

Tipos de Datos y Expresiones

Identificadores, variables, tipos, expresiones y evaluación de expresiones

Jonatan Gómez Perdomo, Ph. D.

jgomezpe@unal.edu.co

Arles Rodríguez, Ph.D.

aerodriguezp@unal.edu.co

Camilo Cubides, Ph.D. (c)

eccubidesg@unal.edu.co

Carlos Andres Sierra, M.Sc.

casierrav@unal.edu.co

Research Group on Artificial Life – Grupo de investigación en vida artificial – (Alife)

Computer and System Department

Engineering School

Universidad Nacional de Colombia

Agenda

1 Identificadores y variables

2 Tipos de datos primitivos

- Numéricos: Enteros y Reales
- Booleanos
- Caracteres y Cadenas de Caracteres

3 Operadores

- Operadores aritméticos
- Operadores de asignación
- Operadores lógicos
- Operadores de igualdad y relacionales
- Precedencia de operadores

4 Evaluación de secuencias de expresiones

5 Tráza de un programa



Identificadores I

Definición

Un *identificador* es una secuencia de símbolos que se utilizan como nombres de variables, funciones, arreglos, clases y otras estructuras de los lenguajes de programación.

Los identificadores en Python se escriben como secuencias de caracteres alfanuméricos del alfabeto inglés o el guión bajo (*underscore*) (`_`), tales que su primer símbolo no es un dígito. Un identificador en Python puede tener cualquier longitud que sea positiva.



Identificadores II

Un identificador válido debe cumplir con la condición adicional de que no pertenezca a las palabras reservadas para el lenguaje, a continuación se listan las palabras reservadas del lenguaje Python:

and	as	assert	break	class
continue	def	del	elif	else
except	finally	False	for	from
global	if	import	in	is
lambda	nonlocal	None	not	or
pass	raise	return	True	try
with	while	yield		



Identificadores III

Ejemplo

Las siguientes secuencias de caracteres son ejemplos de identificadores válidos:

```
i  
x  
n  
suma  
sumando1  
sumando2  
Edad  
paisDeNacimiento
```

```
_nombre  
area_circulo  
false  
Variable  
snake_case  
MACRO_CASE  
camelCase  
CapWords
```



Identificadores IV

Ejemplo

Las siguientes secuencias de caracteres son ejemplos de secuencias que no son identificadores, ¿por qué?:

```
1er_mes  
primer nombre  
while  
p@dre  
día  
velocidad-maxima  
True  
Var#1  
$var
```



Identificadores V

Nota

Una nota importante para el lenguaje Python es que éste es sensible a mayúsculas y minúsculas, esto quiere decir que por ejemplo los identificadores

dia	Dia	DIA
-----	-----	-----

sirven para declarar entidades (variables, funciones, etc.) que son diferentes, pues al ser la misma palabra, difiere en que algunas letras son mayúsculas en unos identificadores y en los otros no.

Nota

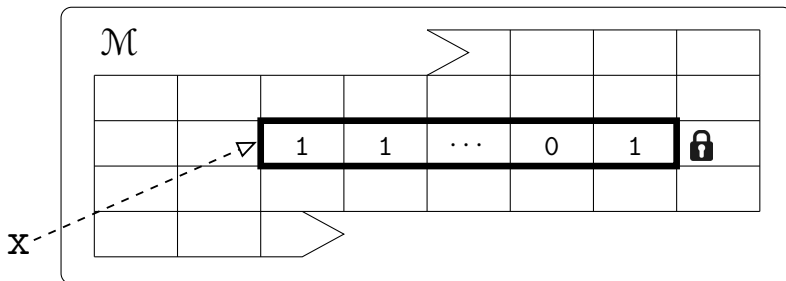
Cabe anotar que existen lenguajes que no son sensibles a mayúsculas y minúsculas, tales como DFD, BASIC, FORTRAN y HTML.



Variables I

Definición

Una *variable* es un espacio de memoria donde se almacena un dato, un espacio donde se guarda la información necesaria para realizar las acciones que ejecutan los programas.



Variables II

Para declarar una variable se necesitan principalmente dos componentes: el nombre y el tipo de dato (opcional en algunos lenguajes). Los tipos de variables se estudian en la siguiente sección, con respecto al nombre, este simplemente debe ser un identificador válido que no sea una palabra reservada.

En general una variable se declara así

$x: T$

Donde T es el tipo de dato o conjunto al que pertenece la variable y x es el identificador que es el nombre de la variable.



Variables III

Una buena práctica de programación* es asignarle el nombre a una variable de tal manera que indique por un lado el papel que desempeña dicha variable en el algoritmo y por otro los posibles valores que almacena. Nombres de variables recomendados dependiendo del tipo de problema pueden ser:

velocidad	espacio	masa	aceleracion
exponente		termino1	valor_maximo
area_circulo		nombre_estudiante	last_name



Por buenas o mejores prácticas se entiende un conjunto coherente de acciones que han rendido bien o incluso excelente servicio en un determinado contexto.



Agenda

- 1 Identificadores y variables
- 2 Tipos de datos primitivos
 - Numéricos: Enteros y Reales
 - Booleanos
 - Caracteres y Cadenas de Caracteres
- 3 Operadores
 - Operadores aritméticos
 - Operadores de asignación
 - Operadores lógicos
 - Operadores de igualdad y relacionales
 - Precedencia de operadores
- 4 Evaluación de secuencias de expresiones
- 5 Oruga de un programa



En programación existe un tipo de dato que permite aproximar el conjunto de los números enteros conocido como `int`.

Otro tipo de dato que aproxima el conjunto de los números reales se conoce como `float`.

Otro tipo de dato conocido como `str` es el que sirve para representar letras y cadenas.

Para la representación de los valores de verdad se tendrán los booleanos representados en el tipo de dato `bool`.

Estos tipos de datos son conocidos como primitivos o escalares, pues están definidos en el lenguaje de programación Python y porque de ellos se pueden derivar otros tipos de datos definidos por el programador.



Agenda

- 1 Identificadores y variables
- 2 Tipos de datos primitivos
 - Numéricos: Enteros y Reales
 - Booleanos
 - Caracteres y Cadenas de Caracteres
- 3 Operadores
 - Operadores aritméticos
 - Operadores de asignación
 - Operadores lógicos
 - Operadores de igualdad y relacionales
 - Precedencia de operadores
- 4 Evaluación de secuencias de expresiones
- 5 Traza de un programa



Enteros I

Los enteros en Python se codifican con la palabra `int` y su declaración es la siguiente

Si x es una variable algebraica que varia en el conjunto \mathbb{Z} , para definir x en el lenguaje Python se utiliza la expresión

`x: int`

lo que sirve para declarar que la variable x pertenece a los enteros que son representables en el lenguaje Python.



Enteros II

Literales enteros

Ejemplo

Los literales enteros, es decir, la sintaxis de los valores que pueden ser asignados a las variables de tipo `int` que soporta Python son por ejemplo:

-32768	-0	-1	-127
32768	0	1	127
+32768	+0	+1	+127



Enteros III

Ejemplo

Cuando se declara una variable de tipo entero, no se sabe que valor tiene, por eso es necesario inicializar la variable. Los siguientes son ejemplos de inicializaciones de variables de tipo `int` en Python.

```
i = 0  
j : int = 1  
n = 5  
p : int = -10  
k = -1
```



Reales I

Los reales en Python se codifican con la palabra `float` y su declaración es la siguiente

Si x es una variable algebraica que varia en el conjunto \mathbb{R} , para definir x en el lenguaje Python se utiliza la expresión

```
x: float
```

lo que sirve para declarar que la variable x pertenece a los reales que son representables en el lenguaje Python.



Reales II

El subconjunto de los números reales que pueden ser representados en el lenguaje Python, es un subconjunto propio de los racionales, que se representan con 64 bits (8 bytes) y que usan un tipo de codificación definida por el *IEEE standard for Binary Floating-Point Arithmetic 754* de 1985, los valores distintos de 0 de este conjunto varían en el rango

$$-1.7976931348623157 \times 10^{+308} \leq x \leq -2.2250738585072014 \times 10^{-308}$$

y

$$2.2250738585072014 \times 10^{-308} \leq x \leq 1.7976931348623157 \times 10^{+308}$$

que dan una precisión científica de 15 dígitos.



Reales III

Los números reales de máquina I

Los números reales de máquina son finitos y por lo tanto, existen números reales que no se pueden representar. Además, la mayoría de los números se acumulan alrededor del 0 y hacia los extremos superior e inferior se encuentran más dispersos.

Ejemplo

Si se tiene una máquina muy sencilla que utiliza una representación en base 2 de 6 bits, tres para la mantisa (uno de estos para el signo), y 3 para el exponente (uno de estos para el signo), se tiene que el conjunto de los números de esta máquina son los siguientes:

Mantisas y exponentes

$$\{-3, -2, -1, -0, 1, 2, 3\}$$



Reales IV

Los números reales de máquina II

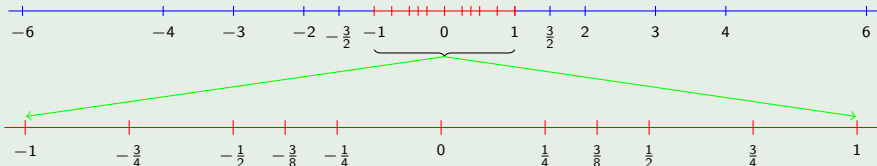
Ejemplo (continuación)

No Negativos

$$\left\{ 0, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{3}{4}, 1, \frac{3}{2}, 2, 3, 4, 6, 8, 12, 16, 24 \right\}$$

Negativos

$$\left\{ -24, -16, -12, -8, -6, -4, -3, -2, -\frac{3}{2}, -1, -\frac{3}{4}, -\frac{1}{2}, -\frac{3}{8}, -\frac{1}{4} \right\}$$



Reales V

Literales reales

Ejemplo

Los literales reales, es decir, la sintaxis de los valores que pueden ser asignados a las variables de tipo `float` que soporta Python son por ejemplo:

-3.14159265	-0.0	-6.02214129E+23	-6.674287e-11
3.14159265	0.0	6.02214129E23	6.674287E-11
+3.14159265	+0.0	+6.02214129e+23	+6.674287E-11



Reales VI

Ejemplo

Cuando se declara una variable de tipo real, no se sabe que valor tiene, por eso es necesario inicializar la variable. Los siguientes son ejemplos de inicializaciones de variables de tipo float

```
e = 2.7182818284
gamma: float = 0.577215664901
phi = +1.61803398874989
X = -1.0
const_Boltzmann = 1.3806488E-23
Luz: float = 2.998e+8
Avogadro = +6.02214129e+23
G: float = 6.67384e-11
Plank = 6.62606896E-34
```



Agenda

- 1 Identificadores y variables
- 2 Tipos de datos primitivos
 - Numéricos: Enteros y Reales
 - Booleanos
 - Caracteres y Cadenas de Caracteres
- 3 Operadores
 - Operadores aritméticos
 - Operadores de asignación
 - Operadores lógicos
 - Operadores de igualdad y relacionales
 - Precedencia de operadores
- 4 Evaluación de secuencias de expresiones
- 5 Oruga de un programa



Booleanos I

Los booleanos en Python se codifican con la palabra `bool` y su declaración es la siguiente

Si x es una variable algebraica que varia en el conjunto \mathbb{B} , para definir x en el lenguaje Python se utiliza la expresión

`x: bool`

lo que sirve para declarar que la variable x pertenece al conjunto de los booleanos ($\mathbb{B} = \{V, F\}$).



Booleanos II

Literales booleanos

Como sólo hay dos valores de verdad V y F , en Python sólo hay dos literales para representar los valores lógicos, estos son:

True

False

donde la cadena True representa el valor de verdad V y la cadena False representa el valor de verdad F .



Booleans III

Ejemplo

Cuando se declara una variable de tipo booleano, no se sabe que valor tiene, por eso es necesario inicializar la variable. Los siguientes son ejemplos de inicializaciones de variables de tipo bool

```
b = True  
flag: bool = True  
exp = False  
isPrime: bool = False
```



Agenda

- 1 Identificadores y variables
- 2 Tipos de datos primitivos
 - Numéricos: Enteros y Reales
 - Booleanos
 - Caracteres y Cadenas de Caracteres
- 3 Operadores
 - Operadores aritméticos
 - Operadores de asignación
 - Operadores lógicos
 - Operadores de igualdad y relacionales
 - Precedencia de operadores
- 4 Evaluación de secuencias de expresiones
- 5 Tráza de un programa



Carácter

Un **carácter** es el elemento mínimo de información usado para representar, controlar, transmitir y visualizar datos. Al conjunto de caracteres usados con este fin se le llama **Esquema de codificación**. Los esquemas de codificación en general usan un número de bits o bytes fijos. Por ahora solo consideraremos el **ASCII**. En **Python** son representados como cadenas de caracteres de un sólo carácter. Más adelante profundizaremos en ellas.



Esquemas de Codificación - ASCII

Caracteres ASCII de control			Caracteres ASCII imprimibles				ASCII extendido (Página de código 437)									
00	NULL	(carácter nulo)	32	espacio	64	@	96	`	128	Ç	160	á	192	Ł	224	Ó
01	SOH	(inicio encabezado)	33	!	65	A	97	a	129	ü	161	í	193	ł	225	ô
02	STX	(inicio texto)	34	"	66	B	98	b	130	é	162	ó	194	Ł	226	ö
03	ETX	(fin de texto)	35	#	67	C	99	c	131	â	163	ú	195	ł	227	õ
04	EOT	(fin transmisión)	36	\$	68	D	100	d	132	ä	164	ñ	196	Ł	228	ö
05	ENQ	(consulta)	37	%	69	E	101	e	133	à	165	Ñ	197	ł	229	ó
06	ACK	(reconocimiento)	38	&	70	F	102	f	134	á	166	ª	198	Ł	230	µ
07	BEL	(timbre)	39	'	71	G	103	g	135	ç	167	º	199	Ł	231	þ
08	BS	(retroceso)	40	(72	H	104	h	136	ê	168	¿	200	Ł	232	þ
09	HT	(tab horizontal)	41)	73	I	105	i	137	ë	169	®	201	Ł	233	Û
10	LF	(nueva línea)	42	*	74	J	106	j	138	è	170	™	202	Ł	234	Ü
11	VT	(tab vertical)	43	+	75	K	107	k	139	ĩ	171	¼	203	Ł	235	Ù
12	FF	(nueva página)	44	,	76	L	108	l	140	î	172	½	204	Ł	236	Ý
13	CR	(retorno de carro)	45	-	77	M	109	m	141	ï	173	¾	205	Ł	237	Ÿ
14	SO	(desplaza afuera)	46	.	78	N	110	n	142	Ā	174	«	206	Ł	238	—
15	SI	(desplaza adentro)	47	/	79	O	111	o	143	Ă	175	»	207	Ł	239	·
16	DLE	(esc.vínculo datos)	48	0	80	P	112	p	144	Ĕ	176	⋈	208	Ł	240	≡
17	DC1	(control disp. 1)	49	1	81	Q	113	q	145	æ	177	⋈	209	Ł	241	±
18	DC2	(control disp. 2)	50	2	82	R	114	r	146	Æ	178	⋈	210	Ł	242	⋈
19	DC3	(control disp. 3)	51	3	83	S	115	s	147	ó	179	⋈	211	Ł	243	¼
20	DC4	(control disp. 4)	52	4	84	T	116	t	148	ô	180	⋈	212	Ł	244	¶
21	NAK	(conf. negativa)	53	5	85	U	117	u	149	õ	181	⋈	213	Ł	245	§
22	SYN	(inactividad sinc)	54	6	86	V	118	v	150	û	182	⋈	214	Ł	246	÷
23	ETB	(fin bloque trans)	55	7	87	W	119	w	151	ü	183	⋈	215	Ł	247	°
24	CAN	(cancelar)	56	8	88	X	120	x	152	ÿ	184	⋈	216	Ł	248	°
25	EM	(fin del medio)	57	9	89	Y	121	y	153	Ų	185	⋈	217	Ł	249	°
26	SUB	(sustitución)	58	:	90	Z	122	z	154	Ų	186	⋈	218	Ł	250	°
27	ESC	(escape)	59	;	91	[123	{	155	ø	187	⋈	219	Ł	251	°
28	FS	(sep. archivos)	60	<	92	\	124		156	ø	188	⋈	220	Ł	252	°
29	GS	(sep. grupos)	61	=	93]	125	}	157	ø	189	⋈	221	Ł	253	°
30	RS	(sep. registros)	62	>	94	^	126	~	158	×	190	⋈	222	Ł	254	■
31	US	(sep. unidades)	63	?	95	_			159	f	191	⋈	223	Ł	255	nbsp
127	DEL	(suprimir)														

Figure: Imagen tomada de <https://elcodigoascii.com.ar/>

Esquemas de Codificación - ASCII

Código Estadounidense Estándar para el Intercambio de Información
(*American Standard Code for Information Interchange*)

- En su versión original usa 7 bits, definiendo 128 caracteres.
- En la versión extendida usa 8 bits (esto es 1 byte), definiendo 256 caracteres.
- Es la base de los archivos de texto plano (o sin formato).
- Es el esquema base para la escritura de programas en casi todos los lenguajes de programación (incluido **Python**).



Cadenas de caracteres (str)

Una **cadena de caracteres** `str` es una secuencia de cero o más caracteres. Una cadena de caracteres se delimita por el carácter `'` o por el carácter `"`. Una cadena de caracteres es una estructura de datos inmutable, esto significa que no puede ser cambiada. Más adelante profundizaremos en ellas.

- `'ejemplo de cadena'`
- `"Cadena con un tabulado \t y una nueva \n línea"`
- `'Cadena con un carácter unicode \u01F4 y una comilla doble''`
- `"Cadena con una comilla simple \', una comilla doble \" y una diagonal invertida \\"`
- La cadena vacía `""` o `''`




Agenda

- 1 Identificadores y variables
- 2 Tipos de datos primitivos
 - Numéricos: Enteros y Reales
 - Booleanos
 - Caracteres y Cadenas de Caracteres
- 3 Operadores
 - Operadores aritméticos
 - Operadores de asignación
 - Operadores lógicos
 - Operadores de igualdad y relacionales
 - Precedencia de operadores
- 4 Evaluación de secuencias de expresiones
- 5 Oruga de un programa



Agenda

- 1 Identificadores y variables
- 2 Tipos de datos primitivos
 - Numéricos: Enteros y Reales
 - Booleanos
 - Caracteres y Cadenas de Caracteres
- 3 **Operadores**
 - **Operadores aritméticos**
 - Operadores de asignación
 - Operadores lógicos
 - Operadores de igualdad y relacionales
 - Precedencia de operadores
- 4 Evaluación de secuencias de expresiones
- 5  Traza de un programa

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 explícita, 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.

$$\begin{array}{c} m \overline{) n} \\ \hline r \quad c \end{array} \longrightarrow m \div n$$

\downarrow

$$m \bmod n$$

$$\begin{array}{c} 9 \overline{) 4} \\ \hline 1 \quad 2 \end{array} \longrightarrow 9 // 4$$

\downarrow

$$9 \% 4$$

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



Agenda

- 1 Identificadores y variables
- 2 Tipos de datos primitivos
 - Numéricos: Enteros y Reales
 - Booleanos
 - Caracteres y Cadenas de Caracteres
- 3 Operadores
 - Operadores aritméticos
 - Operadores de asignación
 - Operadores lógicos
 - Operadores de igualdad y relacionales
 - Precedencia de operadores
- 4 Evaluación de secuencias de expresiones
- 5 Tráza de un programa



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

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

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

```
F = 6.67384 × 10-11 (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

- 1 Identificadores y variables
- 2 Tipos de datos primitivos
 - Numéricos: Enteros y Reales
 - Booleanos
 - Caracteres y Cadenas de Caracteres
- 3 Operadores
 - Operadores aritméticos
 - Operadores de asignación
 - Operadores lógicos
 - Operadores de igualdad y relacionales
 - Precedencia de operadores
- 4 Evaluación de secuencias de expresiones



5 Traza de un programa

Operadores lógicos

not : Operador \neg de la negación en Python. Si la expresión α es falsa su negación es verdadera y si es verdadera su negación es falsa.

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

and : Operador \wedge de la conjunción en Python. La expresión $\alpha \wedge \beta$ solo es verdadera cuando ambas α y β son verdaderas. En otro caso es falsa.

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

or : Operador \vee de la disyunción en Python. La expresión $\alpha \vee \beta$ solo es falsa cuando ambas α y β son falsas. En otro caso es verdadera.

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



Agenda

- 1 Identificadores y variables
- 2 Tipos de datos primitivos
 - Numéricos: Enteros y Reales
 - Booleanos
 - Caracteres y Cadenas de Caracteres
- 3 Operadores
 - Operadores aritméticos
 - Operadores de asignación
 - Operadores lógicos
 - Operadores de igualdad y relacionales
 - Precedencia de operadores
- 4 Evaluación de secuencias de expresiones
- 5 Traza de un programa



Operadores de igualdad y relacionales I

== : Devuelve V si dos valores son iguales.

$$\alpha == \beta \Leftrightarrow \alpha = \beta$$

!= : Devuelve V si dos valores son distintos.

$$\alpha != \beta \Leftrightarrow \alpha \neq \beta$$

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

$$\alpha > \beta \Leftrightarrow \alpha > \beta$$



Operadores de igualdad y relacionales II

$<$: Menor que, devuelve V si el primer operando es estrictamente menor que el segundo.

$$\alpha < \beta \Leftrightarrow \alpha < \beta$$

$>=$: Mayor o igual, devuelve V si el primer operando es mayor o igual que el segundo.

$$\alpha >= \beta \Leftrightarrow \alpha \geq \beta$$

$<=$: Menor igual, devuelve V si el primer operando es menor o igual que el segundo.

$$\alpha <= \beta \Leftrightarrow \alpha \leq \beta$$



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 puede utilizar la sentencia

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

La anterior expresión es equivalente a

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

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

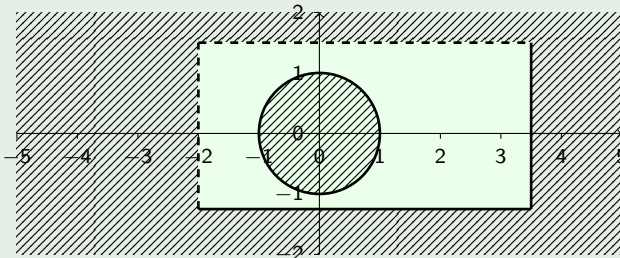


Operadores de igualdad y relacionales IV

Ejemplo

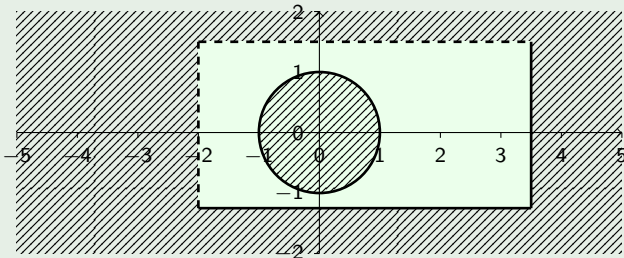
Para decidir si la pareja ordenada (a, b) pertenece al siguiente conjunto, con universo \mathbb{R}^2

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



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

- 1 Identificadores y variables
- 2 Tipos de datos primitivos
 - Numéricos: Enteros y Reales
 - Booleanos
 - Caracteres y Cadenas de Caracteres
- 3 Operadores
 - Operadores aritméticos
 - Operadores de asignación
 - Operadores lógicos
 - Operadores de igualdad y relacionales
 - Precedencia de operadores
- 4 Evaluación de secuencias de expresiones
- 5 Oruga de un programa



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	-(<i>signo menos</i>)		+(<i>signo más</i>)			2
**(<i>potencia</i>)						
	*	/	//	%		3
		+	-			4
	<	>	<=	>=		5
		==	!=			6
and						7
or						8
=	+=	-=	*=	/=	//=	9
%=						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)

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

A ver unos ejercicios:



Precedencia de operadores IV

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 V

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 VI

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 VII

Ejemplo (continuación)

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

A ver unos ejercicios:



Agenda

- 1 Identificadores y variables
- 2 Tipos de datos primitivos
 - Numéricos: Enteros y Reales
 - Booleanos
 - Caracteres y Cadenas de Caracteres
- 3 Operadores
 - Operadores aritméticos
 - Operadores de asignación
 - Operadores lógicos
 - Operadores de igualdad y relacionales
 - Precedencia de operadores
- 4 Evaluación de secuencias de expresiones



5 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: la variable a la que se le asignar el valor de la expresión y la expresión. Se espera que el resultado de la evaluación de la expresión sea del mismo tipo de la variable. En este sentido se podría extender a expresiones de tipo lógico, aritmético o 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

Suponga 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:

$$x = 3, \quad y = 4 \quad y \quad z = -2$$

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 sería:

$$x = 0, \quad y = 4 \quad y \quad z = -2$$

Evaluación de expresiones V

Ejemplo

Supóngase 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 está 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

- 1 Identificadores y variables
- 2 Tipos de datos primitivos
 - Numéricos: Enteros y Reales
 - Booleanos
 - Caracteres y Cadenas de Caracteres
- 3 Operadores
 - Operadores aritméticos
 - Operadores de asignación
 - Operadores lógicos
 - Operadores de igualdad y relacionales
 - Precedencia de operadores
- 4 Evaluación de secuencias de expresiones
- 5  Traza de un programa

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 $x = 3$ y $y = 5$.

t	x	y
0	3	5
1	-4	5

A ver unos ejercicios:



Traza de un programa III

Ejemplo

Suponga que se desea ejecutar la siguiente secuencia de instrucciones

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

cuando $k = 1$. 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