Introducción a la Programación

Prof. Agustín Gravano

Primer semestre de 2022

Clase teórica 2: Tipos de datos y expresiones

Datos

En general, un programa recibe datos de entrada, los procesa y genera datos de salida.

Vamos a empezar por entender qué son los datos que un programa puede manejar. A partir de la próxima clase, veremos qué hacer con ellos.

Hay 4 tipos de datos básicos:

- Números enteros
- Números de punto flotante (o coma flotante)
- Caracteres
- Booleanos (verdadero/falso)

También existen los tipos compuestos: cadenas de caracteres, listas, conjuntos, diccionarios, etc. Y es posible definir nuestros propios tipos, para modelar los datos del problema a resolver.

Hoy vamos a ver algunos de estos tipos de datos. El resto, más adelante.

Números enteros (`int`)

Los enteros (`int`) para una computadora son similares a los enteros matemáticos (el conjunto $\mathbb{Z}$):

$$\ldots, -4,\ -3,\ -2,\ -1,\ 0,\ 1,\ 2,\ 3,\ 4, \ldots$$

con la *pequeña* salvedad de que no son infinitos: están acotados inferior y superiormente, según cuántos bits se usen para representarlos (típicamente, 32 o 64 bits). Esto depende del lenguaje y de la plataforma.

Más adelante en la materia veremos cómo se representan los enteros en una computadora. Por ahora enfoquémosnos en cómo usarlos.

Números enteros (`int`) · Operaciones

Símbolo	Operación	Ejemplo	Resultado
+	Suma	`5 + 2`	`7`
-	Resta	`5 - 2`	`3`
*	Producto	`5 * 2`	`10`
/	División	`5 / 2`	`2.5`
//	División entera	`5 // 2`	`2`
%	Resto	`5 % 2`	`1`
-	Negación	`-5`	`-5`
abs	Valor absoluto	`abs(-5)`	`5`
==	Igual	`5 == 2`	`False`
!=	Distinto	`5 != 2`	`True`
>	Mayor estricto	`5 > 2`	`True`
<	Menor estricto	`5 < 2`	`False`
>=	Mayor o igual	`5 >= 2`	`True`
<=	Menor o igual	`5 <= 2`	`False`

`2.5` es de tipo `float`; `True` y `False` son de tipo `bool`.

Números enteros (`int`) · Ejercicios

Evaluar las siguientes expresiones primero en papel y después en una consola `ipython`:

a) `3 + 4`
b) `3 + 2 * 2`
c) `3 + (2 * 2)`
d) `3 < 4`
e) `3 == 4 - 1`
f) `123 / 10`
g) `123 // 10`
h) `123 % 10`
i) `123 / 0`
j) `(-1) * (-5)`
k) `abs(133)`
l) `abs(10 - 133)`

Números de punto flotante (`float`)

Los números de punto flotante de una computadora son **muy diferentes** de los reales matemáticos (el conjunto $\mathbb{R}$).

Está acotada su cantidad (son finitos) y también su precisión.

Por ejemplo, $\sqrt{2}$ tiene infinitos dígitos decimales (sin período):

$$1.41421356237309504880168872420969807856967187538\ldots$$

Para representar a $\sqrt{2}$ como un número de punto flotante, necesitamos truncarlo en algún lugar. Por ejemplo:

$$1.41421356238$$

Al trabajar con `float` debemos lidiar con errores de representación.

Más adelante veremos cómo se representan en una computadora los números de punto flotante. Por ahora enfoquémosnos en cómo usarlos.

Números de punto flotante (`float`) · Operaciones

Símbolo	Operación	Ejemplo	Resultado
+	Suma	`5.0 + 2.0`	`7.0`
-	Resta	`5.0 - 2.0`	`3.0`
*	Producto	`5.0 * 2.0`	`10.0`
/	División	`5.0 / 2.0`	`2.5`
-	Negación	`-5.0`	`-5.0`
abs	Valor absoluto	`abs(-5.0)`	`5.0`
==	Igual	`5.0 == 2.0`	`False`
!=	Distinto	`5.0 != 2.0`	`True`
>	Mayor estricto	`5.0 > 2.0`	`True`
<	Menor estricto	`5.0 < 2.0`	`False`
>=	Mayor o igual	`5.0 >= 2.0`	`True`
<=	Menor o igual	`5.0 <= 2.0`	`False`

(*) No conviene usar la igualdad exacta entre floats. Podríamos querer que dos números muy cercanos sean considerados iguales (ej: 0.66666667 y 0.66666666). En lugar de `x==y`, convendría hacer `abs(x - y) < 0.000001`.

En la materia *Métodos Computacionales* se ven este y otros temas relacionados.

Números de punto flotante (`float`) · Ejercicios

Evaluar las siguientes expresiones primero en papel y después en una consola `ipython`:

a) `3.0 + 4.0`

b) `1 / 3`

c) `1 / 3 == 0.33333`

d) `1 / 3 <= 0.33333`

e) `1.0 / 0.0`

Ejercicio grupal: Cada estudiante de la mesa diga el día de su fecha de nacimiento (del 1 al 31). Computar (en Python) el promedio de esos números y reportarlo al docente.

Números de punto flotante (`float`) · Módulo `math`

En Python, muchas funciones matemáticas útiles están implementadas en el módulo `math` (que se importa con `import math`).

Algunos ejemplos:

- `math.pi`: constante *pi*.
- `math.e`: constante *e* (número de Euler).
- `math.sin(x)`, `math.cos(x)`, `math.tan(x)`, `math.asin(x)`, etc.: funciones trigonométricas.
- `math.log(x)`, `math.log10(x)`, `math.log2(x)`: logaritmos en distintas bases.
- `math.sqrt(x)`: raíz cuadrada.

Ejercicio: Evaluar en una consola `ipython` las siguientes expresiones, luego de ejecutar el comando `import math`:

a) `math.pi`

b) `math.sqrt(4.0)`

Cadenas de caracteres (`str`)

Un carácter es un símbolo válido en la computadora:

- `abcdefghijklmnopqrstuvwxyz`
- `ABCDEFGHIJKLMNOPQRSTUVWXYZ`
- `áéíóúüñÁÉÍÓÚÜÑ`
- `0123456789¡!#$%*()-_=+~``':;,."<>¿?/\`
- etc.

Internamente, un carácter se representa mediante un número entero (y a su vez, en forma binaria).

En Python, la operación `ord` permite conocer la representación entera de un carácter, y `chr` es la operación inversa.

Ejercicio: Evaluar en una consola `ipython` las siguientes expresiones:

`ord('a')` `chr(97)` `chr(ord('a')+1)`

Una cadena de caracteres o string (`str` en Python) es una secuencia de 0 o más caracteres, delimitados por un símbolo especial (típicamente comillas simples o dobles): `'¡Hola, mundo!'`

Cadenas de caracteres (`str`) · Operaciones

Símbolo	Operación	Ejemplo	Resultado
`len(·)`	Longitud	`len('hola')`	`4`
`+`	Concatenación	`'algo' + 'ritmo'`	`'algoritmo'`
`*`	Multiplicación	`'ji' * 3`	`'jijiji'`
`·[·]`	i-ésimo carácter	`'aeiou'[2]`	`'i'`
`in`	Pertenencia	`'o' in 'aeiou'`	`True`
`==`	Igual	`'hola' == 'mundo'`	`False`
`!=`	Distinto	`'hola' != 'mundo'`	`True`
`>`	Mayor estricto	`'hola' > 'mundo'`	`False`
`<`	Menor estricto	`'hola' < 'mundo'`	`True`
`>=`	Mayor o igual	`'hola' >= 'mundo'`	`False`
`<=`	Menor o igual	`'hola' <= 'mundo'`	`True`

Las comparaciones por mayor y menor se basan en las representaciones enteras de los caracteres.

Hay muchísimas más operaciones con strings, pero vayamos de a poco. Por ahora con estas nos alcanza.

Cadenas de caracteres (`str`) · Ejercicios

Evaluar las siguientes expresiones primero en papel y después en una consola `ipython`:

a) `'prog' + 'rama'`

b) `'3' + '4'`

c) `'ta' * 5`

d) `'universidad'[3]`

e) `'ver' in 'universidad'`

f) `'VER' in 'universidad'`

Ejercicio grupal: Armar un string con todos los nombres de pila de la mesa, unidos sin espacios (`'juanatomasmariaelena...'`). Calcular la cantidad total de caracteres y reportarlo al docente. ¡Incluir todos los nombres de pila!

Expresiones

Una expresión es una combinación de valores, variables y operadores (incluyendo llamados a funciones).

La evaluación de una expresión arroja como resultado un valor.

Ejercicio: ¿Qué valores resultan de evaluar estas expresiones, y cuál es el tipo de cada una?

a) `2 + 3 * 4`

b) `1 / 2 + 5.0`

c) `'2' + 'dos' * 2`

Primero **pensar** las respuestas. Después evaluarlas en una consola `ipython` y averiguar sus tipos usando la operación `type()`.

Prestar atención a la precedencia de los operadores, que determina su orden de evaluación.

Sobrecarga de operadores

Se dice que operadores como + o == están sobrecargados: denotan operaciones en más de un tipo de datos. Se desambiguan por el contexto:

- `1 + 2` devuelve `3` (`int`)
- `1.0 + 2.0` devuelve `3.0` (`float`)
- `'1' + '2'` devuelve `'12'` (`str`)

Expresiones potencialmente ambiguas como `1+'2'` o `1=='2'` pueden arrojar un error de tipos o resultados inesperados.

Es importante tener **control** sobre el tipo de cada término de las expresiones que escribimos.

Conversión de tipos

A veces tiene sentido convertir una expresión de cierto tipo a otro tipo.

Ejemplos:

- `int('1234')` convierte el string `'1234'` en el entero `1234`.
- `float(5)` convierte el entero `5` en el número de punto flotante `5.0`.
- `str(9876)` convierte el entero `9876` en el string `'9876'`.
- `int(3.1416)` convierte el float `3.1416` en el entero `3`. En este caso, decimos que la parte decimal del número *se trunca*.

Cuando la conversión no tenga sentido, obtendremos un error. Ejemplo:

- `int('x')` → `ValueError: invalid literal for int() [...]`

Ejercicio: Averiguar el tipo de la siguiente expresión y evaluarla:

```
str(int(float('123.45')))[1] * 3
```

Resolverlo primero a mano, y después revisar en una consola `ipython`.

Repaso de la clase de hoy

- Tipos de datos: `int`, `float`, `str`.
- ¡Cuidado con las representaciones de los números en la computadora! En especial, cuiado con los errores numéricos del tipo `float`.
- Evaluación de expresiones; sobrecarga de operadores; errores de tipos; conversión de tipos.

Bibliografía complementaria:

- APPP2, secciones 2.1, 2.5 a 2.10 (ignorar variables por ahora)
- HTCSP3, secciones 2.1, 2.5 a 2.9 (ignorar variables por ahora)

Con lo visto, ya pueden resolver hasta el Ejercicio 8 (inclusive) de la Guía de Ejercicios 1.