Práctica 2 - Python avanzado

Protocolos para la Transmisión de Audio y Vídeo en Internet

Versión 11.0 – 13.10.2020

Nota: Esta práctica se puede entregar para su evaluación como parte de la nota de prácticas, pudiendo obtener el estudiante hasta un punto. Para las instrucciones de entrega, mira al final del documento. Para la evaluación de esta entrega se valorará el correcto funcionamiento de lo que se pide y el seguimiento de la guía de estilo de Python.

1 Introducción

La programación orientada a objetos en un paradigma de programación muy utilizado en la actualidad y que conviene conocer para la realización de las prácticas de la asignatura. En esta práctica se pretenden ver los conceptos más importantes de este paradigma, implementando funcionalidad en Python.

2 Objetivos de la práctica

- Conocer y practicar la programación orientada a objetos (con conceptos como clase, objeto, herencia, instanciación).
- Usar varios módulos Python, importando funcionalidad creada en otros módulos.
- Conocer y seguir la guía de estilo de programación recomendada para Python (ver PEP8).
- Utilizar el sistema de control de versiones git en GitLab.

3 Conocimientos previos necesarios

1. Nociones de Python3 (las de la primera práctica)

Tiempo estimado: 10 horas

4 Ejercicios

1. Comprueba que puedes entrar en tu cuenta del GitLab de la ETSIT (`https://gitlab.etsit.urjc.es`).

2. Con el navegador, dirígete al repositorio `ptavi-p2` en la cuenta del profesor en GitLab[1] y realiza un `fork`[2], de manera que consigas tener una copia del repositorio en tu cuenta de GitLab. Clona en tu ordenador local el repositorio que acabas de crear a local para poder editar los archivos. Trabaja a partir de ahora en ese repositorio, sincronizando (commit) los cambios que vayas realizando según los ejercicios que se comentan a continuación.

   Como tarde al final de la práctica, deberás realizar un `push` para subir tus cambios a tu repositorio en GitLab.

3. Investiga el archivo `calc.py`. Comprueba que en él se implementa una calculadora sencilla que permite sumar y restar. El programa tiene las siguientes características:

   - Ha de ser llamado de la siguiente manera por línea de instrucciones (*shell*):

     ```
     $ python3 calc.py operando1 operación operando2
     ```

     donde operación podrá ser `suma` o `resta`. Para tomar los parámetros del programa, se podrá hacer uso del módulo sys (`import sys`), en particular de la lista `sys.argv`. Se comprobará que los parámetros que el usuario pasa son numéricos (`integer` o `float`), imprimiendo `Error: Non numerical parameters` por pantalla en caso contrario.
   - Tener dos funciones (métodos): `sumar` y `restar`.
   - Imprimir el resultado por pantalla.

4. Crea en el archivo `calcoo.py` un programa Python que implemente la misma funcionalidad (y se ejecute de la misma manera) que `calc.py`, pero orientada a objetos. Para tal fin, crea una clase llamada `Calculadora` (las mayúsculas son importantes), que tenga los métodos suma y resta (suma y resta han de devolver el resultado, ¡pero no imprimirlo por pantalla!). A su vez, el programa principal deberá tomar los parámetros que el usuario ha dado en la línea de comandos (con `sys.argv`), instanciar un objeto de la clase Calculadora, llamar al método correspondiente e imprimir por pantalla el resultado.

[1] `http://gitlab.etsit.urjc.es/ptavi/ptavi-p2`

[2] Tienes instrucciones de cómo realizar un `fork` en `https://gitlab.etsit.urjc.es/help/gitlab-basics/fork-project.md`.

[Al terminar el ejercicio es recomendable hacer `commit` de los ficheros modificados]

5. Crea en el archivo `calcoohija.py` un programa Python que además de la misma funcionalidad de `calcoo.py`, pueda multiplicar y dividir. Para tal fin, crea una clase `CalculadoraHija` (las mayúsculas son importantes) que herede de `Calculadora`, y que además tenga los métodos multiplicar y dividir. En el caso de dividir, ha de capturar la excepción que saltará si `operando2` es cero, imprimiendo el siguiente mensaje de error por pantalla: `Division by zero is not allowed`. El programa será llamado de la siguiente manera desde la línea de instrucciones (shell):

```
$ python3 calcoohija.py operando1 operación operando2
```

donde operación podrá ser `suma`, `resta`, `multiplica` o `divide`.

[Al terminar el ejercicio es recomendable hacer `commit` de los ficheros modificados]

6. Crea el archivo `calcplus.py` que será llamado de la siguiente manera desde la línea de instrucciones (shell):

```
$ python3 calcplus.py fichero
```

donde `fichero` es un fichero de texto posiblemente multilínea con formato `CSV` (`comma-separated-value`), esto es, cada línea del mismo tendrá la siguiente forma:

```
operación,operando1,operando2,operando3,...,operandoN
```

`calcplus.py` deberá tomar la operación línea a línea y realizarla de manera secuencial con todos los operandos (el primero con el segundo, el resultado de la operación con el tercero, y así sucesivamente) imprimiendo el resultado por pantalla. Así, para las siguientes líneas:

```
suma,1,2,3,4,5
resta,31,6,4,3,2,1
multiplica,1,3,5
divide,300,10,2
```

el resultado que se imprimirá por pantalla será, para el ejemplo que se da, 15 en todos los casos. En el caso de que la operación sea de división y uno de los operandos sea cero, se imprimirá por pantalla

`Division by zero is not allowed`. El fichero `calcplus.py` deberá hacer uso de la funcionalidad implementada en `calcoohija.py`.

[Al terminar el ejercicio es recomendable hacer `commit` de los ficheros modificados]

7. Crea el archivo `calcplusplus.py` que será llamado de la siguiente manera por línea de instrucciones (shell):

```
$ python3 calcplusplus.py fichero
```

`calcplusplus.py` ha de tener la misma funcionalidad que `calcplus.py` (ver ejercicio anterior), pero deberá hacer uso del módulo `csv` de Python3[3] y de la sentencia `with` de Python.

[Al terminar el ejercicio es recomendable hacer `commit` de los ficheros modificados]

8. Aunque el intérprete de Python admite ciertas libertades a la hora de programar, los programadores de Python con la finalidad de mejorar principalmente la legibilidad del código han acordado seguir una guía de estilo. Esta guía de estilo se encuentra en el `Python Enhancement Proposal 8` (PEP 8), y contiene instrucciones sobre cómo situar los espacios en blanco, cómo nombrar las variables, etc. Puedes encontrar la guía (original, en inglés, y una versión parcial en castellano) en el Moodle de la asignatura.

   Existe un programa de ayuda de línea de comandos llamado `pycodestyle`[4] que te permite comprobar si se siguen la mayoría de las indicaciones de PEP8. Pásale la herramienta a tus programas de Python3 hasta que no dé ningún error.

   [Al terminar el ejercicio es recomendable hacer `commit` de los ficheros modificados]

   [Al terminar la práctica, realiza un `push` para sincronizar tu repositorio GitLab]

5 ¿Qué deberías tener al finalizar la práctica?

La entrega de práctica se deberá hacer antes del miércoles 21 de octubre de 2020 a las 23:59. Para ello, se deberá contar a esa fecha con

1. Tener un repositorio git en GitLab con:
   - 5 módulos Python: `calc.py`, `calcoo.py`, `calcoohija.py`, `calcplus.py`, `calcplusplus.py`.
   - 2 clases: `Calculadora` y `CalculadoraHija`.

[3] https://docs.python.org/3.8/library/csv.html

[4] Antiguamente este programa se llamaba `pep8`, pero se cambió el nombre para no dar a enteder que el programa es equivalente a la guía de estilo.

- Todo ello siguiendo la guía de estilo PEP8.

Se han de tener en cuenta las siguientes consideraciones:

- Se valorará que al menos haya diez `commits` realizados.
- Se valorará que el código entregado siga la guía de estilo de Python (véase PEP8).
- Se valorará que los programas se invoquen correctamente y que muestren los errores correctamente, según se indica en el enunciado de la práctica.

Se puede comprobar la correcta entrega de la práctica utilizando el programa `check-p2.py` incluido en el repositorio de la práctica. Este programa se ejecuta desde la línea de comandos de la siguiente manera:

```
$ python3 check-p2.py login_gitLab
```

donde `login_gitLab` es tu nombre de usuario en el GitLab de la ETSIT. El programa comprueba que se han entregado los ficheros que se solicitan (y sólo esos), y parcialmente si se sigue la guía de estilo PEP8.