# Trabajo Practico

## INGENIERIA EN SOFTWARE 2

UADER – Facultad de Ciencia y Tecnología

**Profesor:** Pedro Colla

**Carrera:** Licenciatura en Sistemas de Información

**Alumno:** Franco Leonardo Camen.

**Github:** https://github.com/FrancoCamen/UADER\_IG2\_-Camen-.git

**Ciclo lectivo:** 2024.

# Consignas

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.

3) 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.

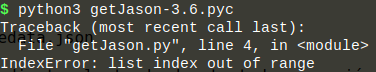
1. Manteniendo las modificaciones realizadas en el punto anterior analice una intervención que permita aplicar programación orientada a objetos de forma que transforme al programa.
2. La transformación consistirá en reorganizar el código para utilizar un patrón de diseño “Singleton”.
3. La clase resultante 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.
4. Para hacer la convergencia entre el programa original y su versión re factorizada utilice una estrategia “Branching by abstraction”.
5. 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).
6. 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.
7. 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”.
8. 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.

# 2.a)

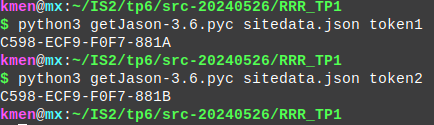
Dentro de la carpeta src del programa encontramos; diferentes versiones para el programa getJason.pyc (2.7, 3.6, 3.7), un archivo README.md que describe como ejecutar el programa, un archivo jason sitedata.json y un archivo compile.sh.



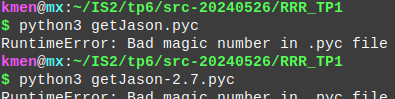
Al ejecutar getJason.pyc con python3.6 obtenemos:



Al ejecutar ./getJason-3.6.pyc ./sitedata.json como nos indica el README obtenemos:



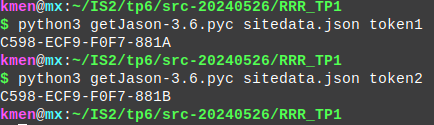
Al ejecutar otras versiones con python3.6 obtenemos:



# 2.b)

La documentacion nos dice que al ejecutar el programa nos debe devolver el token para acceso API del bano XXX. Al ejecutar esto obtenemos par token1 y token2 sus respectivas claves de acceso.

Al ejecutar ./getJason-3.6.pyc ./sitedata.json como nos indica el README obtenemos:

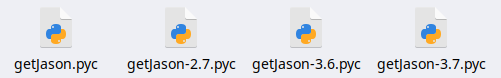


## PASO 1: coleccion de datos

* 4 ejecutables del programa en diferentes versiones (*getJason.pyc*, *getJason-3.6.pyc, getJason-3.7.py, getJason-2.7.pyc*)
* Archivo **README.md** que indica brevemente la finalidad del programa y como ejecutarlo.
* Archive **sitedata.json** con los tokens en formato json que utiliza el ejecutable.
* Archivo **compile.sh** que al parecer es un script con informacion sobre la compilacion del programa.

## PASO 2: Extraer información

### Ejecutables binarios:



### Resultados de ejecución:

Al ejecutar **./getJason.pyc ./sitedata.json** como nos indica el README.md obtenemos las claves de acceso correspondientes para cada token.

### Documentacion:

La documentacion nos dice que el programa es un extractor de token para acceso API servicios de Banco XXX. Permite extraer la clave de acceso api para utilizar los servicios del banco. El en archivo nos dice la forma de ejecutar el programa.

### Discrepancias:

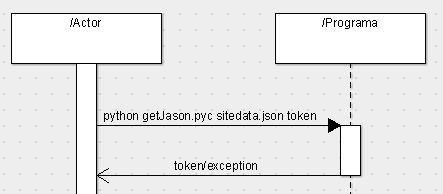
En el archivo README.md que nos dice como ejecutar el archivo está incompleto el comando de ejecución. Para poder hacer que funcione el programa tenemos que agregar al final de este que token queremos buscar (token1/token2)

## Paso 3: extraer estructura

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:

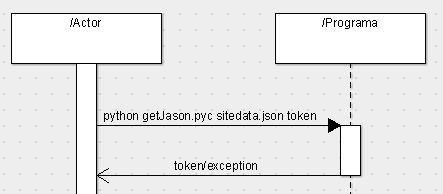


### Excepciones

Algunas de las excepciones que maneja el programa son:

* El parámetro ingresado en token es incorrecto (“KeyError”)
* Falta de parametro sitedata.json y/o token (“IndexError”)

## PASO 5: REGISTRA FLUJOS DE datos



## Paso 6: documentacion

La documentación incluirá lo siguiente:

# 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

1. Asegúrate de tener instalado Python en tu sistema.

2. Coloca el archivo `sitedata.json` en el mismo directorio que `getJason.py`.

3. Ejecuta el programa usando el siguiente comando en la consola de comandos:

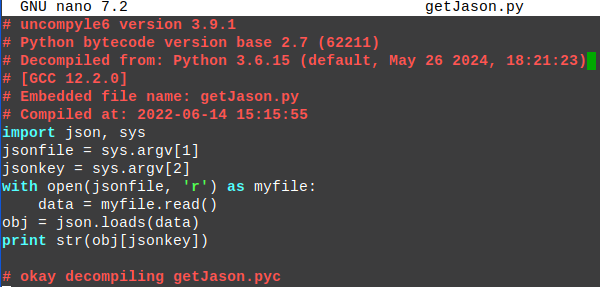
```sh

python getJason.py sitedata.json <keytoken>

# 2.c)



# 2.d)



# 2.e)

Verificamos que se comporta igual a getJason.pyc



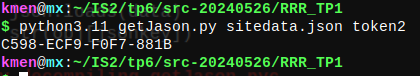
# 2.f)

No se encuentran diferencias de funcionamiento entre getJason.pyc y getJason.py

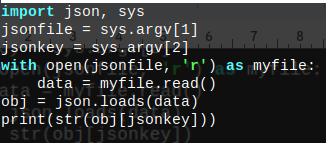
# 2.g)

No se encuentran diferencias entre el resultado de la ejecución de getJason.py y lo que dice la documentación de getJason.pyc

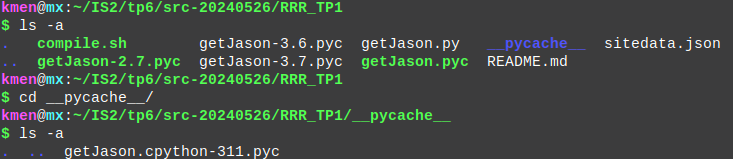
# 2.h)



# 2.i)



# 2.j) 2.k)



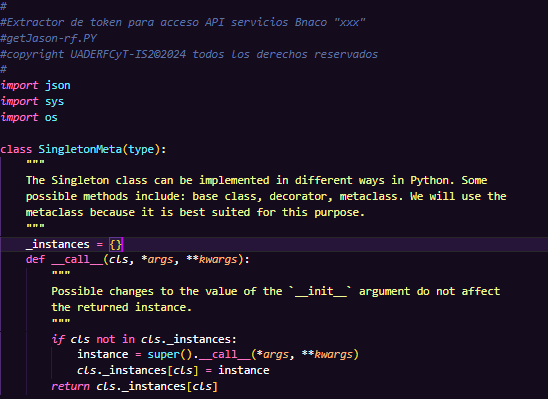
# 3

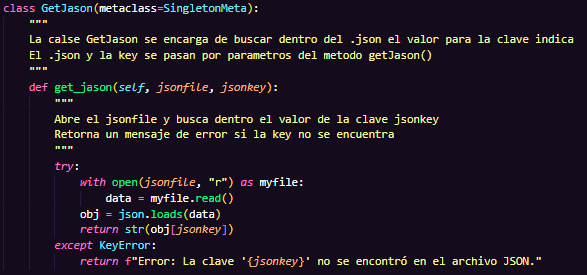
Se hicieron modificaciones al código de getJason.py obteniendo getJason-rf.PY. Se implemento un patrón Singleton, creando la clase getJason. Se mejoro la validación de parámetros para una ejecución más robusto, controlando las validaciones y excepciones de los argumentos ingresados. Se agregaron comentarios y propiedad intelectual. Se añadió opción para mostrar la version del programa (1.1). Se verifico el resultado del programa según los estándares de pylint.

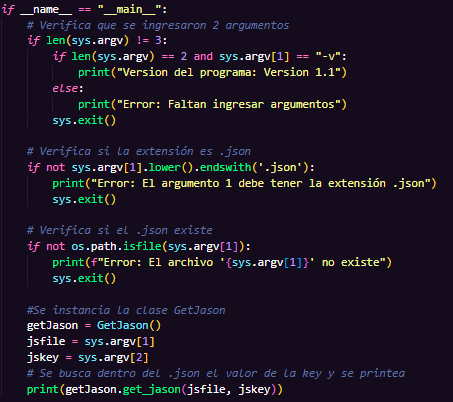
### Verifiacion con pylint

### 

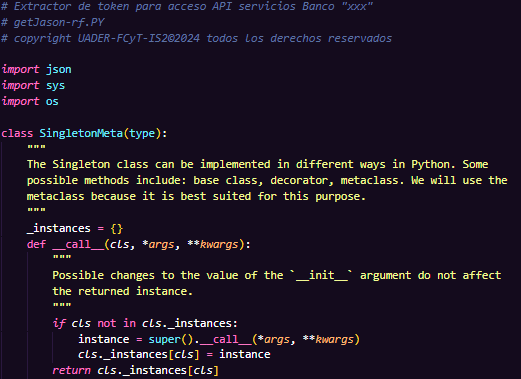
### Getjason-rf.PY

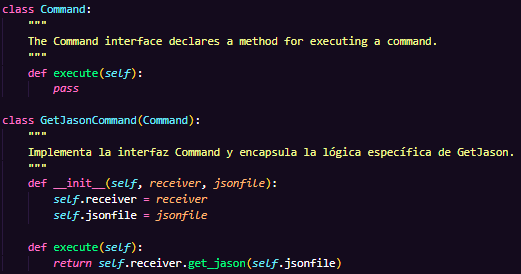
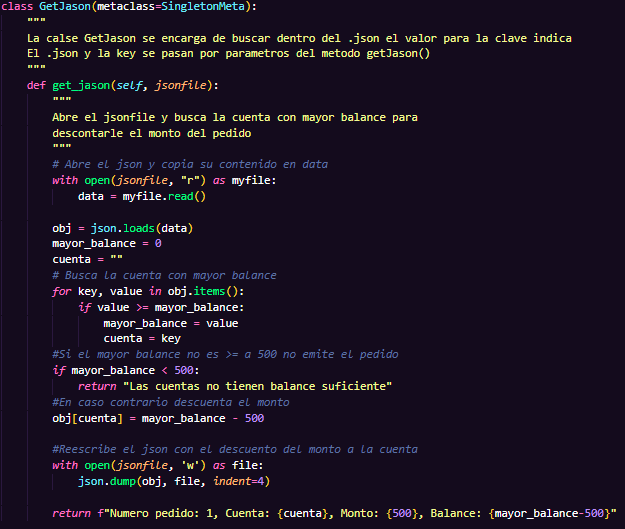


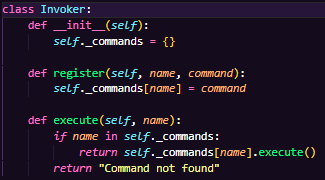


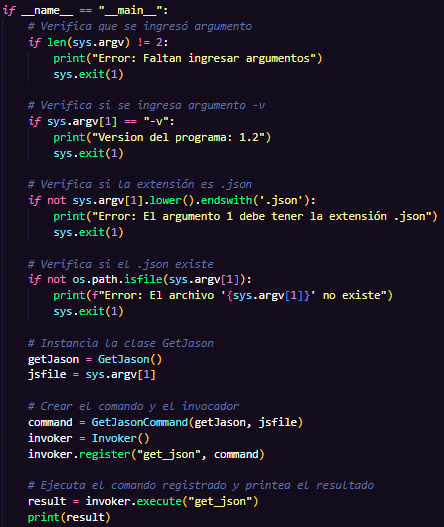


# 4

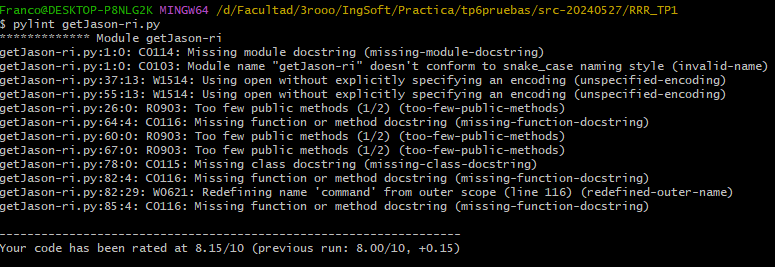








## PRUEBA CON PYlint



## Documentación del código

# Extractor de Token para Acceso API Servicios Banco "xxx"

## Descripción

Este programa se encarga de extraer el token necesario para acceder a los servicios API del banco "xxx". El programa busca dentro de un archivo JSON (`sitedata.json`) el valor asociado a una clave específica y realiza un pedido de pago de $500 desde la cuenta con el mayor balance.

## Archivos del Proyecto

- \*\*getJason-ri.py\*\*: Script principal que realiza la extracción del token y el pedido de pago.

- \*\*README.md\*\*: Archivo de documentación del proyecto.

- \*\*COMPILE.sh\*\*: Script para compilar los archivos.

- \*\*sitedata.json\*\*: Archivo JSON que contiene dos tokens (`token1` y `token2`) y sus valores.

## Uso

Para ejecutar el programa, usa el siguiente comando en la terminal:

"python getJason-ri.py sitedata.json"

## Argumentos

- \*\*sitedata.json\*\*: Archivo JSON que contiene las cuentas y sus balances.

## Ejemplo

python getJason-ri.py sitedata.json

Este comando procesará el archivo sitedata.json, buscará la cuenta con el mayor balance y descontará $500 de dicha cuenta. El resultado será impreso en la terminal.

## Características

- \*\*Singleton Pattern\*\*: La clase principal utiliza el patrón de diseño Singleton para asegurar que solo haya una instancia del objeto que maneja el archivo JSON.

- \*\*Command Pattern\*\*: Se implementa el patrón de diseño Command para encapsular la lógica de negocio y facilitar la ejecución de comandos.

- \*\*Versión\*\*: Si el programa se ejecuta con el argumento -v, mostrará la versión actual del programa.

## Manejo de Errores

-Si no se proporciona un archivo JSON como argumento, el programa mostrará un mensaje de error y terminará la ejecución.

-Si el archivo proporcionado no tiene la extensión .json, el programa mostrará un mensaje de error y terminará la ejecución.

-Si el archivo JSON no existe, el programa mostrará un mensaje de error y terminará la ejecució n.

-Si el mayor balance en el archivo JSON es menor a $500, el programa devolverá un mensaje indicando que las cuentas no tienen balance suficiente.

## Versión

Este programa está en la versión 1.2. Para mostrar la versión del programa, puedes ejecutar el siguiente comando:

"python getJason-ri.py -v"

### TODOS LOS ARCHIVOS DEL PROGRAMA SE ENCUENTRAN EN MI GITHUB

https://github.com/FrancoCamen/UADER\_IG2\_-Camen-.git