INF-253 Lenguajes de Programación
Tarea 1: Python

Profesor: José Luis Martí Lara
Ayudante Cátedras: Sebastián Godínez San Martín
Ayudante Tareas: Gabriel Carmona Tabja, Sebastián Campos Muñoz

14 de abril de 2020

1. Super Programming Bros.

Los hermanos Mario tenian un tercer hermano llamado Giuseppe, este en vez de ser un fontanero que salva princesas, es un aficionado de la informática. Viendo a sus hermano teniendo aventuras logro observar que podía escribir un lenguaje de programación llamado Yahooo. Por esto, Giuseppe les pide a ustedes crear un programa en Python, utilizando expresiones regulares, que permita verificar la sintaxis de un programa en el lenguaje Yahooo y permita ejecutarlo.

Se debe utilizar Python 3 y la librería RegEx para las expresiones regulares de la tarea, en caso de que alguna de éstas dos condiciones no se cumpla no se revisará la tarea.

2. Yahooo Reglas

2.1. Estructura Básica

Todo programa debe contener al principio un **Start Game** y al final un **Game Over** para indicar donde comienza y termina la ejecución del código.

Code 1: Comienzo y fin

```
Start Game
<Codigo a ejecutar>
Game Over
```

Code 2: Ejemplo malo

```
Start Game TA MALO
<Codigo a ejecutar>
Game OverMUY MUY MALO
```

2.2. Variables

Dentro de su programa se pueden declarar variables. Dado que Giuseppe baso el lenguaje en las aventuras de sus hermanos, donde por alguna extraña razón sus aventuras eran siempre en un grupo de máximo 4 personas, decidió que en este lenguaje solo puede existir como máximo 4 variables declaradas, donde cada variable puede llevar consigo un valor.

Code 3: Declaración y asignación de variable

```
Add Player <Nombre variable>
<Nombre variable> took <Value>
```

El nombre de la variable puede empezar solo por letras, pero el resto del nombre puede contener numeros y letras. No se aceptan variables con carácteres especiales.

El valor solo puede ser numérico, tanto positivo como negativo.

Code 4: Ejemplo de declaración y asignación

```
Start Game
Add Player mar10
mar10 took 2
Game Over
```

2.3. Input y Output

A veces las variables necesitan valores por lo que pueden pedir un variable utilizando la palabra **needs a power up**. Y para mostrar valores por consolas se usa la palabra clave **show**.

Cuando se ejecute un input se debe mostrar **Input:** para indicarle al usuario que debe ingresar un valor. Se mostrará más adelante como se debe ver en consola.

Code 5: Expresión para input y output

```
<Nombre variable> needs a power up
show <Nombre de variable o numero>
```

Code 6: Ejemplo de input y output

```
Start Game
Add Player mar10
mar10 needs a power up
show mar10
show −10
Game Over
```

2.4. Operadores

2.4.1. Matemáticos

Una operación matemática se describirá con la siguiente expresión:

Code 7: EBNF de la operación matemática

```
<operador> = '+' | '−' | '*' | '/'
<expresionMate> = '(' ( <expresionMate> <operador> <expresionMate>
                      | <expresionMate> <operador> ( <variable> | <numero> )
                      | ( <variable> | <numero> ) <operador> <expresionMate>
                      | ( <variable> | <numero> ) <operador> ( <variable> | <numero> ) ) ')'
```

Ojo: Para la operación el numero no puede ser negativo. Además no puede existir espacios en la operación

Por otro lado, para almacenar esta operación, se utiliza las palabras **jumps to**.

Code 8: Expresión para la operación matemática

```
Start Game
<Nombre variable> jumps to <expresionMate>
Game Over
```

Code 9: Ejemplo de utilización de operación matemática

```
Start Game
Add Player lu1g1
lu1g1 took 12
lu1g1 jumps to ((lu1g1+12)*lu1g1)
Game Over
```

Code 10: Ejemplo de expresiones incorrectas de operación matemática

```
lu1g1 jumps to (1 +2)
lu1g1 jumps to (1+−3)
lu1g1 jumps to (1+2+3)
```

2.4.2. Lógicas

Una operación lógica se describirá con la siguiente expresión:

Code 11: EBNF de la operación matemática

```
<comparador> = '>' | '>=' | '==' | '<' | '<='
<operador> = 'and' | 'or'

<expresionLogica> = '(' ( <variable> | <numero> ) <comparador> ( <variable> | <numero> ) ')'

<expresionLogicaGeneral>= '(' (<expresionLogica> <operador> <expresionLogica>
                        | <expresionLogica> <operador> <expresionLogicaGeneral>
                        | <expresionLogicaGeneral> <operador> <expresionLogica>
                        | <expresionLogicaGeneral> <operador> <expresionLogicaGeneral> )')'
                        | <expresionLogica>
```

La expresión lógica solo se puede utilizar en los condicionales o ciclos que se detallarán más adelante. Al igual que la expresión matemática, la expresión lógica no puede tener espacios.

Code 12: Ejemplo de expresión lógica correcta

```
(10==2)
((10>=30)and(((60>10)and(1<=2))or(12<1)))
```

Code 13: Ejemplo de expresión lógica incorrecta

```
(10 == 2)
((HOLAAMIGO=>30)and (((60>10)and(1<=2))or(12<1))(OLAESTONODEBERIAIRAQUI>432)
```

2.5. Condicionales

Durante el código pueden existir condicionales que representan el camino a seguir dependiendo de la respuesta a una expresión lógica. Donde **It's a me a conditional** y **Yahoo** indicaría el inicio de un condicional, luego el else será indicado por **Mamma Mia...** y el fin de la expresión condicional será indicado por **Let's Go!**.

Code 14: Expresión de un condicional

```
1  It's a me a conditional <expresionLogicaGeneral> Yahoo
2  <Codigo a ejecutar si la expresionLogicaGeneral es verdadera>
3  Mamma Mia...
4  <Codigo a ejecutar si la expresionLogicaGeneral es Falsa>
5  Let's Go!
```

El condicional no puede estar vacío, si existe el condicional, entonces si o si debe haber un código para el caso verdadero y para el caso falso. Si la expresion lógica general está incorrecta, entonces todo el condicional está incorrecto.

Code 15: Condicional correcto

```
1  Start Game
2  Add Player Lucio
3  Add Player GODinez
4  GODinez took 123
5  Lucio took 432
6  It's a me a conditional (GODinez>2) Yahoo
7  Lucio jumps to (Lucio+GODinez)
8  Mamma Mia...
9  Lucio jumps to (Lucio-GODinez)
10 Let's Go!
11 Game Over
```

Code 16: Condicional incorrecto

```
1  Start Game
2  Add Player Lucio
3  Add Player GODinez
4  GODinez took 123
5  Lucio took 432
6  It's a me a conditional (GODinez>2) Yahoo
7  Mamma Mia...
8  Lucio jumps to (Lucio-GODinez)
9  Let's Go!
10 It's a me a conditional (GODinez > 2) Yahoo
11 Lucio jumps to (Lucio+GODinez)
12 Mamma Mia...
13 Lucio jumps to (Lucio-GODinez)
14 Let's Go!
15 Game Over
```

Ambos condicionales son incorrectos. El primero porque no tiene código si la condición es verdadera y el segundo porque la expresión lógica general es incorrecta.

2.6. Ciclos

Dentro del código existen los ciclos, estos comienzan con un **YA MA** dando entrada a la condición, mientras esta sea verdadera se ejecutará el código que hay dentro del ciclo, indicando el final con un **AH HA!**. Al igual que los condicionales, si la expresión lógica general es incorrecta todo el ciclo es incorrecto.

Code 17: Expresión de un ciclo

```
1  YA MA <expresionLogicaGeneral> YAHOO!
2  <codigo que se ejecuta mientras la condicion sea verdadera>
3  AH HA!
```

Code 18: Ejemplo de ciclo correcto

```
Start Game
Add Player MrPato
MrPato took 2
MrPato jumps to (MrPato+4)
YA MA (MrPato>=3) YAHOO!
MrPato jumps to (MrPato-3)
AH HA!
Game Over
```

Code 19: Ejemplo de ciclo incorrecto

```
Start Game
Add Player MrGanzo
MrGanzo took 2
MrPato jumps to (MrGanzo+4)
YA MA (MrGanzo>=3) YAHOO!
MrGanzo jumps to (MrGanzo-3)
AH HAHAHAHAHAHAHAHAHA!
Game Over
```

2.7. Funciones

Las funciones se definirán con la palabra **Secret Level** continuado con el nombre de la función y el nombre de sus parámetros. El final de una función, por otro lado, estará indicado con la palabra **Return to Level**. Las definición de funciones deben estar antes de un **Start Game** o despues de un **Game Over**.

Para llamar a la función dentro de el código utilizará la palabra enters. Esto hará que la variable que llame a la función almacene el valor que retorne. Sí, esto implica que las funciones solo retornan enteros.

Los parametros pueden ser numeros o variables y dos o más parametros deben tener un guión entre ellos.

Code 20: Expresión para las funciones

```
Secret Level <NombreFuncion> <Parametros>
<Codigo a ejecutar>
Return to Level <Valor o Variable a retornar>

Start Game
<NombreVariable> enters <NombreFuncion> <Parametros>
Game Over
```

Code 21: Ejemplo de utilización de función

```
Secret Level JLML nota1-nota2-nota3
Add Player jeje
jeje jumps to ((nota1+nota2)+nota3)
jeje jumps to (jeje/100)
Return to Level jeje
Secret Level LP tarea1
It's a me a conditional (tarea1>2) Yahoo
tarea1 jumps to (tarea1-2)
Mamma Mia...
tarea1 jumps to (tarea1+2)
```

```
Let's Go!
Return to Level tarea1
Start Game
Add player Mari10
Mario enters JLML 12-13-21
Mario enters LP 1
Game Over
```

3. Revisión y Ejecución del programa en Yahooo

Si al revisar el código se encuentrar errores de sintaxis no se debe ejecutar el código y se debe printear todos las lineas donde encontraron errores de la siguiente forma:

Code 22: Ejemplo de presentación de errores

```
Start Game
Add Player 1Mar
Mar tooook 2
Game Over

----- Consola ----
woooooooohhhh! 1: Add Player 1Mar
woooooooohhhh! 2: Mar tooook 2
```

Si el código tiene la sintaxis completamente correcta, entonces se procede a la ejecución, debe mostrarse por consola que comenzó la ejecución del código y cuando termino esta. Agregando obviamente todos los **show** que aparescan por el código.

Code 23: Ejemplo de ejecución

```
Start Game
Add Player Mar
Mar needs a power up
It's a me a conditional (Mar>2) Yahoo
Mar jumps to (Mar-10)
show Mar
Mamma Mia...
Mar jumps to (Mar-10)
Let's Go!
YA MA (Mar>=3) YAHOO!
Mar jumps to (MrPato-3)
show Mar
AH HA!
Game Over

----- Consola ----
Ejecutando codigo...
Input: 22
12
9
6
3
0
Termino la ejecucion.
```

4. Datos de vital importancia

- Exceptuando las funciones, cualquier código antes de Start Game y después de Game Over debe considerarse cómo un error.
- Los condicionales, operadores matemáticos, asignaciones, etc Sólo pueden tener la información específicada en acada un de sus secciones, y no pueden tener expresiones en líneas que no corresponden (se considera cómo error):

  Ejemplo:

  Code 24: Error

```
1  Star Game
2  Add Player nono (1+2)
3  Game Over
```

- No se pueden anidar dos o más condicionales. En cambio si se pueden anidar ciclos.
- Se debe asumir que si o si habrán como máximo 4 **Add Player**, para que no revisen que las variables sean exactamente 4. Eso siempre va a suceder.
- Si la sintaxis está correcta, entonces se asume que el código se ejecutará correctamente, no habrá errores en tiempo de ejecución.
- El código Yahoo será entregado en un archivo llamado yahooo.txt.

5. Sobre Entrega

- Se debera entregar un programa llamado yahooo.py.
- Las funciones implementadas deben ser comentadas de la siguiente forma. **SE HARÁN DESCUENTROS POR FUNCIÓN NO COMENTADA**

  ”’

  Nombre de la función

  Parametro 1 : Tipo
  Parametro 2 : Tipo
  Parametro 3 : Tipo
  .....

  Breve descripción de lo que realiza la función y lo que retorna ”’
- Se debe trabajar de forma individual obligatoriamente.
- **La entrega debe realizarse en tar.gz y debe llevar el nombre: Tarea1LP_RolIntegrante-1.tar.gz**
- **El archivo README.txt debe contener nombre y rol del alumno e instrucciones detalladas para la correcta utilización de su programa.**
- **El no cumplir con las reglas de entrega conllevará un descuento maximo de 30 puntos en su tarea.**
- La entrega será vía aula y el plazo máximo de entrega es hasta el **Viernes 24 de Abril a las 23:55 hora aula**.

- Por cada día de atraso se descontarán 20 puntos (10 puntos dentro la primera hora).
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.

6. Calificación

6.1. Entrega

- Orden (5 puntos)
- Uso correcto de expresiones regulares (25 puntos)
- Detección de errores
  - Estructura Básica (2 puntos)
  - Variables (2 puntos)
  - Input y Output (2 puntos)
  - Operadores Matemáticos (5 puntos)
  - Operadores Lógicos (5 puntos)
  - Codicionales (5 puntos)
  - Ciclos (7 puntos)
  - Funciones (7 puntos)
- Ejecución correcta
  - No ejecuta nada tiene 0 puntos
  - Porcentaje de ejecución correcta tiene un % de 35 puntos
  - Ejecuta correctamente en su totalidad tiene 35 puntos

6.2. Descuentos

- Falta de comentarios (-10 puntos c/u MAX 30)
- Falta de README (-20 puntos)
- No respeta formato de entrega (-15 puntos)