: no objeto do tipo m com atributo m: 47
método h: FIM
```

Falso

Considere a linha de código 22:

```
m.h()
```

A linha de código origina o seguinte output:

```
método h: INÍCIO
método h: execução do método h
método h: no objeto do tipo m com atributo m: 27
método h: FIM
```

Pergunta 3

Verdadeiro

Considere a linha de código 25:

```
print(type(x))
```

A linha de código origina o seguinte output:

```
<class '__main__.M'>
```

Falso

Considere a linha de código 25:

```
print(type(x))
```

A linha de código origina o seguinte output:

```
<class 'int'>
```

Em Pyhton 3, construa a classe `X` com os seguintes requesitos:

- Para além de self, o construtor da classe `X`, tem o argumento `x`;
- Os objetos do tipo `X` têm o atributo `x`;
- O atributo `x` é inicializado, no construtor, com o valor do argumento `x`;
- Os objetos do tipo `X` têm um método `y`. O método `y` possui somente o argumento self e retorna o valor do atributo `x`.
- Os objetos do tipo `X` têm um método `z`. O método `z` recebe como argumento o self e um argumento `y`. O método `z` realiza a afetação do atributo `x` pelo valor do argumento `y`.
- Os objetos do tipo `X` têm um método `obter_contagem_string`. O método além do self, recebe um argumento `lista` do tipo list, e um argumento `letra` do tipo string. O método `obter_contagem_string` retorna o número de ocorrências do argumento letra na lista.
- A função `seed`, é utilizada para inicializar um gerador de número aleatórios.

Considere a execução do seguinte código Python 3.

```
1  import random
2  import string
3
4  random.seed(18094)
5
6  lista = [''.join(random.choice(string.ascii_lowercase) for i in range(2)) for
       i in range(300)]
7  x = X('y')
8  y = X('y')
9  z = X('y')
10 print(x.obter_contagem_string(lista,x.y()))
11 print(x.y())
12 print(y.y())
13 print(z.y())
14 print(type(x.y()))
15 x.z('o')
16 y.z('j')
17 print(x.y())
18 print(y.y())
19 print(z.y())
20 print(type(y))
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output produzido na linha 10 é:
25

Falso

O output produzido na linha 10 é:

26

Pergunta 2

Verdadeiro

O output produzido na linha 11, 12 e 13 é:
y
y
y

Falso

O output produzido na linha 11, 12 e 13 é:
o
o
o

Pergunta 3

Verdadeiro

O output produzido na linha 17, 18 e 19 é:
o
j
y

Falso

O output produzido na linha 17, 18 e 19 é:
y
y
y

Pergunta 4

Verdadeiro

O output produzido na linha 14 é:
`<class 'str'>`

Falso

O output produzido na linha 14 é:
`<class '__main__.X'>`

Pergunta 5

Verdadeiro

O output produzido na linha 20 é:
`<class '__main__.X'>`

Falso

O output produzido na linha 20 é:
`<class 'str'>`

Considere a execução do seguinte código Python 3.

```
1  class T:
2
3      def __init__(self):
4
5          self.t = None
6
7  class Q:
8
9      def __init__(self):
10
11         self.q = None
12
13 m=46
14 m=[]
15 print(type(m))
16 m = T()
17 m = Q()
18 print(type(m))
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output do print na consola na linha 15 equivale a <class '`list`'>.

Falso

O output do print na consola na linha 15 equivale a <class '`int`'>.

Pergunta 2

Verdadeiro

O output do print na consola na linha 18 equivale a: <class '`__main__.Q`'>.

Falso

O output do print na consola na linha 18 equivale a: <class '`__main__.T`'>.

Pergunta 3

Verdadeiro

A classe `T` tem um atributo `t`.

Falso

A classe `T` não tem atributos.

Considere a execução do programa Python 3, que se segue.

```
1  class O:
2
3  def __init__(self):
4    self.l = 0
5
6  def get(self):
7    return self.l
8
9  v = O()
10 h = v.get()
11 print(h)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

A linha de código número 3 `gera` um erro.

Falso

A linha de código número 3 `não gera` um erro.

Pergunta 2

Verdadeiro

A linha de código número 6 `não gera` um erro.

Falso

A linha de código número 6 `gera` um erro.

Em Pyhton 3, adicione ao seguinte código o construtor da classe `Numero` e o método `ordenar_lista_decresente`:

- Para além de self, o construtor da classe `Numero`, tem o argumento `valor`;
- Os objetos do tipo `Numero` têm o atributo `valor`;
- O atributo `valor` é inicializado, no construtor, com o valor do argumento `valor`;
- os objetos do tipo `Numero` têm um método `ordenar_lista_decresente`. O método `ordenar_lista_decresente` recebe um argumento chamado `lista` do tipo `list` que ordena uma lista de forma decrescente e retorna-a.
- a função `seed`, é utilizada para inicializar um gerador de número aleatórios.

Considere a execução do seguinte código Python 3.

```
1  import random
2  random.seed(215)
3
4  class Numero:
5
6      def generate_random_lists(self, lists_length):
7          lista = []
8          for i in range (self.valor):
9              lista_i = (random.sample(range(1, 40), lists_length))
10             lista_ordenada = self.ordenar_lista_decresente(lista_i)
11             lista.append(lista_ordenada)
12
13         return lista
14
15
16 a = Numero(75)
17 print (a.generate_random_lists(4)[56][3])
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O print na consola na linha de código 17 dá o seguinte output: `3`

Falso

O print na consola na linha de código 17 dá o seguinte output: `10`

Pergunta 2

Verdadeiro

Considere que o seguinte método foi adicionado à classe :

```
1  def triplo():
2    3 * valor
```

O programa `gera` um erro.

Falso

Considere que o seguinte método foi adicionado à classe :

```
def triplo():
  3 * valor
```

O programa `não gera` um `erro`.

Considere a execução do programa Python 3, que se segue.

```
1  class Letras:
2
3      def __init__(self, uma_string):
4
5          self.uma_string = uma_string
6
7  def primeira_letra(self):
8
9      return self.uma_string[0]
10
11 h = Letras('dom')
12 print(h.uma_string)
13 print(h.primeira_letra())
```

Indique se as seguintes perguntas são verdadeiras ou falsas.

Pergunta 1

Verdadeiro

O output do print na consola na linha de código 12 é `dom`.

Falso

O output do print na consola na linha de código 12 é `d`.

Pergunta 2

Verdadeiro

A linha de código 13 `dá` erro.

Falso

A linha de código 13 `não dá` erro

Em Pyhton 3, complete a classe `Pessoa`:

- Para além de self, o construtor da classe `Pessoa`, tem o argumento `nome`, `altura` e `peso`;
- Os objetos do tipo Pessoa têm os atributos `nome`, `altura` e `peso`;
- Os atributos `nome`, `altura` e `peso` são inicializados, no construtor, com o valor dos argumentos `passados no constru`
- Os objetos do tipo `Pessoa` têm um método `get_dados_pessoa`. O método `get_dados_pessoa` retorna uma lista contendo os atributos ordenados da seguinte maneira: `nome`, `altura` e `peso`.

Considere a execução do programa Python 3, que se segue.

```
import random

random.seed(98861)

class Pessoa:

  def __str__(self):
    return "Nome: "+str(self.get_dados_pessoa()[0])+", com altura: "+str(self
    .get_dados_pessoa()[1])+" e peso: "+str(self.get_dados_pessoa()[2])


nome = random.choice(["Marta", "Fonseca", "Madorna", "Beatriz", "Pedro", "
    Eduardo", "Duarte", "Leonor", "Rita", "Miguel", "Margarida", "Diogo", "
    Massibas"])
altura = round(random.uniform(1.12, 1.72), 2)
peso = random.randint(32,119)

p = Pessoa(nome, altura, peso)
print(p)
print(p.get_dados_pessoa())
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O programa gera o seguinte output:
Nome: Beatriz, com altura: 1.72 e peso: 56
['Beatriz', 1.72, 56]

Falso

O programa gera o seguinte output:
Nome: Duarte, com altura: 1.23 e peso: 48
['Duarte', 1.23, 48]

Pergunta 2

Verdadeiro

Considere a execução do programa Python 3, que se segue.

```
1 class Animal:
2   def __init__(self, nome):
3     self.nome = nome
4
5 p = animal('Gohan')
```

O programa `gera` um erro na linha 5.

Falso

Considere a execução do programa Python 3, que se segue.

```
1 class Animal:
2   def __init__(self, nome):
3     self.nome = nome
4
5 p = animal('Gohan')
```

O programa `não gera` um erro na linha 5.

A execução do programa Python 3, que se segue, gera um erro. Qual?

```
classe Animal:

    def __init__(self, especie):

        self.especie = especie

x = Animal('pato')
s = Animal('papagaio')
```

Indique se a seguinte pergunta é verdadeira ou falsa.

Pergunta 1

Verdadeiro

O programa gera um erro de `SyntaxError`.

Falso

O programa gera um erro do tipo `IndentationError`.

Considere a execução do programa Python 3, que se segue.

```
1 Class Livro:
2
3   def __init__(self, titulo):
4     self.titulo = titulo
5
6 x = Livro('To Kill a Mockingbird')
7 f = Livro('Crime and Punishment')
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

Na linha de código 1 dá um erro do tipo `SyntaxError`.

Falso

Na linha de código 1 dá um erro do tipo `ZeroDivisionError`.

Considere a execução do programa Python 3, que se segue.

```
1 class Livro:
2
3     def __init__(self, titulo):
4
5         self.titulo = titulo
6
7 r = Livro('Cryptography and Network Security')
8 print(r.titulo)
9 print(r.autor)
```

Indique se as seguintes perguntas são verdadeiras ou falsas.

Pergunta 1

Verdadeiro

O output do print na consola na linha de código 8 equivale a `Cryptography and Network Security`.

Falso

O output do print na consola na linha de código 8 equivale a `" "`.

Pergunta 2

Verdadeiro

Na linha de código 9 `dá` erro.

Falso

Na linha de código 9 `não dá` erro.

Considere a execução do programa Python 3, que se segue.

```
1  class Livro:
2
3    def __init__(self, titulo, numero_paginas):
4
5      self.titulo = titulo
6      self.numero_paginas = numero_paginas
7
8
9  class Cancao:
10
11   def __init__(self, titulo, duracao):
12
13     self.titulo = titulo
14     self.duracao = duracao
15
16
17 x1 = Livro('To the Lighthouse', 341)
18 x2 = Cancao('Clocks', 2.041)
19 x3 = x1
20 print(x3.titulo)
21 print(x3.numero_paginas)
22 x3 = x2
23 print(x3.titulo)
24 print(x3.numero_paginas)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output produzido pela linha 20 e 21 é:
`To the Lighthouse`
`341`

Falso

O output produzido pela linha 20 e 21 é:
`Clocks`
`2.041`

Pergunta 2

Verdadeiro

O linha de código 23 `não dá` erro.

Falso

O linha de código 23 `dá` erro.

Pergunta 3

Verdadeiro

O linha de código 24 `dá` erro.

Falso

O linha de código 24 `não dá` erro.

Em Pyhton 3, escreva a classe `GestorCodigoPostais`, no ficheiro `gestor_codigos_postais.py`:

- Para além de self, o construtor da classe `GestorCodigoPostais`, tem o argumento `codigos_postais`;
- Os objetos do tipo `GestorCodigoPostais` têm o atributo `codigos_postais`, que é uma `lista` de objetos do tipo `CodigoPostal` ;
- O atributo `codigos_postais` é inicializado, no construtor, com o valor do argumento `codigos_postais`;
- Os objetos do tipo `codigos_postais` têm os métodos `validar_codigos_postais` e `obter_codigos_postais_por_l`
- O método `validar_codigos_postais` valida o atributo `codigos_postais`, de forma ao atributo `codigos_postais` conter apenas códigos postais válidos;
- Para um código postal ser válido o primeiro número tem que ter quatro digitos, o segundo número tem que ter três digitos e a localidade tem que estar apenas em letras capitais;
- O método `obter_codigos_postais_por_localidade`, recebe o argumento `localidade`, o método retorna uma lista de objetos do tipo CodigoPostal do atributo `codigos_postais` cujo a localidade é igual ao argumento `localidade` recebido;
- A função `seed`, é utilizada para inicializar um gerador de número aleatórios;

```
import random
import string
from gestor_codigos_postais import GestorCodigoPostais

random.seed(27071)

class CodigoPostal:

    def __init__(self, digitos4, digitos3, localidade):

        self.digitos4             = digitos4
        self.digitos3             = digitos3
        self.localidade           = localidade
        self.separador_digitos    = '-'
        self.separador_localidade = ' '

    def print_codigo_postal(self):

        print(f"{self.digitos4}{self.separador_digitos}{self.digitos3}{self.separador_localidade}{self.localidade}")



list_localidades = ["LOURES","MAFRA","OEIRAS","CASCAIS","lisboa","ESPINHO","MAIA","Amarante","valongo","OVAR","Pombal","Batalha"]

lista_codigos_postais = []
for i in range (1000):

    numero_4_digitos =  random.randint(1,11625)
    numero_3_digitos =  random.randint(1,1156)
    localidade       =  random.choice(list_localidades)
    codigo_postal_i  =  CodigoPostal(numero_4_digitos,numero_3_digitos,localidade)

    lista_codigos_postais.append(codigo_postal_i)


```

```
36 g = GestorCodigoPostais(lista_codigos_postais)
37 g.validar_codigos_postais()
38 print(len(g.codigos_postais))
39 g.codigos_postais[16].print_codigo_postal()
40 codigos_postais_localidade = g.obter_codigos_postais_por_localidade("MAFRA")
41 print(len(codigos_postais_localidade))
```

Indique se as seguintes perguntas são verdadeiras ou falsas.

Pergunta 1

Verdadeiro

O output do print na consola na linha de código 38 é `333`.

Falso

O output do print na consola na linha de código 38 é `334`.

Pergunta 2

Verdadeiro

O output da linha de código 39 é `7356-410 OEIRAS`.

Falso

O output da linha de código 39 é `5993-901 MAIA`.

Pergunta 3

Verdadeiro

O output do print na consola na linha de código 41 é `46`.

Falso

O output do print na consola na linha de código 41 é `45`.

Considere a execução do programa Python 3, que se segue.

```
1 class Livro:
2
3     def __init__(self, titulo):
4
5         self.titulo = titulo
6
7 r = Livro('Cryptography and Network Security')
8 print(r.titulo)
9 print(r.autor)
```

Indique se as seguintes perguntas são verdadeiras ou falsas.

Pergunta 1

Verdadeiro

O output do print na consola na linha de código 8 equivale a `Cryptography and Network Security`.

Falso

O output do print na consola na linha de código 8 equivale a `" "`.

Pergunta 2

Verdadeiro

Na linha de código 9 `dá` erro.

Falso

Na linha de código 9 `não dá` erro.

Considere a execução do programa Python 3, que se segue.

```
1  class I:
2
3      def init(self, f, o, l):
4
5          self.f = f
6          self.o = o
7          self.l = l
8
9      def get(self):
10
11         return self.f
12
13 i = I(55, 30, 83)
14 print(i.get())
```

Indique se a seguinte pergunta é verdadeira ou falsa.

Pergunta 1

Verdadeiro

O programa `dá erro` na linha 13.

Falso

O programa `dá erro` na linha 14.

Considere a execução do programa Python 3, que se segue.

```
class E:

  def __init__(self, a, i):
    self.a = a
    self.i = i
    return self

  def e(self):
    return self.a + self.i

e = E(26, 82)
print(e.e())
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O erro gerado é no método `construtor`.

Falso

O erro gerado é no método `e`.

Considere a execução do programa Python 3, que se segue.

```
class Carro:

    def __init__(self, marca, modelo, ano):

        self.marca  = marca
        self.modelo = modelo
        self.__ano  = ano

    def marca_modelo(self):

        print(self.marca + ' ' + self.modelo)

h = Carro('Audi', 'R8',  2000)
z = Carro('Mini', '1000', 2020 )

h.marca_modelo()
Carro.marca_modelo(h)
z.marca_modelo()
Carro.marca_modelo(z)
```

Pergunta 1

Verdadeiro

O output do seguinte programa é:

```
Audi R8
Audi R8
Mini 1000
Mini 1000
```

Falso

O output do seguinte programa é:

```
Audi R8
Mini 1000
```

Pergunta 2

Verdadeiro

Considere a execução da seguinte linha de código, a linha de código dá erro.

```
print(h.__ano)
```

Falso

Considere a execução da seguinte linha de código, a linha de código não dá erro.

```
print(h.__ano)
```

Em Pyhton 3, complete a classe Somador, no ficheiro `somador.py`:

- a classe possui um construtor, e dois métodos. O método `soma_lista` e o método estatisticas;
- o método `soma_lista` deve ser implementado, onde recebe como argumento: `self` e uma lista de números designado por `lista`. O retorno do método é o somatório de todos os números dentro da lista;
- dentro do método `soma_lista` deve utilizar os atributos `max`, `min`, `total`, `parcelas` e `listas`. Os dois `underscores` antes do nome de cada atributo deve ser ignorado, servem somente para atribuir visibilidade privada aos atributos;
- o atributo `max` devolve o maior número dentro de todas as listas utilizadas no método `soma_lista`, o atributo `min` devolve o menor número dentro das listas, o atributo `total` devolve a soma de todos os números das listas utilizadas, o atributo `parcelas` corresponde ao número total de índices das listas utilizadas e o atributo `listas` ao número de vezes que foi chamado o método `soma_lista`;

Considere a execução do programa Python 3, que se segue.

```
import random

random.seed(69214)

class Somador:

  def __init__(self):

    self.__max         = 0
    self.__min         = 0
    self.__total       = 0
    self.__parcelas    = 0
    self.__listas      = 0

  def estatisticas(self):

    print("número de listas somadas = "+str(self.__listas))
    print("número parcelas somadas = "+str(self.__parcelas))
    print("total somado = "+str(self.__total))
    print("parcela mínima = "+str(self.__min))
    print("parcela máxima = "+str(self.__max))
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

Perante a seguinte execução de Python 3.

```
somador = Somador()
print(somador.soma_lista([random.randint(14,273), random.randint(24,121),
    random.randint(24,140)]))
print(somador.soma_lista([-1*random.randint(1345,5031), -2*random.randint
    (1189, 6392)]))
print(somador.soma_lista([random.randint(14,156), random.randint(37,297)]))
somador.estatisticas()

somador2 = Somador()
print(somador2.soma_lista([random.randint(15,171), random.randint(28,248)]))
```

```
9  print(somador2.soma_lista([-1*random.randint(1401,6538), -2*random.randint
       (1691, 5503), -3*random.randint(1283,7453)]))
10 print(somador2.soma_lista([random.randint(25,176), random.randint(36,329),
       random.randint(7,159), random.randint(5,313)]))
11 somador2.estatisticas()
```

Após implementar o método em falta da classe o output produzido pelas linhas 2, 3, 4 e 5 deve ser:
296
-14484
152
número de listas somadas = 3
número parcelas somadas = 7
total somado = -14036
parcela mínima = -9618
parcela máxima = 127

O output produzido pelas linhas 8, 9, 10 e 11 é:
176
-20134
455
número de listas somadas = 3
número parcelas somadas = 9
total somado = -19503
parcela mínima = -8505
parcela máxima = 168

Falso

Perante a seguinte execução de Python 3.

```
1  somador = Somador()
2  print(somador.soma_lista([random.randint(14,273), random.randint(24,121),
       random.randint(24,140)]))
3  print(somador.soma_lista([-1*random.randint(1345,5031), -2*random.randint
       (1189, 6392)]))
4  print(somador.soma_lista([random.randint(14,156), random.randint(37,297)]))
5  somador.estatisticas()
6
7  somador2 = Somador()
8  print(somador2.soma_lista([random.randint(15,171), random.randint(28,248)]))
9  print(somador2.soma_lista([-1*random.randint(1401,6538), -2*random.randint
       (1691, 5503), -3*random.randint(1283,7453)]))
10 print(somador2.soma_lista([random.randint(25,176), random.randint(36,329),
       random.randint(7,159), random.randint(5,313)]))
11 somador2.estatisticas()
```

Após implementar o método em falta da classe o output produzido pelas linhas 2, 3, 4 e 5 deve ser:
296
-14484
152
número de listas somadas = 3
número parcelas somadas = 7
total somado = -14036
parcela mínima = -9618
parcela máxima = 127

O output produzido pelas linhas 8, 9, 10 e 11 é:
`175`
`-20135`
`456`
número de listas somadas = `3`
número parcelas somadas = `9`
total somado = `-19504`
parcela mínima = `-8506`
parcela máxima = `169`

Considere a execução do programa Python 3, que se segue.

```
print(89)
print(286)
print('20 14 6')
print(25)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

A `primeira` linha de código imprime na consola o número `89`.

Falso

A `primeira` linha de código imprime na consola o número `286`.

Pergunta 2

Verdadeiro

Na primeira linha de código `não dá` erro.

Falso

Na primeira linha de código `dá erro`.

Pergunta 3

Verdadeiro

O output produzido é :
```
89
286
20 14 6
25
```

Falso

O output produzido é : `89 286 20 14 6 25`

Considere a execução do programa Python 3, que se segue.

```
print('tu  rt  le')
print('Turtle')
print('turtle', 'beauty')
print('TURTLE')
print('t u r t l e ')
print('\n')
print('turtle')
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O programa produz o seguinte output:

```
tu  rt  le
Turtle
turtle beauty
TURTLE
t u r t l e

   turtle
```

Falso

O programa produz o seguinte output:

```
turtle
Turtle
turtle beauty
TURTLE
turtle
\n
turtle
```

Pergunta 2

Verdadeiro

O programa produz 7 prints na consola.

Falso

O programa produz 8 prints na consola.

Considere o output da execução de um programa Python 3, que se segue.

```
m
mo
mon
monk
monke
monkey
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O seguinte programa python 3, que se segue, produz o output anterior.

```
str_python = "monkey"

for i in range (1, len(str_python)+1):

    print(str_python[0:i])
```

Falso

O seguinte programa python 3, que se segue, produz o output anterior.

```
str_python = "monkey"

for i in range (0, len(str_python)+1):

    print(str_python[1:i])
```

Considere o programa Python 3, que se segue.

```
str_python = "animal"

for i in range (0, len(str_python)+1):

   num_spaces = len(str_python) -i
   print(" " * num_spaces + str_python[0:i])
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O seguinte programa Python 3, produz o seguinte output:

```

     a
    an
   ani
  anim
 anima
animal
```

Falso

O seguinte programa Python 3, produz o seguinte output:

```
     a
    an
   ani
  anim
 anima
animal
```

Considere a execução do programa Python 3, que se segue.

```
1 print((18 + 30 -14)/30 - 30 * 18)
2 print((18 + 30 -14)/(30 - 30) * 18)
3 print((18 + 30 - 16)/(30 - 16) * 18)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O seguinte programa gera erro na linha de código 2.

Falso

O seguinte programa gera erro na linha de código 1.

Considere a execução do programa Python 3, que se segue.

```
(22 + 27 - 12)
6
17.0
'Hello World'
'print(Hello World)'
# print('Hello World')
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O programa `não gera` nenhum output.

Falso

O programa gera o seguinte output:

```
Hello World
```

Considere a execução do programa Python 3, que se segue.

```
1 ['TRUE', 'FALSE', 'NONE']
2 ['true','false','none']
3 [True,False,None]
4 '[true, false, none]'
5 [true, false, none]
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O programa gera um erro na linha 5.

Falso

O programa gera um erro na linha 4.

Considere a execução do programa Python 3, que se segue.

```
print(8/14)
print(8-14)
print(type(8))
print(type(14))
print(type(8/14))
print(type(8-14))
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O programa gera o seguinte output:

```
0.5714285714285714
-6
<class 'int' >
<class 'int' >
<class 'float' >
<class 'int' >
```

Indique se é verdadeiro ou falso.

Falso

O programa gera o seguinte output:

```
0.5714285714285714
-6
<class 'int' >
<class 'int' >
<class 'int' >
<class 'int' >
```

Indique se é verdadeiro ou falso.

Considere a execução do programa Python 3, que se segue.

```
print('w39cblebvp')
print('w39cblebvp'[6])
print([91, 869, 717, 760, 681, 348, 503, 264, 186])
print([91, 869, 717, 760, 681, 348, 503, 264, 186][8])
print((91, 869, 717, 760, 681, 348, 503, 264, 186)[7])
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output produzido é:
w39cblebvp
e
[91, 869, 717, 760, 681, 348, 503, 264, 186]
186
264

Falso

O output produzido é:
w39cblebvp
l
[91, 869, 717, 760, 681, 348, 503, 264, 186]
264
503

Considere a execução do programa Python 3, que se segue.

```
print('ld8luleb37'[4])
print([632, 399, 604, 637, 199, 673, 483, 62, 613][8])
print((632, 399, 604, 637, 199, 673, 483, 62, 613)[9])
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output produzido pela primeira linha é:
u

Falso

O output produzido pela primeira linha é:
l

Pergunta 2

Verdadeiro

A linha 3 `dá` erro.

Falso

A linha 3 `não dá` erro.

Considere a execução do programa Python 3, que se segue.

```
import random
import string

random.seed(50810)

lista_strings = []

for i in range(1172):
  lista_strings.append(''.join([random.choice(string.ascii_lowercase) for i
   in range(6)]))

print((2, [1, 12], (16, 9, 4),lista_strings[500]))
print((2, [1, 12], (16, 9, 4),lista_strings[500])[1])
print((2, [1, 12], (16, 9, 4),lista_strings[500])[1][1])
print((2, [1, 12], (16, 9, 4),lista_strings[500])[2][2])
print((2, [1, 12], (16, 9, 4),lista_strings[500])[3][3])
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

No elemento `366` da lista `lista_strings` o seu valor é:
ddidrs Falso

No elemento `366` da lista `lista_strings` o seu valor é:
pwcfhc

Pergunta 2

Verdadeiro

O programa anterior gera o seguinte output:

```
(2, [1, 12], (16, 9, 4), 'vhjoyj')
[1, 12]
12
4
o
```

Falso

O programa anterior gera o seguinte output:

```
(2, [1, 12], (16, 9, 4), 'vhjoyj')
2
2
12
4
```

Considere a execução do programa Python 3, que se segue.

```
print((983, [453, 836], (221, 129, 544))[1][0])
print((983, [453, 836], (221, 129, 544))[2][0])
print((983, [453, 836], (221, 129, 544))[0])
print((983, [453, 836], (221, 129, 544))[2][1])
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output produzido é:
453
221
983
129

Falso

O output produzido é:
453
221
[453, 836]
129

Considere a execução do programa Python 3, que se segue.

```
print('98ci9kzfujy'[-1])
print('98ci9kzfujy'[-2])
print([310, 407, 399, 234, 852, 590, 411, 295, 518, 761, 709][-11])
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output produzido é:
y
j
310

Falso

O output produzido é:
j
u
IndexError: list index out of range

Considere a execução do programa Python 3, que se segue.

```
print('toocpwxjuif'[-10])
print('toocpwxjuif'[-11])
print('toocpwxjuif'[-12])
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O programa gera um erro do tipo `IndexError`.

Falso

O programa gera um erro do tipo `TabError`.

Considere a execução do programa Python 3, que se segue.

```
import random
import string

random.seed(2392)

lista_strings = []

for i in range(1057):
  lista_strings.append(''.join([random.choice(string.ascii_lowercase) for i
   in range(7)]))


print((342, [846, 362], (256, 969, 85), '4eoe9v')[-1][-1])
print((342, [846, 362], (256, 969, 85), '4eoe9v')[-2][-1])
print((342, [846, 362], (256, 969, 85), '4eoe9v')[-3][0])
print((342, [846, 362], (256, 969, 85), '4eoe9v')[1][-1])
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output produzido é:
v
85
846
362

Falso

O output produzido é:
969
846
342
846

Pergunta 2

Verdadeiro

O index número 121 da lista `list_strings` tem o seguinte valor dephwyt.

Falso

O index número 121 da lista `list_strings` tem o seguinte valor vhqrsit.

Considere a execução do programa Python 3, que se segue.

```
import random
import string

random.seed(3386)

lista_strings = []

for i in range(1707):
  lista_strings.append(''.join([random.choice(string.ascii_lowercase+string.
   ascii_uppercase) for i in range(3)]))

print('hello python world!')
print('hello python world!'[7:12])
print('hello python world!'[7:13])
print('hello python world!'[7:-8])
print('hello python world!'[-14:-8])
print('hello python world!'[:-8])
print('hello python world!'[7:])
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O elemento 182 da lista `lista_strings` é:
MiU Falso

O elemento 182 da lista `lista_strings` é:
OSc

Pergunta 2

Verdadeiro

O programa anterior gera o seguinte output:

```
hello python world!
ython
ython
ytho
 pytho
hello pytho
ython world!
```

Falso

O programa anterior gera o seguinte output:

```
hello python world!
thon
```

```
thon w
thon
python
hello pytho
ython world!
```

O output que segue corresponde ao número total de números possíveis de criar com 6, 7 e 8 bits, respetivamente.
64
128
256

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O código que se segue produz o output pretendido.

```
print(2**6)
print(2**7)
print(2**8)
```

Falso

O código que se segue produz o output pretendido.

```
print(6**2)
print(7**2)
print(8**2)
```

Tendo em conta os caracteres "á", "à", "ç", "ã" e "é".

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

A execução do programa Python 3 que se segue dá o `output` dos códigos Unicode.

```
print(ord('ç'))
print(ord('ã'))
print(ord('é'))
print(ord('á'))
print(ord('à'))
```

Falso

A execução do programa Python 3 que se segue dá o `output` dos códigos Unicode.

```
print(bytes('ç', "ISO-8859-1"))
print(bytes('ã', "ISO-8859-1"))
print(bytes('é', "ISO-8859-1"))
print(bytes('á', "ISO-8859-1"))
print(bytes('à', "ISO-8859-1"))
```

Pergunta 2

Verdadeiro

Os caracteres "á", "à", "ç", "ã" e "é" `não fazem parte` da codificação ASCII (7 bits). Falso

Os caracteres "á", "à", "ç", "ã" e "é" `fazem parte` da codificação ASCII (7 bits).

Considere a execução do programa Python 3, que se segue.

```
b1 = None
b2 = '323.0'
b3 = [606, 549, 323]
b4 = True
b5 = False
b6 = ['[]']
b7 = 2.0
b8 = 606
b9 = (False, True)
b10 = 'True'
print(type(b1))
print(type(b2))
print(type(b3))
print(type(b4))
print(type(b5))
print(type(b6))
print(type(b7))
print(type(b8))
print(type(b9))
print(type(b10))
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output produzido é:
<class 'NoneType'>
<class 'str'>
<class 'list'>
<class 'bool'>
<class 'bool'>
<class 'list'>
<class 'float'>
<class 'int'>
<class 'tuple'>
<class 'str'>

Falso

O output produzido é:
<class 'None'>
<class 'string'>
<class 'list'>
<class 'bool'>
<class 'bool'>
<class 'list'>
<class 'float'>
<class 'int'>
<class 'tuple'>
<class 'string'>

Considere a execução do programa Python 3, que se segue.

```
1 y              = 13
2 two_x          = 9 * y
3 _2x            = 9 * y
4 a_string       ='hello'
5 AnotherString ='world'
6 A_CONSTANT     = 23
7 max_value      = 12
8 $amount        = 20
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O programa gera um erro na linha 8.

Falso

O programa gera um erro na linha 7.

Considere a execução do programa Python 3, que se segue.

```
1 d = 71
2 j = 418
3 2d = 2 * d
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

A linha 3 dá um erro do tipo `SyntaxError`.

Falso

A linha 3 dá um erro do tipo `RuntimeError`.

Considere a execução do programa Python 3, que se segue.

```
side = 31
square area = side * side
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O programa gera um erro do tipo `SyntaxError`.

Falso

O programa gera um erro do tipo `SystemError`.

Considere a execução do programa Python 3, que se segue.

```
1 j = 40
2 x = 70
3 110 = j + x
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

A linha 3 dá um erro do tipo `SyntaxError`.

Falso

A linha 3 dá um erro do tipo `RuntimeError`.

Considere a execução do programa Python 3, que se segue.

```
v = 4
print(v)
v = v + 5
print(v)
v = v + 4
print(v)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output produzido é:
4
9
13

Falso

O output produzido é:
4
5
4

Considere a execução do programa Python 3, que se segue.

```
1 q = q + 67
2 q = q + 83
3 print(q)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

A linha 1 dá um erro do tipo `NameError`.

Falso

A linha 1 dá um erro do tipo `RuntimeError`.

Considere a execução do programa Python 3, que se segue.

```
b = [70, 70, 70]
print(b)
b[0] = 59
print(b)
b[1] = 80
print(b)
b[2] = 98
print(b)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output produzido é:
[70, 70, 70]
[59, 70, 70]
[59, 80, 70]
[59, 80, 98]

Falso

O output produzido é:
[59, 80, 98]
[59, 80, 98]
[59, 80, 98]
[59, 80, 98]

Considere a execução do programa Python 3, que se segue.

```
1 f = 'bqsak smhht!'
2 print(f[0])
3 print(f[6])
4 f[0] = 'B'
5 f[6] = 'S'
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

A linha 4 dá um erro do tipo `TypeError`.

Falso

A linha 4 dá um erro do tipo `SyntaxError`.

Considere a execução do programa Python 3, que se segue.

```
v = [68, 68, 68]
r = v
r[1] = 52
print(r)
print(v)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output produzido é:
[68, 52, 68]
[68, 52, 68]

Falso

O output produzido é:
[68, 52, 68]
[68, 68, 68]

Considere a execução do programa Python 3, que se segue.

```
1 u = (
2 95 + 86 +
3 30 + 51
4 )
5 h = 95 + 86 +
6 30 + 51
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

Na linha de código 2 `não dá` erro.

Falso

Na linha de código 2 `dá` erro.

Pergunta 2

Verdadeiro

Na linha de código 5 `dá` erro.

Falso

Na linha de código 5 `não dá` erro.

Considere a execução do programa Python 3, que se segue.

```
def f():
  print('t')
  print('o')
  print('i')
print('g')
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output do programa é:
g

Falso

O output do programa é:
t
o
i
g

Considere a execução do programa Python 3, que se segue.

```
def f():
  print('k')
  print('l')
  print('e')
print('p')
f()
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output do programa é:
p
k
l
e

Falso

O output do programa é:
p

Considere a execução do programa Python 3, que se segue.

```
def z(f):
  print(f + f)
z(10)
z(9)
z(21)
z(13)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output do programa é:
20
18
42
26

Falso

O output do programa é:
10
9
21
13

Considere a execução do programa Python 3, que se segue.

```
def k(t, l):
  print(t + t - l + 23)
k(23, 2)
k(2, 23)
k(9, 9)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output do programa é:
67
4
32

Falso

O output do programa é:
69
27
41

Considere a execução do programa Python 3, que se segue.

```
def h(u, n, s):
  print(u * n * s)
h(-5355, -3137, 3342)
h(-5355, -3137, -7970)
h(3342, 3342, 3342)
h(3137, 3137, 3137)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output do programa é:
56141038170
-133885120950
37326677688
30870492353

Falso

O output do programa é:
56141038171
-133885120951
37326677687
30870492352

Considere a execução do programa Python 3, que se segue.

```
def h(u, n, s):
  print(u * n * s)
h(-5355, -3137, 3342)
h(-5355, -3137, -7970)
h(3342, 3342, 3342)
h(3137, 3137, 3137)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output do programa é:
56141038170
-133885120950
37326677688
30870492353

Falso

O output do programa é:
56141038171
-133885120951
37326677687
30870492352

Considere a execução do programa Python 3, que se segue.

```
def v(h):
  z = h + h
  print(z)
print(v(489))
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O output do programa é:
978
None

Falso

O output do programa é:
978

Considere a execução do programa Python 3, que se segue.

```
def l(h):
  t = h + 1
  return h
  print(t)
  return t
f = 101
l(f)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O programa não produz output.

Falso

O output do programa é:
101

Pergunta 2

Verdadeiro

A função `l` retorna 101.

Falso

A função `l` retorna 102.

Considere a execução do programa Python 3, que se segue.

```
def o(v):
    return v + 987
def s(v):
    v[0] = v[1]
m = [o(1), o(2)]
s(m)
print(m)
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O programa gera o seguinte output: [989, 989]

Falso

O programa gera o seguinte output: [988, 989]

Em Pyhton 3, escreva e adicione ao seguinte programa a função **imc**, que retorna o valor do índice de massa corporal, consoante um argumento `altura` e um argumento `massa`.

- O índice de massa corporal, imc, de uma pessoa, é dado pela fórmula $imc = massa \div altura \times altura$. Onde *massa* é o peso da pessoa, em Kg, e a *altura* é a altura da pessoa, em m (metros);
- A função **imc** irá ter como primeiro argumento `altura` e como segundo argumento `massa`;
- O valor retornado para o imc será arredondado para a segunda casa décimal;
- A função `seed`, é utilizada para inicializar um gerador de número aleatórios;
- A função uniform, da biblioteca random, é utilizada para gerar valores do tipo float aleatórios num intervalo especifico.

Considere a execução do programa Python 3, que se segue.

```
1  import random
2
3  random.seed(3418)
4
5  lista_imc = []
6  for i in range (948):
7
8      lista_imc.append(imc(round(random.uniform(1,2),2),round(random.uniform
       (40,80),2)))
9
10 print(lista_imc[320])
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O print na consola na linha 10 produz o seguinte output: 20.06

Falso

O print na consola na linha 10 produz o seguinte output: 13.03

Pergunta 2

Verdadeiro

Uma função python pode retornar múltiplos valores.

Falso

Uma função python só pode retornar um único valor.

Em Pyhton 3, escreva e adicione ao seguinte programa a função **imc**, que retorna o valor do índice de massa corporal, consoante um argumento `altura` e um argumento `massa`.

- Em Python 3, o operador de exponenciação (ou potência) é `**`. Permite obter $potencia = base^{expoente}$ usando potencia = base ** expoente.
- O índice de massa corporal, imc, de uma pessoa, é dado pela fórmula $imc = massa \div altura^2$. Onde *massa* é o peso da pessoa, em Kg, e a *altura* é a altura da pessoa, em m (metros);
- A função **imc** irá ter como primeiro argumento `altura` e como segundo argumento `massa`. A função **imc** retorna o imc correpondente ao peso e à altura. Use o operador de potência para calcular $altura^2$;
- O valor retornado para o imc será arredondado para a segunda casa décimal;
- A função `seed`, é utilizada para inicializar um gerador de número aleatórios;
- A função uniform, da biblioteca random, é utilizada para gerar valores do tipo float aleatórios num intervalo especifico.

Considere a execução do programa Python 3, que se segue.

```
1  import random
2
3  random.seed(2981)
4
5  lista_imc = []
6  for i in range (1204):
7      altura = round(random.uniform(1,2),2)
8      massa  = round(random.uniform(40,80),2)
9      lista_imc.append(imc(altura,massa))
10
11 print(lista_imc[154])
12 print(type(2**10))
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O print na consola na linha 11 produz o seguinte output: 26.55

Falso

O print na consola na linha 11 produz o seguinte output: 16.71

Pergunta 2

Verdadeiro

O print na consola na linha 12 produz o seguinte output:
`<class 'int'>`

Falso

O print na consola na linha 12 produz o seguinte output:
`<class 'float'>`

Em Python 3, o operador ** funciona com bases e expoentes decimais. Como $\sqrt{x} = x^{\frac{1}{2}}$, o operador `**` permite calcular raízes quadradas. Excreva a função `raiz_quadrada`. A função `raiz_quadrada` tem um único argumento, x. A função `raiz_quadrada` retorna a raiz quadrada de x.

Exemplo de utilização da função:

```
a = 25
print(raiz_quadrada(a))
b = 64
print(raiz_quadrada(b))
c = 1000
print(raiz_quadrada(c))
```

O código anterior produz o seguinte output:

```
5.0
8.0
31.622776601683793
```

Pergunta 1

Verdadeiro

Considere o seguinte código Python 3.

```
d = 226
print(raiz_quadrada(d))
e = 4844
print(raiz_quadrada(e))
```

O código anterior produz o seguinte outeput:

```
15.033296378372908
69.59885056522126
```

Falso

Considere o seguinte código Python 3.

```
d = 226
print(raiz_quadrada(d))
e = 4844
print(raiz_quadrada(e))
```

O código anterior produz o seguinte outeput:

```
15.133296378372908
69.69885056522125
```

As raízes da equação de segundo grau $ax^2 + bx + c = 0$ são dadas pela fórmula resolvente,

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Em Python 3, escreva a função `formula_resolvente`. A função `formula_resolvente` tem 3 argumentos, a, b e c. Todos os argumentos são números. A função `formula_resolvente` retorna um tuplo com dois elementos. Cada um dos elementos, do tuplo retornado, é uma das raízes da equação de segundo grau com coeficientes a, b e c. Use o operador `**` para fazer a raiz quadrada.

Exemplo de utilização da função:

```
# x**2 + 4x -21 = 0
a = 1
b = 4
c = -21
raizes = formula_resolvente(a, b, c)
raiz_1 = raizes[0]
raiz_2 = raizes[1]
print('a equação x**2 + 4x -21 = 0 tem as raízes:')
print('x =')
print(raiz_1)
print('e x =')
print(raiz_2)
```

O código anterior produz o seguinte output:

```
a equação x**2 + 4x -21 = 0 tem as raízes:
x =
3.0
e x =
-7.0
```

Pergunta 1

Verdadeiro

Considere o seguinte código Python 3.

```
# -7x**2 + 37x -139
d = -7
e = 37
f = -139
raizes2 = formula_resolvente(d, e, f)
raiz_3 = raizes2[0]
raiz_4 = raizes2[1]
print('a equação -7x**2 + 37x -139 = 0 tem as raízes:')
print('x =')
print(raiz_3)
print('e x =')
print(raiz_4)
```

O código anterior produz o seguinte outeput:

```
a equação -7x**2 + 37x -139 = 0 tem as raízes:
x =
(2.642857142857143-3.587819529964103j)
e x =
(2.642857142857143+3.587819529964103j)
```

Falso

Considere o seguinte código Python 3.

```
# -7x**2 + 37x -139
d = -7
e = 37
f = -139
raizes2 = formula_resolvente(d, e, f)
raiz_3 = raizes2[0]
raiz_4 = raizes2[1]
print('a equação -7x**2 + 37x -139 = 0 tem as raízes:')
print('x =')
print(raiz_3)
print('e x =')
print(raiz_4)
```

O código anterior produz o seguinte outeput:

```
a equação -7x**2 + 37x -139 = 0 tem as raízes:
x =
(2.5428571428571427-3.587819529964103j)
e x =
(2.7428571428571437+3.587819529964103j)
```

Sejam C, a temperatura em graus Celsius, e *F*, a temperatura em graus Fahrenheit. A conversão de graus Celsius para graus Fahrenheit é dada pela fórmula F= 1,8C+ 32. A conversão de graus Fahrenheit para graus Celsius é dada pela fórmula C= `(F-32)/1,8`. Em Pyhton 3, adicione ao seguinte programa e escrevas as funções **c2f** e **f2c**

- A função `c2f` tem um argumento, o valor da temperatura em graus Celsius, e retorna o valor da temperatura em graus Fahrenheit.
- A função `f2c` tem um argumento, o valor da temperatura em graus Fahrenheit, e retorna o valor da temperatura em graus Celsius.
- A função `seed`, é utilizada para inicializar um gerador de número aleatórios;
- A função uniform, da biblioteca random, é utilizada para gerar valores do tipo float aleatórios num intervalo especifico.

Considere a execução do programa Python 3, que se segue.

```
1  import random
2  random.seed(92931)
3
4  lista_f2c = []
5  for i in range (1275):
6
7      lista_f2c.append(f2c(round(random.uniform(30,100),2)))
8
9  lista_c2f = []
10 for i in range (1052):
11
12     lista_c2f.append(c2f(round(random.uniform(1,38),2)))
13
14 print(lista_c2f[687])
15 print(lista_f2c[1011])
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O print na consola na linha 14 produz o seguinte output: 40.5

Falso

O print na consola na linha 14 produz o seguinte output: 91.33

Pergunta 2

Verdadeiro

O print na consola na linha 15 produz o seguinte output: 31.57

Falso

O print na consola na linha 15 produz o seguinte output: 11.59

Considere a execução do programa Python 3, que se segue.

```
seed = 198405
def pseudo_random_integer(min_int, max_int):
    global seed
    seed = (16807*seed) % 2147483647
    return int(min_int + (max_int - min_int) * seed / 2147483646)

t = []
for d in range(76157):
    t.append(pseudo_random_integer(875, 4785))
```

Indique se é verdadeiro ou falso.

Pergunta 1

Verdadeiro

O elemento da lista `t`, no índice `2657`, é `2302`.

Falso

O elemento da lista `t`, no índice `2657`, é `2303`.

Pergunta 2

Verdadeiro

O elemento da lista `t`, no índice `2765`, é `4117`.

Falso

O elemento da lista `t`, no índice `2765`, é `4116`.

Pergunta 3

Verdadeiro

O elemento da lista `t`, no índice `3015`, é `1982`.

Falso

O elemento da lista `t`, no índice `3015`, é `1981`.

Pergunta 4

Verdadeiro

O elemento da lista `t`, no índice `1162`, é `2016`.

Falso

O elemento da lista `t`, no índice `1162`, é `2015`.

Pergunta 5

Verdadeiro

O elemento da lista `t`, no índice `1424`, é `3538`.

Falso

O elemento da lista `t`, no índice `1424`, é `3539`.

Bibliografia

[Python Errors, 2021] Python Errors (2021). Python Errors and Built-in Exceptions in Python programming language.
https://www.programiz.com/python-programming/exceptions

[Learn LaTeX, 2021] Learn LaTeX (2021). Learn the basics about LaTeX.
https://pt.overleaf.com/learn/latex/Creating_a_document_in_LaTeX

[Highlighting code LaTeX, 2021] Highlighting code LaTeX (2021). Highlighting the code with packages on LaTeX.
https://pt.overleaf.com/learn/latex/Code_Highlighting_with_minted

[Ghostscript Documentation, 2021] Ghostscript Documentation (2021). Documentation for Recently Released Ghostscript Versions.
https://www.ghostscript.com/documentation.html