Universidad Nacional de San Luis

Área de Programación y Metodologías de Desarrollo de Software

Ingeniería en Informática – Ingeniería en Computación

Programación II

Práctico N° 8

Lenguaje Python: Tipos, Sentencias, Funciones y Archivos

Ejercicio 1: Definir variables que permitan almacenar literales de los siguientes tipos:

a) Entero.
b) Flotante.
c) Boolean.
d) String.
e) Complejo.

Ejercicio 2: Escribir un programa que convierta:

a) Un entero a un flotante.
b) Un flotante a un string.
c) Un string a un entero.

Ejercicio 3: Escribir un programa que permita que el usuario ingrese dos números enteros y realice las siguientes operaciones: división entera, división con decimales, módulo, potencia.

Ejercicio 4: Explicar y ejemplifique la diferencia entre objetos mutables e inmutables en Python.

Ejercicio 5: Escribir un programa que inicialice una variable con un string ingresado por el usuario y luego realice las siguientes operaciones:

a) Imprimir el segundo carácter del string.
b) Imprimir los caracteres comprendidos entre las posiciones 2 y 5.
c) Elimine los espacios en blanco que están al comienzo o al final si los hubiere.
d) Imprima la longitud del string.
e) Pase a minúscula todo el string.
f) Pase a mayúscula todo el string.
g) Reemplace una letra por otra (las letras son ingresadas por el usuario).
h) Determine si un string ingresado por el usuario es un substring de un string almacenado en una variable.

Ejercicio 6: Dada la siguiente lista frutas = [*"manzana", "banana", "frutilla"*] realice las siguientes tareas:

a) Imprima el segundo ítem.
b) Cambie el valor de *'manzana'* por *'kiwi'*.
c) Incorpore *'naranja'* al final de la lista.
d) Incorpore *'limón'* como segundo ítem de la lista frutas.
e) Elimine *'banana'* de la lista frutas.

Ejercicio 7: Explicar y ejemplifique el tipo de dato lista. Describe el conjunto de métodos predefinidos.

Ejercicio 8: Realizar un programa que permita al usuario ingresar números impares, los almacene y los imprima por pantalla. El programa finaliza cuando el usuario ingresa un número par.

Ejercicio 9: Desarrollar un programa que defina dos listas y devuelva *True* si tienen al menos 1 miembro en común o devuelva *False* de lo contrario.
Nota: debes usar la sentencia de iteración for anidada.

Ejercicio 10: Escribir un programa que permita crear una lista de palabras. Para ello, el programa tiene que pedir un número y luego solicitar esas palabras para crear la lista. Por último, el programa tiene que mostrar por pantalla la lista.

- Ejemplo 1:
  - Cuantas palabras tiene la lista? 3
  - Palabra 1: Tengo
  - Palabra 2: mucha
  - Palabra 3: sed
  - La lista creada es: ['Tengo', 'mucha', 'sed']
- Ejemplo 2:
  - Cuantas palabras tiene la lista? 0
  - Error: la lista debe tener como mínimo un elemento

Ejercicio 11: Escribir un programa que permita crear dos listas de enteros y que a continuación elimine de la primera lista los enteros de la segunda lista. Si hay elementos de la segunda lista que no están en la primera, los debe pasar por alto.
Por ejemplo:

Lista 1	Lista 2	Resultado
[1,5,12,7,9,6,5]	[1,5]	[12,7,9,6]
[1,5,12,7,9,6,5]	[1,5,4]	[12,7,9,6]
[1,5,12,7,9,6,5]	[]	[1,5,12,7,9,6,5]
[1,5,12,7,9,6,5]	[1,5,12,7,9,6,5,8,2]	[]

Ejercicio 12: Escribir un programa que determine si una cadena ingresada por el usuario es un palíndromo.

Ejercicio 13: Desarrollar un programa que reciba una cadena y retorne la misma al revés. Por ejemplo, para la cadena "*Esto*", deberá mostrar: "*otsE*". Realice el programa principal y la correspondiente invocación a la función.

Ejercicio 14: Escribir un programa que solicite al usuario un texto e imprima por pantalla el mismo texto pero con cada palabra invertida. Por ejemplo, para el texto: "*Esto es una prueba*", debe imprimir "*otsE se anu abeurp*".

Ejercicio 15: Escribir un programa que obtenga un número aleatorio secreto, y luego permita al usuario ingresar números y le indique sin son menores o mayores que el número a adivinar, hasta que el usuario

ingrese el número correcto.
Nota: usa la función randint() del módulo random.

Ejercicio 16: Construir un programa que dada una tupla de números enteros imprima por pantalla la posición del número mayor y la posición del número menor.

Ejercicio 17: Escribir una función que reciba una tupla con nombres, y para cada nombre imprima el mensaje "*Estimado, vote por mi*".

Ejercicio 18: Modificar la función anterior para que tenga en cuenta el género del destinatario, para ello, deberá recibir una tupla de tuplas, conteniendo el nombre y el género.

Ejercicio 19: Dar un ejemplo de uso del método get para recuperar un valor de un diccionario.

Ejercicio 20: Desarrollar un programa que solicite un número entero *n* y genere un diccionario de los cuadrados correspondientes a los primeros n enteros y posteriormente lo imprima. Es decir, las entradas del diccionario tendrán la forma (*i, i*i*) donde *0<i<n+1*.
Por ejemplo, si *n* es *4*, el diccionario resultante será {*1:1, 2:4, 3:9, 4:16*}

Ejercicio 21: Desarrollar un programa que construya una agenda telefónica mediante un diccionario con nombres de personas y sus números de teléfono. Posteriormente el usuario puede consultar dicha agenda, ya sea, ingresando el nombre de la persona: el programa debe mostrar el/los números de teléfonos asociados o ingresando un determinado número de teléfono: el programa debe mostrar el usuario al cual pertenece dicho número.

Ejercicio 22: Crear un diccionario que permita almacenar los datos de una persona. Por cada persona se desea registrar la siguiente información: DNI, Nombre, Domicilio y Edad. Ingrese en dicho diccionario cuatro personas.

Ejercicio 23: Escribir un programa que dado un diccionario permita recuperar las claves que dicha estructura de datos contiene.

Ejercicio 24: Definir una función que reciba como parámetros una lista de palabras y retorne como resultado la palabra más larga. Realice el programa principal y la correspondiente invocación a la función.

Ejercicio 25: Desarrollar una función que reciba una lista de enteros y permita "empaquetar" las repeticiones consecutivas de valores en dicha lista mediante una tupla (*<valor>, <cantidad- de-valores>*). Para esto se construye una nueva lista que contenga las nuevas tuplas generadas. Por ejemplo, si la lista inicial es [*1,1,1,3,5,1,1,3,3*], la función debe retornar [*(1,3),(3,1),(5,1),(1,2),(3,2)*]. Realice el programa principal y la correspondiente invocación a la función.

Ejercicio 26: Desarrollar un programa que valide si una dirección de correo electrónico es válida. Es decir si la cadena tiene el siguiente formato *<usuario>@<dominio>.<extension>*. Utilice para *<dominio> y <extensión>* listas con posibles valores, es decir, para *<dominio>*, los valores posibles serían *['gmail','hotmail','outlook'...]*. Si la dirección es válida debe imprimir por pantalla "*Dirección Válida*" caso contrario deberá imprimir "*Dirección Inválida*".

Ejercicio 27: Escribir una función que reciba una lista de tuplas, y que devuelva un diccionario en donde las claves sean los primeros elementos de las tuplas, y los valores una lista con los segundos. Por ejemplo: Para la lista [*('Pedro', 2664889665), ('Pedro', 2665366998), ('Juan', 2657503689)*]
Deberá mostrar: {*'Pedro': [2664889665,2665366998], 'Juan': [2657503689]*}

Ejercicio 28: Escribir un programa que le pida un texto al usuario, y realice una estadística sobre cuantas veces aparece cada carácter. El programa debe mostrar el resultado en orden lexicográfico. Usar diccionarios. Por ejemplo, para "*nosotros no somos como los orozco, yo los conozco, son ocho los monos*", deberá mostrar:

: 12
, : 2
c : 5
h : 1
l : 3
m: 3
n : 5
o : 23
r : 2
s : 9
t : 1
y : 1
z : 2

Ejercicio 29: Definir una función que reciba como parámetro un diccionario y lo convierta a una lista de tuplas. Es decir, la función retornará una lista que contendrá, por cada entrada en el diccionario, una tupla. *Nota:* el diccionario puede tener diccionarios anidados.

Ejercicio 30: Escribir una función que reciba un texto y una longitud y devuelva una lista de cadenas de como máximo esa longitud. Las líneas deben ser cortadas correctamente en los espacios (sin cortar las palabras).

Ejercicio 31: Escribir una función llamada mapear, que reciba una función y una lista y devuelva la lista que resulta de aplicar la función recibida a cada uno de los elementos de la lista recibida.

Ejercicio 32: Escribir un programa que dado un diccionario de personas como el descrito en el ejercicio 21 permita recuperar la información de una persona a partir de un dato ingresado por el usuario.

Ejercicio 33: Escribir un programa que dado un diccionario permita cambiar el valor de una clave ingresada por el usuario.

Ejercicio 34: Construya un programa que dada una lista de números enteros *l* genere una lista *v* que contiene tantos ceros como lo indica la longitud de *l*. Luego el programa debe crear un diccionario donde el primer elemento de *l* es la clave y el primer elemento de *v* es el valor, el segundo elemento de *l* es la clave y el segundo elemento de *v* es el valor y siguiendo.

Ejercicio 35: Construya un programa que obtenga y muestre por pantalla el directorio actual.

Ejercicio 36: Cree un programa que dentro del directorio actual cree otra carpeta de nombre "*ejercicios*".

Ejercicio 37: Construya un programa que dado un path, liste todos los archivos que contiene la carpeta.

Ejercicio 38: Construya un programa que verifique si un archivo (nombre ingresado por teclado) existe; en caso verdadero muestra su tamaño, en caso falso muestra un cartel informando que no existe dicho archivo.

Ejercicio 39: Construya un programa que dado un nombre de archivo imprima por pantalla su contenido.

Ejercicio 40: Construya un programa que cree un archivo y escriba dentro del mismo el nombre de la materia que cursa.

Ejercicio 41: Construya un programa que dada una lista de nombres los guarde en un archivo.

Ejercicio 42: Construya una función que dado un nombre de archivo y un diccionario de personas (ejercicio 21) guarde la información contenida en el archivo.

Ejercicio 43: Construya un programa que además de la funcionalidad planteada en el ejercicio 41 permita leer los datos almacenados de personas y los almacene en un diccionario. Conjuntamente, debe permitir cargar, modificar y eliminar nuevas personas del diccionario.