2.4. Programas en Python

Este apartado va a estar dividido en dos partes principales, la primera es el análisis del problema y la otra es cómo escribir un programa en Python, sin embargo, es necesario tener claro los conceptos de algoritmos y programación en el cotexto de la guía.

> **Programming** is the process of taking an algorithm and encoding it into a notation, a programming language, so that it can be executed by a computer. (Miller & Ranum, 2018)
>
> **(Programar es el proceso de tomar un algoritmo y codificarlo en una notación (un lenguaje de programación), para que este pueda ser ejecutado en un computador)**

> **Algorithms** describe the solution to a problem in terms of the data needed to represent the problem instance and the set of steps necessary to produce the intended result. (Miller & Ranum, 2018)
>
> **(Los algoritmos describen la solución de un problema en términos de los datos necesarios para representar la instancia del problema y el conjunto de pasos necesarios para conseguir el resultado deseado)**

En resumen, para efectos prácticos de esta guía Python es solo un lenguaje de programación, es decir, un lenguaje común entre el programador y la computadora, con el cual se escriben las instrucciones para que puedan ser ejecutadas, al conjunto de aquellas instrucciones se les llama algoritmo y al proceso de escribirlo con Python se le llama programar.

2.4.1 Análisis del Problema

Antes empezar a programar se recomienda realizar un análisis previo del problema que permita esquematizarlo en términos de objetivo, entradas, salidas, y algoritmo, con el fin de tener una guía estructurada que facilite la programación ya sea con Python o con otro lenguaje.

A continuación se describirá cada paso del análisis de un programa sencillo que calcule raíces cuadradas y una recopilación de preguntas guía que serán de ayuda en esta estapa.

2.4.1.1. Objetivo

Definir cuál es el propósito del programa, es decir, "se necesita un programa para que resuelva tal problema o haga tal cosa", además es recomendable describir el tipo de interacción con el usuario.

> ***Ej:*** Se requiere un programa para calcular la raíz cuadrada de varios números, donde el usuario indica las veces que quiere hacer la operación y los números a operar, y una vez terminados los cálculos se despida.

2.4.1.2. Entradas

Definir que información se ingresará al programa y en que tipo de variable y estructura de datos debe almacenarse para ser procesada adecuadamente, tenga en cuenta que, normalmente la información ingresa como texto, pero para procesarse se deben cambiar a `integer` o `float`.

También es recomendable en esta estapa defnir los nombres que estas variables van a tener en el código.

> ***Ej:***

1. Cantidad de cálculos= `veces` - str
2. Número a operar= `num` - str

2.4.1.3. Salidas

Definir según el objetivo que salidas se deben mostrar, y/o guardarse, y en qué tipo de formato deben presentarse o almacenarse.

Igual que en las entradas sugiero que sean establecidos los nombres que tendrán las salidas en el código.

> ***Ej:***

1. Mensaje 1: "CÁLCULO DE RAÍCES CUADRADAS" - str
2. Mensaje 2: "La raíz cuadrada de num es `raiz`" - str
3. Mensaje 3: "Hemos terminado. Adios" - str

Nota: Al ser un programa sencillo la única variable de salida como tal es `raiz`, sin embargo, personalmente me gusta incluir los mensajes que se le muestran al usuario.

2.4.1.4. Algoritmo

Definir los pasos y cálculos necesarios para resolver el problema, es decir, para convertir las salidas en entradas. (Este es el algoritmo)

> **Ej:**

1. Mostrar el nombre del programa
2. Pedir la cantidad de veces
3. Pedir el número para operar
4. Hacer el cálculo de la raíz
5. Mostrar el resultado
    - Repetir paso 3, 4, 5 la cantidad de veces indicada.
6. Despedirse.

Nota: En el análisis de problema de los ejemplos, el algoritmo estará includio en el psudocódigo.

2.4.1.5. Pseudocódigo

El pseudocódigo es la secuencia de pasos detallados donde se escribe el algoritmo a términos cercanos al lenguaje de programación que se va a utilizar, para que cualquier persona lo pueda entender, es casi como un código, pero sin la sintaxis del lenguaje.

Una de las grandes ventajas de Python, es que, si se realiza bien el pseudocódigo, el programa es prácticamente igual, solo que en inglés.

Para hacer el pseudocódigo se debe:

- Definir el o los tipos de estructura de datos donde se va almacenar la información, por ejemplo, variables individuales, listas, listas anidadas, diccionarios, tuplas, matrices...
- Definir las variables de operación con su respectivo tipo, y con las variables de entrada y salida, en caso de no haberlo hecho en los pasos anteriores.

Nota: Las variables de operación, son las variables internas con las cuales vamos a realizar las operaciones del algoritmo, hay casos en las que estas son las mismas que las de entrada.

- Definir pasos y cálculos necesarios para resolver el problema como lo se haría en el código, recuerde que se debe ser lo más específico posible, porque estas son las instrucciones que va a recibir la computadora.

Para realizar esta etapa es conveniente estructurar el programa en varias secciones, como se propone a continuación.

Estructura del programa

Normalmente estructuro el programa en los siguientes pasos, aunque claramente no es una camisa de fuerza y puede cambiar dependiendo del tipo de programa, me gusta organizar las secciones de esta manera, ya que, me facilita el proceso de lectura, pruebas y depuración del código.

-2. Importar de módulos / Import Statements
-1. Definir de funciones auxiliares/ Helper Functions

1. Programa principal / Main program

```
1. Ingreso - lectura y almacenamiento de datos
2. Procesamiento o caja negra. (Los pasos del algoritmo)
    ...
    n. Impresión- almacenamiento de resultados.
```

(n+1). Ejecutar.

Nota: Cuando se manejan bucles donde haya interacción con el usuario, se suele presentar una mezcla entre las partes del programa principal, como vio en ejemplo.

> Ej:

```
"""
-2. Importar math.
-1. Crear función para calcular raíz= raiz_cuadrada
0. Definir programa principal.
    1. Imprimir: Mensaje 1.
    2.Ingresar veces.
    3. Convertir veces a integer= times
    4. Ingresar num y calcular raíz.
    Por cada vez en times haga:
        4.1 Ingresar num.
        4.2 Convertir num a float= numero.
        4.3 Calcular raíz= raiz_cuadrada(numero)
```

```
        4.4 Imprimir: Mensaje 2.
    5. Imprimir: Mensaje 3.
6. Ejecutar programa principal.
"""
```

- Preguntas guía

Estas son algunas preguntas guía que es bueno tener en cuenta para realizar un análisis completo.

- ¿Cómo se van a ingresar los datos de entrada?: Si estos van a ser digitados hay que tener la opción de corregir los datos ingresados, o si se van a extraer de una base de datos o de un archivo ya existente, para lo cual hay que tener en cuenta la el tipo de archivo y su ubicación.
- ¿Es un programa personal o lo usará alguien más?: De esto depende que tan detalladas deben ser las instrucciones, el nivel de validación de datos o si es necesaria una interfaz (esta últimas no está en el alcance de la guía).
- ¿Las salidas van a ser utilizados para un cálculo posterior, se deben almacenar, o solo hay que conocerlos?: Hay que saber en qué tipo de archivo es más eficiente que se guarden los resultados.
- ¿Cómo se van a almacenar los datos en una lista de listas o en listas por separado?
- ¿El archivo que contiene los datos debe tener un formato en especial?: Es importante conocer la estructura del archivo que se va a manejar, ya que, si no tiene un formato estándar el programa deberá reconocer y organizar los datos para luego procesarlos.

2.4.2 Código en Python

Los programas en Python son archivos de texto plano, que se graban con la extensión .py, pueden crearse desde cualquier editor de texto plano, pero los IDE para Python tienen varias funcionalidades que facilitan escribir el código, como se describen en el apartado `*` 1.3. Algunos IDE.

Al momento de enfrentar un problema complejo, se debe tener en cuenta que:

- Tener por lo menos un ejercicio pequeño ya resuelto, para crear un set de datos de prueba y poder verificar los resultados a medida que se termine una sección grande, se cree funciones auxiliares, o para hacer pruebas unitarias con el módulo unittest.
- Bautizar las variables de tal manera que sean explicativas por sí mismas, y usar un estándar para bautizarlas, con el fin de mejorar la legibilidad del código y en caso de que se quiera publicar en un foro lo mejor es nombrar las variables en inglés.

- Hacer uso de los comentarios, sobretodo explicando bloques de código que puedan ser complejos, recuerde que un buen código es el que se puede leer fácilmente, sobre todo cuando lo se retoma después de un tiempo o cuando se trabaja en un proyecto colaborativo.

- Si no se teiene idea de cómo abordar el problema, plantear el algoritmo o alguna parte en específico del código, la forma más rápida de enocntra la solución es busarcar en google o postearlo en un foro de la comunidad.

Nota: Una de las grandes ventajas de Python es que tiene una comunidad cooperativa enorme trabajando en proyectos open source y activa en foros.

A continuación, se muestra el código de la Calculadora de raíces cuadradas, en cual se mostrará cómo pasar del pseudocódigo al código en Python, y se explicará sus partes más importantes.

Ejemplo 1: Calculadora de raíces.

```
# Ejemplo1_Calculadora_de_raices.py

# By Leonardo Rivera

## -2. IMPORT STATEMENTS:
import math

## -1. HELPER FUNCTIONS:

def raiz_cuadrada(n):
    raiz=math.sqrt(n)

    return raiz

## 0. MAIN PROGRAM

def main():
    print("CÁLCULO DE RAÍCES CUADRADAS") # 1
    veces= input("Cuántas raíces cuadradas va a calcular: ") # 2
    times= int(veces) # 3

    #4
    for j in range (times):
        num=input("Dígame el número: ")# 4.1
        numero= float(num)# 4.2
        raiz= raiz_cuadrada(numero) #4.3
        print("La raíz cuadrada de ",numero," es:",raiz) #4.4

    print("Hemos terminado. Adiós") #5
```

```
## 6. EJECUTAR:
if __name__=='__main__':
    main()
```

Descargar * Ejemplo-1_Calculadora de raíces.py *

```
# Pase aquí el Ejemplo_1
```

2.4.2.1. Import Statements

En muchas ocasiones será necesario importar módulos del lenguaje que no se ejecutan por defecto. En este ejemplo se usa el módulo `math`, que tiene funciones matemáticas más allá de las más básicas como la raíz cuadrada.

Como práctica en el siguiente bloque ejecute los siguientes comandos para ver los métodos que el módulo `math` conitene y la ayuda para la función de la raíz cuadrada.

```
>>> import math
>>> dir(math)
>>> help(math.sqrt)
```

```
## Escriba aquí:
```

2.4.2.2. Helper Functions

En esta sección se ubica todas las funciones que harán tareas específicas para el cuerpo principal del programa. En este caso se tiene una función, que toma como entrada el parámetro `n` y devuelve como resultado la variable raiz que ha sido calculada.

```
def raiz_cuadrada(number):
    raiz = math.sqrt(number)

    return raiz
```

La primera línea de la función inicia con la palabra clave `def` (quiere decir que se va a definir la función), seguida del nombre de la función y paréntesis. Dentro de los paréntesis se especifica los parámetros de la función, es decir, los valores que debe recibir para hacer su tarea. Si **no** son necesarios los parámetros, los paréntesis estarán vacíos ( ). Se termina la primera línea con `:`

La segunda línea está indentada cuatro espacios, y es la que ejecuta la operación matemática. A la variable `raíz` se le asigna la raíz cuadrada de number.

La última línea empieza con la palabra clave `return`. En esta se especifica que la función retornará el valor de la variable `raíz` a la parte del programa que la haya llamado.

2.4.2.3_Main Program

En esta sección se desarrolla el flujo del programa principal. Cuando se ejecute el programa, es esta la sección que controla toda la ejecución.

Siguiendo la ejecución de nuestro pequeño programa de ejemplo, la primera acción es presentar en pantalla el anuncio '*CALCULO DE RAICES CUADRADAS*'. La segunda línea toma en la variable `veces` el valor que el usuario digite, después de presentar en pantalla la pregunta: '*Cuantas raíces cuadradas va a calcular?:*'. Como el valor almacenado en la variable `veces` es una variable de texto en la siguiente línea convertimos ese valor en un número entero y lo almacenamos en la variable `times`.

Justo después encontramos un bloque de repetición for, en el que usamos la variable `times` para indicarle al ciclo cuantas veces se debe ejecutar. Observemos que todo el bloque de instrucciones que deben repetirse bajo el control del for está indentado cuatro espacios debajo de la palabra clave for.

De aquí en adelante pueden descifrar fácilmente qué hace el resto del programa.

2.4.2.4. Ejecutar

En esta sección se declara que al ejecutar el archivo **Ejemplo-1_Calculadora-de-raices**, la función que controlará la ejecución de todo el archivo será aquella cuyo nombre sea main, es decir, el programa principal.

Anterior	-	*Siguiente*

Home