Introducción a los lenguajes de programación II

Operadores, expresiones aritméticas y evaluación

Jonatan Goméz Perdomo, Ph.D.
jgomezpe@unal.edu.co

Arles Rodríguez, Ph.D.
aerodriguezp@unal.edu.co

Camilo Cubides, Ph.D.(c)
eccubidesg@unal.edu.co

Grupo de investigación en vida artificial – Research Group on Artificial Life – (Alife)
Departamento de Ingeniería de Sistemas e Industrial
Facultad de Ingeniería
Universidad Nacional de Colombia

Agenda

Agenda

Operadores aritméticos I

Para los datos de tipo numérico se pueden utilizar los siguientes operadores infijos, a excepción del operador - que puede actuar también como un operador prefijo:

- `+` : Suma de dos valores, por ejemplo, cuando se evalúa la expresión `2.0 + 3.0` se obtiene el valor 5.0.
- `-` : Resta de dos valores, por ejemplo, cuando se evalúa la expresión `2.0 - 3.0` se obtiene el valor -1.0. También se utiliza para cambiar el signo de un número si se utiliza con un sólo operando, por ejemplo, cuando se evalúa la expresión `-23` se obtiene el valor -23.

Operadores aritméticos II

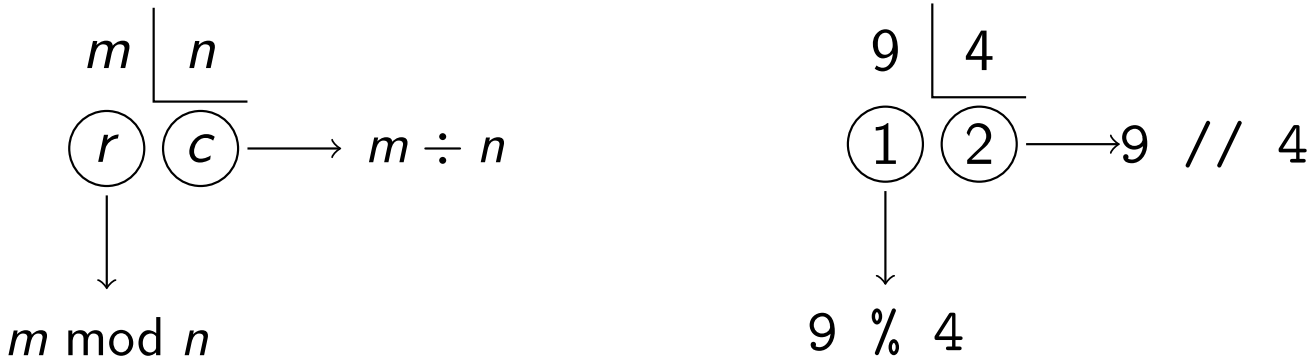
- `*` : Multiplicación de dos valores, por ejemplo, cuando se evalúa la expresión `2.0 * -3.0` se obtiene el valor -6.0. La multiplicación es explicita, es decir no se puede escribir una expresión como `(2.0)(-3.0)`, esto se debe escribir como `(2.0)*(-3.0)`.
- `/` : División de dos valores, independiente de si alguno de los operandos es real, retorna la división exacta, por ejemplo, en Python cuando se evalúa la expresión `-3.0/2` se obtiene el valor -1.5, al igual que si se ejecutará la instrucción `-3/2`. El valor del segundo operando debe ser distinto de 0.
- `//` : División entera de dos valores, se obtiene la parte entera de la división exacta, por ejemplo, cuando se evalúa la expresión `-3//2` se obtiene el valor -1. El valor del segundo operando debe ser distinto de 0.

Operadores aritméticos III

% : El resto de la división de dos números que **deben ser enteros**, representa la operación matemática

$$m \bmod n = r,$$

por ejemplo, cuando se evalúa la expresión `9 % 4` se obtiene el valor 1, que es lo mismo que $9 \bmod 4 = 1$, el residuo de dividir 9 entre 4.



Otro ejemplo es `3 % 4 = 3`, el residuo de dividir 3 entre 4 es 3.

Agenda

Operadores de asignación I

Para asignar valores a variables se pueden utilizar los siguientes operadores infijos:

- `=` : Asignación. La parte de la izquierda que debe ser una variable. Sirve para almacenar un dato en una variable. Asigna el valor de evaluar la parte de la derecha a la variable de la parte de la izquierda. Por ejemplo, cuando se evalúa la expresión `pi = 3.14159265`, entonces se almacena el valor 3.14159265 en la variable `pi`.

Operadores de asignación II

`+=` : Asignación con suma. La parte de la izquierda debe ser una variable. Suma la evaluación de parte de la derecha con el valor almacenado en la variable definida en la parte de la izquierda y guarda el resultado en la variable de parte de la izquierda. Por ejemplo, la expresión `x += 2`, es equivalente a la expresión `x = x + 2`.

`+= 1` : esta operación es una de las más utilizadas y se usa para incrementar el valor de la variable en 1; la expresión `i += 1` es equivalente a la expresión `i = i + 1`; ésta asignación NO se puede utilizar dentro de una expresión.

Operadores de asignación III

`-=` : Asignación con resta. La parte de la izquierda debe ser una variable. Resta al valor almacenado en la variable definida en la parte de la izquierda el resultado de la evaluación de parte de la derecha y guarda el resultado en la variable de parte de la izquierda. Por ejemplo, la expresión `x -= 2`, es equivalente a la expresión `x = x - 2`.

`-= 1` : esta operación es una de las más utilizadas y se usa para decrementar el valor de la variable en 1; la expresión `i -= 1` es equivalente a la expresión `i = i - 1`; ésta asignación NO se puede utilizar dentro de una expresión.

Operadores de asignación IV

`*=` : Asignación con multiplicación. La parte de la izquierda debe ser una variable. Multiplica el valor almacenado en la variable definida en la parte de la izquierda con la evaluación de parte de la derecha y guarda el producto en la variable de parte de la izquierda. Por ejemplo, la expresión `x *= 2`, es equivalente a la expresión `x = x * 2`.

Operadores de asignación V

`/=` : Asignación con división. La parte de la izquierda debe ser una variable. Divide el valor almacenado en la variable definida en la parte de la izquierda entre el valor de la evaluación de la parte de la derecha y guarda el resultado en la variable de parte de la izquierda. Por ejemplo, la expresión `x /= 2`, es equivalente a la expresión `x = x / 2`. El valor de la evaluación de la parte de la derecha debe ser distinto de 0.

`//=` : Asignación con división entera. La parte de la izquierda debe ser una variable. Divide de forma entera el valor almacenado en la variable definida en la parte de la izquierda entre el valor de la evaluación de la parte de la derecha y guarda el resultado entero en la variable de parte de la izquierda. Por ejemplo, la expresión `x //= 3`, es equivalente a la expresión `x = x // 3`. El valor de la evaluación de la parte de la derecha debe ser distinto de 0.

Operadores de asignación VI

`%=` : Asignación con residuo. La parte de la izquierda debe ser una variable. Calcula el residuo de dividir el valor almacenado en la variable definida en la parte de la izquierda entre el valor de la evaluación de la parte de la derecha y guarda el resultado en la variable de parte de la izquierda. Por ejemplo, la expresión `x %= 2`, es equivalente a la expresión `x = x % 2`. El valor de la evaluación de la parte de la derecha debe ser distinto de 0.

Escritura de la Ley de Gravitación Universal I

Problema

La ley de Gravitación Universal

$$F = G\frac{m_1 m_2}{r^2}$$

aplicada a dos cuerpos de masas m_1 y m_2, separados una distancia r, permite calcular la fuerza F con la cual los cuerpos se atraen. Esta ley utiliza adicionalmente la constante de gravitación universal

$$G = 6.67384 \times 10^{-11}\frac{Nm^2}{Kg^2}.$$

Escriba en Python, en una línea, una asignación a una variable de tipo real de tal manera que quede almacenada la fuerza F con la cual se atraen dos cuerpos que están a una distancia dada, escribiendo explícitamente el valor de la constante de gravitación universal y suponiendo que las variables involucradas han sido creadas e inicializadas previamente.

Escritura de la Ley de Gravitación Universal II

Las siguientes expresiones no son soluciones al problema de traducción de la ley de gravitación universal ¿por qué?.

Solución

```
F: float = 6,67384E-11 [(m1 m2) / (r^2)]
```

Solución

```
F = 6.67384e-11 * m1 * m2 / (r * r);
```

Solución

$$\texttt{F = 6.67384} \times 10^{-11} \texttt{ (m1 * m2 / r * r)}$$

Solución

```
F = 6,67384^-11 * m1: float * m2: float ÷ r: float^2
```

Escritura de la Ley de Gravitación Universal III

Las siguientes expresiones son soluciones al problema de traducción de la ley de gravitación universal.

Solución

```
F: float = 6.67384E-11 * ((m1 * m2) / (r * r))
```

Solución

```
F = 6.67384e-11 * m1 * m2 / r ** 2
```

Solución

```
F: float = (6.67384e-11 * m1 * m2) / (r * r)
```

Solución

```
F = 6.67384E-11 * (m1 * m2) * (1 / (r * r))
```

Agenda

De entero a real I

De entero a real: dado un dato o una variable de tipo entero, si se opera o se asigna el dato o la variable con un dato o una variable de tipo real, entonces al realizar la operación o la asignación, el dato entero se convierte (se promueve) a un dato de tipo real de forma automática, simplemente agregándole la parte decimal ".0".

De entero a real II

Ejemplo

Para las asignaciones

```
n = 1
x = float(n)
y = float(0)
z = float(-2)
```

se tiene que las variables almacenan los valores:

n	1
x	1.0
y	0.0
z	−2.0

De entero a real III

Ejemplo

Las siguientes operaciones son equivalentes

- `-2 + 1.0` $\Leftrightarrow$ `-2.0 + 1.0`
- `0.0 * 5` $\Leftrightarrow$ `0.0 * 5.0`
- `(5 // 2) * 2.0` $\Leftrightarrow$ `2 * 2.0` $\Leftrightarrow$ `2.0 * 2.0`
- `(5.0 / 2) * 2` $\Leftrightarrow$ `(5.0 / 2.0) * 2` $\Leftrightarrow$ `2.5 * 2` $\Leftrightarrow$ `2.5 * 2.0`

De real a entero I

De real a entero: dado un dato o una variable de tipo real, si se aplica la función `int( )` al dato o la variable, entonces el valor del dato o la variable se convierte (se promueve) a un dato de tipo entero, simplemente elimina la parte decimal del real y dejando la parte entera.

De real a entero II

Ejemplo

Para las siguientes asignaciones en Python

```
x = 1.0
y = -2.5
n = int(x)
m = int(y)
p = int(3.14159265)
```

se tiene que las variables almacenan los valores:

x	1.0
y	−2.5
n	1
m	−2
p	3

De real a entero III

Ejemplo

Las siguientes operaciones son equivalentes

- `2 * int(2.5)` $\Leftrightarrow$ `2 * 2`
- `int(-3.14) * int(5.5)` $\Leftrightarrow$ `-3 * 5`
- `2 / (2.5 - int(2.5))` $\Leftrightarrow$ `2 / (2.5 - 2)` $\Leftrightarrow$ `2 / (0.5)` $\Leftrightarrow$ `2 / 0.5` $\Leftrightarrow$ `2.0 / 0.5`

De real a entero IV

Ejemplo

Las siguientes sentencias son equivalentes

```
x = 0.4
y = 2.5
n = int(x) * int(y)
```

⇕

```
x = 0.4
y = 2.5
n = 0 * 2
```

Agenda

Operadores lógicos

`not` : Operador $\neg$ de la negación en Python.

$$\texttt{not}\ \alpha \Leftrightarrow \neg\alpha$$

`and` : Operador $\wedge$ de la conjunción en Python.

$$\alpha\ \texttt{and}\ \beta \Leftrightarrow \alpha \wedge \beta$$

`or` : Operador $\vee$ de la disyunción en Python.

$$\alpha\ \texttt{or}\ \beta \Leftrightarrow \alpha \vee \beta$$

Agenda

Operadores de igualdad y relacionales I

== : Devuelve *V* si dos valores son iguales.

$$\alpha \texttt{ == } \beta \Leftrightarrow \alpha = \beta$$

!= : Devuelve *V* si dos valores son distintos.

$$\alpha \texttt{ != } \beta \Leftrightarrow \alpha \neq \beta$$

> : Mayor que, devuelve *V* si el primer operador es estrictamente mayor que el segundo.

$$\alpha \texttt{ > } \beta \Leftrightarrow \alpha > \beta$$

Operadores de igualdad y relacionales II

`<` : Menor que, devuelve *V* si el primer operador es estrictamente menor que el segundo.

$$\alpha \texttt{ < } \beta \Leftrightarrow \alpha < \beta$$

`>=` : Mayor o igual, devuelve *V* si el primer operador es mayor o igual que el segundo.

$$\alpha \texttt{ >= } \beta \Leftrightarrow \alpha \geq \beta$$

`<=` : Menor igual, devuelve *V* si el primer operador es menor o igual que el segundo.

$$\alpha \texttt{ <= } \beta \Leftrightarrow \alpha \leq \beta$$

Operadores de igualdad y relacionales III

Ejemplo

Para decidir si el valor $a \in (0, 1] = \{x : (x \in \mathbb{R}) \wedge (0 < x \leq 1)\}$ en el lenguaje Python se utiliza la sentencia

```
(0 < a and a <= 1)
```

La anterior expresión es equivalente a

```
(0 < a <= 1)
```

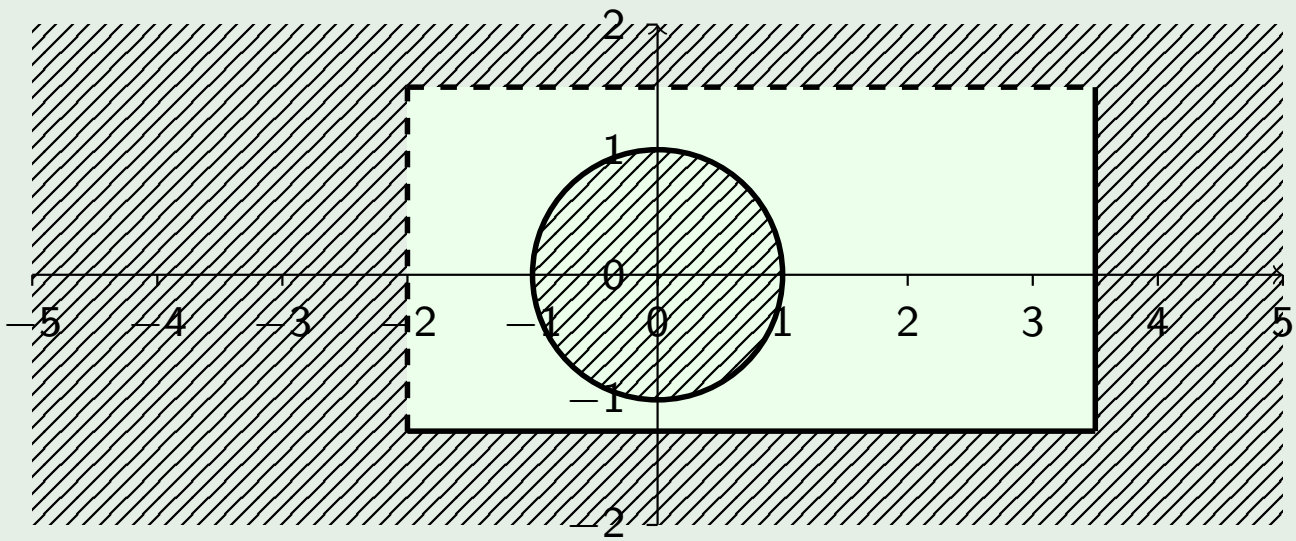
en Python, pero ésta NO es sintácticamente valida en lenguajes como Java o C++.

Operadores de igualdad y relacionales IV

Ejemplo

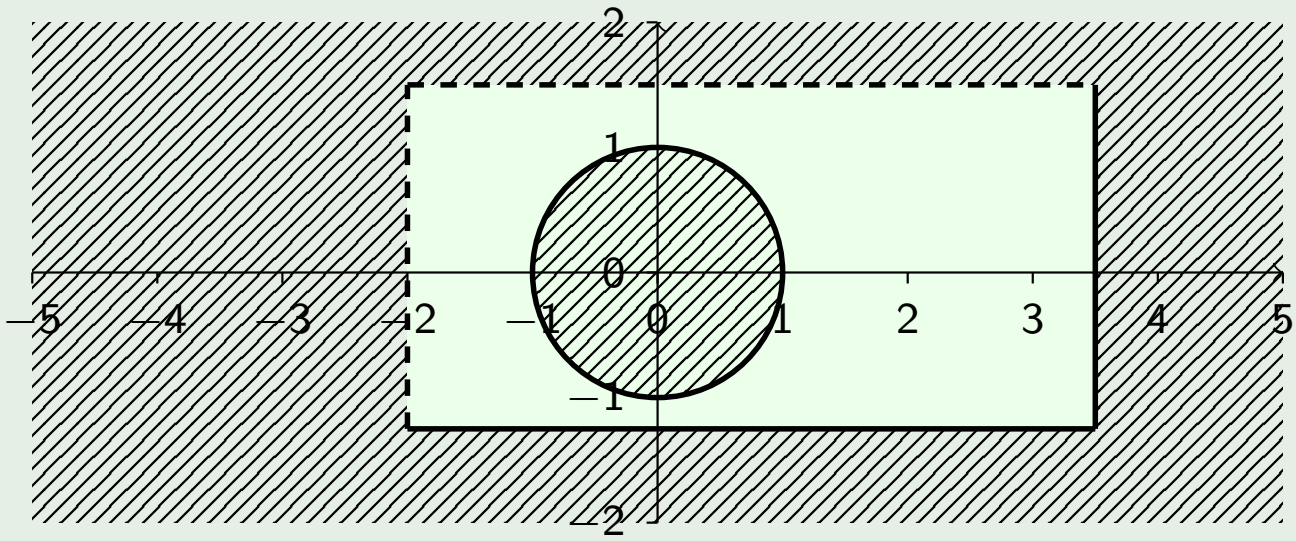
Para decidir si la pareja ordenada (a, b) pertenece al siguiente conjunto, con universo $\mathbb{R}^2$

$$\overline{[-2, 3.5) \times (-1.25, 1.5]} \cup \left\{(x, y) : x^2 + y^2 \leq 1\right\}$$



Operadores de igualdad y relacionales V

Ejemplo



en el lenguaje Python se puede utilizar la sentencia

```
not(a >= -2 and a < 3.5 and b > -1.25 and b <= 1.5)
or (a * a + b * b <= 1)
```

Agenda

Precedencia de operadores I

En la siguiente tabla se presenta la prioridad de los principales operadores de Python, la prioridad más alta es la 1 y la más baja es la 9.

Operador(es)	Prioridad
`()`	1
`not` `-`*(signo menos)* `+`*(signo más)* `**`*(potencia)*	2
`*` `/` `//` `%`	3
`+` `-`	4
`<` `>` `<=` `>=`	5
`==` `!=`	6
`and`	7
`or`	8
`=` `+=` `-=` `*=` `/=` `//=` `%=`	9

Table: Precedencia de los operadores en Python.

Precedencia de operadores II

Ejemplo

Hallar el valor de la siguiente expresión teniendo en cuenta la prioridad de operadores y que los operandos son números enteros

```
42 // 6 + 7 * 3 - 39
```

i) $(42//6) + 7 * 3 - 39$ ($//$ prioridad 3)

ii) $(42//6) + (7 * 3) - 39$ ($*$ prioridad 3)

iii) $((42//6) + (7 * 3)) - 39$ ($+$ prioridad 4)

iv) $(((42//6) + (7 * 3)) - 39)$ ($-$ prioridad 4)

Precedencia de operadores III

Ejemplo (continuación)

$$\begin{aligned} 42 \;//\; 6 + 7 * 3 - 39 &= 7 + 7 * 3 - 39 \\ &= 7 + 21 - 39 \\ &= 28 - 39 \\ &= -11 \end{aligned}$$

Obsérvese la diferencia entre este ejemplo y siguiente ejemplo (¡de la vida real!), donde la diferencia de los resultados está dada por la prioridad con la cual se evalúan los operadores.

Precedencia de operadores IV

Ejemplo

Cuál será la forma correcta de escribir en Python la operación que aparece como miembro izquierdo de la igualdad que se muestra en la siguiente imagen, de tal manera que se obtenga como resultado de la evaluación el número que aparece como miembro derecho de la igualdad

Precedencia de operadores V

Ejemplo (continuación)


```
(((42 // 6) + 7) * 3) - 39
```

Precedencia de operadores VI

Ejemplo

Hallar el valor de la siguiente expresión teniendo en cuenta la prioridad de operadores y que los operandos son tanto números reales como enteros

```
12.0 * 3 - -4.0 + 8 // 2 % 3
```

i) $12.0 * 3 - (-4.0) + 8 \;//\; 2 \;\%\; 3$ ($-$ prioridad 2)

ii) $(12.0 * 3) - (-4.0) + 8 \;//\; 2 \;\%\; 3$ ($*$ prioridad 3)

iii) $(12.0 * 3) - (-4.0) + (8 \;//\; 2) \;\%\; 3$ ($//$ prioridad 3)

iv) $(12.0 * 3) - (-4.0) + ((8 \;//\; 2) \;\%\; 3)$ ($\%$ prioridad 3)

v) $((12.0 * 3) - (-4.0)) + ((8 \;//\; 2) \;\%\; 3)$ ($-$ prioridad 4)

vi) $(((12.0 * 3) - (-4.0)) + ((8 \;//\; 2) \;\%\; 3))$ ($+$ prioridad 4)

Precedencia de operadores VII

Ejemplo (continuación)

$$
\begin{aligned}
12.0 * 3 - -4.0 + 8 \;//\; 2 \;\%\; 3 &= 12.0 * 3 - (-4.0) + 8 \;//\; 2 \;\%\; 3 \\
&= 36.0 - (-4.0) + 8 \;//\; 2 \;\%\; 3 \\
&= 36.0 - (-4.0) + 4 \;\%\; 3 \\
&= 36.0 - (-4.0) + 1 \\
&= 40.0 + 1 \\
&= 41.0
\end{aligned}
$$

Precedencia de operadores VIII

Ejemplo

Hallar el valor de la siguiente expresión teniendo en cuenta la prioridad de operadores y que los operandos son números enteros

```
(-2 + 5 % 3 * 4) // 4 + 2
```

i) $((-2) + 5 \ \% \ 3 * 4) \ // \ 4 + 2$ ($-$ prioridad 2)

ii) $((-2) + (5 \ \% \ 3) * 4) \ // \ 4 + 2$ (% prioridad 3)

iii) $((-2) + ((5 \ \% \ 3) * 4)) \ // \ 4 + 2$ ($*$ prioridad 3)

iv) $(((-2) + ((5 \ \% \ 3) * 4))) \ // \ 4 + 2$ ($+$ prioridad 4)

v) $((((-2) + ((5 \ \% \ 3) * 4))) \ // \ 4) + 2$ ($/$ prioridad 3)

vi) $(((((-2) + ((5 \ \% \ 3) * 4))) \ // \ 4) + 2)$ ($+$ prioridad 4)

Precedencia de operadores IX

Ejemplo (continuación)

$$
\begin{aligned}
(-2 + 5 \ \% \ 3 * 4) \ // \ 4 + 2 &= ((-2) + 5 \ \% \ 3 * 4) \ // \ 4 + 2 \\
&= ((-2) + 2 * 4) \ // \ 4 + 2 \\
&= ((-2) + 8) \ // \ 4 + 2 \\
&= (6) \ // \ 4 + 2 \\
&= 6 \ // \ 4 + 2 \\
&= 1 + 2 \\
&= 3
\end{aligned}
$$

Agenda

Agenda

1. Operadores
   - Operadores aritméticos
   - Operadores de asignación
   - Conversión de tipos de datos numéricos (*typecasting*)
     - De entero a real
     - De real a entero
   - Operadores lógicos
   - Operadores de igualdad y relacionales
   - Precedencia de operadores
2. **Evaluación de secuencias de expresiones**
   - Evaluación de expresiones
   - Traza de un programa

Evaluación de expresiones I

Como se vio previamente las cadenas =, +=, −=, *=, /=, //=, %= sirven para representar los operadores de asignación, es decir, permiten asignar un valor a una determinada variable, donde la variable se encuentra a la izquierda del operador y el valor resulta de evaluar una expresión que se encuentra a la derecha del operador.

Se debe tener cuidado que al evaluar una expresión el resultado sea del mismo tipo de la variable a la que se le esta asignando el valor, esto es, que a una variable de tipo `int` se le asigne un valor entero, que a una variable de tipo `float` se le asigne un valor real, que a una variable de tipo `bool` se le asigne un valor booleano, que a una variable de tipo `str` se le asigne una cadena, etc. Python no controla esto.

Evaluación de expresiones II

Una asignación comprende dos partes: el valor al que se le asignar el valor y la expresión. Se espera que el resultado de la evaluación de la expresión sea del mismo tipo de la expresión. En este sentido se podría extender a expresiones de tipo lógico, aritmético ó cadena.

El proceso de asignar valores a variables es el objetivo central a la hora de construir un programa, ya que un programa no es más que una función que transforma la memoria desde un estado inicial hasta un estado final donde se encuentra el resultado que se quería calcular.

Evaluación de expresiones III

Visto así, en programación, la asignación es una operación temporal, que primero lee el valor de las variables existentes en la memoria, a partir de estos valores se evalúa la expresión a la derecha de la asignación y luego se realiza la asignación a la variable de la izquierda correspondiente al resultado de la evaluación de la expresión, es decir, se actualiza el valor de la variable en la memoria.

Evaluación de expresiones IV

Ejemplo

Supónga que un programa contiene las variables x, y y z en el instante de tiempo t, que sus valores en este instante de tiempo son:

$$\mathtt{x} = 3, \qquad \mathtt{y} = 4 \qquad \text{y} \qquad \mathtt{z} = -2$$

si a partir de estos valores se realiza la asignación

```
x = y + z - 2
```

entonces se tendría que en el instante de tiempo t se evalúa la expresión `y + z - 2` y el resultado de esta evaluación se asignaría a la variable x pero ya en el instante de tiempo $t + 1$. Con lo que los valores de las variables (memoria) en este nuevo instante de tiempo seria:

$$\mathtt{x} = 0, \qquad \mathtt{y} = 4 \qquad \text{y} \qquad \mathtt{z} = -2$$

Evaluación de expresiones V

Ejemplo

Supónga que se desea realizar la asignación

```
x = x + 3 - 2 * y
```

cuando $x = 3$ y $y = 5$.
Para entender como se realiza la asignación es útil subindizar las variables teniendo en cuenta el instante de tiempo en el cual se esta leyendo o modificando la memoria. Con base en lo anterior, la expresión se reescribe de la siguiente manera

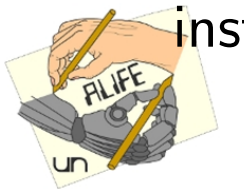
$$x_{t+1} = x_t + 3 - 2 * y_t$$

Así, si en el instante de tiempo t se tiene que $x_t = 3$ y $y_t = 5$, entonces en el instante de tiempo $t + 1$, se tendrá que $x_{t+1} = -4$ y $y_{t+1} = 5$.

Agenda

Traza de un programa I

- Cuando se desea estudiar el comportamiento de una secuencia de asignaciones es común utilizar una tabla de la *traza* de ejecuciones.
- En esta tabla se tiene una columna donde se ubica el instante de tiempo t que inicialmente debe ser igual a 0, en las otras columnas se ubican todas las variables que intervienen en los cálculos. Para las variables que tienen un valor inicial, este valor se ubica en la misma fila del instante de tiempo $t = 0$ y para el resto de variables se utiliza el símbolo — para indicar que aún no están inicializadas.
- Tras iniciar la ejecución de las instrucciones, se va actualizando el valor de las variables a las que se les haya hecho alguna asignación, teniendo en cuenta el instante de tiempo en el cual se realiza la asignación. Esto se realiza hasta que se ejecute toda la secuencia de instrucciones.



Traza de un programa II

Ejemplo

La siguiente tabla es la traza obtenida tras ejecutar la instrucción

```
x = x + 3 - 2 * y
```

cuando $\mathtt{x} = 3$ y $\mathtt{y} = 5$.

t	x	y
0	3	5
1	-4	5

Traza de un programa III

Ejemplo

Supónga que se desea ejecutar la siguiente secuencia de instrucciones

```
i = k + 1
j = 2 * k
i = i * k * j
j = j * k - i
```

cuando $\texttt{k} = 1$. Entonces la traza de la ejecución será la siguiente

t	k	i	j
0	1	—	—
1	1	2	—
2	1	2	2
3	1	4	2
4	1	4	−2