Actividades III: Strings, Listas y Diccionarios

Métodos del objeto String

Actividad 54

Crear un módulo para validación de nombres de usuarios. Dicho módulo, deberá cumplir con los siguientes criterios de aceptación:

- El nombre de usuario debe contener un mínimo de 6 caracteres y un máximo de 12
- El nombre de usuario debe ser alfanumérico
- Nombre de usuario con menos de 6 caracteres, retorna el mensaje “El nombre de usuario debe contener al menos 6 caracteres”
- Nombre de usuario con más de 12 caracteres, retorna el mensaje “El nombre de usuario no puede contener más de 12 caracteres”
- Nombre de usuario con caracteres distintos a los alfanuméricos, retorna el mensaje “El nombre de usuario puede contener solo letras y números”
- Nombre de usuario válido, retorna True

Actividad 55

Crear un módulo para validación de contraseñas. Dicho módulo, deberá cumplir con los siguientes criterios de aceptación:

- La contraseña debe contener un mínimo de 8 caracteres
- Una contraseña debe contener letras minúsculas, mayúsculas, números y al menos 1 carácter no alfanumérico
- La contraseña no puede contener espacios en blanco
- Contraseña válida, retorna True
- Contraseña no válida, retorna el mensaje “La contraseña elegida no es segura”

Actividad 56

Crear un módulo que solicite al usuario el ingreso de un nombre de usuario y contraseña y que los valide utilizando los módulos generados en los dos ejercicios anteriores.

Ayuda: para contar la cantidad de caracteres de una cadena, en Python se utiliza la función incorporada: len(cadena)

Listas y Tuplas

Una **lista** *es una colección ordenada de valores. Puede contener cualquier cosa y pueden mezclarse diferentes tipos de datos dentro de la misma.*

Una **tupla** *es una secuencia de valores agrupados. Sirve para agrupar, como si fueran un único valor, varios valores que, por su naturaleza, deben ir juntos. Las tuplas son inmutables: una tupla no puede ser modificada una vez que ha sido creada.*

En el conjunto de las funciones integradas de Python, podemos encontrar dos funciones que nos permiten convertir listas en tuplas y viceversa. Estas funciones pueden ser muy útiles cuando, por ejemplo, una variable declarada como tupla, necesita ser modificada en tiempo de ejecución, para lo cual, debe convertirse en una lista puesto que las tuplas, son inmutables. Lo mismo sucede en el caso contrario: una variable que haya sido declarada como lista y sea necesario convertirla en una colección inmutable.

Actividad 57

Sin usar el ordenador, indica cuál es el resultado y el tipo de las siguientes expresiones. Luego verifica las respuestas con la consola o el IDE:

```
>>> a = [5, 1, 4, 9, 0]
>>> b = range(3, 10) + range(20, 23)
>>> c = [[1, 2], [3, 4, 5], [6, 7]]
>>> d = ['perro', 'gato', 'jirafa', 'elefante']
>>> e = ['a', a, 2 * a]
```

1. a[2]
2. b[9]
3. c[1][2]
4. e[0] == e[1]
5. len(c)
6. len(c[0])
7. len(e)
8. c[-1]
9. c[-1][+1]
10. c[2:] + d[2:]
11. a[3:10]
12. a[3:10:2]
13. d.index('jirafa')
14. e[c[0][1]].count(5)
15. sorted(a)[2]
16. complex(b[0], b[1])

Actividad 58

Sin usar el ordenador, indica cuál es el resultado y el tipo de las siguientes expresiones. Luego verifica las respuestas con la consola o el IDE:

```
>>> a = (2, 10, 1991)
>>> b = (25, 12, 1990)
>>> c = ('Donald', True, b)
>>> x, y = ((27, 3), 9)
>>> z, w = x
>>> v = (x, a)
```

1. a < b
2. y + w
3. x + a
4. len(v)
5. v[1][1]
6. c[0][0]
7. z, y
8. a + b[1:5]
9. (a + b)[1:5]
10. str(a[2]) + str(b[2])
11. str(a[2] + b[2])
12. str((a + b)[2])
13. str(a + b)[2]

Actividad 59

Escribe un programa que pregunte al usuario cuántos datos ingresará, a continuación, le pida que ingrese los datos uno por uno, y finalmente entregue como salida cuántos de los datos ingresados son mayores que el promedio.

```
Cuantos datos ingresara? 5
Dato 1: 6.5
Dato 2: 2.1
Dato 3: 2.0
Dato 4: 2.2
Dato 5: 6.1
2 datos son mayores que el promedio
```

Diccionarios

Hay diferentes formas de iterar sobre diccionarios:

Iterar sólo sobre *llaves*:

```
for k in telefonos.keys():
   print k
```

Para iterar sobre los *valores*:

```
for v in telefonos.values():
   print v
```

Para iterar sobre las *llaves* y los *valores* simultáneamente:

```
for k, v in telefonos.items():
   print 'El telefono de', k, 'es', v
```

```
for clave, valor in diccionario.iteritems():
   print "El valor de la clave %s es %s" % (clave, valor)
```

Actividad 60

Escriba la función `contar_letras(oracion)` que retorne un diccionario asociando a cada letra la cantidad de veces que aparece en la oración:

```
>>> contar_letras('El elefante avanza hacia Asia')
{'a': 8, 'c': 1, 'e': 4, 'f': 1, 'h': 1, 'i': 2, 'l': 2, 'n': 2, 's': 1,
't': 1, 'v': 1, 'z': 1}
```

Cada valor del diccionario debe considerar tanto las apariciones en mayúscula como en minúscula de la letra correspondiente. Los espacios deben ser ignorados.

Actividad 61

Escriba la función `contar_iniciales(oracion)` que retorne un diccionario asociando a cada letra la cantidad de veces que aparece al principio de una palabra:

```
>>> contar_iniciales('El elefante avanza hacia Asia')
{'e': 2, 'h': 1, 'a': 2}
>>> contar_iniciales('Varias vacas vuelan sobre Venezuela')
{'s': 1', 'v': 4}
```

Actividad 62

Escriba la función `obtener_largo_palabras(oracion)` que retorne un diccionario asociando a cada palabra su cantidad de letras:

```
>>> obtener_largo_palabras('el gato y el pato son amigos')
{'el': 2, 'son': 3, 'gato': 4, 'y': 1, 'amigos': 6, 'pato': 4}
```