1. La instalación dispone de un programa llamado getJason.pyc legado de un sistema ya obsoleto y del cual no existen fuentes. El mismo fue compilado utilizando Python versión 3.6, la cual es una versión ya deprecada.
2. El mismo permite recuperar el API token para acceder a microservicios bancarios del Banco XXX contenido en un archivo JSON. Se desea reusar el programa asegurando que puede recuperarse cualquier clave existente en el archivo JSON el que ahora contiene claves múltiples indicándola como argumento (siendo el default “token1”).

a) Obtenga el programa getJason.pyc junto con el archivo JSON de prueba provisto y la documentación existente. Utilice Python 3.6. Documente lo que encuentre.

b) Ejecútelo, verifique la correspondencia entre lo que indica la documentación y el comportamiento real. Realice brevemente pasos 1 a 6 de la metodología explicada para ingeniería reversa.

c) Instale el paquete Python uncompyle6.

d) Ejecute uncompyle6 y obtenga el código fuente getJason.py.

e) Ejecútelo para verificar se comporta como getJason.pyc en el paso 2.b.

f) Identifique las razones para las diferencias encontradas en el punto anterior entre el código obtenido y la documentación del mismo.

g) Modifique el programa getJason.py para que actúe como indica la documentación y satisfaga los requerimientos de reuso.

h) Valide y verifique su nuevo comportamiento bajo una versión mas avanzada de Python (3.11 por ejemplo).

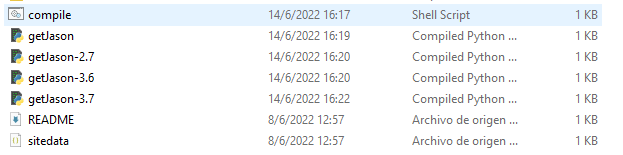
i) Remueva los comentarios hechos por el proceso de de-compilación.

j) Genere la nueva versión de getJason.pyc mediante compilación

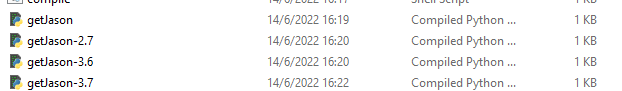
k) Utilice python –m compileall getJason.py

l) Verifique y valide.

1. Continuando con el programa getJason.pyc (y su versión fuente getJason.py) del ejercicio anterior se ha decidido que el programa tiene un código deficiente para lo crítico que resulta para la instalación y por lo tanto se lo someterá a un proceso de re-factoría.
2. Manteniendo las modificaciones realizadas en el punto anterior analice una intervención que permita aplicar programación orientada a objetos de forma que transforme al programa.
3. La transformación consistirá en reorganizar el código para utilizar un patrón de diseño “Singleton”.
4. La clase resultante del punto anterior debe poder ser ejecutada desde línea de comandos, al efecto deberán mejorarse las condiciones de chequeo de los parámetros externos de ejecución de forma de hacer a la ejecución más robusta. El objetivo primario es que el programa nunca termine con un error de sistema y siempre lo haga con un error del programa controlado.
5. Para hacer la convergencia entre el programa original y su versión re factorizada utilice una estrategia “Branching by abstraction”.
6. Aproveche a agregar comentarios al programa de su funcionamiento y una carátula donde se especifique es propiedad de la compañía (“copyright UADERFCyT-IS2©2024 todos los derechos reservados).
7. También mejore el funcionamiento controlando que los argumentos de ejecución sean correctos y en caso de no serlos la terminación sea con un error controlado y no una excepción del lenguaje.
8. Agregue además que si el programa se ejecuta con el argumento “-v” emita la versión del mismo. En éste caso lo denominará “versión 1.1”.
9. Haga una corrida de revisión con el analizador estático de código pylint y corrija las observaciones hasta que el mismo otorgue un puntaje de 8 o superior.
10. 1. Dentro de la carpeta src del programa encontramos; diferentes versiones para el programa getJason.pyc (2.7, 3.6, 3.7), un archivo README.md que describe cómo funciona el programa, un archivo compile.sh y un archivo jason sitedata.json



* + - Al ejecutar getJason.pyc con python3.6 obtenemos un error de Python no controlado
    - Al ejecutar ./getJason-3.6.pyc ./sitedata.json <token> como nos indica el README nos retorna el token especifico de manera correcta
    - Al ejecutar otra versión del programa, obtenemos un error de “Magic Number” (Versión incorrecta de Python)
  1. La documentación nos dice que al ejecutar el programa nos debe devolver el token para acceso API del banco XXX. Al ejecutar esto con el comando proporcionado ({path ejecutable}/getJason.pyc {path archivo JSON}/{nombre archivo JSON}.json) obtenemos para token1 y token 2 sus respectivas claves de acceso de manera correcta. Se debe mencionar que también incluye un comando para cito “mensajes de ayuda detallado” (./getJason.pyc -h), el cual retorna un error de Python no controlado, como también afirma que “Todas las condiciones de error del programa deben producir un mensaje de error bajo su control antes de terminar”, por lo tanto, esta afirmación es errónea.
     + Paso 1: Recolección de datos:
       1. 4 ejecutables del programa en diferentes versiones de Python (getJason.pyc, getJason-3.6.pyc, getJason-3.7.py, getJason-2.7.pyc)
       2. Archive sitedata.json con los tokens en formato json que utiliza el ejecutable.
       3. Archivo compile.sh que al parecer es un script con informacion sobre la compilación del programa.
       4. Archivo README.md que indica brevemente la finalidad del programa y como ejecutarlo.
     + Paso 2:
       1. Ejecutables binarios:



* + - 1. Documentación: Este programa permite extraer la clave de acceso API para utilizar los servicios del Banco XXX.

Funciona como un micro servicio y debe ser invocado con la siguiente estructura:  
{path ejecutable}/getJason.pyc {path archivo JSON}/{nombre archivo JSON}.json

El token se recuperará a través del standard output en el siguiente formato:

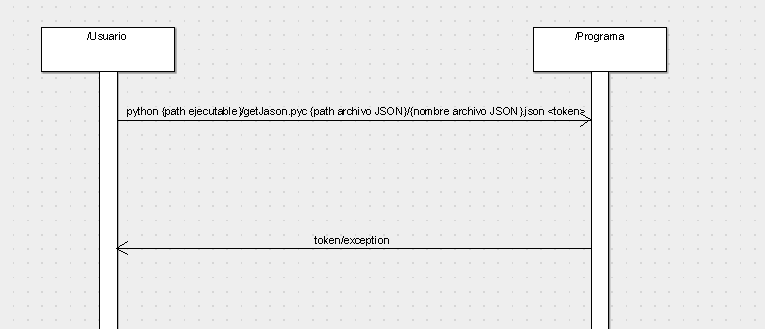
{1.0}XXXX-XXXX-XXXX-XXXX

Mensaje de ayuda:

./getJason.pyc –h

Y que todas las excepciones deben producir un mensaje de error controlado antes de cerrar

* + - 1. Discrepancias:
         1. El comando para recuperar el token no especifica que debe indicar que token se recuperara agregándole un <token> al final del comando
         2. El formato de recuperación es XXXX-XXXX-XXXX-XXXX
         3. El mensaje de ayuda no funciona, retorna excepción no controlada
         4. Existen excepciones no controladas, a pesar de que se detalla en el README que no debería hacerlo Ej: Index Error
      2. Resultantes de ejecución, al ejecutar cualquier de los comandos retorna índex error, al especificar el token para recuperarlo, como se especificó en las discrepancias, el programa funciona de manera correcta
    - Paso 3: Hasta ahora no podemos ver el código fuente del ejecutable. Asemejado a variables tenemos un json con dos tokens y sus respectivos valores.
    - Paso 4: Registra funcionalidad

Diagrama de secuencia para ver el comportamiento del programa  


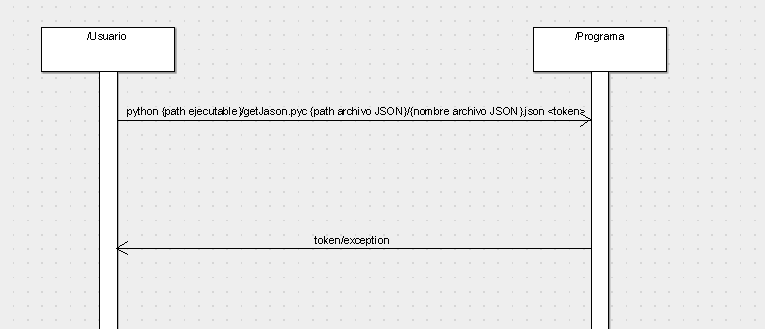
* + - 1. Excepciones: Ambas excepciones encontradas retornan errores genéricos de Python:
         1. Key Error:



* + - * 1. Índex Error:



* + - Paso 5: Registra funcionalidad



* + - Paso 6: Documentación

Token Extractor para Servicios API del Banco XXX

Este proyecto contiene un extractor de token que permite obtener la clave de acceso API para utilizar los servicios del Banco XXX. El extractor es compatible con diferentes versiones de Python.

Archivos del Proyecto

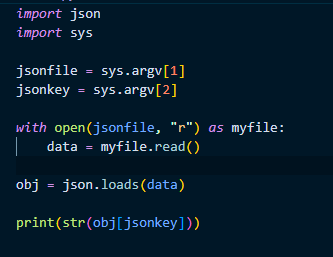
* getJason.py - Script ejecutable para diferentes versiones de Python.
* README.md - Este archivo de documentación.
* COMPILE.sh - Script para compilar el proyecto.
* sitedata.json - Archivo JSON que contiene los tokens y sus valores.

Requisitos

* Python 3.x (Asegúrate de tener la versión adecuada de Python instalada en tu sistema).

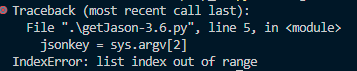
Uso del Programa

* Asegúrate de tener instalado Python en tu sistema.
* Coloca el archivo sitedata.json en el mismo directorio que getJason.py.
* Ejecuta el programa usando el siguiente comando en la consola de comandos: Python {path ejecutable}/getJason.pyc {path archivo JSON}/{nombre archivo JSON}.json <token>
  1. 



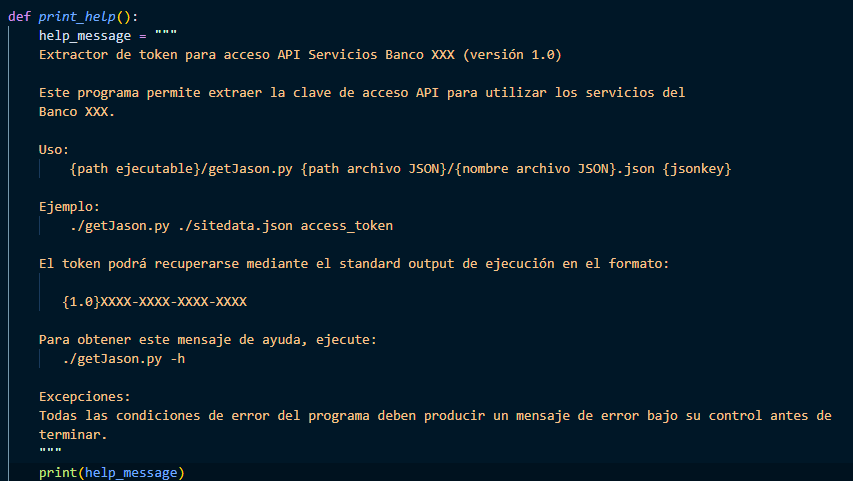


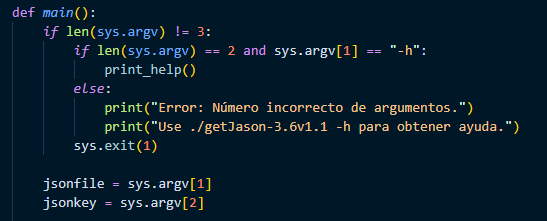


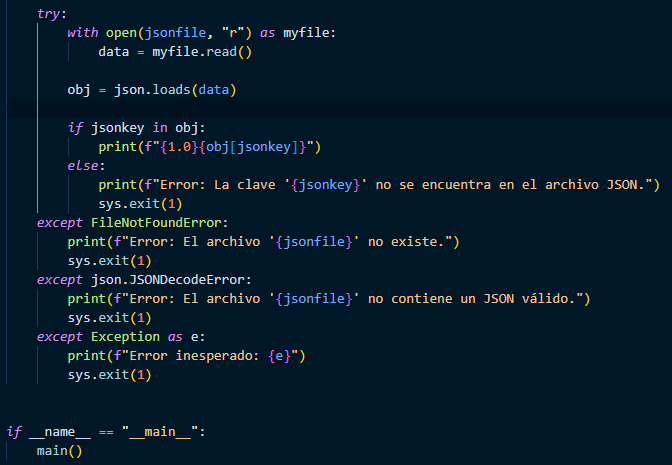


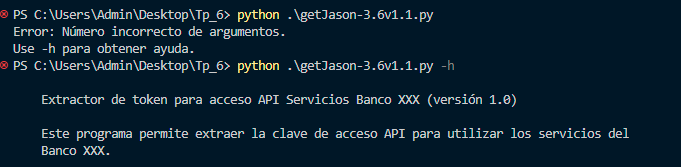
Funciona de la misma manera

El código requiere el nombre del token para funcionar ya que sería inconveniente que se retorne todos los tokens al no introducir parámetro, además no tiene implementado la capacidad de tratar excepciones

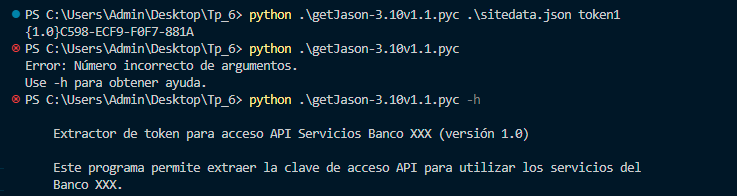
* 1. Nuevo code  
     



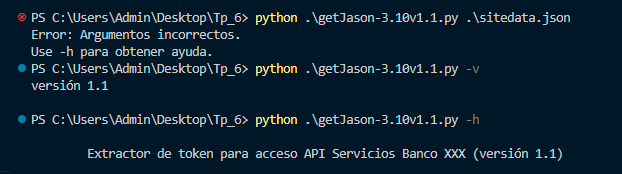


* 1. Comando ayuda. Comando incompleto  
     

Respuesta token  


* 1. Ya fueron removidos
  2. 
  3. 

Ejecución:





Pylint final:



* 1. El código y documentación(README-ri.md) pueden encontrarse en mi GitHub:
  2. Pylint final:
     + 