Introdução à Programação e Resolução de Problemas

2009/2010

Ficha #8

Ernesto Costa,
Luís Macedo, Rogério de Lemos, Vasco Pereira

Ficha 8

Enforcado

Objectivos

- ✔ Exercitar conceitos de programção descendente e incremental
- ✔ Exercitar a definição de interfaces e a reutilização de código
- ✔ Consolidar o uso de ficheiros, dicionários, entrada e saída de dados e, ainda, de listas e de cadeias de caracteres
- ✔ Motivar para o recurso a **excepções**.

Requisitos

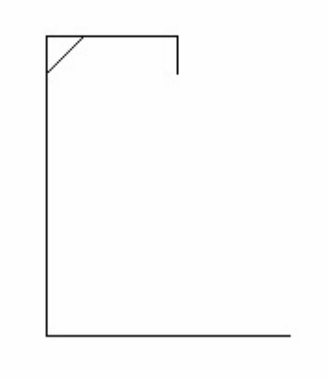
- ✔ Conceitos anteriores consolidados
- ✔ Módulos: **cTurtle**

Material de Apoio

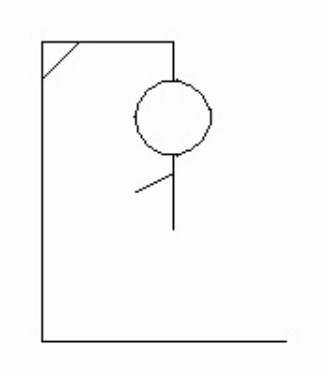
- ✔ É recomendada a consulta dos slides das aulas teóricas, dos textos de apoio que se encontram no **WoC** e da bibliografia indicada.

8.1 Introdução

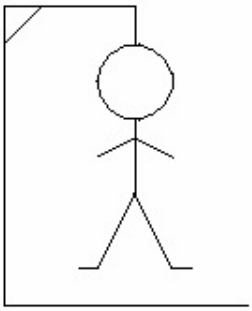
Todos jogámos em criança ao jogo do enforcado (*hangman*): uma pessoa escolhe uma palavra que uma segunda tenta adivinhar indicando, caracter a caracter, a sua composição. Por cada caracter falhado, aparece uma parte do nosso corpo agarrado a uma forca (ver figura 8.1). Simpático! O objectivo desta ficha é desenvolver um programa que permita jogar no computador o jogo. O computador define a palavra, o utilizador tenta adivinhar. O problema serve de pretexto para aprofundar o conhecimento da linguagem **Python** e exercitar competências no desenvolvimento de programas. Vamos ter que **dominar a complexidade** aproximando-nos passa a passo de uma solução razoável. Não se espere uma solução única! Havendo várias alternativas para diversos passos do programa estas devem ser ponderadas.



(a) No início



(b) No meio



(c) No fim??

Figura 8.1: **Enforcado**

Ao longo da ficha iremos fazer melhoramentos a partir das versões anteriormente desenvolvidas.

8.2 Questões marginais . . . ou talvez não!

Antes de passarmos à resolução do problema do enforcado vamos tentar resolver algumas questões. Estes problemas devem ser resolvidos **antes** da aula.

Problema 8.1 F

As listas e as cadeias de caracteres são tipos de objectos. Em certas circunstâncias, é possível passar de uma cadeia de caracteres a uma lista e vice-versa. Exemplifique, isto é, teste no computador como pode:

- transformar uma cadeia de caracteres numa lista
- transformar uma lista de cadeias de caracteres numa única cadeia da caracteres.

Problema 8.2 F

Suponha que está a pedir dados ao utilizador. Admita ainda que esses dados devem ser números inteiros num dado intervalo. Cada vez que o utilizador se engana, indicando um número fora do intervalo, o número não é guardado e o pedido deve ser repetido. Desenvolva um pequeno programa que permita fazer esta **protecção da entrada dos dados**. a listagem 8.1 ilustra o que se pretende.

```
1  >>>
2  Evaluating untitled-1.py
3  Um número inteiro no intervalo [-30,30]: 20
4  Um número inteiro no intervalo [-30,30]: -50
5  ERRO: o número deve estar no intervalo [-30,30]!!
6  Um número inteiro no intervalo [-30,30]: 60
7  ERRO: o número deve estar no intervalo [-30,30]!!
8  Um número inteiro no intervalo [-30,30]: 4
9  Um número inteiro no intervalo [-30,30]: 3
10 [20, 4, 3]
11 >>>
```

Listagem 8.1: Proteger os dados.

Problema 8.3 F

Pretende-se melhorar o problema 8.2, filtrando também os valores repetidos. Implemente o respectivo programa. A listagem 8.2 ilustra o que se pretende.

```
>>>
Evaluating ficha8.py
Um número inteiro no intervalo [-30,30]: 4
Um número inteiro no intervalo [-30,30]: 4
ERRO: o número 4 já existe!!
Um número inteiro no intervalo [-30,30]: 3
Um número inteiro no intervalo [-30,30]: -50
ERRO: o número deve estar no intervalo [-30,30]!!
Um número inteiro no intervalo [-30,30]: 2
[4, 3, 2]
>>>
```

Listagem 8.2: Proteger os dados (2).

Problema 8.4 F

Suponha que tem duas listas. Uma tem nomes de pessoas do sexo masculino, a outra tem nomes de pessoas do sexo feminino. Vai organizar uma festa e quer convidar **igual** número de homens e de mulheres. Não tem lugar para todos e, por isso, apenas vai convidar alguns. Como não tem critério para os distinguir, afinal são todos seus amigos, a escolha de quem virá será **aleatória**. O mesmo procedimento será adoptado para a distribuição pela mesa de jantar. A única **restrição** é que vai querer juntar um homem com uma mulher. Escreva um programa para resolver esta questão, isto é, dadas as duas listas, devolve uma lista de pares (mulher, homem).

Problema 8.5 F

Problema semelhante ao 8.4, só que agora a informação sobre os nomes encontra-se em dois ficheiros: um com nomes de mulheres, outro com nomes de homens.

Problema 8.6 M

Em ficha anterior vimos como podemos fazer um **dicionário de frequências de ocorrências**. Por exemplo, se tivermos uma lista com números, queremos construir um dicionário em que cada chave é um dos números da lista e o valor associado é o número de vezes que o número está na lista. Vamos fazer algo de semelhante. No entanto, agora o que queremos é associar a cada chave os **índices** da sua ocorrência na lista. O programa deve funcionar seja com números, seja com cadeias de caracteres. A listagem 8.3 ilustra o

pretendido.

```
>>>print dic_freq('abaabcba')
{'a': [0, 2, 3, 7], 'c': [5], 'b': [1, 4, 6]}
>>>
```

Listagem 8.3: Dicionário de frequências

8.3 Solução básica

Retomemos o problema do enforcado. Antes de começar a resolver o exercício 8.7 leia, por favor, todas as perguntas para perceber melhor o que se entende por solução básica.

Problema 8.7 M

Pretende-se desenvolver um programa simples para o jogo do enforcado. Nesta fase apenas se pretende poder jogar o jogo usando o computador. **Não** temos pretensões de ter um **interface gráfico** desenvolvido. Vejamos o que se pretende através de pequenas listagens resultantes da interacção.

Quando tudo corre bem para o utilizador.

```
>>>
Evaluating hang_0.py
Palavra:
_ _ _ _ _ _ _ _ _ _

Escolha uma letra: a
Palavra:
a _ a _ a _ a _ _ a

Escolha uma letra: b
Palavra:
a b a _ a _ a b _ a

Escolha uma letra: c
Palavra:
a b a c a _ a b _ a

Escolha uma letra: d
Palavra:
a b a c a d a b _ a

```

```
22 Escolha uma letra: r
23
24 Uau! Acertou!
25 Palavra:
26 a b a c a d a b r a
```

Listagem 8.4: Quando se acerta.

A listagem acima mostra que o programa tem um comportamento repetitivo. Vemos, por exemplo, que entre a linha 3 e 6 é-nos mostrado o estado actual da palavra que queremos adivinhar (linha 3-4) e a escolha que fazemos para a letra (linha 6). Quando na etapa seguinte retomamos a visualização da palavra já se reflecte o facto de termos acertado na letra (linha 8).

Quando tudo corre mal:

```
1  >>>
2  Evaluating hang_0.py
3  Palavra:
4  _ _ _ _ _ _ _ _ _ _ _ _
5
6  Escolha uma letra: x
7  Palavra:
8  _ _ _ _ _ _ _ _ _ _ _ _
9
10 ... semelhante
11
12 Escolha uma letra: g
13 Palavra:
14 _ _ _ _ _ _ _ _ _ _ _ _
15
16 Escolha uma letra: k
17 Palavra:
18 _ _ _ _ _ _ _ _ _ _ _ _
19
20 Escolha uma letra: s
21 Oops! Esgotaram-se as suas hipóteses...
22 A palavra secreta era:
23 Palavra:
24 r i n o c e r o n t e
25
26 See you!
```

Listagem 8.5: Quando se erra (listagem incompleta).

Na linha 10 **escrevemos** semelhante para indicar que não se mostram muitas das escolhas. Note-se a mensagem final (linhas 21-26)

Quando repetimos letras:

```
1  >>>
2  Evaluating hang_0.py
3  Palavra:
4  _ _ _ _ _ _ _ _ _
5
6  Escolha uma letra: a
7  Palavra:
8  _ _ _ _ _ _ _ _ a
9
10 Escolha uma letra: a
11
12 *** Letra já usada. Escolha outra sff!***
13 Escolha uma letra: b
14 Palavra:
15 _ _ _ _ _ _ _ _ a
16
17 Escolha uma letra: e
18 Palavra:
19 e _ _ _ _ _ _ _ a
```

Listagem 8.6: Quando se erra (listagem incompleta).

Entre as linhas 10 e 13 exemplificamos o que acontece quando indicamos uma letra que já anteriormente tinha sido escolhida.
Agora é preciso meter as mãos ao trabalho. Para escrever programas com alguma dimensão e complexidade é preciso **pensar** antes de começar a escrever o código. Convém esquecer **todos** os detalhes e iniciar a busca da solução identificando os objectos e uma solução global. Por exemplo, precisamos de determinar:

- Como escolhe o computador a palavra secreta?
- Como representar **internamente** a palavra secreta, fixa ao longo do programa, e a solução parcial que o utilizador foi construindo?
- Em que situações pode terminar o programa?
- Qual a lógica geral do programa?

e outras coisas que poderão surgir. No início não deve ter receio de colocar muitas questões. Mais tarde vai *arrumar a casa*, ficando apenas com algumas dessas questões para responder. Um esqueleto de programa principal pode ser o da listagem 8.7.

```
1  def hangman():
2      # inicialização
3      # --- palavra secreta e representacão dos objectos
4      # --- parâmetros
5
6    repete :
7          # estado actual
8          # escolhe letra
9          # analisa resposta
10         # actualiza estado
11
12     # mensagem final
```

Listagem 8.7: Enforcado: aspecto geral.

Não se iniba de procurar algo diferente! O que cada um destes passos deve fazer é, mais ou menos, auto explicativo. Durante a aula irá ser chamado a fazer o que falta, discutindo as alternativas que se colocam a cada passo. **Atenção**: se não dominar Python a sua solução será seguramente mais pobre!

8.4 Complementos fáceis

Obtido um programa funcional, está na hora de melhorar a solução inicial. Será tanto mais fácil, quanto mais modular for o código. As alterações sugeridas não são as únicas possíveis e, mesmo estas não têm necessariamente uma solução única. Programar significa também estar sempre a fazer **opções**!

Problema 8.8 F

Altera a solução para que seja possível jogar repetidas vezes o jogo e não apenas uma vez.

Problema 8.9 F

Tente introduzir o conceito de **estado** do jogo. Por exemplo, o estado pode caracterizar o número de tentativas já efectuadas, as letras usadas e as erradas, a palavra com as letras já adivinhadas, etc. Use essa noção para

tornar o programa mais legível e modular.

Problema 8.10 F

Altere o programa para que diferentes jogadores o possam usar, guardando informação **estatística** sobre o desempenho dos diferentes jogadores. Por exemplo: número de jogos efectuados, quantas vezes acertou, número de jogadas necessário para acertar, etc.

Problema 8.11 F

Introduza o conceito de **nível** do jogador. Por exemplo, os jogadores podem ser divididos em três categorias: iniciado, normal, perito. A escolha de um nível pelo utilizador tem implicações quanto à dificuldade da palavra a adivinhar e ao número de tentativas que são dadas ao utilizador. Pense numa solução simples e equilibrada.

Problema 8.12 F

Altere o programa de modo a detectar quando o jogador já não tem hipótese de acertar em função do número de tentativas e das letras por adivinhar. Deve então terminar o jogo.

8.5 Introduzir visualização

Problema 8.13 M

Bom, está na hora de pensar numa interação **parcialmente** visual. Adapte o código de modo a que vá sendo desenhado o homem na forca. Tem liberdade de escolher a solução. Na figura 8.1 mostramos uma possibilidade. Para melhor ter a noção das proporções a figura 8.2 dá pistas para construir o desenho.

8.6 Usar excepções

Problema 8.14 D

Em que medida o seu programa é robusto? Tente fazer a protecção dos dados. Pense na possibilidade de usar o mecanismo de **excepções**. Não tendo sido matéria dada deve procurar estudar o que são socorrendo-se dos docentes para tirar dúvidas.

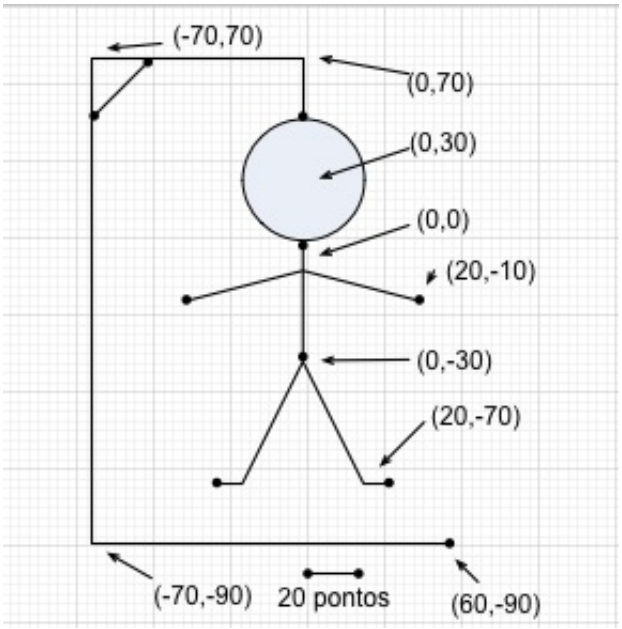


Figura 8.2: **Enforcado: desenho**