TÉCNICO LISBOA

Fundamentos da Programação

Ano letivo 2017-18

Primeiro Projeto

Palavra GURU

O jogo *Palavra Guru* é um quebra-cabeças que requer a formação de palavras a partir de um conjunto de letras.

Figura 1: Exemplo de um quebra-cabeças do Palavra Guru.

O objetivo do jogo é descobrir um conjunto de palavras de tamanhos variados, formadas a partir de 5 ou 6 letras. Por exemplo, na Fig. 1 a partir do conjunto de letras *{A, E, L, M, T}* é necessário descobrir três palavras de três letras e duas palavras de quatro letras, obtendo-se as palavras *{ELA, EMA, TAL, META, TEMA}*.

Para resolver um enigma, é preciso criar palavras dos tamanhos requeridos e descobrir

as palavras escondidas de entre as palavras válidas. No exemplo anterior, *TEM* e *TELA* são palavras válidas mas não correspondem às palavras escondidas.

1 Trabalho a Realizar

O objetivo do primeiro projeto é escrever um programa em Python que permita verificar a validade de uma palavra, dado um conjunto de letras.

Para tal, deverá definir um conjunto de funções que implementem um programa que verifique se uma palavra pertence à linguagem definida a seguir.

1.1 Gramática Guru

```
<palavra>::=<monossilabo>|<silaba>*<silaba_final>
<silaba>::=<vogal>|<silaba_2>|<silaba_3>|<silaba_4>|<silaba_5>
<silaba_final>::=<monossilabo_2>|<monossilabo_3>|<silaba_4>|<silaba_5>
<silaba_5>::=<par_consoantes><vogal>NS
<silaba_4>::=<par_vogais>NS|<consoante><vogal>NS|<consoante><vogal>IS
              |<par_consoantes><par_vogais>
              |<consoante><par_vogais><consoante_final>
<silaba_3>::=QUA|QUE|QUI|GUE|GUI|<vogal>NS|<consoante><par_vogais>
              |<consoante><vogal><consoante_final>
              |<par_vogais><consoante_final>
              |<par_consoantes><vogal>
<silaba_2>::=<par_vogais>|<consoante><vogal>|<vogal><consoante_final>
<monossilabo>::=<vogal_palavra>|<monossilabo_2>|<monossilabo_3>
<monossilabo_3>::=<consoante><vogal><consoante_terminal>
              |<consoante><ditongo>
              |<par_vogais><consoante_terminal>
<monossilabo_2>::=AR|IR|EM|UM|<vogal_palavra>S|<ditongo_palavra>
              |<consoante_freq><vogal>
<par_consoantes>::=BR|CR|FR|GR|PR|TR|VR|BL|CL|FL|GL|PL
<consoante>::=B|C|D|F|G|H|J|L|M|N|P|Q|R|S|T|V|X|Z
<consoante_final>::=N|P|<consoante_terminal>
<consoante_terminal>::=L|M|R|S|X|Z
<consoante_freq>::=D|L|M|N|P|R|S|T|V
<par_vogais>::=<ditongo>|IA|IO
<ditongo>::=AE|AU|EI|OE|OI|IU|<ditongo_palavra>
<ditongo_palavra>::=AI|AO|EU|OU
<vogal>::=I|U|<vogal_palavra>
<vogal_palavra>::=<artigo_def>|E
<artigo_def>::=A|O
```

1.2 Funções a implementar

- *e_silaba*: *cad. caracteres* → *booleano* **(7 val.)**

  Esta função recebe uma *cadeia de caracteres* e devolve um *booleano*. Se a cadeia de caracteres formar uma sílaba válida de acordo com a gramática dada, ou seja, se satisfizer a regra **silaba**, a função devolve *True*, senão devolve *False*. A função deve verificar a validade do seu argumento, gerando um `ValueError` com a mensagem `'e_silaba:argumento invalido'` caso o argumento introduzido seja inválido.

- *e_monossilabo* : *cad. caracteres* → *booleano* **(7 val.)**

  Esta função recebe uma *cadeia de caracteres* e devolve um *booleano*. Se a cadeia de caracteres formar uma palavra válida de acordo com a gramática dada e contiver apenas com uma sílaba, ou seja, se satisfizer a regra **monossilabo**, a função devolve *True*, senão devolve *False*. A função deve verificar a validade do seu argumento, gerando um `ValueError` com a mensagem `'e_monossilabo:argumento invalido'` caso o argumento introduzido seja inválido.

- *e_palavra* : *cad. caracteres* → *booleano* **(6 val.)**

  Esta função recebe uma *cadeia de caracteres* e devolve um *booleano*. Se a cadeia de caracteres formar uma palavra válida de acordo com a gramática dada, ou seja, se satisfizer a regra **palavra**, a função devolve *True*, senão devolve *False*. A função deve verificar a validade do seu argumento, gerando um `ValueError` com a mensagem `'e_palavra:argumento invalido'` caso o argumento introduzido seja inválido.

Exemplo de interação:

```
3.3.2 (v3.3.2:d047928ae3f6, May 13 2013, 12:45:22)
[GCC 4.2.1 (Apple Inc. build 5577)]
Python Type "help", "copyright", "credits" or "license" for more infor...
[evaluate gramatica.py]
>>> e_silaba("a")
False
>>> e_silaba("A")
True
>>> e_silaba("AI")
True
>>> e_silaba("QUE")
True
>>> e_silaba("TAIS")
True
>>> e_silaba("TRANS")
True
>>> e_monossilabo("AR")
True
>>> e_monossilabo("AE")
```

```
False
>>> e_monossilabo("RIO")
False
>>> e_monossilabo("TAO")
True
>>> e_palavra("A")
True
>>> e_palavra("I")
False
>>> e_palavra("AO")
True
>>> e_palavra("EO")
False
>>> e_palavra("RIO")
False
>>> e_palavra("RIE")
False
>>> e_palavra("CONFORTO")
True
>>> e_palavra("TRANSPORTAVA")
True
>>> e_palavra("CONSTITUCIONAL")
True
```

1.3 Sugestões

1. Leia o enunciado completo, procurando perceber o objetivo das várias funções pedidas. Em caso de dúvida de interpretação, utilize o horário de dúvidas para esclarecer as suas questões.
2. No processo de desenvolvimento do projeto, comece por implementar as várias funções pela ordem apresentada no enunciado, seguindo as metodologias estudadas na disciplina. Ao desenvolver cada uma das funções pedidas, comece por perceber se pode usar alguma das anteriores.
3. Para verificar a funcionalidade das suas funções, utilize os exemplos fornecidos como casos de teste.
4. Tenha o cuidado de reproduzir fielmente as mensagens de erro e restantes *outputs*, conforme ilustrado nos vários exemplos.

2 Aspetos a Evitar

Os seguintes aspetos correspondem a sugestões para evitar maus hábitos de trabalho (e, consequentemente, más notas no projeto):

1. Não pense que o projeto se pode fazer nos últimos dias. Se apenas iniciar o seu trabalho neste período irá ver a Lei de Murphy em funcionamento (todos os problemas são mais difíceis do que parecem; tudo demora mais tempo do que nós pensamos; e se alguma coisa puder correr mal, ela vai correr mal, na pior das alturas possíveis).
2. *Não duplique código*. Se duas funções são muito semelhantes é natural que estas possam ser fundidas numa única, eventualmente com mais argumentos.
3. Não se esqueça que as funções excessivamente grandes são penalizadas no que respeita ao estilo de programação.
4. A atitude "vou pôr agora o programa a correr de qualquer maneira e depois preocupo-me com o estilo" é totalmente errada.
5. Quando o programa gerar um erro, preocupe-se em descobrir qual a causa do erro. As "marteladas" no código têm o efeito de distorcer cada vez mais o código.

3 Classificação

A avaliação da execução será feita através do sistema *Mooshak*, onde existem vários testes configurados no sistema. O tempo de execução de cada teste está limitado, bem como a memória utilizada. O sistema não deverá ser utilizado para debug e como tal, só poderá efetuar uma nova submissão pelo menos 15 minutos depois da submissão anterior. Só são permitidas 10 submissões em simultâneo no sistema, pelo que uma submissão poderá ser recusada se este limite for excedido.[1] Nesse caso tente mais tarde.

Os testes considerados para efeitos de avaliação podem incluir ou não os exemplos disponibilizados, além de um conjunto de testes adicionais. O facto de um projeto completar com sucesso os exemplos fornecidos não implica, pois, que esse projeto esteja totalmente correto, pois o conjunto de exemplos fornecido não é exaustivo. É da responsabilidade de cada aluno garantir que o código produzido está correto.

Não será disponibilizado qualquer tipo de informação sobre os casos de teste utilizados pelo sistema de avaliação automática. Os ficheiros de teste usados na avaliação do projeto serão disponibilizados na página da disciplina após a data de entrega.

A nota do projeto será baseada nos seguintes aspetos:

1. Execução correta (70%).

   Esta parte da avaliação é feita recorrendo ao sistema *Mooshak* que sugere uma nota face aos vários aspetos considerados.

[1] Note que o limite de 10 submissões simultâneas no sistema *Mooshak* implica que, caso haja um número elevado de tentativas de submissão sobre o prazo de entrega, alguns grupos poderão não conseguir submeter nessa altura e verem-se, por isso, impossibilitados de submeter o código dentro do prazo.

2. Estilo de programação e facilidade de leitura, nomeadamente a abstração procedimental, nomes bem escolhidos, qualidade (e não quantidade) dos comentários e tamanho das funções (30%). Os seus comentários deverão incluir, entre outros, a assinatura de cada função definida.

4 Condições de Realização e Prazos

A entrega do 1º projeto será efetuada exclusivamente por via eletrónica. Deverá submeter o seu projeto através do sistema *Mooshak*, até às **23:59 do dia 3 de Novembro de 2017**. Depois desta hora, não serão aceites projetos sob pretexto algum.

Deverá submeter um único ficheiro com extensão `.py` contendo todo o código do seu projeto. O ficheiro de código deve conter em comentário, na primeira linha, o número e o nome do aluno.

No seu ficheiro de código não devem ser utilizados caracteres acentuados ou qualquer carácter que não pertença à tabela ASCII. Isto inclui comentários e cadeias de caracteres. Programas que não cumpram este requisito serão penalizados em três valores.

Duas semanas antes do prazo, serão publicadas na página da cadeira as instruções necessárias para a submissão do código no *Mooshak*. Apenas a partir dessa altura será possível a submissão por via eletrónica. Nessa altura serão também fornecidas a cada um as necessárias credenciais de acesso. Até ao prazo de entrega poderá efetuar o número de entregas que desejar, sendo utilizada para efeitos de avaliação a última entrega efetuada. Deverá portanto verificar cuidadosamente que a última entrega realizada corresponde à versão do projeto que pretende que seja avaliada. Não serão abertas exceções.

Pode ou não haver uma discussão oral do trabalho e/ou uma demonstração do funcionamento do programa (será decidido caso a caso).

Lembre-se que no Técnico, a fraude académica é levada muito a sério e que a cópia numa prova (projetos incluídos) leva à reprovação na disciplina. O corpo docente da cadeira será o único juiz do que se considera ou não copiar num projeto.