Introducción a la Programación

Prof. Agustín Gravano

Primer semestre de 2022

Clase teórica 3: Variables

Programas y datos

La clase pasada empezamos a ver cómo son los datos que un programa puede manejar. Vimos los tipos de datos:

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

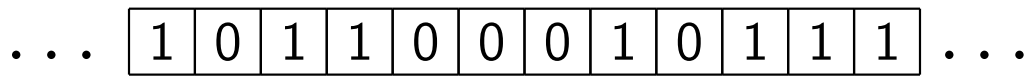
Hoy vamos a empezar a ver qué se puede hacer con los datos.

En el paradigma de programación imperativa, un programa es una secuencia finita de instrucciones.

Memoria

Durante la ejecución de un programa, sus datos se almacenan en la memoria.

La memoria de una computadora es una secuencia numerada de celdas. La unidad elemental es el bit, que toma valores 0 o 1. (El soporte físico puede ser electrónico, magnético u óptico.)

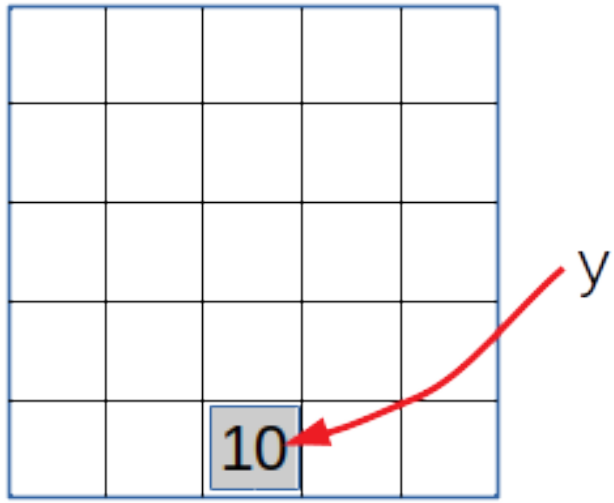


Habitualmente, para referirnos a su tamaño usamos otras unidades:

- 8 bits = 1 byte (unidad mínima más usada)
- 1024 bytes = 1 KB (kilobyte)
- 1024 KB = 1 MB (megabyte)
- 1024 MB = 1 GB (gigabyte)
- 1024 GB = 1 TB (terabyte)
- 1024 TB = 1 PB (petabyte)
- . . .

Variable

En un programa, una variable es un nombre que denota una posición de la memoria, en la cual se almacena un valor.



En esa celda de memoria es posible:

- leer el valor almacenado;
- escribir un valor nuevo, que reemplaza al anterior.

Variable

En Python, los nombres de las variables deben respetar las siguientes reglas sintácticas:

1. Solo puede contener caracteres alfanuméricos y guiones bajos (`A-Z`, a-z, 0-9, _)
2. Tiene que empezar con una letra o un guión bajo
3. Son *case sensitive* (`edad`, `Edad` y `EDAD` son tres variables distintas)

`nrotelefono` ✓	`2nro_telefono` ✗
`nro-telefono` ✗	`nro_telefono` ✓
`_nro_telefono` ✓	`nro.telefono` ✗
`nroTelefono` ✓	`nro telefono` ✗
`nrotelefono2` ✓	

Además, cada lenguaje tiene una lista de palabras reservadas que no pueden usarse como nombres de variables. Ejemplos en Python: `int`, `float`, `True`, `False`, `str`, `def`, `if`, `while`, `for`, `class`, `None`, ...

Asignación

Una asignación es una instrucción que se escribe como una *ecuación orientada*:

VARIABLE = EXPRESIÓN

que almacena en la posición de memoria denotada por la VARIABLE el valor resultante de evaluar la EXPRESIÓN.

Ejemplos:

`x = 1000`	✓	
`1000 = x`	✗	(1000 no es una variable)
`y = x`	✓	
`x = x`	✓	(no tiene efecto)
`x = x + y * 22`	✓	
`x + 1 = y`	✗	(`x + 1` no es una variable)

Declaraciones de tipos

Al crear una variable nueva, algunos lenguajes de programación exigen declarar su tipo.

Por ejemplo, en Java o C++:

```
1  int edad = 27;          // Creo la variable edad, de tipo
2                          // entero, y le asigno el valor 27.
3
4  float peso = 64.9;      // Creo la variable peso, de tipo
5                          // float, y le asigno el valor 64.9.
```

En esos lenguajes, a una variable de tipo `T` solo se le pueden asignar valores de tipo `T`.

Declaraciones de tipos

En otros lenguajes, como Python, no es necesario declarar el tipo de las variables. A una variable se le pueden asignar valores de cualquier tipo:

```
1  x = 27      # Creo la variable x; le asigno un valor entero.
2  type(x)     # Devuelve 'int'.
3
4  x = 64.9    # A la misma variable le asigno un valor float.
5  type(x)     # Devuelve 'float'.
6
7  x = 'Ian'   # A la misma variable le asigno un valor string.
8  type(x)     # Devuelve 'str'.
```

Esta flexibilidad es un arma de doble filo.

Permite escribir código más rápido, pero aumenta el riesgo de cometer errores de tipos, y hace muy difícil encontrarlos.

Sugerencias de tipo

A partir de su versión 3.6, Python permite incluir sugerencias de tipo *(type hints)* para indicar el tipo esperado de las variables:

```
1  edad:int = 27          # Creo la variable edad; indico que es
2                         # de tipo entero; le asigno un valor.
3
4  peso:float = 64.9      # Creo la variable peso; indico que es
5                         # de tipo float; le asigno un valor.
6
7  nombre:str = 'Ian'     # Creo la variable nombre; indico que
8                         # es de tipo string; le asigno un valor.
```

En Python, las sugerencias de tipo son opcionales y no tienen efecto sobre la ejecución del código: el intérprete las ignora por completo.

Están pensadas como ayuda para lxs programadorxs, y algunos (pocos) entornos de desarrollo las están empezando a aprovechar.

En esta materia las sugerencias de tipo son obligatorias. Indicamos el tipo de todas las variables (también de las funciones y sus parámetros). Esto nos aportará mayor claridad, control y confianza en nuestro código.

Expresiones (revisitadas)

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 valor imprime el siguiente código?

```
1  s:str = '123' + '45'
2  n:int = len(s) * 10
3  x:float = n / 2
4
5  print(s + str(n) + str(x))
```

Primero, resolverlo a mano. Después revisar en Python, pero ya no en una consola `ipython`. Ahora escribir un programa (en el editor de texto de Spyder), guardarlo y ejecutarlo.

Estado

Se denomina estado al valor de todas las variables de un programa en un punto de su ejecución.

El estado es una foto de la memoria en un momento determinado.

Ejemplo:

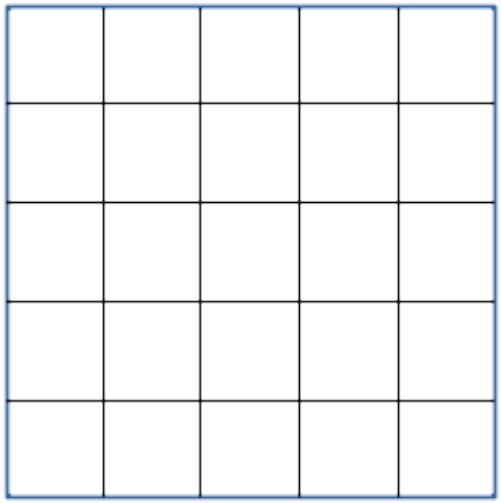
Programa

```
y:int = 10

x:int = y * 2

y = y + 1
```

Memoria



Estado

Se denomina estado al valor de todas las variables de un programa en un punto de su ejecución.

El estado es una foto de la memoria en un momento determinado.

Ejemplo:

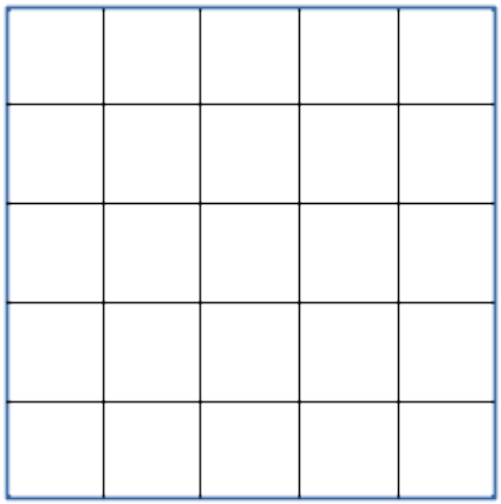
Programa

```
y:int = 10

x:int = y * 2

y = y + 1
```

Memoria



Estado

Se denomina estado al valor de todas las variables de un programa en un punto de su ejecución.

El estado es una foto de la memoria en un momento determinado.

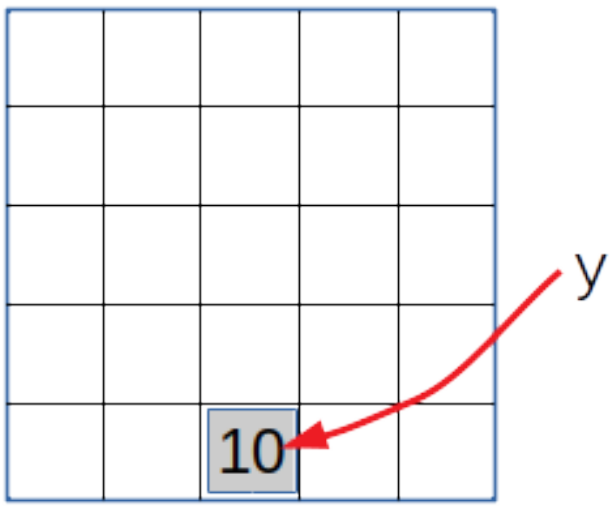
Ejemplo:

Programa

```
y:int = 10
                                    ⇐
x:int = y * 2

y = y + 1
```

Memoria



Estado

Se denomina estado al valor de todas las variables de un programa en un punto de su ejecución.

El estado es una foto de la memoria en un momento determinado.

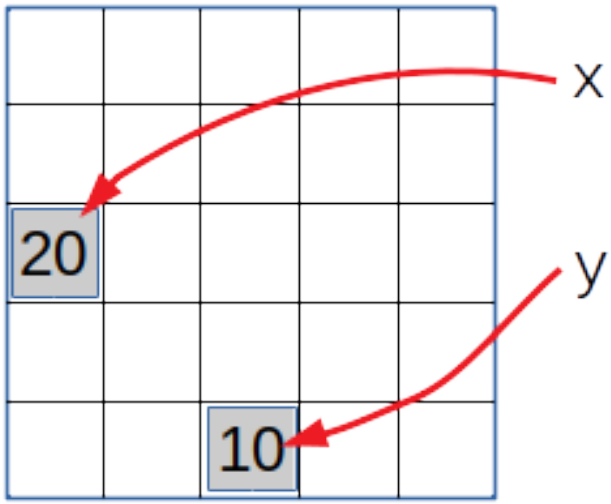
Ejemplo:

Programa

```
y:int = 10

x:int = y * 2
-------------------- ⇐
y = y + 1
```

Memoria



Estado

Se denomina estado al valor de todas las variables de un programa en un punto de su ejecución.

El estado es una foto de la memoria en un momento determinado.

Ejemplo:

Programa

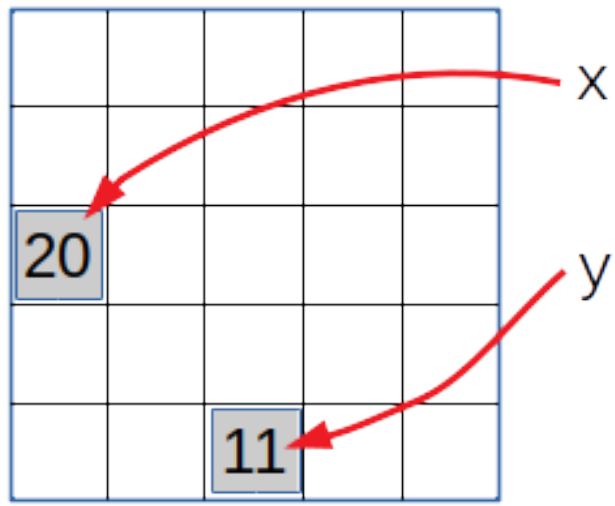
```
y:int = 10

x:int = y * 2

y = y + 1
```

⇐

Memoria



Ejercicio:

Completar el siguiente código, de manera que convierta una distancia en millas a kilómetros. Tener en cuenta que 1mi $\approx$ 1.60934km.

```
1  mi:... = 50.0
2  km:... = ...
3  print(...)
```

La salida debe decir: `50 millas equivalen a 80 kilómetros.`

Repaso de la clase de hoy

- Memoria, variable, asignación.
- Declaraciones de tipos y sugerencias de tipos.
- Estado de un programa.

Bibliografía complementaria:

- APPP2, capítulo 2
- HTCSP3, capítulo 2

Con lo visto, ya pueden resolver toda la Guía de Ejercicios 1.