Un primer programa

Juan Álvarez

El objetivo de la programación es escribir programas que resuelvan problemas. A modo de ejemplo, se abordará el problema de calcular los resultados en porcentajes de un plebiscito entre dos opciones. El problema es sencillo de resolver siempre que se conozca la cantidad total de votos de cada una de las opciones.

Consecuentemente, un programa que resuelva el problema debe establecer un diálogo ("conversación") con una persona (usuario) que permita obtener los datos necesarios y escribir los resultados de una manera legible. Por ejemplo, un posible diálogo es el siguiente:

```
votos de opción 1?
470
votos de opción 2?
330
porcentaje de opción 1: 58.75
porcentaje de opción 2: 41.25
```

Los números 470 y 330 deben ser ingresados por la persona que usa el programa. Todas las otras líneas las escribe el computador para solicitar el ingreso de los números y mostrar los resultados.

Algoritmo

La Real Academia de la Lengua española (RAE) define algoritmo como un "conjunto ordenado y finito de operaciones que permite hallar la solución de un problema". El ejemplo clásico es considerar una receta de cocina como un algoritmo porque contiene las instrucciones que debe seguir una persona para preparar un determinado plato. De hecho, las recetas están redactadas de una manera que pueda ser realizada por cualquier persona con una mínima experiencia en cocina.

Si queremos instruir al computador para que resuelva un problema, entonces el algoritmo correspondiente debe formularse en términos de sus capacidades. Respecto del problema de calcular los porcentajes de hombres y mujeres, el diálogo que queremos que establezca el programa permite deducir cuál es el algoritmo computacional que lo resolverá. En otras palabras, a partir del diálogo podemos redactar las acciones que debe realizar (ejecutar) el computador para obtener los datos, calcular los resultados intermedios y mostrar los resultados finales.

El algoritmo computacional para resolver nuestro problema se puede expresar de distintas maneras y puede redactarse combinando palabras y fórmulas matemáticas. Por ejemplo, el algoritmo puede formularse de la siguiente manera:

1. escribir (mostrar) una línea en la pantalla con la frase "**votos de opción 1**?"
2. leer (obtener) el número ingresado por la persona usando el teclado. Y asignar el nombre simbólico v1 al número.
3. escribir (mostrar) en la pantalla la frase "**votos de opción 2**?"
4. **leer** el número ingresado por la persona (nombre simbólico v2)
5. calcular el porcentaje de v1 con respecto a la suma de v1 y v2 como

$$\frac{v1}{v1 + v2} \times 100$$ (nombre simbólico p)

6. escribir una línea con la frase "porcentaje de opción 1:" y el valor de p
7. escribir una línea con "porcentaje de opción 2:" y el número que resulta de calcular 100-p

El algoritmo se ha expresado de una manera sistemática describiendo siete acciones que debe realizar el computador. Cada acción se ha redactado con un verbo, que indica la orden que debe realizar el computador, y una explicación complementaria. Pero el algoritmo podría expresarse de una manera distinta, lo que podría prestarse para confusiones y ambigüedades. Por lo tanto, se necesita expresar los algoritmos computacionales de una manera precisa y no ambigua de modo que no se preste a distintas interpretaciones.

Programa

Un programa es un algoritmo computacional expresado en un lenguaje de programación. Considerando que un lenguaje tiene una sintaxis con estrictas reglas gramaticales y una semántica (significado, interpretación) muy precisa, entonces un programa no se presta para confusiones y el computador interpreta y realiza las instrucciones de una manera única.

Existen muchos lenguajes de programación. Python es un lenguaje multipropósito, es decir apropiado para resolver problemas de distinto tipo. Python es también moderno, en el sentido que fue diseñado para operar adecuadamente en el entorno Internet/Web. A diferencia de Java, que es un lenguaje que tiene similares características, Python es más pedagógico y resulta apropiado para aprender a programar, puesto que es más sencillo y los programas se pueden escribir de manera más concisa y legible.

El algoritmo para resolver el problema de los porcentajes se expresa en el lenguaje Python en un programa con las siguientes instrucciones:

```
print("votos de opción 1?")
v1 = int(input())
print("votos de opción 1?")
v2 = int(input())
p = v1/(v1+v2)*100
print("porcentaje de opción 1:",p)
print("porcentaje de opción 2:",100-p)
```

Se observa que un programa se escribe combinando palabras en inglés (print, input, int), frases textuales (encerradas entre comillas) y elementos de la notación matemática: variables como v1, v2 y p; constantes como 100; expresiones o fórmulas, por ejemplo v1/(v1+v2)*100 y 100-p; y funciones como input(), int(…) y print(…). En general se usan letras minúsculas porque facilitan la legibilidad de los programas, aunque también se pueden usar letras mayúsculas, por ejemplo, para nombrar las variables.

Por otra parte, cada una de las siete acciones del algoritmo se tradujo en una instrucción de Python, situación que no siempre es así. A continuación, se explicará en detalle las instrucciones del programa

Instrucción de asignación

En el programa el porcentaje de votos de la primera opción del plebiscito se calcula con la instrucción de asignación **`p=v1/(v1+v2)*100.`** El objetivo de una instrucción de asignación es asignar un valor a una variable. Al respecto, al igual que en las matemáticas, una variable es la representación simbólica de un número. Por ejemplo, las variables v1, v2 y p representan la cantidad de votos de las dos opciones del pelbiscito y el porcentaje de hombres respectivamente.

Desde el punto de vista computacional, una variable representa una ubicación (casilla) de la memoria del computador donde se puede guardar un número. A cada variable se le asocia un tipo según el valor que se almacena: tipo **int** si es un número entero, tipo **float** si es un número real (número con un punto decimal). Por otra parte, el nombre o identificador de una variable debe comenzar con una letra seguida de letras o dígitos o el subrayado (guion bajo). Por ejemplo, v1, V1, votos1, votos_opcion1 o votosOpcion1 son identificadores válidos para representar la cantidad de votos de la primera opción del plebiscito. Independiente del tamaño del identificador, lo importante es que el nombre sea descriptivo, significativo y nemotécnico.

De la instrucción **`p=v1/(v1+v2)*100`** se puede deducir la sintaxis y la semántica de la instrucción de asignación. En efecto, la sintaxis o forma general es **`variable=expresión`** (con o sin espacios antes y después del signo igual y de cada operador). El significado, o semántica, de la instrucción establece que primero se debe evaluar la expresión aritmética y, segundo, guardar el resultado del cálculo en la variable.

En el programa hay otras dos instrucciones de asignación: v1=int(input()) y v2=int(input()). En estos casos, la expresión `int(input())` entrega como resultado el valor que se lee con la función input y que se convierte a número entero con la función int.

Expresión aritmética

Una expresión aritmética debe escribirse de acuerdo a estrictas reglas de sintaxis. En primer lugar, una expresión se escribe en una línea y no en varios niveles como en la notación matemática. Por ejemplo, A dividido por B se escribe A/B y A elevado a B se escribe A**B. Por otra parte, todos los operadores deben indicarse explícitamente. Por ejemplo, A multiplicado por B se escribe A*B y no AB, que se interpretaría como el nombre de una variable.

Una expresión aritmética también tiene reglas semánticas que aseguran un resultado único. Tal como ocurre con las reglas algebraicas habituales, primero se evalúa el operador de elevación a potencia (**). La segunda prioridad la tienen los operadores unarios (+, -). La tercera prioridad corresponde a los operadores "multiplicativos" (*, /, //, %). Y la última prioridad los operadores "aditivos" (+, -). Por ejemplo, en la expresión –A+B*C**D, primero se calcula –A, después C**D, en seguida B*(C**D) y finalmente la suma entre –A y B*C**D.

La expresión –A+B*C**D también podría escribirse usando paréntesis: (-A)+(B*(C**D)). Los paréntesis se pueden utilizar para modificar o confirmar el orden de prioridades. Por ejemplo, en (A+B)*C los paréntesis alteran el orden de evaluación de los operadores. En cambio, en A+(B*C) los paréntesis confirman las prioridades predefinidas. En caso de operadores de la misma prioridad, la evaluación es de izquierda a derecha. Por ejemplo, A-B+C es equivalente a (A-B)+C.

Para determinar el tipo del resultado de la evaluación de una expresión aritmética también hay reglas predefinidas muy precisas. En caso de realizarse una operación entre dos operandos del mismo tipo, el resultado es del tipo común. Por ejemplo, como 2 y 3 son de tipo int entonces 2+3 es 5 (tipo int). Asimismo, 2.0+3.0 es 5.0 (float). Por otra parte, si se realiza una operación entre operandos de distinto tipo el resultado será real (tipo float). Por ejemplo, 2.0+3 es 5.0 (tipo float) y 2+3.0 también es 5.0. Por otra parte, en el caso de 2**-3, en que los dos operandos son de tipo int, el resultado 0.125 de tipo float.

La operación de división merece una explicación especial. El resultado del operador / es siempre real. Por ejemplo, aunque las constantes 7 y 2 son de tipo int, sin embargo 7/2 es 3.5 y 6/2 es 3.0. Si se necesita un resultado entero truncado entonces se debe usar el operador //. Por ejemplo, 7//2 es 3 y 6//2 es 3. El resto o residuo de la división entre enteros se obtiene con el operador %. Por ejemplo, 7%2 es 1 y 6%2 es 0.

Todas las reglas anteriores explican que el valor de p=v1/(v1+v2)*100 será siempre un número real (tipo float). Así, si v1 es 470 y v2 es 330, entonces p será 58.75 (tipo float) y 100-p será 41.25 (tipo float). Al calcular con otros valores de v1 y v2 los porcentajes podrían tener más o menos dígitos después del punto decimal.

Funciones de lectura y escritura

La instrucción **`print("votos de opción 1?")`** escribe o muestra una línea en la pantalla con la frase escrita entre comillas. El nombre print es un homenaje a los primeros lenguajes de programación de la época en que los programas imprimían los resultados en papel.

La instrucción **`print("porcentaje de opción 1:",100-p)`** permite apreciar que print es una función que recibe como argumentos una o más expresiones. Su semántica es evaluar las expresiones y escribir los resultados en una misma línea, pero separadas por un espacio. Si una expresión es una frase encerrada entre comillas entonces se escribe la frase textualmente. Después de escribir los valores de todas las expresiones la función print posiciona el cursor al comienzo de la siguiente línea en la pantalla.

La instrucción **v1=int(input())** lee un número entero ingresado por una persona y lo asigna a la variable h. El mismo efecto se lograría con las instrucciones **s=input()** y **v1=int(s)**. Al respecto, input es una función que no recibe ningún argumento y que entrega como resultado lo que la persona ingresa usando el teclado. Por ejemplo, si la persona ingresa 470 y la tecla enter (o return), entonces input() entrega como resultado '470' que es un valor del tipo str. Los valores del tipo str se pueden encerrar entre apóstrofes o dobles comillas, es decir '470' es equivalente a "470".

La función int recibe un argumento y entrega como resultado el número entero correspondiente. Por ejemplo, int('470') entrega 470 (de tipo int) e int(4.7) entrega 4, es decir el número real se trunca. Por otra parte, la función float convierte valores a números reales (de tipo float). Por ejemplo, float('4.7') entrega 4.7 y float(47) entrega 47.0. Existe también la función str, de modo que str(470) entrega '470' y str(4.7) entrega '4.7', ambos resultados de tipo str.

El diálogo, que establece el programa con un usuario, se podría modificar para ingresar los datos en la misma línea en que aparece la pregunta y, además, entregar los resultados con dos decimales y redondeados. Por ejemplo:

```
votos opción 1?2
votos opción 2?1
porcentaje de opción 1: 66.67
porcentaje de opción 2: 33.33
```

La función input admite un argumento de tipo str que se escribe en la pantalla delante del valor que ingresará el usuario. Así por ejemplo, la instrucción **v1=int(input("votos de opción 1?"))** es similar a las instrucciones **print("votos de opción 1?")** y **v1=int(input())** con la diferencia que el usuario ingresa el valor en la misma línea del mensaje. El programa será el siguiente:

```
v1 = int(input("votos de opción 1?"))
v2 = int(input("votos de opción 2?"))
p = v1/(v1+v2)*100
pr = round(p,2)
print("porcentaje de opción 1:",pr)
print("porcentaje de opción 2:",100-pr)
```

La instrucción pr=round(p,2) calcula el porcentaje redondeado a dos decimales. La función predefinida round recibe 2 argumentos: el valor a redondear y la cantidad de decimales. Por ejemplo, round(66.666,2) entrega el resultado 66.67.