UNIVERSIDAD NACIONAL DE AVELLANEDA

Carrera: Ingeniería en Informática

Materia: Informática

Enunciados de ejercicios de práctica obligatoria y opcional

J. Daniel SABELLA ROSA

El presente trabajo responde al ordenamiento temático de los primeros capítulos del libro "Algoritmos y Programación I con lenguaje Python". Los contenidos se nutrieron inicialmente del aporte de Rosita Wachenchauzer y, adicionalmente, de Andrés Otaduy, Silvina Busto, J. Daniel Sabella Rosa, Juan Pablo Peña, Ricardo Tomsic, Juan Roberto Alunni, Pedro Martin Salaberry, Hugo Pirón y Jorge Riú, bajo la supervisión de Adriana Galli.

Recopilación y reelaboración de enunciados realizada sobre la base de ejercicios propuestos (ciclos lectivos 2013 - 2014) y revisados (ciclo lectivo 2014) por los docentes de la materia Informática, de la carrera Ingeniería en Informática, de la Universidad Nacional de Avellaneda – **Marzo de 2015**.

1. Conceptos básicos: *asignación, secuencia, iteración, estado de variables*

Ejercicio 1.1. Escribir un programa que pregunte al usuario:

a) su nombre, y luego lo salude.

b) dos números y luego muestre el producto.

Ejercicio 1.2. Mostrar el resultado de ejecutar en el intérprete de Python 2, la siguiente secuencia de instrucciones; sacar conclusiones sobre qué sucede y escribirlas:

a) >>> alfa = 10

b) >>> beta = 15

c) >>> gama = alfa + beta

d) >>> print “el valor de alfa es: “, alfa

e) >>> print “el valor de beta es: “, beta

f) >>> print “el valor de gama es: “, gama

g) >>> print "el valor de 'alfa + beta' es: ", alfa + beta

h) >>> alfa = alfa + 100

i) >>> print “el valor de alfa es: “, alfa

j) >>> print “el valor de gama es: “, gama

k) >>> beta = beta * 2

l) >>> print “el valor de beta es: “, beta

m) >>> print “el valor de gama es: “, gama

n) >>> print "el valor de 'alfa + beta' es: ", alfa + beta

o) >>> gama = alfa + beta

p) >>> print “el valor de gama es: “, gama

q) >>> gama = gama / 2

r) >>> print “el valor de gama es: “, gama

s) >>> print "el valor de 'alfa + beta' es: ", alfa + beta

t) >>> gama = gama / 3

u) >>> print “el valor de gama es: “, gama

Ejercicio 1.3. Expresar la situación que se describe a continuación, a través de *asignación* sobre *variables* (Python 2) identificadas con nombres adecuados a su significado:

Bartolomé vende sombreros y paraguas. Tiene 10 paraguas, 5 sombreros y $1000.

a) Escriba instrucciones que permitan ver en pantalla la tenencia inicial de dinero, sombreros y paraguas, y las respectivas variaciones que resultan de la venta de 4 paraguas, a $40 cada uno, y 2 sombreros, a $90 cada uno.

b) Modifique la solución anterior para permitir que sean variables los precios de venta de paraguas y sombreros.

c) Agregue instrucciones que representen y muestren la venta de la mitad (entera) de sombreros y paraguas restantes, con las respectivas variaciones de dinero.

Ejercicio 1.4. Implementar algoritmos que permitan:

a) Obtener el perímetro y área de un rectángulo, dada su base y su altura.

b) Obtener el perímetro y área de un círculo, dado su radio.

c) Dados los catetos de un triángulo rectángulo, obtener su hipotenusa.

d) Dados dos números *n1* y *n2*, indicar la suma, resta, división y multiplicación de ambos. Analizar el resultado, con los pares de números: 5.0 y 2 ; 5 y 2 ; 2 y 5 ; 5 y 0.

Ejercicio 1.5. Mostrar el resultado de ejecutar en el intérprete de Python 2, los siguientes bloques de código, sacar conclusiones sobre qué sucede en cada caso y escribirlas:

a)
```
>>> for  i  in  range( 5 ):
        print  i * i
```

b)
```
>>> for  i  in  range( 1, 6 ):
        print  i,  i / 3,  i / 3.0
```

c)
```
>>> for  i  in  range( 0, 6, 2 ):
        print  i,  "-", 2 ** i
```

d)
```
>>> for  d  in [ 3, 1, 4, 1, 5 ]:
        print  d,
```

Ejercicio 1.6. Implementar algoritmos que resuelvan los siguientes problemas:

a) Dado un número natural n, imprimir su tabla de multiplicar hasta n.

b) Dado un número natural n, imprimir la suma de los naturales desde 1 hasta n.

c) Dado un número natural n, imprimir su factorial < n! = 1 x 2 x 3 x …x (n-1) x n >.

Ejercicio 1.7. Escribir un programa que le pida una palabra al usuario, para luego imprimirla 50 veces, separada por un espacio intermedio, sin forzar cambio de renglón (salto de línea).

Ejercicio 1.8. Para la siguiente secuencia de código Python 2, dados los valores de los identificadores que se enuncian a continuación, complete respectivamente el estado de las variables y la salida que se despliega en pantalla (sólo si ésta ocurre):

n1 = 3
n2 = 5

Instrucción Python	Estado de variables	Salida en pantalla
>>> for x in range (n1, n2): print "Valor ==> ", x * x	n1 → … n2 → … x → …	
	n1 → … n2 → … x → …	
>>> print "Valor final de x: ", x	n1 → … n2 → … x → …	
>>> n2 = n2 / n1	n1 → … n2 → … x → …	
>>> print "Valor de n2: ", n2	n1 → … n2 → … x → …	

2. Programas sencillos (metodología de construcción de programas)

Ejercicio 2.1. Desarrolle los primeros 5 pasos de la metodología de construcción de programas, para el caso que se enuncia a continuación: Escribir un programa que le solicite al usuario una cantidad de pesos (capital), una tasa de interés anual (tasa) y un número entero de años (tiempo), para mostrar, como resultado, el monto final a obtener, de acuerdo a la fórmula:

$$C_n = C \times (1 + t / 100)^n$$

Donde **C** es el capital inicial, **t** es la tasa de interés y **n** es el tiempo en años. Identifique las variables con nombres adecuados a su significado.

Ejercicio 2.2. Ídem anterior: Escribir un programa Python que le solicite al usuario que ingrese el costo (en pesos) de cada uno de los siguientes dispositivos de almacenamiento:

- ✓ Memoria RAM de 4GB para PC
- ✓ Pendrive 16 GB
- ✓ Disco rígido interno 2 TB

Considerando que en el Sistema Internacional de Unidades (Decimal) un Terabyte (TB) equivale a 10^{12} bytes y un Gigabyte (GB) equivale a 10^9 bytes, el programa deberá informar en la pantalla:

a) ¿Cuánto costaría almacenar 1 GB en cada uno de esos dispositivos?

b) ¿Cuánto más cara (en forma relativa) es la memoria RAM que el pendrive?

c) ¿Cuánto más cara (en forma relativa) es la memoria RAM que el disco rígido?

d) ¿Cuánto más caro (en forma relativa) es el pendrive que el disco rígido?

Ejercicio 2.3. Ídem anterior: Escribir un programa que convierta un valor dado en grados Fahrenheit a grados Celsius, considerando que la fórmula para conversión de Celsius a Fahrenheit es:

$$F = 9 / 5 \times C + 32$$

Ejercicio 2.4. Modifique el programa anterior para que genere una tabla de conversión de temperaturas, desde 0° F hasta 120° F, de 10° en 10°.

Ejercicio 2.5. Escribir un programa que imprima todos los números pares comprendidos entre dos números pares solicitados al usuario (considerar incluidos ambos números).

Ejercicio 2.6. Escribir un programa que reciba un número *n* por parámetro e imprima los primeros *n* números triangulares, junto con su índice. Considerar que los números triangulares se obtienen mediante la suma de los números naturales desde 1 hasta *n*. Es decir, si se piden los primeros 5 números triangulares, el programa debería imprimir:

1 - 1
2 - 3
3 - 6
4 - 10
5 - 15

[Nota: hacerlo sin usar la ecuación $\sum_{i=1}^{n} i = n * (n + 1) / 2$]

Ejercicio 2.7. Escribir un programa que imprima por pantalla todas las fichas de dominó, de una por línea y sin repetir.

Opcionales:

Ejercicio 2.8. Modificar el programa anterior para que pueda generar fichas de un juego que puede tener números de 0 a n.

Ejercicio 2.9. Escribir un programa que tome una cantidad *m* de valores ingresados por el usuario, a cada uno le calcule el factorial e imprima el resultado junto con el número de orden correspondiente.

3. Funciones

Ejercicio 3.1. Definir (y documentar) con un nombre apropiado, una función:

a) Que calcule e imprima el valor del número 10 elevado al cubo.

b) Modificar (y documentar) la solución anterior, para que la función calcule el cubo de cualquier número dado y devuelva el resultado al programa principal. (Observe si el nombre de la función sigue siendo apropiado)

c) Modificar la solución anterior, para que la función calcule cualquier potencia dada, de cualquier número dado. Pruebe la función con valores: 2^{0}, 2^{1}, 2^{2}, 2^{3} y $36^{0.5}$. Consultar la documentación mediante la función *help* provista por Python ¿todavía describe adecuadamente lo que hace la función?

Ejercicio 3.2. Definir (y documentar) con un nombre apropiado, una función que:

a) Calcule la cantidad de segundos de un tiempo dado en horas, minutos y segundos, y devuelva el resultado al programa principal.

b) Calcule la cantidad de horas, minutos y segundos de un tiempo dado en segundos, y devuelva el resultado al programa principal.

Ejercicio 3.3. Escribir un programa que lea de teclado dos tiempos expresados en horas, minutos y segundos, los sume y muestre por pantalla el resultado en horas, minutos y segundos. El programa deberá utilizar las funciones del ejercicio anterior.

Ejercicio 3.4. Definir (con nombre apropiado) una función que, dada la cantidad de bytes descargados y el tiempo expresado en minutos y segundos, devuelva la velocidad de descarga expresada en kB/seg (kilobytes por segundo). [Considere el Sistema Internacional de Unidades (Decimal) donde un kilobyte (kB) equivale a 10^3 bytes].

Ejercicio 3.5. Definir una función que muestre en pantalla los números impares desde el 1 hasta un número (inclusive) ingresado por teclado. Pruebe la función con valores: 4, 7 y 1.

Ejercicio 3.6. Definir (y documentar) con un nombre apropiado, una función que reciba dos números, cuyos valores representan la base y la altura de un triángulo, y devuelva como resultado el área respectiva. Pruebe la función con el par (3, 5) y con el par (3, 5.0).

Ejercicio 3.7. Definir (y documentar) una función

a) denominada “numero_pi”, que devuelva como resultado el valor redondeado del número PI: 3.14159. [Utilice el dato *math.pi* del módulo *math* y la función *round(n, d)*]

b) con nombre apropiado, que dado el radio de un círculo, devuelva como resultado el valor del área respectiva. Utilice la función del ejercicio anterior.

c) denominada “factorial”, que dado un natural, devuelva como resultado su factorial.

Opcionales:

Ejercicio 3.8. Definir una función denominada “pinta_cuadro”, que dibuje con el carácter ‘x’ un cuadrado relleno, de lado igual al parámetro. Ej: pinta_cuadro (3)

xxx
xxx
xxx

Ejercicio 3.9. Definir una función denominada “suma_n”, que reciba como parámetro un número natural *n* y devuelva como resultado la suma de naturales desde 1 hasta *n*.

Ejercicio 3.10. Para la siguiente secuencia de código Python, complete respectivamente el estado de las variables y la salida que se despliega en pantalla (sólo si ésta ocurre):

Instrucción Python	Estado de variables	Salida en pantalla

`def f_calc (n):`
`  resulta = n * 2 / 3`
`  return resulta`		

`acum = 1.0`	acum → …..	

`for ix in range (2,5,2):`
`  acum = acum * f_calc(ix)`
`  print ix, "==>", acum`
	ix → …..
	acum → …..	

	ix → ….. acum → …..	
`print "Final:",ix,"y",acum`	ix → ….. acum → …..	

4. Decisiones

Ejercicio 4.1. Definir y documentar:

a) una función denominada “es_par” que, dado un número entero, devuelva un resultado booleano que indique si es, o no es, par. Dé un ejemplo de su invocación.

b) Modificar la solución anterior, para que devuelva como resultado “S” si es par, “N” si es impar o “0” si es cero. Consulte la documentación mediante la función *help*.

c) una función denominada “es_primo” que, dado un número entero, devuelva un resultado booleano que indique si es, o no es, primo. [Un número natural es primo, si sólo es divisible por sí mismo y por 1]

d) una función denominada “val_abs” que, dado un número, devuelva como resultado su valor absoluto. [No utilizar la función abs ( ) provista por Python]

Ejercicio 4.2. Definir una función denominada “mayor_de_3” que, dados tres números, devuelva como resultado el mayor de ellos. Pruebe la función con seis ternas de valores.

Ejercicio 4.3. Escribir un programa que pida al usuario que ingrese dos números naturales, *n1* y *n2*, e imprima en pantalla la secuencia de enteros comprendida entre *n1* y *n2* (incluidos) con la siguiente particularidad: si el número es múltiplo de 3, en lugar del número debe imprimir “TRES”, si es múltiplo de 5, en vez del número debe imprimir “CINCO” y si es múltiplo de 3 y de 5, en lugar del número debe imprimir “TRES Y CINCO”.

Ejercicio 4.4. Definir una función denominada “es_bisiesto” que, dado un número que representa un año, devuelva un resultado booleano que indique si es, o no es, bisiesto. Pruebe eficientemente la función para múltiplos de 4, de 100 y de 400. [Nota: un año es bisiesto si es un número divisible por 4, pero no es divisible por 100, excepto que también sea divisible por 400. Ejemplos: 1984 y 2000 son bisiestos, 1800 no es bisiesto].

Ejercicio 4.5. Definir una función “cant_dias_mes” que, dados dos números, que representan el mes y el año, devuelva como resultado la cantidad de días correspondientes al mes. Invocar la función del enunciado 4.4.

Ejercicio 4.6. Definir una función “fecha_valida” que, dada una fecha en números (día, mes, año), devuelva como resultado si es válida o no. ¿Qué función debería invocar?

Ejercicio 4.7. Definir una función “fecha_siguiente” que, dada una fecha en números (día, mes, año), devuelva como resultado la fecha (día, mes, año) del día siguiente.

Opcionales:

Ejercicio 4.8. Definir una función “matriz_identidad” que reciba como parámetro una dimensión *n*, e imprima la matriz identidad correspondiente a esa dimensión. [Nota: La matriz identidad de dimensión *n* (de *n* filas, por *n* columnas), es la matriz diagonal que tiene valor 1 en cada elemento de su diagonal principal y 0 en el resto de los elementos]. Ejemplo para dimensión 3:

1 0 0
0 1 0
0 0 1

Ejercicio 4.9. Definir una función “nombre_dia” que, dado un número de día de la semana, devuelva como resultado su nombre. Considerar que “domingo” es el 1er. día.

Ejercicio 4.10. Escribir un programa de astrología que pida al usuario que ingrese el número de día y el número de mes correspondientes a su fecha de cumpleaños, e imprima en pantalla el signo del zodíaco al que pertenece. Considere las siguientes fechas: Aries: 21 de marzo al 20 de abril.

Tauro: 21 de abril al 20 de mayo.

Geminis: 21 de mayo al 20 de junio.

Cáncer: 21 de junio al 21 de julio.

Leo: 22 de julio al 22 de agosto.

Virgo: 23 de agosto al 22 de septiembre.

Libra: 23 de septiembre al 22 de octubre.

Escorpio: 23 de octubre al 22 de noviembre.

Sagitario: 23 de noviembre al 21 de diciembre.

Capricornio: 22 de diciembre al 20 de enero.

Acuario: 21 de enero al 19 de febrero.

Piscis: 20 de febrero al 20 de marzo.

Ejercicio 4.11. Definir una función “tira_generala” que, dados cinco números que representan, cada uno, el valor de un dado, devuelva como resultado la combinación obtenida. Considere las siguientes opciones: Generala: todos los números iguales; Póker: 4 números iguales; Full: tres números iguales y los dos restantes iguales entre sí; Escalera: todos los números distintos; Trío: tres números iguales; Par: dos números iguales.

Ejercicio 4.12. Para la siguiente secuencia de código Python, complete respectivamente el estado de las variables y la salida que se despliega en pantalla (sólo si ésta ocurre):

Instrucción Python	Estado de variables	Salida en pantalla

>>> for boole in (True,False):
b1 = boole
b2 = b1
for cant in range(2):
print b1, 'y', b2, '=', b1 and b2
b2 = not b2
	boole → …..
	b1 → …..
	b2 → …..
	cant → …..
	
	b2 → …..	

	boole → ….. b1 → ….. b2 → ….. cant → ….. b2 → …..	
	boole → ….. b1 → ….. b2 → ….. cant → ….. b2 → …..	
	boole → ….. b1 → ….. b2 → ….. cant → ….. b2 → …..	
>>> print b1, 'o', b2, 'es', b1 or b2	b1 → ….. b2 → …..	

5. Ciclos

Ejercicio 5.1. Definir y documentar:

a) una función denominada “prod_en_sumas” que, dados dos números enteros, devuelva como resultado su multiplicación, obtenida por sumas sucesivas. Incluya instrucciones de depuración (debugging) que muestren todas las sumas que se realizan hasta obtener el resultado final. Pruebe el comportamiento de la función cuando el primero de los dos números es negativo. ¿Qué sucede si el número negativo es el segundo?

b) una función denominada “poten_en_prod” que, dados dos números naturales *a* y *n*, devuelva como resultado la potenciación del primero elevado al segundo (a^n), calculando la potencia por multiplicaciones sucesivas. Utilice la función “prod_en_sumas” del ejercicio anterior.

Ejercicio 5.2. Escribir un programa que reciba, una a una, las calificaciones del usuario, preguntando a cada paso si desea ingresar más notas; finalmente, el programa debe imprimir el promedio correspondiente.

Ejercicio 5.3. Escribir un programa:

a) que contenga una contraseña de 4 caracteres inventada, que le pregunte al usuario la contraseña y no le permita continuar hasta que la haya ingresado correctamente.

b) Modificar el programa anterior para que solamente permita una cantidad fija de intentos. Al finalizar, deberá imprimir en pantalla “Eureka” si acertó la clave o, en caso contrario, “Clave incorrecta” y la cantidad de intentos fallidos.

c) Modificar el programa anterior para que después de cada intento agregue una pausa cada vez mayor, utilizando la función *time.sleep(…)* del módulo *time*.

Ejercicio 5.4. Escribir un programa que elija un número al azar, entre 1 y 99, y lo mantenga en secreto (utilice la función *random.randrange (1,100)* del módulo *random*). El programa debe solicitar al usuario que adivine el número. Si el número que ingresa el usuario es mayor, el programa debe responder "Incorrecto, mi número es menor"; si el número ingresado es menor, el programa debe responder "Incorrecto, mi número es mayor". En ambos casos deberá solicitar otro número hasta que el usuario acierte el correcto. Mostrar la cantidad de intentos realizados para adivinar.

Ejercicio 5.5. Definir una función “max_com_div” que, dados dos números *n* y *m*, devuelva como resultado el Máximo Común Divisor entre ambos, implementando el algoritmo de Euclides. Se describen a continuación los pasos de ese algoritmo matemático:

1. Teniendo n y m, se obtiene r, el resto de la división entera de m / n.
2. Si r es cero, n es el MCD de los valores iniciales.
3. Se reemplaza m ← n, n ← r, y se vuelve al primer paso.

Hacer un seguimiento del algoritmo implementado, para los pares de números: (15,9); (9,15); (10,8) y (12,6).

Ejercicio 5.6. Definir y documentar:

a) una función “es_potencia_de_dos” que reciba como parámetro un número natural, y devuelva *True* si el número es una potencia de 2, y *False* en caso contrario.

b) una función que, dados dos números naturales pasados como argumentos (parámetros de la función), devuelva la suma de todas las potencias de 2 que hay en el rango formado por esos números (0, si no hay potencia de 2 entre ellos). Utilice la función “es_potencia_de_dos” del ejercicio anterior.

Ejercicio 5.7. Escribir un programa que le pida al usuario que ingrese una sucesión de números naturales (primero uno, luego otro, y así hasta que el usuario ingrese ’-1’ como condición de salida). Al final, el programa debe imprimir cuántos números fueron ingresados, la suma total de los valores y el promedio.

Opcionales:

Ejercicio 5.8. Definir una función que reciba un número natural e imprima todos los números primos que hay hasta ese número. Utilice la función del ejercicio 4.1.c).

Ejercicio 5.9. Modificar el programa del enunciado 5.4, para que el usuario pueda dar por terminado el juego, ingresando el valor 0.

Ejercicio 5.10. Definir una función “reloj” que, dados tres números naturales que representan la hora, los minutos y los segundos iniciales, imprima en pantalla, a cada segundo, la información horaria actualizada. Considere válidas las horas de 0:00:00 hasta 23:59:59. Utilice la función *time.sleep*(1) del módulo *time*.

Ejercicio 5.11. Para la siguiente secuencia de código Python, complete respectivamente el estado de las variables y la salida que se despliega en pantalla (sólo si ésta ocurre):

Instrucción Python	Estado de variables	Salida en pantalla
`>>> n = 12`	n → ….	

`>>> while n > 0:`
`  if n % 4 == 0:`
`  print "Cuatro"`
`  print 'DEBUG:', n`
`  n = n / 3`
	n → ….	

	n → ….	
	n → ….	
`>>> print 'n es:', n`		

Ejercicio 5.12. Para la siguiente porción de código Python, dado el valor del identificador que se enuncia, complete respectivamente el estado de la variable y la salida que se despliega en pantalla (sólo si ésta ocurre):

c = 1

Instrucción Python	Estado de variables	Salida en pantalla

`>>> while c < 20:`
`  c = c * 2  + 3`
`  print c/4`
`  if c % 4 > 0:`
`  print "R"`
`  print 'x' * (c / 4)`
	c → …	

Complete en hoja aparte, los estados de la variable y las salidas en pantalla, para la ejecución completa del ciclo.

6. Cadenas de caracteres

Ejercicio 6.1. Definir una función “segm_3_txt” que, dados como parámetros una cadena de caracteres y un carácter selector,

a) imprima los tres primeros caracteres de la cadena, si el valor del selector es la letra ‘P’, o los tres últimos caracteres si el valor del selector es ‘U’, o el mensaje ‘Error en el selector’ si el valor del selector no es ‘P’ ni ‘U’.

b) modificar la solución anterior, para que sólo imprima el primero o el último carácter, respectivamente, cuando la longitud de la cadena sea menor que 3.

Ejercicio 6.2. Definir una función que, dada una cadena de caracteres como parámetro,

a) devuelva como resultado la cadena invertida. [Ej: para el argumento ‘¡Hola Undav!’ debería devolver al programa principal ‘!vadnU aloH¡’]

b) imprima la cadena original yuxtapuesta a la cadena invertida. [Ej: para el argumento ‘espejo’ debería imprimir ‘espejoojepse’]

Ejercicio 6.3. Definir una función “intercala_chr” que, dados como parámetros una cadena de caracteres y un carácter, devuelva como resultado la cadena con el carácter insertado entre sus caracteres originales. [Ej: para los argumentos ‘veamos’ y ‘-‘, debería devolver al programa principal ‘v-e-a-m-o-s’]

Ejercicio 6.4. Definir una función “sustituye_chr” que, dados como parámetros una cadena de caracteres *txt* y dos caracteres *c1* y *c2*, devuelva como resultado la sustitución en la cadena *txt*, de todos los caracteres iguales a *c1*, por el carácter *c2*. [Ej: pasados como argumentos el texto ‘mi primer modulo.py’, el carácter ‘ ‘ y el carácter ‘_‘, debería devolver al programa principal ‘mi_primer_modulo.py’]

Ejercicio 6.5. Definir una función ‘oculta_digitos’ que, dada una cadena de caracteres como parámetro, devuelva como resultado la cadena con todos sus dígitos reemplazados por el carácter ‘*’. [Ej: para el argumento ‘su clave es: 1540’, debería devolver al programa principal ‘su clave es: ****’]

Ejercicio 6.6. Definir una función ‘separa_miles’ que, dada una cadena que contiene un largo número entero, devuelva como resultado la cadena con las separaciones de miles incluidas en el número. [Ej: para el argumento ‘1234567890’ debería devolver al programa principal ‘1.234.567.890’]

Ejercicio 6.7. Definir una función que, dadas dos cadenas de caracteres como parámetros, devuelva como resultado la cadena que sea anterior en orden alfabético. [Ej: para los argumentos 'kde' y 'gnome' debería devolver al programa principal 'gnome']

Opcionales:

Ejercicio 6.8. Definir una función 'letras_iniciales' que, dada una cadena de caracteres, devuelva como resultado las primeras letras de cada palabra de la cadena. [Ej: para el argumento 'Universal Serial Bus' debería devolver al programa principal 'USB']

Ejercicio 6.9. Definir (y documentar) una función 'inicial_mayuscula' que, dada una cadena de caracteres, devuelva como resultado dicha cadena con la primera letra de cada palabra en mayúsculas. [Ej: para el argumento 'república argentina' debería devolver al programa principal 'República Argentina'].
Utilice la función *ord(…)*, que devuelve el número ordinal de un carácter, y la función *chr(…)*, que devuelve el carácter correspondiente a un ordinal del rango [0,256]. Ejemplos: *ord ('a')* es 97 y *chr (65)* es 'A'.

Ejercicio 6.10. Definir una función 'sin_vocales' que, dada una cadena de caracteres, devuelva la cadena sin las letras vocales. [Ej: para el argumento 'República Argentina' debería devolver al programa principal 'rpblc rgntn'].

Ejercicio 6.11. Definir una función 'es_palindromo' que, dada una cadena de caracteres, devuelva un resultado booleano que indique si se trata de un palíndromo. [Ej: 'arroz a la zorra' es palíndromo, porque se lee igual de izquierda a derecha que de derecha a izquierda].

Ejercicio 6.12. Definir una función 'es_subcadena' que, dadas dos cadenas de caracteres, devuelva un resultado booleano que indique si la primera cadena está contenida en la segunda. [Ej: la cadena 'bandera' es subcadena de 'abanderado'.]

Ejercicio 6.13. Definir una función 'binario_a_decimal' que reciba una cadena de unos y ceros (es decir, un número en representación binaria) y devuelva como resultado el valor decimal correspondiente. [Ej: para el argumento '0111' debería devolver al programa principal el valor 7, para '1000' el valor 8, etc.]

Ejercicio 6.14. Definir una función 'cuenta_letra' que, dados como parámetros un carácter y una cadena de caracteres, devuelva como resultado la cantidad de veces que se encuentra ese carácter en la cadena. [Ej: en 'república argentina' hay 3 'a'.]

Ejercicio 6.15. Para las siguientes secuencias de código Python, complete respectivamente el estado de las variables y la salida que se despliega en pantalla (sólo si ésta ocurre):

a)

Instrucción Python	Estado de variables	Salida en pantalla

`tope = 8`
`mensaje = 'triangulo'`
`while tope > 0:`
`  print mensaje[:tope]`
`  tope = tope - 3`
 	
`print mensaje[tope]`
	`tope` → …
	`mensaje` → …
	
	…	

b)

Instrucción Python	Estado de variables	Salida en pantalla

`n = 3`
`argum = "canibales"`
`largo = len (argum)`
`while largo > 0:`
`  st = argum[-largo:]`
`  resu = ""`
`  for i in range (0,len(st),n):`
`  resu = resu + st[i]`
`  print st, '>', resu`
`  largo = largo - n`
 	
`print "Final:", st`
	`n` → …
	`argum` → …
	`largo` → …
	`st` → …
	`resu` → …
	`i` → …
	…	

7. Tuplas y Listas

Ejercicio 7.1. Definir una función

a) que reciba una tupla *tup* con nombres, y para cada nombre imprima el mensaje "Estimado <nombre>, vote por mí ".
b) que reciba una tupla *tup* con nombres, una posición de origen *p* y una cantidad *n*, e imprima el mensaje anterior para los *n* nombres que se encuentran a partir de la posición *p*.
c) Modificar la función del caso a) para que distinga el género del destinatario, considerando que *tup* es una tupla integrada por tuplas de la forma (nombre, género). [El valor 'M' denota género masculino y el valor 'F', femenino]

Ejercicio 7.2. Definir una función 'coincide_domino', que reciba dos tuplas que representan fichas de dominó y devuelva un resultado booleano que indique si encajan o no. [Ej: las fichas (3, 4) y (4, 1) encajan, porque coinciden en el número 4. Ídem las fichas (4, 4) y (5, 4)]

Ejercicio 7.3. Definir una función 'es_tupla_ordenada', que reciba una tupla de elementos y devuelva un resultado booleano que indique si se encuentran ordenados de menor a mayor.

Ejercicio 7.4. Definir una función que reciba dos listas, que representan conjuntos, y

a) devuelva como resultado una lista que incluya, sin repeticiones, los elementos comunes a ambas (Intersección).
b) devuelva como resultado una lista que incluya, sin repeticiones, los elementos que pertenezcan a una u otra lista (Unión).

Ejercicio 7.5. Definir una función que, dada una lista de números enteros,

a) devuelva como resultado una lista que incluya solamente los que sean primos.
b) devuelva como resultado una lista con el factorial de cada uno de esos números.
c) devuelva como resultado una tupla de dos elementos, compuesta por la sumatoria y el promedio de los números de la lista.

Ejercicio 7.6. Definir una función que, dada una lista de números enteros y un entero *k*,

a) devuelva como resultado una lista que incluya sólo los que sean múltiplos de *k*.
b) devuelva como resultado una lista compuesta por tres listas: la primera con los mayores a *k*, la segunda con los iguales a *k* y la tercera con los menores.

Ejercicio 7.7. Definir una función 'lista_de_minimos' que, dadas dos listas de igual longitud, compuestas por números, devuelva como resultado una nueva lista que contenga en cada posición el menor entre los elementos correspondientes de ambas listas.

Ejercicio 7.8. Definir una función que reciba una lista de tuplas, cada una de la forma (apellido, nombre, inicial_segundo_nombre), y devuelva como resultado una lista de cadenas de caracteres, cada una de las cuales contenga primero el nombre, un espacio, luego la inicial con un punto, un espacio y luego el apellido.

Ejercicio 7.9. Definir una función que, dada como parámetro una lista,

a) devuelva como resultado una nueva lista con los mismos elementos, pero en orden invertido. [Ej: para el argumento [‘Di’, ‘buen’, ‘dia’, ‘a’, ‘papi’], debería devolver al programa principal [‘papi’, ‘a’, ‘dia’, ‘buen’, ‘Di’], sin modificar la lista original.]

b) invierta la lista (sin usar listas auxiliares) y la devuelva modificada al programa principal. ¿Es necesaria la instrucción *return* en la función?

Opcionales:

Ejercicio 7.10. Definir una función

a) ‘suma_vectores’ que, dadas dos tuplas que representan vectores, devuelva como resultado la tupla que representa su suma. [La suma de vectores (u1, u2, u3) y (v1, v2, v3), se puede calcular como el vector que resulta de sumar sus componentes homólogas: (u1 + v1, u2 + v2, u3 + v3)]

b) ‘producto_escalar’ que, dadas dos tuplas que representan vectores, devuelva como resultado el valor de su producto escalar. [El producto escalar de los vectores (u1, u2, u3) y (v1, v2, v3), se puede calcular como el valor numérico que resulta de sumar los productos de las componentes homólogas: u1 * v1 + u2 * v2 + u3 * v3]

Ejercicio 7.11. Definir una función ‘tanto_envido’ que, dadas dos tuplas que representan naipes españoles de la forma (palo, numero), devuelva como resultado el valor de los puntos del envido en el juego del truco. [Si ambos palos son iguales, el envido se calcula con la suma del “valor para envido” de los números numero1 y numero2, a lo cual se suman 20 puntos. Si ambos palos difieren, el envido se calcula con el mayor “valor para envido” entre numero1 y numero2. El valor para envido de las figuras (sota, caballo o rey) es cero, mientras que para el resto de los naipes españoles, es el propio número.]

Ejercicio 7.12. Definir una función ‘encera_lista’ que, dadas una tupla compuesta por unos y ceros, y una lista compuesta por números, ambas de igual longitud, modifique la lista, de modo que contenga valor cero en cada posición correspondiente a un cero de la tupla. [Ej: sean los argumentos (1, 0, 0, 1, 0) y [123, 234, 345, 456, 567], la lista modificada [123, 0, 0, 456, 0] debería ser devuelta al programa principal]

8. Anexo a capítulo 7: Seguimiento de estado de variables (Listas)

Para las siguientes porciones de código Python, dados los valores de los identificadores que se enuncian en cada caso, complete respectivamente el estado de las variables y la salida que se despliega en pantalla (sólo si ésta ocurre):

Ejercicio 8.1. Sean los valores de los identificadores:

lis1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

lis2 = [[10, 20, 30], [40, 50, 60], [70, 80, 90]]

lis3 = [ ]

Instrucción Python	Estado de variables

>>> if len(lis1) == len(lis2) :
  tope = len(lis1)
else:
  tope = 0
	
	tope → …

>>> for f in range (tope) :
  fila = [ ]
  for c in range(tope) :
  fila.append(lis1 [f] [c] + lis2 [f] [c])
  lis3.append (fila)
	f → …
	fila → …
	c → …
	fila → …
	lis3 → …

Complete en hoja aparte, el estado de las variables para la ejecución completa del ciclo.

Ejercicio 8.2. Sean los valores de los identificadores:

luno = [ ]

ldos = luno

Instrucción Python	Estado de variables	Salida en pantalla

>>> `for i in range(3):`
`  luno.append(i)`
`  print luno`
`  ldos.insert(0,i)`
`  print ldos`
	luno → …
	ldos → …	

Complete en hoja aparte, el estado de las variables y las salidas en pantalla, para la ejecución completa del ciclo.

9. Diccionarios

Ejercicio 9.1. Definir una función que reciba una cadena y que devuelva como resultado

a) un diccionario, que dé cuenta de la cantidad de apariciones de cada carácter en la cadena.
   Por ejemplo, si recibe "el mejor ejemplo"
   debiera resultar el diccionario: { ‘e’ : 4, ‘l’ : 2, ’ ’ : 2, ’m’ : 2, ‘j’ : 2, ‘o’ : 2, ‘r’ : 1, ‘p’ : 1 }

b) un diccionario, en donde las claves sean las palabras de la cadena y los valores, la respectiva cantidad de apariciones de la palabra en la cadena.
   Por ejemplo, si recibe "el mejor ejemplo es el segundo"
   debiera resultar: { ‘el’ : 2, ‘mejor’ : 1, ’ ejemplo’ : 1, ‘es’ : 1, ‘segundo’ : 1 }

Ejercicio 9.2. Definir una función que reciba una lista de tuplas y que devuelva como resultado un diccionario en donde las claves sean los primeros elementos de las tuplas, y los valores, una lista con los segundos elementos respectivos.

Por ejemplo, dada la lista: [(´Hola´,´don Pepito’), (‘Hola’, ‘don Jose’), (‘Buenos’, ‘dias’)]
Debiera resultar el diccionario: { ‘Hola’ : [‘don Pepito’, ‘don Jose’], ‘Buenos’ : [‘dias’] }

Ejercicio 9.3. Escribir un programa que vaya solicitando al usuario que ingrese nombres a una agenda implementada con un diccionario.

a) Si el nombre se encuentra en la agenda, debe mostrar el número de teléfono y, opcionalmente, permitir modificarlo si no es correcto.
b) Si el nombre no se encuentra, debe permitir ingresar el teléfono correspondiente.

El usuario puede utilizar la cadena “ * “ para salir del programa.

Ejercicio 9.4. Definir una función que reciba un texto y que devuelva como resultado un diccionario, donde a cada carácter presente en el texto se le asocie como valor la cadena más larga en la que se encuentra ese carácter.

Ejercicio 9.5. Definir una función “costo_total”, que reciba como parámetros dos diccionarios, a saber:

1. Listado de precios: con artículos y sus respectivos precios.
2. Inventario: con artículos y sus respectivas existencias (cantidad disponible)

y que devuelva como resultado, cuánto se recaudaría por vender todos los artículos del inventario.

Opcionales:

Ejercicio 9.6 Dado el siguiente diccionario, cuyos elementos asocian a cada "provincia" argentina las respectivas cantidades de habitantes y de km^2 de superficie (en ese orden):

dic_censo = { 'Buenos Aries' : (15625100, 307571), 'Mendoza' : (1739000, 148827), 'Tucuman' : (1448200, 22524), … }

a. Eliminar del diccionario los datos correspondientes a la provincia "Rioja", si existe, y asociarlos a una nueva provincia "La Rioja".

b. Recorrer el diccionario mostrando, en cada renglón de pantalla, el nombre de la provincia, la superficie y la densidad poblacional (cantidad de habitantes por km2).

c. ¿Cómo ordenaría alfabéticamente los resultados por provincia?

Ejercicio 9.7 Considerando los diccionarios del enunciado 9.5
Definir una función "factura_venta" que reciba dos diccionarios y una lista de tuplas (que representa un Pedido de Compras) de la forma artículo y cantidad a comprar, y que devuelva como resultado el costo total de la factura (que resulta de recorrer la lista de compras, sumando los precios de cada artículo multiplicado por la cantidad). Además, la función deberá imprimir los pedidos de los artículos que no se encuentren disponibles (sin stock) o que no existan en el inventario.

Ejercicio 9.8 Definir una función "reparte_letras" que reciba una tupla compuesta por letras distintas (sin repeticiones) y una cadena de caracteres compuesta por palabras, y que devuelva como resultado una lista integrada por una lista de letras y un diccionario. La lista de letras debe contener las letras de la tupla que no pertenecen a palabra alguna de la cadena de caracteres. Las letras de la tupla que pertenecen a palabras de la cadena, deben ser clave en el diccionario, teniendo como valor la lista de palabras que la contienen.

Por ejemplo, si recibe los siguientes argumentos:
tuplita = ('a', 'j', 'k', 's') cadenita = "este texto tiene muchas letras"
debiera resultar la lista:
[['j' , 'k'] , { 'a' : ['muchas', 'letras'] , 's' : ['este', 'muchas', 'letras'] }]