Introducción a Tareas y Tips del Lenguaje Python

Álvaro Rojas V.

UTFSM

Agosto 2024-2

Table of Contents

1. Lenguaje Python
   - Globalización y List Comprehesion
   - Manejo de Excepciones
   - Librearia RegEx

Intérprete de un Intérprete

La tarea 1 se tiene que crear un intérprete en lenguaje Python. Lenguaje ya conocido de antes.
Pero hay algunas cosas que no se habían enseñado. Por ejemplo: manejo de excepciones, y la librería RegEx.

Globalización y Main?

El ámbito de acceso de Python es global para todas las variables que son declaradas fuera de una función. Una forma de acercar Python a un lenguaje más parecido a C, es definir una función `main()`. Para ello se puede usar dos líneas de código para que esta función solamente se llame si y solo si es el archivo principal.

Función Main

```
# Ambito Global
ListaGlobal = "HolaMundo"

def main():
    # Ambito Local a Main
    VariableLocal = "HolaMain"
    return
# Convencion Main
if __name__ == "__main__":
    main()
```

Globalización y Main?

Esto son útiles para la modularización de códigos en Python. Cuando se importan librerías o códigos, el intérprete ejecuta todo el código del archivo. Por lo que se puede evitar ejecuciones no deseadas al momento de importar, pero se quiere mantener para hacer pruebas.
Se puede encontrar un poco más sobre importación de módulo aquí: https://www.freecodecamp.org/espanol/news/python-if-name-main/

List Comprehesion

List Comprehesion es una forma de reducir código. Es útil para filtrar elementos de una lista que tenga una condición deseada:

Ejemplo:

```
l1 = [1, 2, 3, 4, 5]
# Iteracion equivalente
l2 = []
for i in range(len(l1)):
    if l1[i] % 2 == 0:
        l2.append(l1[i])
# List Comprehesion
l3 = [x for x in l1 if x % 2 == 0]
print(l2)
print(l3)
```

Manejo de Excepciones

Al momento de ejecutar un programa de Python, existe la posibilidad de que se detenga y levante una excepción debido a un error de sintaxis o de ejecución.
Como programadores, uno le gustaría tener el control de estas excepciones e incluso levantar nuevas excepciones personalizadas para un manejo y orden más expedito del código.
En Python 3 se puede hacer ambas.

Bloques Try Except

Una excepción que el intérprete levanta es el **TypeError**, si se quiere evitar que el programa se detenga, se tiene que usar las líneas `try:` `except:`.

Ejemplo

```
try:
    print("Hola " + 45)
except:
    print("Error Captado")
print("Todo bien :)")
```

- La ejecución contenida en `try` se detendrá al levantar una excepción.
- Luego, en cambio de detener el programa, continuará en las líneas contenidas en `expect`.
- Finalmente, la ejecución continuará fuera de la sentencia `try`

Capturas de Excepciones

Las sentencias excpet permiten captar excepciones en específicas si se le es indicado:

Ejemplo

```
try:
    x = int(input("Dame un Numero: "))
    result = 256 // x
except ValueError:
    print("Ups, Eso no era un numero")
except ZeroDivisionError:
    print("Ups, Intente dividir por 0")
except:
    print("Atrape otro error no considerado")
```

Creación de Excepciones

Otra opción disponible es la de crear nuevas excepciones. Para ello, se tiene que crear una clase que herede los atributos de la superclase `Exception`.

Ejemplo

```
class MyError(Exception):
    # Inicializador
    def __init__(self, valor):
        self.valor = valor
```

Luego, pueden ser levantados de manera manual usando la palabra `raise`

```
try:
    raise(MyError("Hola soy un error"))
```

Expresiones Regulares

Durante las clases, se vio la **Jerarquía de Chomsky** para la clasificación de tipos de gramáticas formales. Uno de ellos son los lenguajes regulares. Durante la Tarea 1, se le pide utilizar **expresiones regulares** (ER) para analizar las sintaxis del lenguaje. Para ello, la librería estándar de Python cuenta con **RE** (Expresiones Regulares), un módulo que proporciona operaciones de **Matching** de patrones en cadenas de textos.

Expresiones Regulares

En pocas palabras, una ER es una expresión de un patrón a buscar en una palabra. Por ejemplo: la expresión `a(ab)*` indica el lenguaje L de palabras que comienzan con `a`, seguida de **0 o más** `ab`.
Por lo que, las palabras `a`, `aab`, `aababab` $\in L$. Pero `ab`, `aba`, `hola` $\notin L$.
Otros ejemplos de ER serían:

- `[A-Z][a-z]*`: Cualquier palabra que comience con mayúscula seguida de letras en minúscula.
- `([1-9][0-9]*|0)(\.[0-9][1-9]*)?`: Números enteros y decimales.
- `\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,7}\b`: Direcciones de correos electrónicos.

Símbolos

La simbología de las expresiones regulares de la librería de Python tienen diferentes significados, estos son algunos:

- `.` (Punto): Calza a cualquier carácter excepto al salto de línea.
- `*` : Calza **0 o más** repeticiones de la ER que le precede.
- `+` : Calza **1 o más** repeticiones de la ER que le precede.
- `?` : Calza solamente **0 o 1** repeticiones de la ER que le precede.
- `()` : Agrupa el contenido de una ER.
- `|` : Alterna entre dos ER arbitraria. (`A|B` equivale calzar A o B)
- `[]` : Indica un conjunto de caracteres a calzar. Ej: `[A-B]`, `[aeiou]`

Estos son algunas de las características de la sintaxis de las ER. Se pueden ver más en la documentación!

Objeto de Match

Las funciones más importante del módulo son aquellas que usan los ER para encontrar patrones.

Código Ejemplo

```
import re
digits = re.compile("(0|([1-9][0-9]*))")
digits.match("5390") # Patron en [0-3]
digits.fullmatch("abc1902") # None
digits.search("abc1902") # Patron en [3-6]
```

Para este caso, se compila el patrón para crear un **objeto de expresión regular** para reutilizar un patrón. Las funciones de calces (`.match`, `.fullmatch`, `.search`) retorna un objeto llamado `Match`.

Objeto de Match

Los objetos Match contiene información importante del calce encontrado. Indican información de agrupaciones y posiciones relativas al string original. Estos son:

- `Match.group()` : Retorna el calce encontrado, o subgrupos en una tupla. Los grupos son identificados usando las agrupaciones en la expresión regular.
- `Match.span()` : Entrega la tupla-2 `(m.inicio(), m.fin())` para una coincidencia con la información de los índices donde se encuentra la coincidencia en el string original. También se puede buscar por grupos.
- `Match.groups()` : Retorna una tupla que contiene todos los subgrupos de la coincidencia encontrada.

Límites de las ER

Las expresiones regulares tiene un límite de los lenguajes que pueden identificar. Por ejemplo, tiene problemas para identificar una expresión aritmética que contiene paréntesis anidados y se autocontiene. No todas las EBNF se pueden expresar en expresiones regulares, ya que este está en un nivel superior en los lenguajes que puede identificar.
Este tema se trata con más detalle en las clases de **Informática Teórica**.

Fuente

- https://docs.python.org/es/3/reference/executionmodel.html#exceptions
- https://docs.python.org/es/3/reference/compound_stmts.html#the-try-statement
- https://docs.python.org/es/3/library/re.html
- Para aprender algunas buenas prácticas: https://docs.python-guide.org/writing/style/