

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Tarea 1 y 2 unidad 2. Estructuras de Datos y algoritmos II.

Prof. Gerardo Tovar Tapia.

Alumna. Monroy Velázquez Alejandra Sarahí

Grupo 6

- 1) Del libro “python para informáticos”. Leer el capítulo 6, capítulo 7, hacer los ejercicios que se proponen en de dichas sección.

CAPITULO 6

Ejercicio 6.1 Escribe un bucle while que comience en el último carácter de la cadena y haga su recorrido hacia atrás hasta el primer carácter de la misma, mostrando cada letra en una línea separada.

```
1 print("--Ejercicio 6.1--\n")
2 fruta = 'banana'
3 print("Cadena original: ",fruta)
4 i = len(fruta)-1
5 while i >= 0:
6     letra = fruta[i]
7     print(letra)
8     i-=1
9
```

```
--Ejercicio 6.1--
Cadena original: banana
a
n
a
n
a
b
```

Ejercicio 6.2 Dado que fruta es una cadena, ¿qué significa fruta[:]?

```
10
11 print("\n\n--Ejercicio 6.2--\n")
12 print(fruta[:])
13 print("El significado de fruta[:] es
14     la cadena completa.")
15
```

```
--Ejercicio 6.2--
banana
El significado de fruta[:] es la cadena completa.
```

Ejercicio 6.3 Encapsula el código anterior en una función llamada contador, y generalízala, de modo que acepte la cadena y la letra como argumentos.

```
15
16 print("\n\n--Ejercicio 6.3--\n")
17 def contador(palabra,a):
18     contador = 0
19     for letra in palabra:
20         if letra == a:
21             contador = contador + 1
22     print("En la cadena \"{0}\" hay: {1} {2}'s".format
23           (palabra,contador,a))
23 contador(fruta,'a')
```

```
--Ejercicio 6.3--

En la cadena "banana" hay: 3 a's
```

Ejercicio 6.4 Existe un método de cadena llamado count, que es similar a la función que vimos en el ejercicio anterior. Lee la documentación de este método en <https://docs.python.org/2/library/stdtypes.html#string-methods> y escribe una invocación que cuente el número de veces que aparece la letra "a" en 'banana'.

```
25
26 print("\n\n--Ejercicio 6.4--\n")
27 print("Usando el metodo count en ",fruta)
28 print("Count para 'a' = ",fruta.count('a'))
29
```

```
--Ejercicio 6.4--

Usando el metodo count en  banana
Count para 'a' = 3
```

Ejercicio 6.5 Toma el código en Python siguiente, que almacena una cadena: 'cad = 'X-DSPAM-Confidence: 0.8475' Usa find y rebanado de cadenas (slicing) para extraer la porción de la cadena después del carácter punto, y luego usa la función float para convertir la cadena extraída en un numero en punto flotante.

```
30
31 print("\n\n--Ejercicio 6.5--\n")
32 cad = 'X-DSPAM-Confidence: 0.8475'
33 print("Cadena original: ",cad)
34 num =float(cad[slice(cad.find('.'),len(cad),1)])
35 print("El valor floatante extraido: ", num)
36
```

```
--Ejercicio 6.5--

Cadena original: X-DSPAM-Confidence: 0.8475
El valor floatante extraido: 0.8475
```

Ejercicio 6.6 Lee la documentación de los métodos de cadena que está en <https://docs.python.org/2/library/stdtypes.html#string-methods>. Puede que quieras experimentar con algunos de ellos para asegurarte de que comprendes cómo funcionan. `strip` y `replace` resultan particularmente útiles. La documentación utiliza una sintaxis que puede resultar confusa. Por ejemplo, en `find(sub[, start[, end]])`, los corchetes indican argumentos opcionales. De modo que `sub` es necesario, pero `start` es opcional, y si incluyes `start`, entonces `end` es opcional.

```

37
38 print("\n\n--Ejercicio 6.6--\n")
39 str1 = "0000000Cadena de ejemplo0000000";
40 print("Strip\nCadena original: ",str1)
41 print("Despues de usar Strip:",str1.strip( '0' ))
42 str2 = "Los perros y amigos felices cantaban";
43 print("\nReplace\nCadena original: ",str2)
44 print("Despues del replace total:",str2.replace('o','a'))
45 print("Despues del replace parcial:",str2.replace('o','a',2))
46
47

```

--Ejercicio 6.6--

Strip

Cadena original: 0000000Cadena de ejemplo0000000
Despues de usar Strip: Cadena de ejemplo

Replace

Cadena original: Los perros y amigos felices cantaban
Despues del replace total: Las perras y amigas felices cantaban
Despues del replace parcial: Las perras y amigos felices cantaban

NOTA: Los programas del capítulo 6 están contenidos en el archivo `EjerciciosCapitulo6.py`

CAPITULO 7

Ejercicio 7.1 Escribe un programa que lea un fichero e imprima en pantalla su contenido (línea a línea), todo en mayúsculas. La ejecución del programa debería ser algo similar a esto:

`python shout.py`

Introduzca el nombre del fichero: `mbox-short.txt`

`FROM STEPHEN.MARQUARD@UCT.AC.ZA SAT JAN 5 09:14:16 2008`

`RETURN-PATH: <POSTMASTER@COLLAB.SAKAIPROJECT.ORG>`

`RECEIVED: FROM MURDER (MAIL.UMICH.EDU [141.211.14.90]) BY`

`FRANKENSTEIN.MAIL.UMICH.EDU (CYRUS V2.3.8) WITH LMTPA; SAT, 05 JAN 2008 09:14:16 -0500`

```

Ejercicio7.1.py
1 name=input("Introduzca el nombre del archivo: ")
2 try:
3     manf=open(name)
4 except:
5     print("No se pudo abrir el fichero: ",name)
6     exit()
7
8 print("\n>>\n\n")
9 for linea in manf:
10     linea = linea.rstrip()
11     print (linea.upper())
12 manf.close()
13 print("\n<<")
14

```

```

C:\> Símbolo del sistema
El volumen de la unidad C es Windows
El número de serie del volumen es: 8E5B-3C79

Directorio de C:\Users\Alexandra\Desktop

28/10/2017 12:21 p. m. <DIR> .
28/10/2017 12:21 p. m. <DIR> ..
28/10/2017 12:29 p. m. 259 Ejercicio7.1.py
26/10/2017 10:50 a. m. 1,227 EjerciciosCapitulo6.py
28/10/2017 12:29 p. m. 6,374 mbox-short.txt
27/10/2017 12:42 a. m. 6,687,002 mbox.txt
28/10/2017 12:24 p. m. <DIR> Nueva carpeta
26/10/2017 11:03 a. m. 789,267 Tarea4,5,Unidad2.docx
5 archivos 7,484,129 bytes
3 dirs 163,652,476,928 bytes libres

C:\Users\Alexandra\Desktop>python Ejercicio7.1.py
Introduzca el nombre del archivo: mbox-short.txt

>>

FROM STEPHEN.MARQUARD@UCT.AC.ZA SAT JAN 5 09:14:16 2008
RETURN-PATH: <POSTMASTER@COLLAB.SAKAIPROJECT.ORG>
RECEIVED: FROM MURDER (MAIL.UMICH.EDU [141.211.14.90])
BY FRANKENSTEIN.MAIL.UMICH.EDU (CYRUS V2.3.8) WITH LMTPA;
SAT, 05 JAN 2008 09:14:16 -0500
X-SIEVE: CMU SIEVE 2.3
RECEIVED: FROM MURDER ([UNIX SOCKET])
BY MAIL.UMICH.EDU (CYRUS V2.2.12) WITH LMTPA;
SAT, 05 JAN 2008 09:14:16 -0500
RECEIVED: FROM HOLES.MR.ITD.UMICH.EDU (HOLES.MR.ITD.UMICH.EDU [141.211.14.79])
BY FLAWLESS.MAIL.UMICH.EDU ( ) WITH ESMTP ID M05EEFR1013674;
SAT, 5 JAN 2008 09:14:15 -0500
RECEIVED: FROM PAPLOO.UHI.AC.UK (APP1.PROD.COLLAB.UHI.AC.UK [194.35.219.184])
BY HOLES.MR.ITD.UMICH.EDU ID 477F90B0.2DB2F.12494 ;
5 JAN 2008 09:14:10 -0500
RECEIVED: FROM PAPLOO.UHI.AC.UK (LOCALHOST [127.0.0.1])
BY PAPLOO.UHI.AC.UK (POSTFIX) WITH ESMTP ID 5F919BC2F2;
SAT, 5 JAN 2008 14:10:05 +0000 (GMT)
MESSAGE-ID: <200801051412.M05ECIAH010327@NAKAMURA.UITS.IUPUI.EDU>
MIME-VERSION: 1.0
CONTENT-TRANSFER-ENCODING: 7BIT
RECEIVED: FROM PROD.COLLAB.UHI.AC.UK ([194.35.219.182])
BY PAPLOO.UHI.AC.UK (JAMES SMTP SERVER 2.1.3) WITH SMTP ID 899
FOR <SOURCE@COLLAB.SAKAIPROJECT.ORG>;
SAT, 5 JAN 2008 14:09:50 +0000 (GMT)
RECEIVED: FROM NAKAMURA.UITS.IUPUI.EDU (NAKAMURA.UITS.IUPUI.EDU [134.68.220.122])
BY SHMI.UHI.AC.UK (POSTFIX) WITH ESMTP ID A215243002
FOR <SOURCE@COLLAB.SAKAIPROJECT.ORG>; SAT, 5 JAN 2008 14:13:33 +0000 (GMT)
RECEIVED: FROM NAKAMURA.UITS.IUPUI.EDU (LOCALHOST [127.0.0.1])
BY NAKAMURA.UITS.IUPUI.EDU (8.12.11.20060308/8.12.11) WITH ESMTP ID M05ECJVP010329
FOR <SOURCE@COLLAB.SAKAIPROJECT.ORG>; SAT, 5 JAN 2008 09:12:19 -0500
RECEIVED: (FROM APACHE@LOCALHOST)
BY NAKAMURA.UITS.IUPUI.EDU (8.12.11.20060308/8.12.11/SUBMIT) ID M05ECIAH010327
FOR SOURCE@COLLAB.SAKAIPROJECT.ORG; SAT, 5 JAN 2008 09:12:18 -0500
DATE: SAT, 5 JAN 2008 09:12:18 -0500
X-AUTHENTICATION-WARNING: NAKAMURA.UITS.IUPUI.EDU: APACHE SET SENDER TO STEPHEN.MARQUARD@UCT.AC.ZA USING -F
TO: SOURCE@COLLAB.SAKAIPROJECT.ORG

```

Cabe mencionar que el archivo que se va a leer tiene que estar guardado en la misma dirección dónde se encuentra guardado el programa que se corre.

Ejercicio 7.2 Escribe un programa que pida el nombre de un fichero y después lea ese fichero, buscando líneas que tengan la forma:

X-DSPAM-Confidence: 0.8475

Cuando encuentres una línea que comience por “X-DSPAM-Confidence:”, separa esa línea para extraer el número en punto flotante que figure en ella. Cuenta esas líneas y calcula también el total de los valores de probabilidad de spam (spam confidence) de estas líneas. Cuando alcances el final del archivo, muestra en pantalla el valor medio de probabilidad de spam.

Introduzca el nombre del fichero: mbox.txt

Valor medio de probabilidad de spam: 0.894128046745

Introduzca el nombre del fichero: mbox-short.txt

Valor medio de probabilidad de spam: 0.750718518519

Prueba tu programa con los archivos mbox.txt y mbox-short.txt.

```
Ejercicio7.2.py x
1 xdspam=[]
2 cspam=0
3 cemail=0
4
5 name=input("Introduzca el nombre del archivo: ")
6 try:
7     manf=open(name)
8 except:
9     print("No se pudo abrir el fichero: ",name)
10    exit()
11
12 for linea in manf:
13     if linea.startswith('X-DSPAM-Confidence:') :
14         num =float(linea[slice(linea.find('.'),len(linea),1)])
15         xdspam.append(num)
16         cspam+=1
17     if linea.startswith('Received'):
18         cemail+=1
19 manf.close()
20
21 if(cemail!=0):
22     probabilidad=cspam/cemail
23     print("\nEn la bandeja hubo {0} coincidencias en los {1} emails totales.".format(cspam,cemail))
24     print("Valor medio de probabilidad de spam :",probabilidad)
25 else:
26     print("\nLa bandeja esta vacia.")
27
```

```
Simbolo del sistema
C:\Users\Alexandra\Desktop>python Ejercicio7.2.py
Introduzca el nombre del archivo: mbox.txt

En la bandeja hubo 1797 coincidencias en los 16173 emails totales.
Valor medio de probabilidad de spam : 0.11111111111111111

C:\Users\Alexandra\Desktop>python Ejercicio7.2.py
Introduzca el nombre del archivo: mbox-short.txt

En la bandeja hubo 27 coincidencias en los 243 emails totales.
Valor medio de probabilidad de spam : 0.11111111111111111

C:\Users\Alexandra\Desktop>
```

Ejercicio 7.3 Algunas veces, cuando los programadores se aburren o quieren divertirse un poco, añaden un inofensivo Huevo de Pascua (Easter Egg) en sus programas ([es.wikipedia.org/wiki/Huevo_de_pascua_\(virtual\)](https://es.wikipedia.org/wiki/Huevo_de_pascua_(virtual))). Modifica el programa que pide al usuario el nombre del fichero para que imprima un mensaje divertido cuando el usuario escriba el nombre exacto “na na boo boo”. El programa debe comportarse normalmente con todos los demás cheros, tanto los que existan como los que no. A continuación, una muestra de la ejecución del programa:

python egg.py

Introduzca el nombre del fichero: mbox.txt

Hay 1797 líneas subject en mbox.txt

python egg.py

Introduzca el nombre del fichero: missing.tyxt

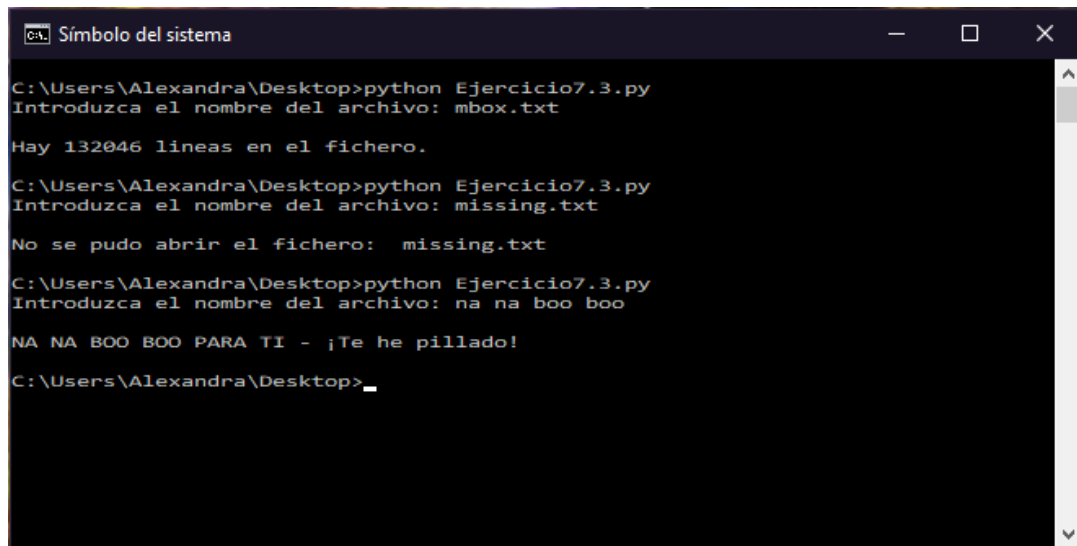
No se pudo abrir el fichero: missing.tyxt

python egg.py

Introduzca el nombre del fichero: na na boo boo

NA NA BOO BOO PARA TI - ¡Te he pillado!

```
Ejercicio7.3.py
1  cont=1
2  name=input("Introduzca el nombre del archivo: ")
3
4  if(name=='na na boo boo'):
5      print("\nNA NA BOO BOO PARA TI - ¡Te he pillado!")
6      exit()
7  try:
8      manf=open(name)
9  except:
10     print("\nNo se pudo abrir el fichero: ",name)
11     exit()
12
13     for linea in manf:
14         cont+=1
15     manf.close()
16
17     print("\nHay {0} líneas en el fichero.".format(cont))
18
19
```



```
C:\Users\Alexandra\Desktop>python Ejercicio7.3.py
Introduzca el nombre del archivo: mbox.txt

Hay 132046 líneas en el fichero.

C:\Users\Alexandra\Desktop>python Ejercicio7.3.py
Introduzca el nombre del archivo: missing.txt

No se pudo abrir el fichero: missing.txt

C:\Users\Alexandra\Desktop>python Ejercicio7.3.py
Introduzca el nombre del archivo: na na boo boo

NA NA BOO BOO PARA TI - ¡Te he pillado!

C:\Users\Alexandra\Desktop>
```

2) Y hacer comentario por capítulo como evidencia de la lectura.

CAPITULO 6

Este capítulo se trató de cadenas, como su mismo título lo dice, una cadena es una secuencia de caracteres y podemos acceder a los caracteres de uno en uno con el operador corchete. La expresión entre corchetes recibe el nombre de índice. El índice indica a que carácter de la secuencia se desea acceder. La función len devuelve el número de caracteres de una cadena. Cabe destacar que las cadenas son inmutables, lo cual significa que no se puede cambiar una cadena existente.

El capítulo señala que las cadenas son un ejemplo de objetos en Python, un objeto contiene tanto datos (la propia cadena en si misma) como métodos, que en realidad son funciones que están construidas dentro de los propios objetos y que están disponibles para cualquier instancia del objeto.

Por ultimo un segmento de una cadena recibe el nombre de rebanada (slice). Seleccionar una rebanada es similar a seleccionar caracteres, el operador [n:m] devuelve la parte de la cadena desde el “n-esimo” carácter hasta el “m-esimo”, incluyendo el primero pero excluyendo el último.

CAPITULO 7

El capítulo 7 trata de ficheros, dice que cuando se desea leer o escribir en un archivo en el disco duro primero debemos abrir el fichero. Al abrir el fichero nos comunicamos con el sistema operativo, que sabe dónde se encuentran almacenados los datos de cada archivo. Cuando se abre un fichero, se está

pidiendo al sistema operativo que lo busque por su nombre y se asegure de que existe.

Un fichero de texto puede ser considerado una secuencia de líneas, para leer un archivo se puede construir un bucle for para ir leyendo y contabilizando cada una de las líneas de un fichero o si el fichero es pequeño se puede leer completo en una cadena usando el método read. Para buscar datos dentro de un fichero, un diseño muy común consiste en ir leyendo el archivo completo, ignorando la mayoría de las líneas y procesando únicamente aquellas que cumplen alguna condición particular. O si queremos buscar una línea que empiece por un sufijo, utilizamos el método de cadena startswith(). Para escribir en un fichero se debe abrir utilizando el modo write "w" como segundo parámetro, si el fichero ya existe, abrirlo en modo escritura eliminará los datos contenidos y escribirá sobre él; si no existe, se creará nuevo. Otra de las cosas que señala el capítulo es la función exit(), utilizada en los catch para hacer finalizar el programa.

3) Del libro que se anexa "Algoritmos y Programación I Con lenguaje Python", leer el capítulo 8, hacer comentario de la lectura, y hacer los programas que se van exponiendo en dicho capítulo.

Ejercicio8.1 - Busca utilizando index e in provistos por Python

```
1 def buscar_con_index(xs,x):
2     if x in xs:
3         return(xs.index(x))
4     else:
5         return -1
6
7 A=[1,4,54,3,0,-1]
8 print("Lista: ",A)
9 x=int(input("Introduzca el elemento que busca saber su posicion: "))
10 b=buscar_con_index(A,x)
11 if(b>=0):
12     print("El elemento se encuentra en la lista en el indice: ",b)
13 else:
14     print("El elemento no se encuentra en la lista")
15     print(b)
16
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
Lista: [1, 4, 54, 3, 0, -1]
Introduzca el elemento que busca saber su posicion: 3
El elemento se encuentra en la lista en el indice: 3
>
Lista: [1, 4, 54, 3, 0, -1]
Introduzca el elemento que busca saber su posicion: 44
El elemento no se encuentra en la lista
-1
>
```


Ejercicio8.2 - Función de búsqueda lineal

```
1 def busqueda_lineal(lista,x):
2     i=0
3     for z in lista:
4         if z==x:
5             return i
6         else:
7             i+=1
8     return -1
9
10 A=[1,4,54,3,0,-1]
11 print("Lista: ",A)
12 x=int(input("Elemento que desea buscar saber su posicion: "))
13 b=busqueda_lineal(A,x)
14 if(b>=0):
15     print("El elemento se encuentra en la lista en el indice: ",b)
16 else:
17     print("El elemento no se encuentra en la lista.")
18     print(b)
19
20
```

```
Python 3.6.1 (default, Dec 2015, 19:03:11)
[GCC 4.8.2] on linux
>
Lista: [1, 4, 54, 3, 0, -1]
Elemento que desea buscar saber su posicion: 44
El elemento no se encuentra en la lista.
-1
>
Lista: [1, 4, 54, 3, 0, -1]
Elemento que desea buscar saber su posicion: 0
El elemento se encuentra en la lista en el indice: 4
>
```

Ejercicio8.3 - Función de búsqueda binaria

```
1 def busqueda_binaria(lista,x):
2     print("Lista",lista)
3     izq=0
4     der=len(lista)-1
5     while izq <= der:
6         medio=int((izq+der)/2)
7         print("DEBUG: Izq:{0} der:{1} medio:{2}".format(izq,der,medio))
8
9         if lista[medio]==x:
10             return medio
11         elif lista[medio]>x:
12             der=medio-1
13         else:
14             izq=medio+1
15     return -1
16
17 def main():
18     lista=input("Dame una lista ordenada([[]] para terminar): ")
19     while lista != '[]':
20         x=input("¿Valor buscado?:")
21         resultado=busqueda_binaria(lista[1:len(lista)-1:2],x)
22         print("Resultado:",resultado)
23         lista=input("Dame una lista ordenada([[]] para terminar): ")
24
25 main()
26
27
```

```
Python 3.6.1 (default, Dec 2015, 19:03:11)
[GCC 4.8.2] on linux
>
Dame una lista ordenada([[]] para terminar): [1,3,5]
¿Valor buscado?: 2
Lista 135
DEBUG: Izq:0 der:2 medio:1
DEBUG: Izq:0 der:0 medio:0
Resultado: -1
Dame una lista ordenada([[]] para terminar): [1,3,5]
¿Valor buscado?: 3
Lista 135
DEBUG: Izq:0 der:2 medio:1
Resultado: 1
Dame una lista ordenada([[]] para terminar): [1,3,5]
¿Valor buscado?: 5
Lista 135
DEBUG: Izq:0 der:2 medio:1
DEBUG: Izq:2 der:2 medio:2
Resultado: 2
Dame una lista ordenada([[]] para terminar): [1,3,5]
¿Valor buscado?: 6
Lista 135
DEBUG: Izq:0 der:2 medio:1
DEBUG: Izq:2 der:2 medio:2
Resultado: -1
Dame una lista ordenada([[]] para terminar): []
¿Valor buscado?: 0
Lista
Resultado: -1
Dame una lista ordenada([[]] para terminar): [1]
¿Valor buscado?: 1
Lista 1
DEBUG: Izq:0 der:0 medio:0
Resultado: -1
Dame una lista ordenada([[]] para terminar): [[]]
>
```

COMENTARIO.

El capítulo nos expone algunos algoritmos de búsqueda, señalándonos los algoritmos de búsqueda lineal, y binaria. Como también de la función `index` proporcionada en python.

Como habíamos visto en clase y en el capítulo vuelve a mencionar, la búsqueda lineal consiste en ir recorriendo mediante un bucle cada elemento dentro de una lista y verificando si el elemento actual es igual al elemento que se busca, no necesariamente la lista tiene que ser ordenada, este algoritmo es útil para el caso de que la lista sea pequeña. El algoritmo de búsqueda binaria si necesita que la lista este ordenada, y una vez esto, procede a utilizar un método que va descartando la mitad de la lista cada vez y verificando si el número a buscar es mayor o menor a la mitad de va a descartar, este algoritmo es útil cuando la lista es demasiado larga, es mucho más eficiente que el anterior; pero siempre hay que tener presentes el mejor y peor caso para aplicarlos.

Nota: Los programas estan contenidos en el zip "grupo6_EDA_ProgramasTarea4,5"