Seguridad Informática

Programación con Python

Virgilio Castro Rendón
Fernando Marcos Parra Arroyo

Comprensión de listas

- Es una forma de crear listas de forma rápida, siempre y cuando estas listas sigan una regla específica.
- Por ejemplo:

```
nones = []
for n in range(1000):
    if n % 2 != 0:
            nones.append(n)

nones = [n for n in range(1000) if n%2 != 0]
```

Comprensión de listas

- Lista con el cubo de todos los nones menores a 30.

- Lista de tuplas, a partir de una lista de cadenas. Cada tupla generada contendrá la cadena en mayúsculas, en minúsculas y el número de caracteres de la cadena.

  ['aaa','qwerty']
  [('AAA','aaa',3),('QWERTY','qwerty',6)]

- Diccionario donde las llaves son los nombres de los becarios y los valores son sus calificaciones (un número aleatorio entre 0 y 10).

Ejercicio de clase 6

- Hacer un diccionario por comprensión.

- Las llaves son los números odiosos menores a 50 y el valor es una tupla de dos elementos: su representación en binario y su representación en hexadecimal.

- Un número odioso (odious number) es todo aquél que su representación en binario tiene un número impar de unos.

Excepciones

- Ya hemos visto las excepciones en acción durante el curso.
- Una excepción ocurre cuando algo falla en la ejecución del código.
- Por ejemplo, si hacemos una división entre 0, se genera la excepción ZeroDivisionError.

```
>>> 7 / 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
>>>
```

Excepciones

- Algunas excepciones comunes son:
  - ImportError: falló una importación
  - IndexError: se indexó una lista con un número fuera del rango
  - NameError: se utilizó una variable desconocida
  - SyntaxError: el código no se pudo analizar de forma correcta
  - TypeError: una función es llamada con un valor de un tipo inapropiado
- Python tiene más excepciones, además de las mencionadas.
- Las bibliotecas creadas por terceros, comúnmente definen sus propias excepciones.

Excepciones

- Existen tres bloques de código que, en conjunto, sirven para manejar excepciones:
  - **try**:
    - Contiene el código que puede generar una excepción
    - Si ocurre una excepción, este código deja de ejecutarse
  - **except:**
    - Contiene el código que se ejecuta si ocurre una excepción en el bloque “try”
    - Si no ocurre una excepción, no se ejecuta este bloque de código
  - **finally:** (opcional)
    - El código contenido en este bloque se ejecuta sin importar si ocurre o no una excepción

Excepciones

- Una excepción en particular se maneja a través de su nombre.

```
>>> try:
...         num1 = 10
...         num2 = 0
...         print num1/num2
...         print 'calculo terminado'
... except ZeroDivisionError:
...         print 'Error por division entre cero'
...
Error por division entre cero
>>>
```

Excepciones

- Se pueden poner múltiples bloques except

```
>>> try:
...         variable = 10
...         print variable + "hola"
... except ZeroDivisionError:
...         print 'Division entre cero'
... except (ValueError,TypeError):
...         print 'Ocurrio un error'
...
Ocurrio un error
```

Excepciones

- Si se omite el nombre de la excepción, se atrapa cualquier excepción que ocurra.
- Se recomienda poner este bloque al final de los demás bloques except.

```
>>> try:
...     variable = 3/0
... except TypeError:
...     print 'Error de tipos'
... except:
...     print 'Ocurrio un error'
...
Ocurrio un error
>>>
```

Excepciones

- El bloque finally ayuda en los casos en que queramos ejecutar una instrucción sin importar la presencia o falta de errores.

```
>>> try:
...     f1 = open('test.txt','w')
...     f1.write("hola" ** 2)
... except:
...     print 'Ocurrio un error'
... finally:
...     print 'Se ejecuta siempre'
...     f1.close()
...
Ocurrio un error
Se ejecuta siempre
>>>
```

PIP

- PIP es un sistema de gestión de paquetes.
- Es el sistema más usado para instalar y administrar paquetes de software escritos en Python.
- Puede ser instalado a través del manejador de paquetes del sistema operativo:
  - apt
  - yum
  - pacman

PIP

- Con pip es posible instalar una gran cantidad de módulos de Python que facilitan considerablemente el desarrollo de aplicaciones grandes.

- Tiene varias opciones:
  - pip install _____
  - pip uninstall _____
  - pip search _____
  - pip --help

Módulos populares (y muy útiles)

Nombre	Descripción
Requests	Sirve para entablar comunicaciones usando el protocolo HTTP.
Scrapy	Uno de los mejores módulos para *webscraping* (extracción de información de sitios web).
BeautifulSoup	*Parser* de formatos XML y HTML.
NumPy	Agrega diversas funcionalidades matemáticas a Python.
SciPy	Contiene herramientas matemáticas y algoritmos que lo han vuelto popular entre científicos.
Matplotlib	Permite hacer múltiples tipos de gráficas. Útil para el análisis de datos.
Pygame	Útil para el desarrollo de videojuegos en segunda dimensión.
Scapy	Sniffer y analizador de paquetes.

Un poco de teoría de HTTP

- Protocolo para transferencia de hipertexto (básicamente para visualizar páginas web).
- Tiene diversos métodos, los cuales tienen diferentes aplicaciones.
  - GET
  - HEAD
  - POST
  - PUT
  - DELETE
- Para hacer más sencilla la comunicación entre clientes y servidores, implementa diversos códigos de respuesta.
  - 1xx – Respuestas informativas
  - 2xx – Peticiones correctas
  - 3xx – Redirecciones
  - 4xx – Errores del cliente
  - 5xx – Errores del servidor

Un poco de teoría de HTTP

- Existen diversas formas de autenticarse ante un servidor HTTP.
- La más usada, es usando formularios HTML, con lo que la autenticación de los usuarios se controla con el lenguaje de backend.
- Existen otras formas, en las que se puede autenticar, siendo el servidor quien controla la autenticación.
  - Basic
  - Digest

Requests

- Este módulo permite hacer peticiones HTTP de manera muy sencilla, lo cual puede ser muy útil. Algunas aplicaciones (de seguridad) pueden ser:
  - Análisis de cabeceras HTTP
  - Análisis de certificados (HTTPS)
  - Web Scraping
  - Análisis de vulnerabilidades
  - Ataques de fuerza bruta
  - Búsqueda de información sensible

Descargar: req.py

Ejercicio de clase 7

- Modificar el archivo req.py para que, en lugar de recibir la contraseña y usuario directamente, reciba un archivo con una lista de contraseñas y un archivo con lista de usuarios.
- Debe de validar todos los usuarios con todas las contraseñas de las listas.

Tarea 5

- Modificar el archivo req.py del ejercicio anterior.
- Implementar la bandera del modo verboso(-v)
- Se debe de poder indicar un usuario o una lista de usuarios.
- Se debe de poder indicar una contraseña o una lista de contraseñas.
- Agregar bandera de reporte para indicar el archivo donde escribirá los hallazgos.
- Debe de tener todas las validaciones necesarias para que funcione correctamente.

Tarea 6

ANONIMATO

- Modificar el archivo req.py de la tarea anterior.

- Implementar una bandera para que todas las peticiones se hagan a través de TOR (investigar cómo lograr esto).

- Agregar bandera para cambiar el agente de usuario a uno diferente al de Python durante el ataque.