Instituto Superior Técnico

Fundamentos da Programação
2016/2017

Enunciado do 1º Projecto

Data de entrega: 4 de Novembro de 2016 às 23h59

Encriptação de Mensagens

Pretende-se com este trabalho a implementação em Python de um algoritmo de encriptação. A criptografia (do grego "kryptos", oculto, e "graphein", escrever) estuda formas de ocultar informação. Julga-se que já os Gregos Antigos e os Espartanos usavam formas de encriptação para enviar mensagens durante as campanhas militares. Actualmente, a criptografia é essencial em aplicações que utilizam os cartões Multibanco ou de crédito e o comércio electrónico. Neste contexto, existem nos dias de hoje vários algoritmos de encriptação, continuando matemáticos e informáticos activamente a trabalhar no desenvolvimento de algoritmos que se pretendem cada vez mais seguros.
Um algoritmo de encriptação transforma um texto normal num texto encriptado. Para recuperar o texto original é depois necessário aplicar-lhe um outro algoritmo que faça a operação inversa. Este tipo de algoritmos existem portanto sempre aos pares, constituindo aquilo a que se chama uma cifra. Para decifrar um texto encriptado é necessário não só o algoritmo correcto, mas também *chave* usada na encriptação. Por exemplo, uma forma muito simples de encriptar um texto é substituir cada letra pela letra que está i posições mais adiante no alfabeto. Embora seja relativamente simples escrever algoritmos de encriptação, e que descodifiquem um texto encriptado, só é possível decifrar um texto encriptado se for conhecida a chave usada na encriptação, neste caso o valor de i.

1 Trabalho a Desenvolver

O algoritmo que vamos utilizar baseia-se no **quadrado de Polybius**, criado pelo historiador grego do mesmo nome, nascido cerca de 200 A.C. No quadrado de Polybius as letras do alfabeto encontram-se dispostas numa matriz 5×5:

A	B	C	D	E
F	G	H	I/J	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

Na encriptação de uma mensagem cada caractere é substituído por dois dígitos, correspondentes à linha e à coluna do caractere no quadrado de Polybius. Por exemplo, se considerarmos que a numeração das linhas e das colunas começa em zero, a encriptação da mensagem 'PROGRAMACAO' daria origem à mensagem encriptada '24312311310021000200 23'. Para obter a mensagem original segue-se o processo inverso.
Para facilitar o desenvolvimento do programa em Python, nas secções seguintes descreve-se duas versões do algoritmo que deverá implementar: a primeira, uma versão simplificada, e a segunda, a versão final, que deverá obter complementando a versão simplificada.

1.1 Versão Simplificada

1.1.1 Geração das Chaves

A geração de chaves consiste em a partir de uma sequência de 25 caracteres, gerar uma tabela 5 × 5 que contem os caracteres da sequência. Por exemplo, a tabela gerada a partir da sequência ('A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', ' ', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'X', 'Z', '.'), onde ' ' representa um espaço em branco, seria:

A	B	C	D	E
F	G	H	I	J
	L	M	N	O
P	Q	R	S	T
U	V	X	Z	.

(2.0 val.) Escreva a função `gera_chave1` que recebe um argumento, `letras`, que consiste num tuplo de 25 caracteres. A função devolve um tuplo de 5 tuplos de caracteres, cada um com 5 elementos, cujos elementos são os caracteres de `letras`, dispostos por linhas. Por exemplo:

```
>>> letras = ('A', 'B', 'C', 'D', 'E', 'F', 'G', \
...           'H', 'I', 'J', ' ', 'L', 'M', 'N', \
...           'O', 'P', 'Q', 'R', 'S', 'T', 'U', \
...           'V', 'X', 'Z', '.')
>>> gera_chave1(letras)
...  (('A', 'B', 'C', 'D', 'E'),
...   ('F', 'G', 'H', 'I', 'J'),
...   (' ', 'L', 'M', 'N', 'O'),
...   ('P', 'Q', 'R', 'S', 'T'),
...   ('U', 'V', 'X', 'Z', '.'))
```

Na função `gera_chave1` considere que o argumento é correcto, isto é, `letras` é um tuplo com exactamente 25 caracteres distintos. Os caracteres possíveis são os caracteres visíveis do código ASCII. Não necessita por isso fazer a gestão de eventuais erros.
Note que a chave pode ser no máximo uma matrix 10×10 visto que usaremos os dígitos de 0 a 9 que identificam as linhas e colunas na encriptação da mensagem. Esta chave permite no entanto codificar os 96 caracteres ASCII visíveis.
Nos restantes exemplos da Secção 1.1, assume-se que o valor da variável `chave` é o tuplo de tuplos com a chave gerada no exemplo anterior.

1.1.2 Encriptação de Mensagens

Para encriptar uma mensagem, usando determinada chave, cada caractere deve ser substituído por uma cadeia de 2 caracteres, correspondendo o primeiro ao número da linha e o segundo ao número da coluna. As linhas e as colunas são numeradas a partir de zero.

(2.5 val.) Escreva a função `obtem_codigo1` que recebe dois argumentos, `car`, consistindo num caractere, e `chave`, consistindo numa chave. A função devolve uma cadeia de 2 caracteres, correspondendo ao código do caractere de acordo com a `chave`. Assuma que o caractere pertence à chave. Por exemplo:

```
>>> obtem_codigo1('Q', chave)
'31'
```

(1.5 val.) Escreva a função `codifica1` que recebe dois argumentos, `cad`, uma cadeia de caracteres, e `chave`, uma chave. A função utiliza a função `obtem_codigo1` e devolve a cadeia de caracteres correspondente à encriptação de `cad`. Assuma que todos os caracteres da cadeia pertencem à chave. Por exemplo:

```
>>> codifica1('FUNDAMENTOS DA PROGRAMACAO', chave)
'1040230300220423342433200300203032241132002200020024'
```

1.1.3 Desencriptação de Mensagens

Para desencriptar uma mensagem, usando determinada chave, basta seguir o processo inverso ao seguido na encriptação.

(2.5 val.) Escreva a função `obtem_car1` que recebe dois argumentos, `cod`, consistindo numa cadeia de 2 dígitos, e `chave`, consistindo numa chave. A função devolve o caractere correspondente ao código `cod`. Assuma que a linha e a coluna de `cod` existem em `chave`. Por exemplo:

```
>>>obtem_car1('31', chave)
'Q'
```

(1.5 val.) Escreva a função `descodifica1` que recebe dois argumentos, `cad_codificada`, uma cadeia de caracteres encriptada, e `chave`, a chave usada na encriptação. A função utiliza a função `obtem_car1` e devolve uma cadeia de caracteres correspondente à mensagem original. Assuma que todos os códigos pertencem à chave. Por exemplo:

```
>>>descodifica1('1040230300220423342433200300203032241132002200020024',\
... chave)
'FUNDAMENTOS DA PROGRAMACAO'
```

1.2 Versão Final

A versão final extende as funcionalidades da versão simplificada. Neste contexto, **deve, sempre que possível, reutilizar na versão final código desenvolvido na versão simplificada.**

1.2.1 Geração das Chaves

A geração de chaves deve agora aceitar sequências com qualquer número de caracteres, e gerar um tuplo de tuplos, em que o último tuplo pode ser mais curto que os restantes. O número de tuplos deve ser a raiz quadrada do menor quadrado perfeito não inferior ao comprimento da sequência.

(2.0 val.) Escreva a função `gera_chave2` que recebe um argumento, `letras`, consistindo num tuplo de caracteres. A função devolve um tuplo de tuplos, cujos elementos são os de `letras`, dispostos por linhas. Por exemplo:

```
>>>letras = ('A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',\
...           'I', 'J', 'L', 'M', 'N', 'O', 'P', 'Q',\
...           'R', 'S', 'T', 'U', 'V', 'X', 'Z')
>>>gera_chave2(letras)
...  (('A', 'B', 'C', 'D', 'E'),
...   ('F', 'G', 'H', 'I', 'J'),
...   ('L', 'M', 'N', 'O', 'P'),
...   ('Q', 'R', 'S', 'T', 'U'),
...   ('V', 'X', 'Z'))
```

Nos restantes exemplos da Secção 1.2, assume-se que o valor da variável `chave` é a chave gerada no exemplo anterior.

1.2.2 Encriptação de Mensagens

Para encriptar uma mensagem deve usar-se o método descrito na Secção 1.1.2.

(1.5 val.) Escreva a função `obtem_codigo2` que recebe dois argumentos, `car`, consistindo num caractere, e `chave`, consistindo numa chave. A função devolve uma cadeia de 2 caracteres, correspondendo ao código do caractere de acordo com `chave`. Se o caractere não pertencer à chave, a função deve devolver 'XX'. Por exemplo:

```
>>>obtem_codigo2('Q', chave)
'30'
>>>obtem_codigo2(':', chave)
'XX'
```

(0.5 val.) Escreva a função `codifica2` que recebe dois argumentos, `cad`, uma cadeia de caracteres, e `chave`, uma chave. A função utiliza a função `obtem_codigo2` e devolve uma cadeia de caracteres, correspondendo à encriptação de `cad`. Por exemplo:

```
>>>codifica2('FUNDAMENTOS DA PROGRAMACAO', chave)
'10342203002104223323 32XX0300XX2431231131002100020023'
```

1.2.3 Desencriptação de Mensagens

Para desencriptar uma mensagem, usando determinada chave, basta seguir o processo inverso ao seguido na encriptação.

(1.5 val.) Escreva a função `obtem_car2` que recebe dois argumentos, `cod`, consistindo numa cadeia de 2 dígitos ou 'XX', e `chave`, consistindo numa chave; a função devolve o caractere correspondente ao código `cod`, ou '?' se `cod` for 'XX'. Por exemplo:

```
>>>obtem_car2('30', chave)
'Q'
>>>obtem_car2('XX', chave)
'?'
```

(0.5 val.) Escreva a função `descodifica2` que recebe dois argumentos, `cad_codificada`, a encriptação de uma cadeia de caracteres, e `chave`, a chave usada na encriptação. A função utiliza a função `obtem_car2` e devolve uma cadeia de caracteres, correspondendo à mensagem original. Por exemplo:

```
>>> descodifica2('10342203002104223323 32XX0300XX2431231131002100020023',\
... chave1)
'FUNDAMENTOS?DA?PROGRAMACAO'
```

2 Avaliação

O projecto é individual e a avaliação do projecto terá duas componentes:

1. **Avaliação automática (16 valores)**;
2. **Avaliação manual (4 valores)**.

Na **avaliação automática**, que avaliará a **execução do programa**, será usado o sistema **Mooshak** que testará o programa usando um conjunto de **testes privados** seguindo as cotações indicadas no enunciado para cada função a desenvolver. Não será disponibilizado qualquer tipo de informação sobre os casos de teste privados utilizados pelo sistema na avaliação do projecto. Estes serão disponibilizados na página da disciplina após a entrega.

Uma semana antes do prazo de entrega serão disponibilizados um conjunto de **testes públicos** que deve usar para testar o funcionamento do programa antes de o submeter para avaliação no Mooshak. No caso dos testes públicos será disponibilizado o input do programa e o respectivo output para que possa detectar eventuais erros semânticos.

Na **avaliação manual**, que avaliará a **qualidade do código** desenvolvido, serão usados os seguintes critérios e cotações:

- Tamanho das funções e repetição de código **(2 valores)**;
- Clareza dos nomes das variáveis e das funções **(1 valor)**;
- Qualidade (e não quantidade) dos comentários **(1 valor)**.

3 Condições de Realização e Prazos

A entrega do 1º projecto será efectuada exclusivamente por via electrónica. Deverá submeter o seu projecto **através do sistema Mooshak** até às **23:59** do dia **4 de Novembro de 2016**. Projectos em atraso não serão aceites seja qual for o pretexto.

Deverá submeter um **único ficheiro com extensão .py** contendo todo o código do seu projecto (implementação das funções pedidas). O ficheiro de código deve conter em **comentário na primeira linha o número e o nome do aluno**.

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

Uma semana antes do prazo de entrega *serão publicadas na página da disciplina as instruções necessárias para a submissão do código no Mooshak. Apenas a partir dessa altura será possível a submissão por via electrónica. Nessa altura serão também fornecidas as credenciais de acesso individuais. Até ao prazo de entrega poderá efectuar o número de entregas que desejar, sendo utilizada para efeitos de avaliação a última entrega efectuada.*

Deverá portanto verificar cuidadosamente que a versão submetida no Mooshak corresponde à versão do projecto que pretende que seja avaliada. Não serão abertas excepções.

Os testes considerados para efeitos de avaliação podem incluir ou não os exemplos disponibilizados no enunciado e/ou nos testes públicos, além de um conjunto de testes adicionais. O facto de um projecto completar com sucesso os exemplos fornecidos no enunciado não implica, pois, que esse projecto esteja totalmente correcto, pois o conjunto de exemplos fornecido não é exaustivo. **É da responsabilidade do programador garantir que o código produzido está correcto**.

Caso o corpo docente considere necessário poderá existir uma discussão oral do trabalho e/ou uma demonstração do funcionamento do programa (decidida caso a caso).

Projectos iguais, ou muito semelhantes, serão penalizados com a reprovação à disciplina. O corpo docente da disciplina será o único juiz o que se considera ou não copiar num projecto. **Será usado um sistema automático de detecção de cópias**.