INF-253 Lenguajes de Programación
Tarea 1: Python

Profesor: José Luis Martí Lara, Jorge Ariel Díaz Matte,
Wladimir Elías Ormazabal Orellana.
Ayudante Cátedra: Norman Cortés Vega, Lucas Morrison Osorio,
Jhossep Martínez Velásquez.
Ayudante Coordinador: Álvaro Rojas Valenzuela.
Ayudante Tareas: Fernanda Araya Zárate, Bryan González Ramírez,
Andrés Jablonca Peña, Blas Olivares Gutiérrez, Nicolás Paz Tralma,
Bastián Salomon Ávalos, Cristian Tapia Llantén, Cristóbal Tirado Morales.

08 de Agosto de 2024

Es Turing Completo?

1. Un Lenguaje Nuevo

Unos investigadores se preguntan cómo sería programar en un lenguaje más simple de leer. Para ello crean un nuevo lenguaje llamado **PySimplex**. Este viene con los elementos necesario para crear programas básicos. Pero durante la producción, notaron que no tienen forma de ejecutar los códigos de pruebas creados. Para ello se le acerca usted, alumno del curso de Lenguaje de Programación de la UTFSM, para crear un programa que pueda interpretar los códigos en este intrincado lenguaje.

Para ello, se le pide usar *Python3* junto con la librería RegEx que pueda revisar, interpretar y ejecutar los códigos creados por los investigadores.

2. PySimplex

PySimplex es un lenguaje interpretado con ligado de tipo dinámico. La sintaxis de este lenguaje incluye:

- **Declaración de Variables**
  La declaración de variables está dado por las siguientes características:

- **Tipos de Datos**
  El ligado de tipo es **dinámico**. Es decir, que las variables pueden cambiar el tipo de dato que almacenan.
  El lenguaje ofrece soporte para los siguientes tipos:
  - **Enteros**: Corresponde a un número entero sin 0 extras al comienzo. Ej.: 0, 123 y 987010.
  - **Booleanos**: Corresponde a los valores `True` y `False`.
  - **Strings**: Corresponde a cualquier secuencia de caracteres y espacios entre dos símbolos #. Ej.: `#HolaMundo#`, `#aA bCD103902 CC#`.
- **Nombre de variable**
  Los nombres de las variables comienzan con un símbolo $, seguido por un guion bajo _, una letra mayúscula, seguido por cualquier combinación de letras en mayúscula y minúsculas. Ej.: `$_Hola`, `$_EstoEsUnaVariable`.

Para declarar una variable, se usa la palabra clave `DEFINE`. Ej.:

- Correcto: `DEFINE $_Var`
- Correcto: `DEFINE $_Varvar`
- Incorrecto: `define notvar`

- **Operaciones de procesamientos de datos y asignación**
  Estas instrucciones solamente pueden hacer una operación a la vez, tienen una variable de destino y uno o dos operandos que pueden ser una variable o una constante de tipo entero, booleano o string. Las soportadas son:

  - **Asignación**: Está dado por la palabra `ASIG`, es de tipo unaria y corresponde a que el destino se guarde el valor de una variable, número, booleano u string.
  - **Suma**: Está dado por el símbolo `+`, es de tipo binaria y corresponde a que el destino guarde el resultado de la suma de los operandos. No soporta de tipo booleano. Y cuando se opera un string con un entero, este último se concadena al string.
  - **Multiplicación**: Está dado por el símbolo `*`, es de tipo binaria y corresponde a que el destino guarde el resultado de la multiplicación de los operandos. No soporta de tipo booleano ni string.
  - **Mayor que**: Está dado por el símbolo `>`, es de tipo binaria y corresponde a que el destino guarde el resultado de la operación *mayor que* de los operandos (True o False). Solo es soportado por el tipo entero y el resultado es de tipo booleano.
  - **Igual que**: Está dado por el símbolo `==`, es de tipo binaria y corresponde a que el destino guarde el resultado de la operación *igual que* de los operandos (True o False). Es soporta por el tipo entero y tipo string. El resultado es de tipo booleano.

Para hacer una instrucción de procesamientos de datos y asignación, se usa la palabra clave `DP`, seguido por la variable destino (donde se guarda la operación), luego la operación, y los operandos. Ej.:

- Correcto: `DP $_Vardest ASIG 3654`
- Correcto: `DP $_Var + $_Var 1`
- Correcto: `DP $_Nombre + #hola# #mundo#`
- Incorrecto: `DP var + 5`

- **Estructura de Control Condicional**
  Los condicionales del lenguaje está dado por la palabra `if` y `else` escrito de la siguiente forma:

```
if(condicion) {
    codigo
} else {
    codigo
}
```

  La condición está dada por una variable que debe contener `True o False`.

- **Salida por Texto**
  El lenguaje contiene una función llamada `MOSTRAR(variable)`, en la que escribe el contenido de la variable, seguido por un salto de línea, en un archivo de texto llamado `output.txt`. (El archivo inicia vacío en cada ejecución del programa)

- Cada línea de código debe terminar con un salto de línea. La indentación no afecta la ejecución del programa.

- La cantidad de condicionales anidadas es hasta 3 veces.

2.1. EBNF General

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

digito_o_cero ::= digito | '0'

entero ::= (digito {digito_o_cero}) | '0'

booleano ::= 'True' | 'False'
```

```
string ::= '#' {A-Z | a-z | digito_o_cero | ' '} '#'

tipo ::= 'int' | 'bool' | 'string'

oper_un ::= 'ASIG'

oper_bin ::= '+' | '*' | '>' | '=='

operando ::= variable | entero | booleano | string

linea ::= declaracion | procesar | condicional | mostrar

variable ::= '$_' (A-Z) {A_Z | a-z}

declaracion ::= 'DEFINE' variable

procesar ::= 'DP' variable ((oper_un operando) | (oper_bin operando operando))

condicional ::= 'if(' variable ') {' {linea} '} else {' {linea} '}'

mostrar ::= 'MOSTRAR(' variable ')'
```

Se destaca que la cantidad de espacios entre palabras es inderteminada. Y no van a ver líneas vacías.

3. Interprete.py

Se debe crear un código en Python llamado **Interprete.py**, que lea y ejecute el archivo de texto llamado **codigo.txt** que contendrá el código en PySimplex. En caso de encontrar un error de sintaxis o de ejecución, se debe mostrar por la terminal el **tipo de error, en qué número de línea y detener la ejecución del código.** Algunos tipos de errores son:

- `Mal Sintaxis`: La línea (N) no está bien escrita.
- `Variable No Definida`: La variable de nombre (nombre) no ha sido definida o no se le ha asignado valor en la línea (N).
- `Tipo Incompatible`: La operación DP o condicional en la línea (N) es incompatible al tipo de dato.
- `Variable Ya Definida`: La variable de nombre (nombre) ya se encuentra definida.

Se deja en libertad el formato del tipo de error, siendo preciso y en pocas palabras.

- Ejemplo 1:
  - **codigo.txt**

```
DEFINE $_Largo
DEFINE $_Ancho
DEFINE $_Area
DEFINE $_Texto
DP $_Largo ASIG 6
DP $_Ancho ASIG 5
DP $_Area * $_Largo $_Ancho
DP $_Texto ASGI #La altura del rectangulo es #
DP $_Texto + $_Texto $_Area
MOSTRAR($_Texto)
```

  - **output.txt**

```
La altura del rectangulo es 15
```

- Ejemplo 2:
  - **codigo.txt**

```
DEFINE $_NumeroUno
DEFINE $_NumeroDos
DP $_NumeroUno ASIG 670
DP $_NumeroDos ASIG 670
DEFINE $_Cond
DP $_Cond > $_NumeroUno $_NumeroDos
if ($_Cond) {
    DEFINE $_texto
    DP $_texto ASIGN #Numero Uno es mayo a Numero Dos#
    MOSTAR($_texto)
} else {
    DP $_Cond == $_NumeroUno $_NumeroDos
    if ($_Cond) {
        DEFINE texto
        DP texto ASIGN #Numero Uno es igual a Numero Dos#
        MOSTRAR(texto)
    } else {
        DEFINE $_texto
        DP $_texto ASIGN #Numero Uno es menor a Numero Dos#
```

```
            MOSTAR($_texto)
        }
    }
```

- **output.txt** (vacío)

- **terminal**
  Error durante ejecución: Mala Sintaxis de variable en DEFINE en la línea 14.

4. Sobre la Entrega

- El código debe venir indentado y ordenado.
- Se debe entregar los siguientes archivos:
  - Interpetre.py
  - README.txt
- Se puede crear otros archivos **.py**.
- Al no hacer uso de expresiones regulares, la tarea no será revisada.
- Solo se permite el uso de la biblioteca estándar de Python (docs.python.org/library) (Incluye RegEx). No se permite usar otros módulos externos.
- Todas las funciones deben ir comentadas con el siguiente formato:

```
'''
***
Parametro 1 : Tipo
Parametro 2 : Tipo
...
***
Tipo de Retorno o None
***
Breve descripción de la función y el retorno
'''
```

- **La entrega debe realizarse en un archivo comprimido en tar.gz y debe llevar el nombre: Tarea1LP_RolAlumno.tar.gz**.
  Ej.: `Tarea1LP_202273000-k.tar.gz`.

- El archivo **README.txt** debe contener **nombre y rol del alumno**, junto a instrucciones detalladas para la correcta **ejecución** del programa.
- La entrega es vía aula y el plazo máximo de entrega es hasta el **23 de Agosto a las 23:55 hora aula**.
- Por cada día (o fracción) de atraso se descontará 20 puntos. 10 puntos dentro de la primera hora.
- Las copias serán evaluadas con nota 0 y se informará a las respectivas autoridades.

5. Clasificación

5.1. Entrega

Para la clasificación de su tarea, se debe realizar una entre con requerimientos mínimos que otorgarán hasta 30 pts base. Luego se le entregará puntaje dependiendo de los otros requerimientos que llegue a cumplir.

5.1.1. Entrega Mínima

Para obtener el puntaje mínimo en la entrega, el programa de cumplir con los siguientes requerimientos:

- Crea diferentes expresiones regulares y las utiliza de manera correcta para verificar la sintaxis del código antes de su ejecución, aprovechándose de su modularización generalizada. (10 pts)
- El intérprete ejecuta código básico de declaración de variables y procesamiento de datos de adición numérica según las reglas solicitadas, e imprime en el archivo de salida correctamente según enunciado. (10 pts)
- El intérprete detiene su ejecución al encontrar instrucción que usen variables previamente no definidas. (10 pts)

5.1.2. Entrega

Luego de cumplir con la Entrega Mínima, puede obtener más puntaje cumpliendo con los siguientes puntos (puede haber puntaje parcial por cada punto):

- **Ejecución del Código:** (50 pts Max)
  - El intérprete ejecuta todas las instrucciones de procesamiento de datos correctamente. (13 pts)

- El intérprete ejecuta la estructura de control condicional correctamente sin anidado. (10 pts)
- El intérprete ejecuta estructuras de control anidadas. (12 pts)
- El intérprete pasa múltiples casos de pruebas con todas las instrucciones disponibles correctamente. (15 pts)

- **Detección de Errores:** (20 pts Max)
  - El intérprete detecta todos los errores de sintaxis. (10 pts)
  - El intérprete detecta los errores relacionados con las variables. (5 pts)
  - El intérprete detecta los errores de calce de tipo de datos. (5 pts)

5.2. Descuentos

- Falta de orden (Max -20 pts)
- 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)
- Día de atraso (-20 pts por día (o fracción), -10 pts dentro de la primera hora)
- Mal nombre en algún archivo entregado o información errónea (-5 pts c/u)

En caso de existir nota negativa, esta será remplazada por un 0.