Práctica 10: Introducción a Python — Parte II

Estructuras de Datos y Algoritmos I

Autor: José Mauricio Matamoros de Maria y Campos

1. Objetivo

El alumno conocerá los fundamentos de programación en lenguage Python 3, en particular lo concerniente a graficación a fin de ser capaz de elaborar programas simples de ingeniería en este lenguaje de programación.

2. Material

Se asume que el alumno cuenta con un una computadora con arquitectura Intel x86 o compatible con interprete de python instalado (versión 3.5 o posterior).

2.1. Instalación de Puython 3 en Ubuntu Linux

Para instalar Python3 en Ubuntu Linux ejecute el siguiente comando en la terminal (se requieren privilegios de superusuario).

```
sudo apt-get update
sudo apt-get -y install python3-all
```

O bien, si se desea instalar sólo los paquetes mínimos necesarios

```
sudo apt-get -y install python3 python3-pip python3-numpy python3-matplotlib
```

3. Antecedentes

3.1. Pidiendo datos al usuario

Para pedir datos al usuario y capturarlos en una variable se utiliza la función embebida `input`.

`input(`[*prompt*]**`)`**: Si el argumento prompt está presente, se escribe a la salida estándar sin una nueva línea a continuación. La función lee entonces una línea de la entrada, la convierte en una cadena (eliminando la nueva línea), y retorna eso. Cuando se lee `EOF`, se lanza una excepción `EOFError`. Ejemplo:

```
>>> s = input('--> ')
--> Monty Python's Flying Circus
>>> s
"Monty Python's Flying Circus"
```

Si el módulo readline estaba cargado, entonces `input()` lo usará para proporcionar características más elaboradas de edición de líneas e historiales.

3.2. Otras funciones embebidas

`abs(`x**`)`**: Retorna el valor absoluto de un número. El argumento puede ser un número entero o de punto flotante. Si el argumento es un número complejo, retorna su magnitud. Si x define un método `__abs__()`, `abs(`x`)` retorna `x.__abs__()`.

chr(*i*): Retorna la cadena que representa un carácter cuyo código Unicode es el entero *i*. Por ejemplo, `chr(97)` retorna la cadena 'a', mientras que `chr(8364)` retorna la cadena «€». Esta función es la inversa de `ord()`.

El rango válido para el argumento *i* es desde 0 a 1,114,111 (0×10FFFF en base 16). Se lanzará una excepción `ValueError` si *i* está fuera de ese rango.

globals(): Retorna un diccionario que representa la tabla global de símbolos. Es siempre el diccionario del módulo actual (dentro de una función o método, este es el módulo donde está definida, no el módulo desde el que es llamada).

hex(*x*): Convierte un número entero a una cadena hexadecimal de minúsculas con el prefijo *«0x»*. Si x no es un objeto de la clase Python `int`, tiene que definir un método `__index__()` que retorne un entero.

```
>>> hex(255)
'0xff'
>>> hex(-42)
'-0x2a'
```

max(*iterable*, ∗[, *key*, *default*]**)**
max(*arg*1, *arg*2, ∗*args*[, *key*]**)**: Retorna el elemento mayor en un iterable o el mayor de dos o más argumentos.

Si un argumento posicional es dado, debe ser un iterable. El elemento mayor en el iterable es retornado. Si dos o más argumentos posicionales son indicados, el mayor de los argumentos posicionales será retornado.

Hay dos argumentos de solo palabra clave que son opcionales. El argumento `key` especifica una función de ordenación de un sólo argumento, como la usada para `list.sort()`. El argumento `default` especifica un objeto a retornar si el iterable proporcionado está vacío. Si el iterable está vacío y `default` no ha sido indicado, se lanza un `ValueError`.

Si hay múltiples elementos con el valor máximo, la función retorna el primero que ha encontrado. Esto es consistente con otras herramientas para preservar la estabilidad de la ordenación.

min(*iterable*, ∗[, *key*, *default*]**)**
min(*arg*1, *arg*2, ∗*args*[, *key*]**)**: Retorna el menor elemento en un iterable o el menor de dos o más argumentos.

Si se le indica un argumento posicional, debe ser un iterable. El menor elemento del iterable es retornado. Si dos o más argumentos posicionales son indicados, el menor de los argumentos posicionales es retornado.

Hay dos argumentos de solo palabra clave que son opcionales. El argumento `key` especifica una función de ordenación de un sólo argumento, como la usada para `list.sort()`. El argumento `default` especifica un objeto a retornar si el iterable proporcionado está vacío. Si el iterable está vacío y `default` no ha sido indicado, se lanza un `ValueError`.

Si hay múltiples elementos con el valor mínimo, la función retorna el primero que encuentra. Esto es consistente con otras herramientas que preservan la estabilidad de la ordenación.

ord(*c*): Al proporcionarle una cadena representando un carácter Unicode, retorna un entero que representa el código Unicode de ese carácter. Por ejemplo, `ord('a')` retorna el entero 97 y `ord('€')` (símbolo del Euro) retorna 8364. Esta es la función inversa de `chr()`.

print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False): Imprime objects al flujo de texto `file`, separándolos por `sep` y seguidos por `end`. `sep`, `end`, `file` y `flush`, si están presentes, deben ser dados como argumentos por palabra clave.

Todos los argumentos que no son por palabra clave se convierten a cadenas tal y como `str()` hace y se escriben al flujo, separados por `sep` y seguidos por `end`. Tanto `sep` como end deben ser cadenas; también pueden ser `None`, lo cual significa que se empleen los valores por defecto. Si no se indica objects, `print()` escribirá `end`.

El argumento file debe ser un objeto que implemente un método `write(string)`; si éste no está presente o es `None`, se usará `sys.stdout`. Dado que los argumentos mostrados son convertidos a cadenas de texto, `print()` no puede ser utilizada con objetos fichero en modo binario. Para esos, utiliza en cambio `file.write(...)`.

Que la salida sea en búfer o no suele estar determinado por `file`, pero si el argumento por palabra clave `flush` se emplea, el flujo se descarga forzosamente.

repr(*object*): Retorna una cadena que contiene una representación imprimible de un objeto. Para muchos tipos, esta función intenta retornar la cadena que retornaría el objeto con el mismo valor cuando es pasado a `eval()`, de lo contrario la representación es una cadena entre corchetes angulares («<>») que contiene el nombre del tipo del objeto junto con información adicional que incluye a menudo el nombre y la dirección del objeto. Una clase puede controlar lo que esta función retorna definiendo un método `__repr__()`.

reversed(*seq***)**: Retorna un iterator reverso. `seq` debe ser un objeto que tenga un método `__reversed__()` o que soporte el protocolo de secuencia (el método `__len__()` y el método `__getitem__()` con argumentos enteros comenzando en 0).

round(*number*[, *ndigits*]**)**: Retorna `number` redondeado a `ndigits` de precisión después del punto decimal. Si `ndigits` es omitido o es `None`, retorna el entero más cercano a su entrada.

Para los tipos integrados (built-in) que soportan `round()`, los valores son redondeados al múltiplo de 10 más cercano a la potencia menos `ndigits`; si dos múltiplos están igual de cerca, el redondeo se hace hacia la opción par (así que por ejemplo tanto round(0.5) como round(-0.5) son 0, y round(1.5) es 2).

Para un objeto number general de Python, round delega a `number.__round__`.

sorted(*iterable*, ∗, *key* = *None*, *reverse* = *False***)**: Retorna una nueva lista ordenada a partir de los elementos en iterable.

Tiene dos argumentos opcionales que deben ser especificados como argumentos de palabra clave.

`key` especifica una función de un argumento que es empleada para extraer una clave de comparación de cada elemento en iterable (por ejemplo, `key=str.lower`). El valor por defecto es `None` (compara los elementos directamente).

`reverse` es un valor boleado. Si está puesto a `True`, entonces la lista de elementos se ordena como si cada comparación fuera reversa.

Puedes usar `functools.cmp_to_key()` para convertir las funciones `cmp` a la antigua usanza en funciones `key`.

La función built-in `sorted()` está garantizada en cuanto a su estabilidad. Un ordenamiento es estable si garantiza que no cambia el orden relativo de elementos que resultan igual en la comparación. Esto es de gran ayuda para ordenar en múltiples pases (por ejemplo, ordenar por departamento, después por el escalafón de salario).

3.3. Graficación

Graficar datos es una habilidad esencial para los ingenieros. Los gráficos pueden revelar tendencias en datos y valores atípicos. Los gráficos son una forma de comunicar visualmente los resultados experimentales recabados para su efectiva presentación en reportes técnicos y científicos.

Matplotlib es una biblioteca de gráficos que puede generar líneas, gráficos de barras, histogramas y muchos otros tipos de gráficos utilizando Python.

Para graficar las funciones trigonométricas seno y coseno se comienza importando las librerías `matplotlib` y `numpy`:

Código ejemplo 1: Paquetes `numpy` y `matplotlib`

```
1 import matplotlib.pyplot as plt
2 import numpy as np
```

A continuación, se construye un conjunto 1000 de valores para x en el intervalo $[0, 4\pi)$ almacenándolos en un arreglo. Para esto se pueden usar dos funciones: `arange()` y `linspace()` de *numpy*. La función `arange()` require tres parámetros, el mínimo, el máximo y el incremento, mientras que los parámetros de la función `linspace()` son el mínimo, el máximo y el número de divisiones en los cuales hay que partir ese intervalo. La función `arange()` es la más utilizada, pero se recomienda usar `linspace()` cuando el incremento es menor a 1, por lo cual se usará esa función en el rango $[0, 4\pi)$ con 1000 divisiones. A continuación se define y como el seno de x usando la función `sin()` de *numpy*.

Código ejemplo 2: Graficación de la funció Seno

```
1 x = np.linspace(0, 4*np.pi, 1000)   # start, stop, divisions
2 y = np.sin(x)
```

Para crear el gráfico, usamos la función `plt.plot()` de *matplotlib*. Los dos argumentos son los arreglos x y y. Finalmente, la línea `plt.show()` muestra el gráfico terminado(véanse Figura 1 y Código de Ejemplo 2).

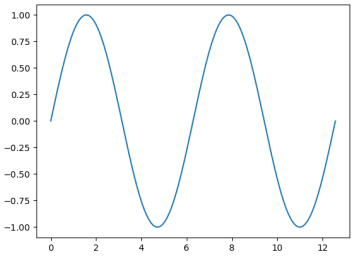

Figura 1: Gráfica de la función seno

Sin embargo ésta gráfica es muy simple y no muestra información que puede ser importante, tal como las etiquetas de los ejes con las unidades utilizadas, el título del gráfico, y la leyenda que explica qué es cada una de las líneas mostradas. *matplotlib* tiene funciones para ésto, como:

- `xlabel`: Etiqueta para el eje x.
- `ylabel`: Etiqueta para el eje y.
- `title`: Título del gráfico.
- `legend`: La leyenda como una lista de cadenas para cada una de las series graficadas.

Es más, es incluso posible agregar una segunda serie al gráfico para contrastar datos. Así, se puede agregar la función coseno al gráfico junto con las funciones anteriores para presentar ambas funciones adecuadamente (véanse Figura 2 y Código de Ejemplo 3).

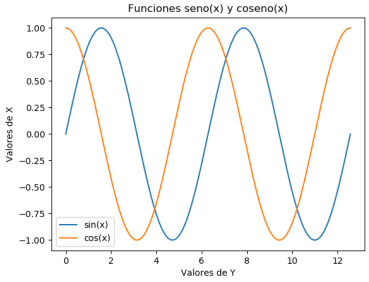

Figura 2: Gráfica de las funciones seno y coseno con leyenda y etiquetas.

Código ejemplo 3: Código para la graficación de las funciones Seno y Coseno

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 4*np.pi, 1000)   # start, stop, divisions
y = np.sin(x)
z = np.cos(x)

plt.plot(x, y, x, z)
plt.xlabel("Valores de Y")
plt.ylabel("Valores de X")
plt.title("Funciones seno(x) y coseno(x)" )
plt.legend(["sin(x)", "cos(x)"])
plt.show()
```

Nótese que, para trazar el seno y el coseno en el mismo conjunto de ejes, se incluye los dos pares de valores como argumentos para `plt.plot()`. El primer par es x, y que corresponde a la función seno, y el segundo par es x, z que corresponde a la función coseno. Si se agregan sólo tres argumentos (i.e. `plt.plot(x, y, z)`), el gráfico no mostrará el seno y el coseno en el mismo conjunto de ejes.

4. Desarrollo de la práctica

Lea cuidadosamente los problemas descritos a continuación y desarrolle los programas propuestos.

4.1. Actividad 1

El programa de la Apéndice A.1 solicita varios números al usuario e imprime el más grande de todos ellos. Ejecútelo con la línea:

```
python3 prog1.py
```

Modifique el programa de la Apéndice A.1 para que imprima el número más pequeño de los introducidos por el usuario en notación hexadecimal, ejecute el programa y responda las preguntas que se presentan a continuación.

¿Qué modificaciones se realizaron al programa? Explique [1 punto]: _____

¿Para qué sirve la función `input`? [1 punto] Explique: _____

¿Para qué sirven las funciones `max` y `min`? [1 punto] Explique: _____

¿Para qué sirve la cláusula `try/catch`? [1 punto] Explique: _____

4.2. Actividad 2

El programa de la Apéndice A.2 grafica las funciones seno y coseno en el intervalo $[0, 4\pi)$. Ejecútelo con la línea:

```
python3 prog2.py
```

Nota: Para terminar el programa debe cerrar la ventana con el gráfico.

Modifique el programa de la Apéndice A.2 para que grafique la función $y = \frac{1}{2}x^2 + 1$ como un conjunto de puntos verdes (*Hint: use "go"*).

¿Qué modificaciones se realizaron al programa? Explique [3 puntos]: _____

4.3. Actividad 3

Modifique el programa de la Apéndice A.2 para que grafique la función $y = e^{-t}\cos(2\pi t)$ como una línea punteada de color naranja (*Hint: use* `plt.plot(x, y, "-", color="orange")`).
¿Qué modificaciones se realizaron al programa? Explique [3 puntos]:

A. Código de ejemplo

A.1. Archivo `src/prog1.py`

src/prog1.py

```
import sys
from math import sin

def read_int():
  try:
    s = input("int? ")
    n = int(s)
    return n
  except:
    return None

def main(argv):
  n = read_int()
  numbers = []
  while n:
    numbers.append(n)
    n = read_int()
  print("El elemento más grande es:", max(numbers) )

if __name__ == '__main__':
  main(sys.argv)
```

A.2. Archivo `src/prog2.py`

src/prog2.py

```
#
# ## ####################################################################
import numpy as np
import matplotlib.pyplot as plt
  plt.plot(t, s, "-", color="orange")
  plt.plot(t, c, "b-")
  plt.legend(["y = cos(t)", "y = sin(t)"])
  plt.xlabel("Valores de t")
  plt.ylabel("Valores de y")
  plt.show()

def get_data():
  t = np.linspace(0, 4*np.pi, 100) # 0 a 12.64 / 100: 0, 0.1264, 0.2528 ... 12.64
  s = np.sin(t)
  c = np.cos(t)
  return t, s, c

def main():
  t, s, c = get_data()
  plot(t, s, c)

if __name__ == '__main__':
```

B. Reporte Escrito

El reporte de la práctica deberá ser entregada en un archivo en formato `PDF` siguiendo las siguientes especificaciones:

- La primera página del documento deberá ser la carátula oficial para prácticas de laboratorio disponible en `lcp02.fi-b.unam.mx/`
- El nombre del documento `PDF` deberá ser `nn-XXXX-L10.pdf`, donde:
  - `nn` es el número de lista del alumno a dos dígitos forzosos (ej. 01, 02, etc.).
  - `XXXX` corresponden a las dos primeras letras del apellido paterno seguidas de la primera letra del apellido materno y la primera letra del nombre, en mayúsculas y evitando cacofonías; es decir, los cuatro primeros caracteres de su RFC o CURP.
- El reporte consiste en un documento de redacción propia donde el alumno explica de forma concisa y a detalle las actividades realizadas en la práctica, y reportando los resultados obtenidos.
- La longitud del reporte no deberá exceder las 3 páginas, sin contar la carátula.
- El reporte deberá seguir todos los lineamientos para documentos escritos establecidos al inicio del curso.
- Todas las referencias deberán estar debidamente citadas.

IMPORTANTE: No se aceptan archivos en otros formatos ni con nombres distintos a los especificados.