INF-253 Lenguajes de Programación

Tarea 1: Python

Profesor: José Luis Martí, Rodrigo Salas, Ricardo Salas, Gabriel Carmona

Ayudante Cátedras: Gonzalo Severín Muñoz, Carlos Bracamonte Espinoza, Daniel German Dueñas Rivera, Matías Uribe Ramírez, Dylan Oteíza Canales

Ayudante Tareas: Ricardo Olalquiaga Ferreira, Pablo Marambio Cuzmar, Cristian Tapia Llantén, Bryan González Ramírez, Sofía Riquelme Flores, Diego Moyano Raggio, Vicente Alvear Miranda, Bayron Valenzuela Galleguillos, Juan Pablo Cassis Oyanadel

02 de agosto de 2023

1. La Hyper Calculator

Un miembro de su comunidad, Complejín, tiene un ritual para comprar la mercadería del mes: primero compra las verduras esenciales, segundo los bebestibles infaltables, luego el pan de cada día, y el dinero sobrante lo reparte entre distintas "cosas pa picar" que se le antojen. Por esto, previo a cada ritual Complejín arma un carrito de compras donde calcula todo el dinero que gastará, utilizando todos los cupones que tiene a disposición, sin embargo, la cantidad de productos y el tamaño de los precios no caben adecuadamente en su Calculator 1050X.

Incapaz de resolver la lista del último mes, Complejín se acerca a usted buscando ayuda, como sabe que es estudiante de Informática, se preguntaba si podría ayudarlo creando una calculadora que se acomode a sus necesidades.

De esta forma, usted decide crear la *Hyper Calculator*, utilizando *Python3* para su desarrollo y la librería RegEx para facilitar su uso.

2. Especificaciones de la Calculadora

La calculadora solicitada posee más capacidades que una común, a continuación, se presenta un listado de sus funcionalidades.

- **Operaciones Binarias**
  - **Suma:** De la forma *a + b,* indica la suma de los enteros *a* y *b*.
  - **Resta:** De la forma *a – b,* indica la resta de los enteros *a* y *b*.
  - **Multiplicación:** De la forma *a * b,* indica la multiplicación de los enteros *a* y *b*.
  - **División entera:** De la forma *a // b,* indica la división entera de *a* con *b*. Al ser división entera, sentencias como *5 // 2* resultan en 2.
  - Ejemplos:
    - Correctos:
      - 1 + 2
      - 2 - 18
      - 5 * 7
      - 3 // 2
      - 18 // 3
    - Incorrectos:
      - 3 +
      - 23 –
      - 8 *
      - 4 //
      - – 34

- **Función CUPON**
  - Al encontrar la palabra clave *CUPON(x)*, la calculadora instancia en su lugar el ***20%*** de *x* **truncado.**
  - Como caso extra, la palabra *CUPON* puede aceptar 2 parámetros: en el caso de tener *CUPON(x, y)*, la instancia en su lugar será el ***y%*** de *x* **truncado**. Note que *x* e *y* se encuentran separados por una *coma*, por lo que da igual cuántos espacios se encuentran al lado de la coma.

- **Variable ANS**
  - Al igual que una calculadora simple, la super calculadora guarda el valor de la última línea resuelta detrás de la variable *ANS*.
  - Note que *ANS* solo guarda valores dentro del mismo problema, al terminar un problema (luego de un *whiteline*), ANS debería devolverse a 0.

- **Revisión de Errores**
    - El archivo "`problemas.txt`" contiene múltiples problemas a resolver por la calculadora. Es importante notar que la resolución de un problema solo se realizará en caso de que no tenga errores.
    - Por cada problema, la calculadora analizará todas sus líneas, revisando que no existan errores de syntax. En caso de encontrar un error, la calculadora no resolverá el problema, pero escribirá en el archivo "`desarrollos.txt`" cuáles sentencias eran las erróneas. (**'= error'** para las líneas con error y **'= Sin resolver'** para las sentencias del problema que no tenían errores)
    - Como caso especial, la división por 0 también es considerado como un error.

- **Paréntesis y orden de operaciones**
    - Expresiones encerradas en paréntesis se ejecutan primero que cualquier otra. En caso de haber múltiples niveles de paréntesis, se deben ejecutar desde adentro hacia afuera.
    - El orden general de ejecución de operaciones es Paréntesis – Multiplicación y División – Suma y Resta. Si se encuentran 2 operaciones con la misma prioridad en una sentencia, se resuelven de **izquierda a derecha**.

3. Ejemplo

A continuación se presentan algunos ejemplos de problemas y su resultado, por favor preste atención al formato. Es muy importante notar que cada problema se compone por una serie de sentencias y que cada problema termina por un *whiteline* (**'\n'** en Python).

- `problemas.txt`

```
200000 – 5000 * 2 + (1320 – 10) // 3
ANS – 4500 + (300 * 5) // (25 * 7)
ANS // 15

ANS +    7 * 4 – 1
27 // 0
3 + 5

350 * 8 - (1500 – CUPON(1500))
ANS * ANS // 5

((3 + 1) * (500 – 34) - 30)
ANS // 3
```

- desarrollos.txt

```
200000 – 5000 * 2 + (1320 – 10) // 3 = 190436
ANS – 4500 + (300 * 5) // (25 * 7) = 185944
ANS // 15 = 123962

ANS +    7 * 4 – 1 = Sin resolver
27 // 0 = Error
3 + CUPON(10, 30) = Sin resolver

350 * 8 - (1500 – CUPON(1500)) = 1600
ANS * ANS // 5 = 512000

((3 + 1) * (500 – 34) - 30) = 1834
ANS // 3 = 611
```

4. EBNF General

```
digito ::= ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’

digito_o_zero ::= digito | ‘0’

entero ::= (digito {digito_o_zero} | ‘0’)

espacio ::= ‘ ‘

clave ::= ‘ANS’ | ‘CUPON(‘ {espacio} (entero | ANS) [ {espacio} ‘,’
{espacio} (entero | ANS) {espacio}] ‘)’

operador ::= {espacio} (‘+’ | ‘-‘ | ‘*’ | ‘//’) {espacio}

operacion ::= (clave | entero) operador (clave | entero)

sentencia ::= operacion { operador (entero | clave)}
```

5. Detalles relevantes

- La calculadora no trabajará con números negativos, en el caso de que un valor sea negativo, se cambia a 0.
- El formato del archivo "`desarrollos.txt`" debe verse igual al presentado en el **Ejemplo**.
- La cantidad de espacios entre un número y un operador es mayor o igual a 0.
- Notar que en la **EBNF General** se establece que "*CUPON*" no acepta obtener el cupón de un cupón, es decir, "*CUPON(CUPON(< número >))*" no es un syntax posible para esta calculadora.
- Aunque no se especificó, la prioridad de ejecución de *ANS* y *CUPON* debe ser tomado en cuenta para el correcto funcionamiento del programa.
- Para *CUPON(x, y)*, *y* no puede ser mayor a 100.
- **Las consultas se hacen en el foro disponible en la sección <u>Tareas</u> de aula.**

6. Sobre la Entrega

- Se deberá entregar un programa llamado calculadora.py.
- Si no existe orden en el código habrá descuento.
- Las funciones implementadas deben ser comentadas de la siguiente forma. **SE HARÁN DESCUENTOS POR FUNCIÓN NO COMENTADA**

```
def my_fun(parametro_1, parametron_2, …):
    '''
    ***
    * parametro_1 : Tipo
    * parametro_2 : Tipo
    * …
    ***
    Breve descripción de qué hace la función y qué retorna.
    '''
```

- La tarea es individual.
- La entrega debe realizarse en tar.gz con el nombre: "Tarea1LP_RolAlumno.tar.gz"
- El archivo README.txt debe contener el nombre y rol del alumno e instrucciones detalladas para la correcta utilización de su programa.
- La entrega será vía aula y el plazo máximo de entrega es hasta el 19 de agosto a las 23:55 **<u>hora aula</u>**.
- Por cada día de atraso se descontarán 20 puntos (10 puntos dentro de la primera hora)
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.
- Solo se pueden realizar consultas respecto a la tarea hasta 2 días antes de la entrega.

7. Calificación

7.1. Entrega

Para la calificación de su tarea, debe realizar una entrega con requerimientos mínimos que otorgarán 30 pts base, luego se entregará puntaje dependiendo de los otros requerimientos que llegue a cumplir.

7.1.1. Entrega Mínima

- Crea diferentes expresiones regulares y las utiliza de manera correcta para resolver los problemas, aprovechándose de la modularización generada
- El programa es capaz de resolver correctamente problemas que solo tienen sumas, restas y la palabra clave ANS.

7.1.2. Entrega

- Resolución de problemas (total 40 pts):
  1. Realiza lo de la entrega mínima (MAX 0 pts)
  2. El programa escribe en "`desarrollos.txt`" el resultado de cada sentencia de acuerdo con el formato establecido. (MAX 10 pts)
  3. La calculadora puede resolver correctamente problemas con multiplicación y división entera, respetando el orden de prioridades. (MAX 25 pts)
  4. El programa puede resolver correctamente problemas que incluyen la palabra clave *CUPON* y paréntesis. (MAX 35 pts)
  5. El programa puede resolver problemas con paréntesis profundos mediante recursión. (MAX 40 pts)

- Detección y notificación de errores (total 30 pts):
  1. No detecta ningún tipo de error. (MAX 0 pts)
  2. Detecta errores generales de syntax, como escribir mal una palabra clave, dejando sin resolver el problema. (MAX 10 pts)
  3. Detecta errores de los operadores binarios, dejando sin resolver el problema. (MAX 15 pts)
  4. Detecta errores de las palabras claves *ANS* y *CUPON*, además del caso especial de división por 0, dejando sin resolver el problema. (MAX 20 pts)
  5. Detecta errores por paréntesis, dejando sin resolver el problema. (MAX 20 pts)
  6. Notifica los errores tal como se mostró en el ejemplo. (MAX 30 pts)

- **NOTA:** Puede existir puntaje parcial, por ejemplo en detección y notificación de errores puede obtener 3 pts

7.2 Descuentos

- Falta de comentarios (-10 pts c/u MAX 30 pts)
- Falta de README (-20 pts)
- Falta de alguna información obligatoria en el README (-5 pts c/u)
- Falta de orden (-20 pts)
- Día de atraso (-20 pts por día, -10 dentro de la primera hora)
- Mal nombre en algún archivo entregado (-5 pts c/u)