CS Bridge

Introducción a Python, Variables y Expresiones

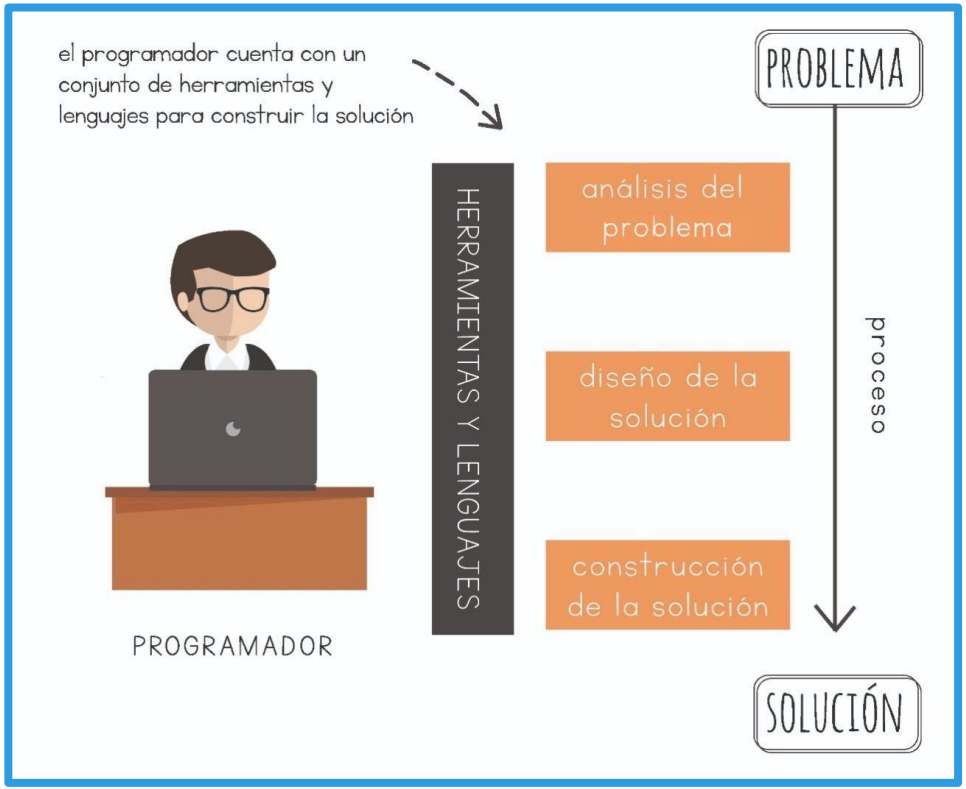
Agenda

1. Bienvenidos a Python: entrada, salida, procesamiento
2. Variables
    - Asignación y uso
    - Tipos
3. Uso de variables en expresiones

ADIOS KAREL

...

Solucionar un problema es construir un programa



DEL ALGORITMO AL PROGRAMA DE COMPUTADOR

1. Inicio
2. Si pasa A, se hace B
3. ....
4. ....

Algoritmo

Java, C++, Python

Lenguaje de máquina

PSEUDOCÓDIGO → CÓDIGO FUENTE → CÓDIGO OBJETO

DEL ALGORITMO AL PROGRAMA DE COMPUTADOR

Herramientas & Lenguajes

Entorno de desarrollo (IDE)

- Editor de código fuente
- Compilador
- Depurador

≠

Lenguaje de programación

- Lenguaje de alto nivel

Bienvenidos a Python

Bienvenidos a Python

Guido van Rossum
(Creador de Python)

Empresas que usan Python:

NETFLIX

Google

Uber

YouTube

Pinterest

Usando Python

- **Python debe ser instalado y configurado antes de usarse**
  - Algo que se instala es el intérprete
- **El intérprete se puede usar en:**
  - Modo interactivo
  - Modo archivo

Ejemplo

1. Supón que una clínica te contrata para que propongas una solución que ayude a monitorear el estado de salud de pacientes no críticos, desde sus casas
    1. Colectar temperatura y nivel de oxígeno
    2. Enviarlos al médico tratante
2. Una solución es a través de un programa de computador

¿Cómo trabaja el computador?

Entrada, Procesamiento y Salida

1. Recibir entrada
    1. Cualquier dato que el programa recibe
2. Procesar la entrada
    1. Por ej., transformarla, hacer un cálculo
3. Producir una salida

¿Cómo el computador produce salidas?

Función `print`

```
print("hola a todos")
```

`print` → imprime texto en la pantalla

El texto debe colocarse entre comillas dobles o simples ('texto')

La selección depende de lo que se quiere imprimir

Comillas dobles cuando el texto tiene comillas simples

`print("Dunkin' donuts")` → `Dunkin' donuts`

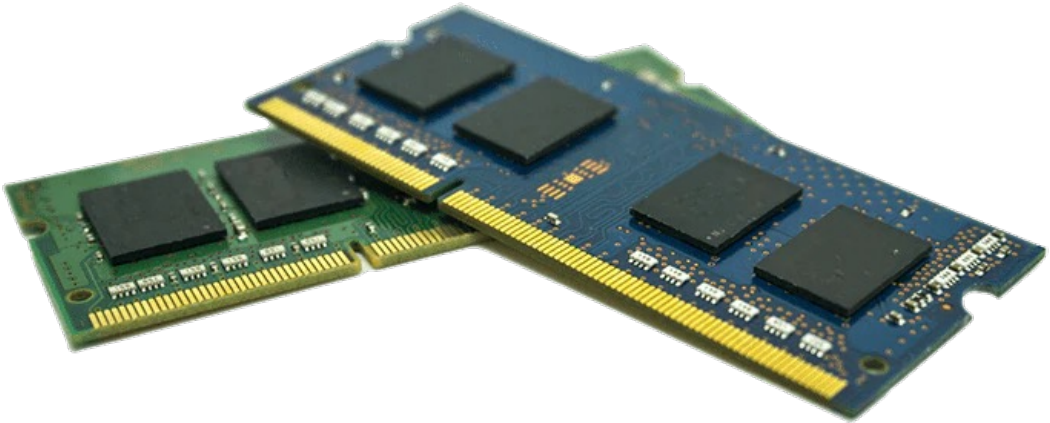
Comillas simples cuando el texto tiene comillas dobles

`print('di "hola" Karel')` → `di "hola" Karel`

¿Cómo el computador almacena datos?

¡Tu computador tiene una memoria!

Los datos son almacenados en la memoria (RAM) del computador



¿Cómo el computador almacena datos en los programas?

¿Cómo el computador almacena datos en los programas?

¡Variables!

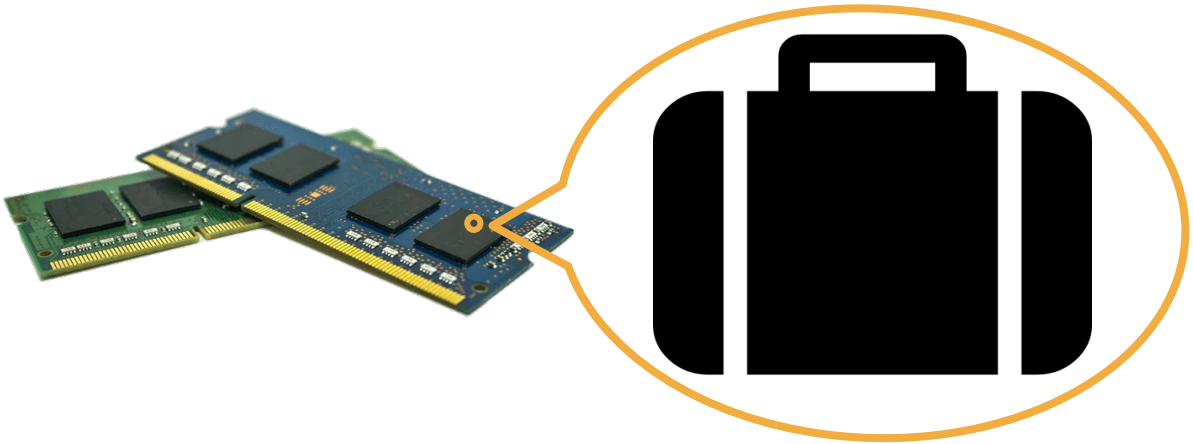
Variable

Definición

variable
Forma en la que un programa guarda información asociando nombre a un valor

Analogía de la maleta

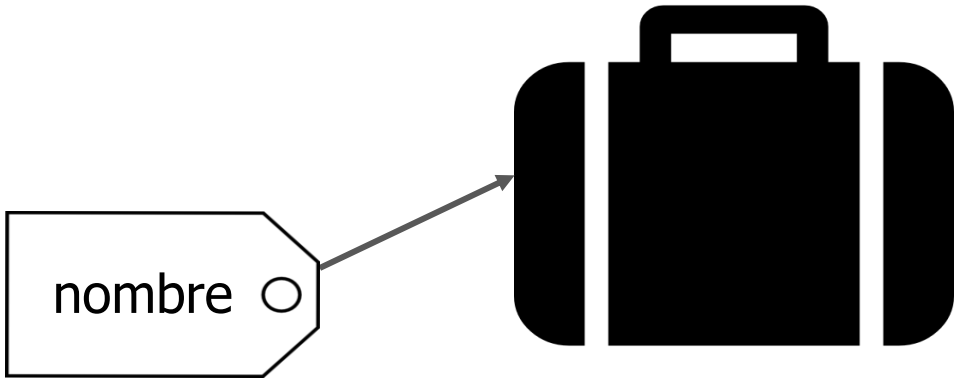
- Cuando almacenas información en Python, se convierte en un **objeto Python**
  - Los objetos son de diferente tamaño y tipo.



Objeto Python almacenado en la memoria

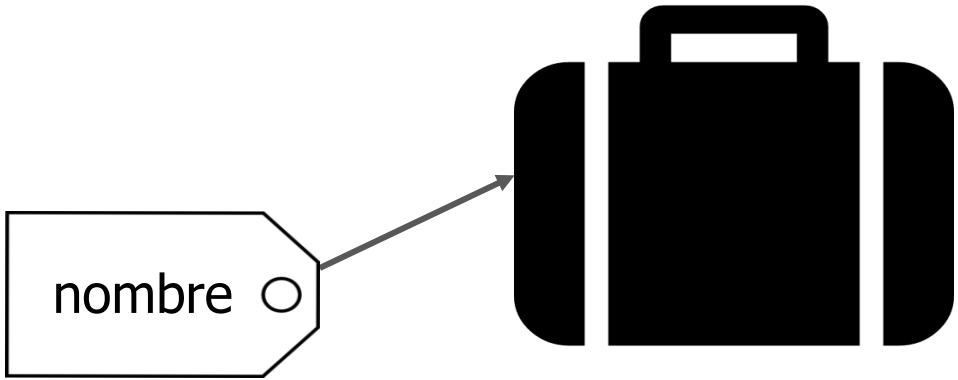
Analogía de la maleta

- Un objeto Python es como una maleta almacenada en la memoria del computador
- El tamaño de la maleta depende de lo que se está almacenando



Analogía de la maleta

- Una variable es una etiqueta para la maleta que le da un nombre



Variable

- **Variable: nombre que representa un valor almacenado en la memoria del computador**
  - Se usa para acceder y manipular datos almacenados en memoria
  - Una variable sirve para referenciar el valor que representa
- **Asignación: usada para crear una variable y hacer referencia a su valor**
  - Formato general `variable = expresión`
    - Ejemplo: `edad = 15`
    - Operador de asignación: signo igual (=)

Reglas para nombrar variables en Python

- El nombre no puede ser una palabra clave de Python

~~while = 5~~

- No puede contener espacios

~~mi edad = 17~~

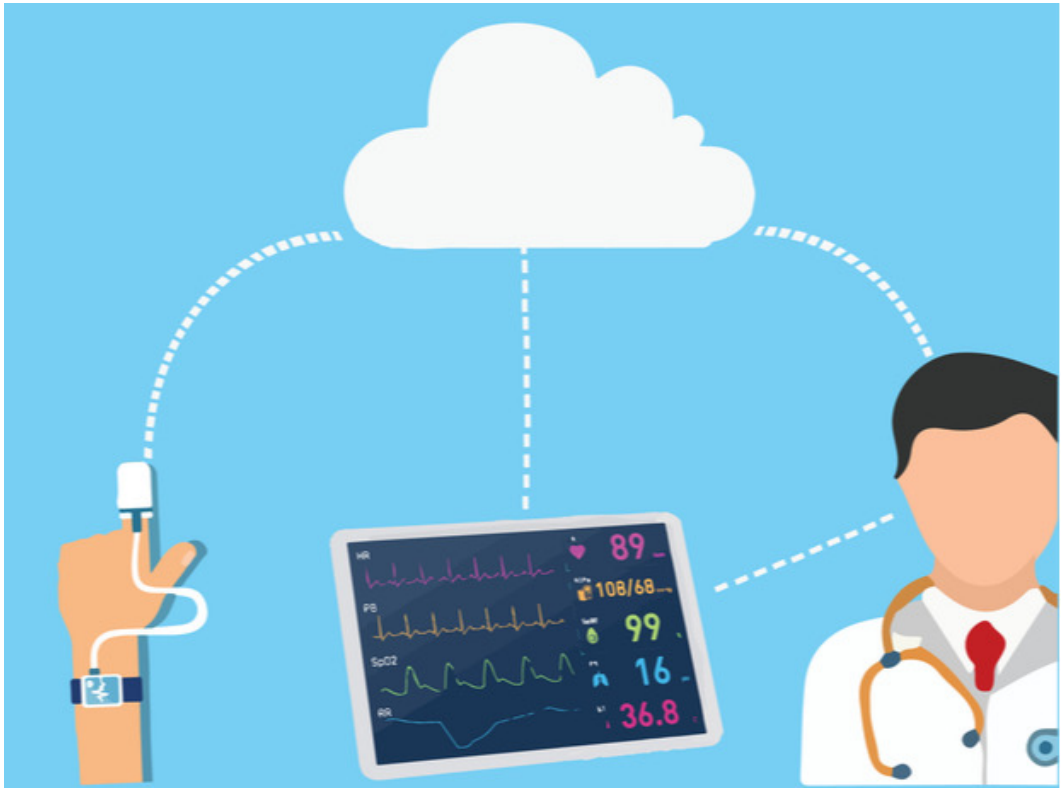
- El 1° carácter debe ser una letra o un guion bajo
- Después del 1° carácter se pueden usar letras, dígitos o guion bajo

~~1x = 5.2~~

- Los nombres de las variables son sensibles a mayúsculas y minúsculas

- **El nombre debe reflejas su uso**
  - x = 10 versus mi_nota = 10

Volvamos al ejemplo de monitoreo de pacientes



Ejemplo de variable

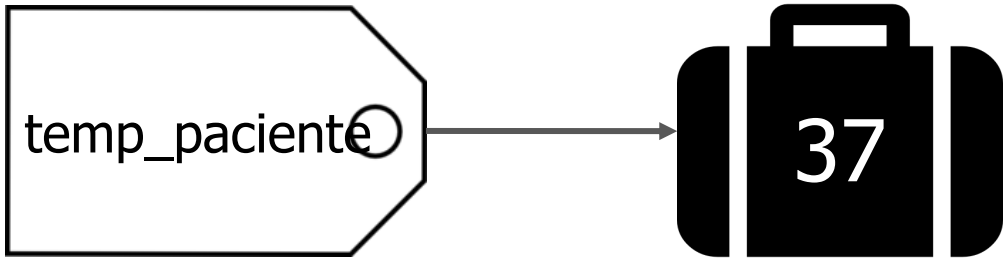
Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente = 37
```

Ejemplo de variable

Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

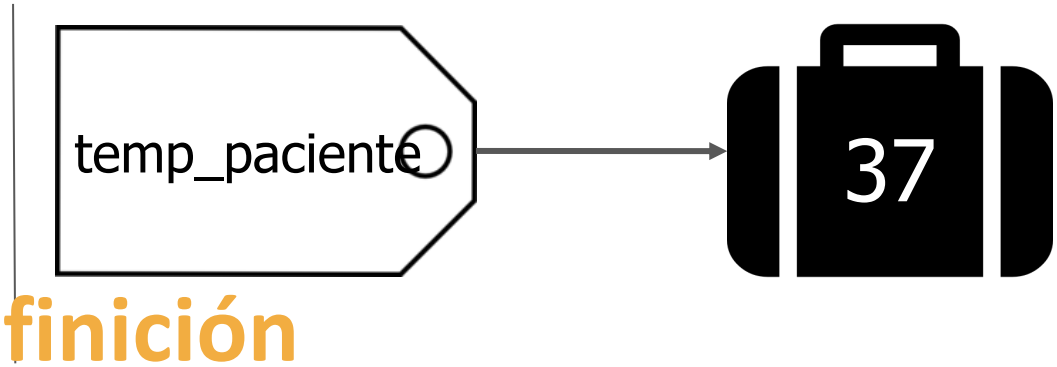
```
temp_paciente = 37
```



Ejemplo de variable

Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente = 37
```

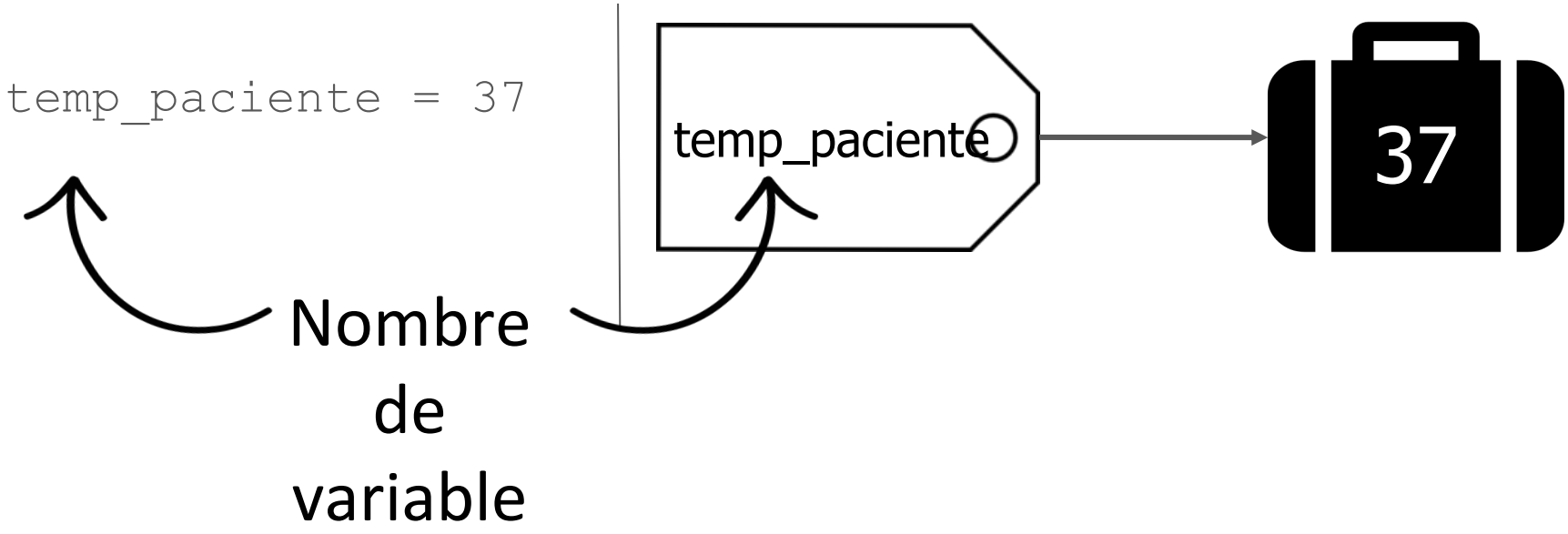


Definición

Asignación de variable
Asociar un nombre a un valor (usa el símbolo =)

Ejemplo de variable

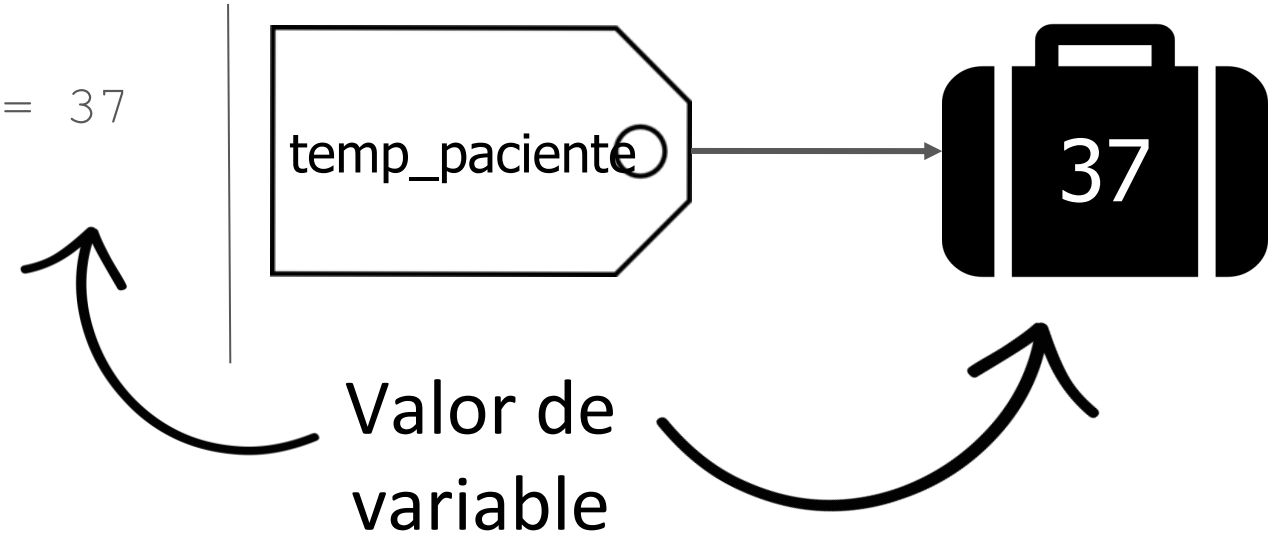
Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:



Ejemplo de variable

Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente = 37
```

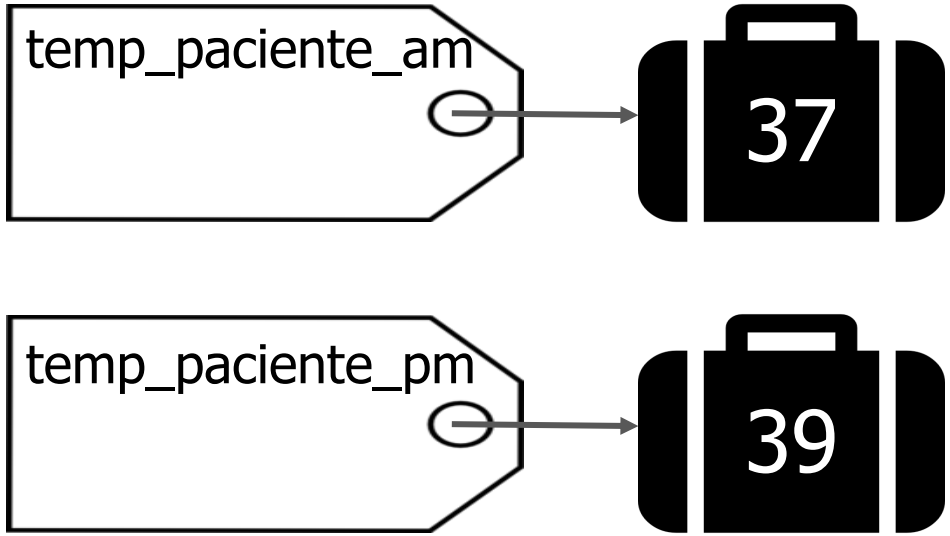


Ejemplo de variable

Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente_am = 37

temp_paciente_pm = 39
```



Ejemplo de variable

Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente_am = 37

temp_paciente_pm = 39

dif_temperatura = temp_paciente_am - temp_paciente_pm
```

Ejemplo de variable

Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente_am = 37

temp_paciente_pm = 39

dif_temperatura = temp_paciente_am - temp_paciente_pm
```

Asignación de variable

Ejemplo de variable

Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente_am = 37

temp_paciente_pm = 39

dif_temperatura = temp_paciente_am - temp_paciente_pm
```

Lado derecho se evalúa primero

Ejemplo de variable

Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente_am = 37

temp_paciente_pm = 39

dif_temperatura = temp_paciente_am - temp_paciente_pm
```

Ejemplo de variable

Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente_am = 37

temp_paciente_pm = 39

dif_temperatura = temp_paciente_am - temp_paciente_pm
```

Definición

Uso de variables

Obtener el valor asociado a un nombre

Ejemplo de variable

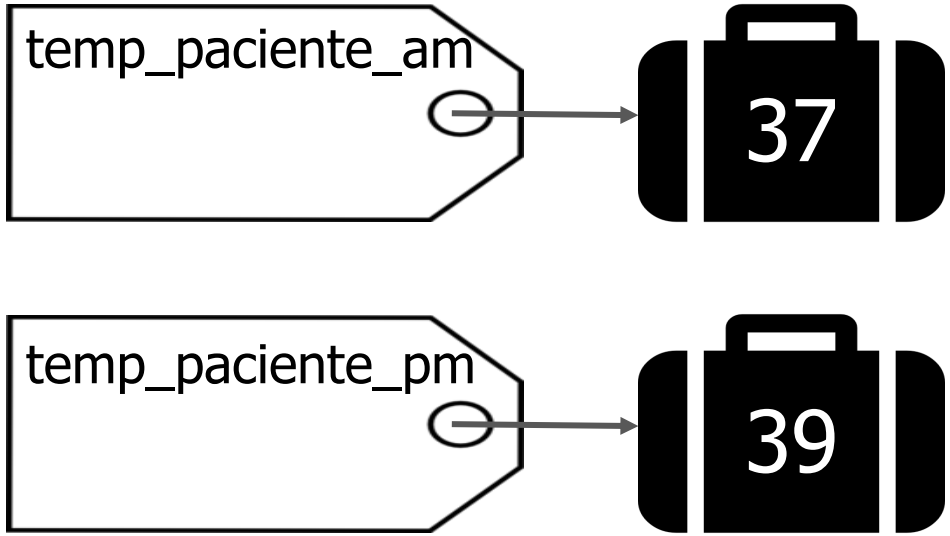
Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente_am = 37
temp_paciente_pm = 39
dif_temperatura = temp_paciente_am
                - temp_paciente_pm
```

37

39

Al usar variables recuperamos los valores guardados en las maletas



Ejemplo de variable

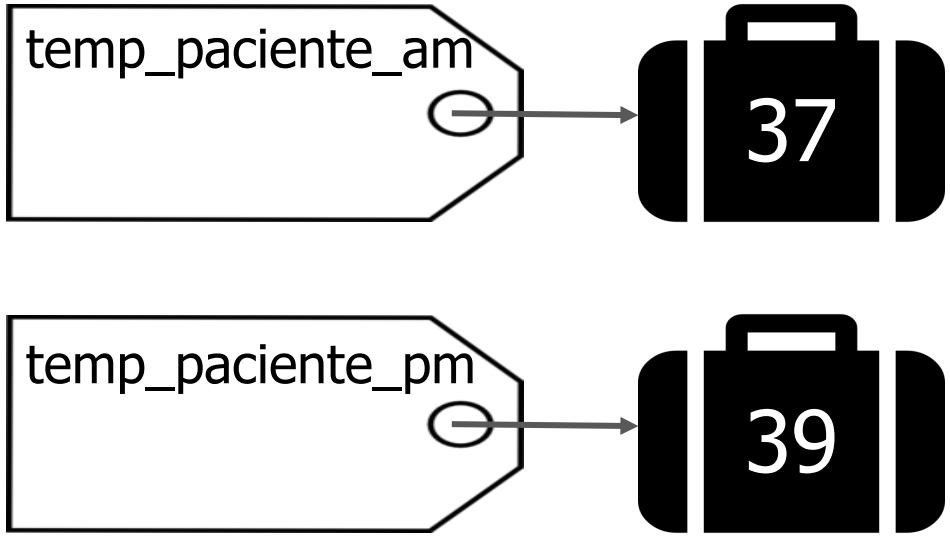
Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente_am = 37
temp_paciente_pm = 39
dif_temperatura = temp_paciente_am
                - temp_paciente_pm
```

37

39

Entonces se puede evaluar el lado derecho de la asignación

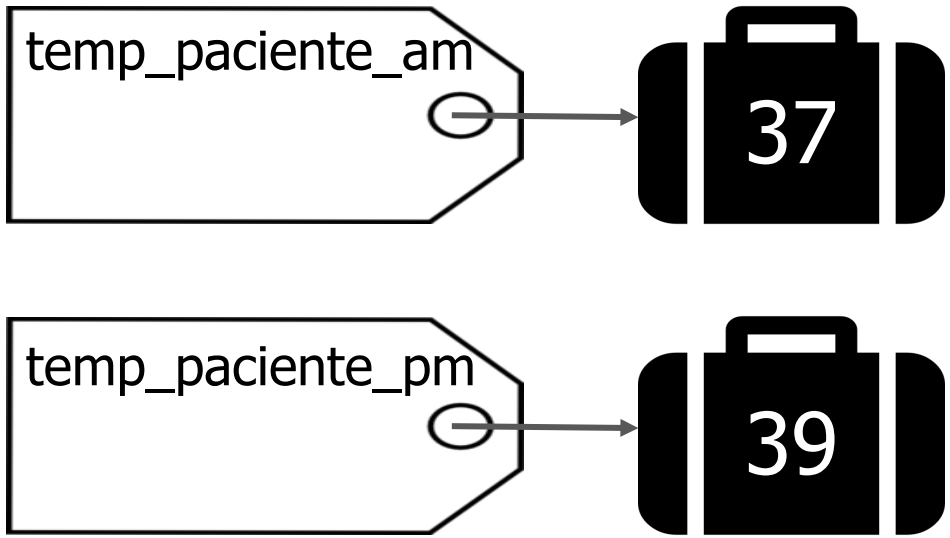


Ejemplo de variable

Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente_am = 37
temp_paciente_pm = 39

dif_temperatura = -2
```



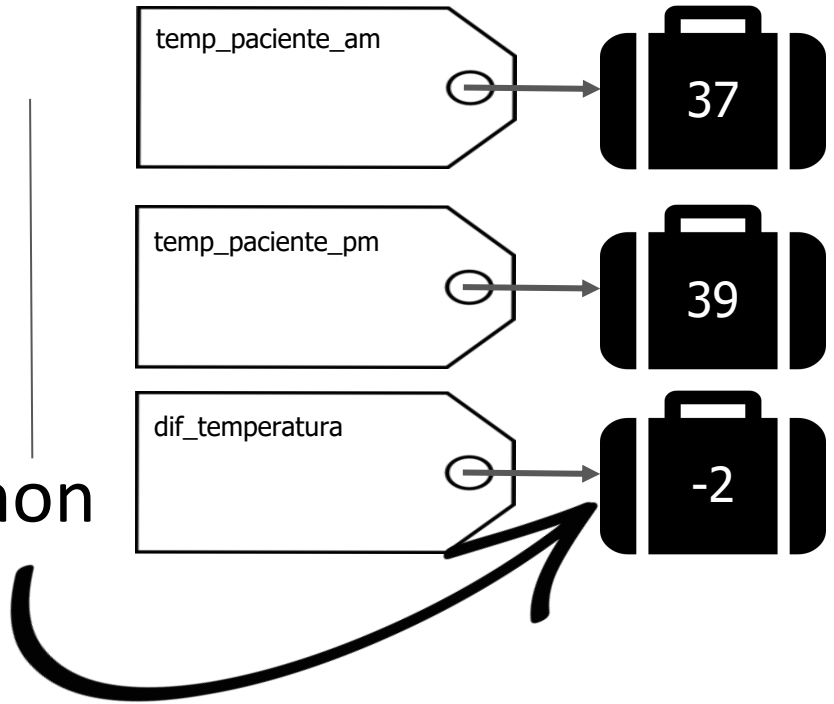
Ejemplo de variable

Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente_am = 37
temp_paciente_pm = 39

dif_temperatura = -2
```

Es un nuevo objeto Python

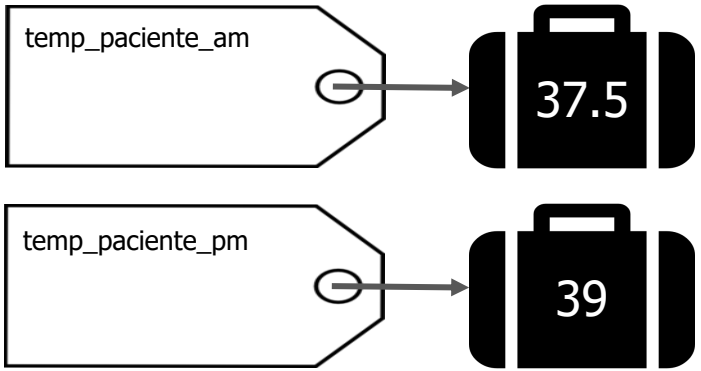


Ejemplo de variable

Supón que el programa que estás construyendo monitorea el comportamiento de la temperatura del paciente:

```
temp_paciente_am = 37
temp_paciente_pm = 39

temp_paciente_am = 37.5
```



Python maneja las maletas por nosotros cuando usamos variables

¿Cómo el computador recibe datos de entrada?

función `input`

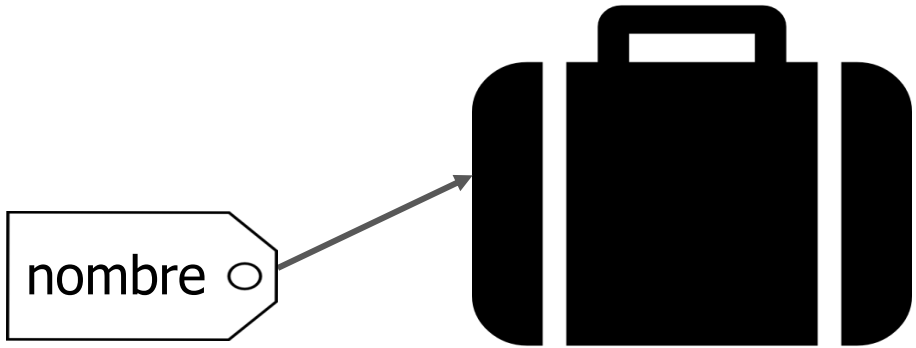
```
temp_paciente_am = input("Introduce la temperatura am: ")
```

- **`input`** obtiene el texto digitado por el usuario
  - Imprime el texto especificado entre comillas
  - Espera la entrada del usuario
  - Asocia la entrada a la variable (`temp_paciente_am`)
  - Considera la entrada como texto aún si el usuario digita un número
- Hablaremos más de input después

Tipos de datos

Analogía de la maleta

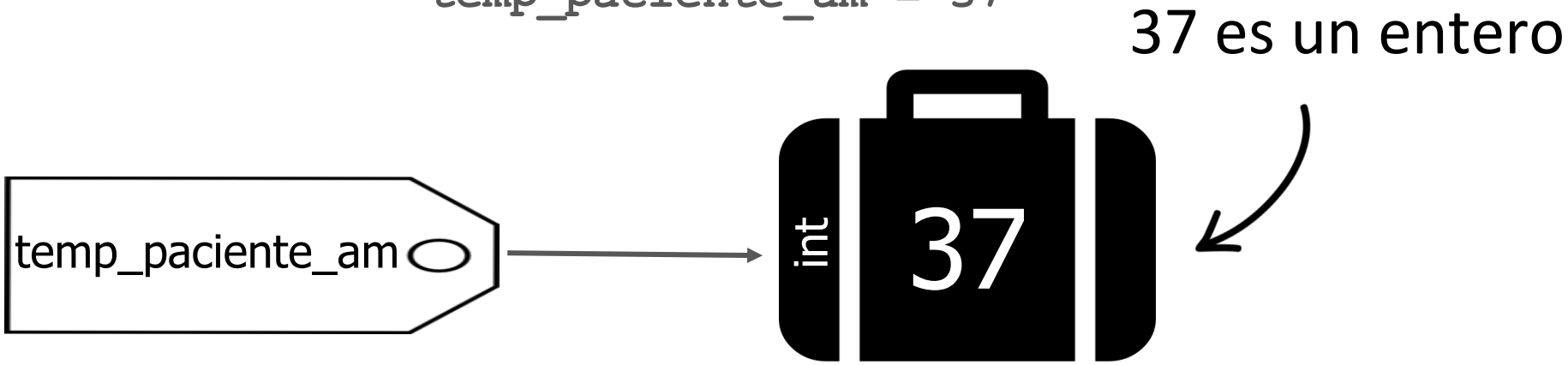
- Cuando almacenas información en Python, se vuelve un objeto Python
  - **Los objetos vienen en tamaños y tipos diferentes**



Analogía de la maleta

- Cuando almacenas información en Python, se vuelve un objeto Python
  - **Los objetos vienen en tamaños y tipos diferentes**

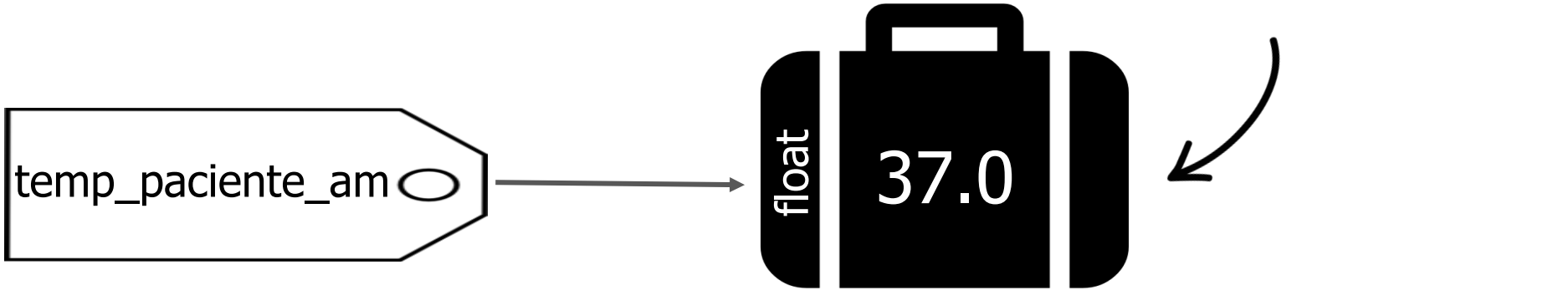
```
temp_paciente_am = 37
```



Analogía de la maleta

- Cuando almacenas información en Python, se vuelve un objeto Python
  - **Los objetos vienen en tamaños y tipos diferentes**

```
temp_paciente_am = 37.0
```



Tipos

Todos los objetos Python tienen un tipo

- Python automáticamente sabe cuál es el tipo a partir del valor
  - Las variables son "**dinámicamente tipadas**": no necesitas especificar el tipo al que tiene que asociarse el objeto Python

Tipos

Todos los objetos Python tienen un tipo

- Enteros (int) – números sin decimales

```
temperatura = 37
```

- Flotantes (float) – números con decimales

```
temperatura = 37.5
```

- Booleanos (boolean) – verdadero (True) o falso (False)

```
soleado_hoy = True
```

- Cadenas de caracteres (string) – conjunto de caracteres

```
mi_nombre = 'María'
```

Tipos

Volvamos al ejemplo de monitoreo de pacientes…

¿Qué tipo usarías para almacenar lo que sigue?

Piensa/Comparte con tus vecinos

Tipos

Volvamos al ejemplo de monitoreo de pacientes…

¿Qué tipo usarías para almacenar lo que sigue?

- Peso del paciente
- Número de días que han pasado desde la última visita al médico
- Si el paciente se ha enfermado de gripa o no
- Número de hijos del paciente

Tipos

Volvamos al ejemplo de monitoreo de pacientes…

¿Qué tipo usarías para almacenar lo que sigue?

- Peso del paciente → **Flotante**
- Número de días que han pasado desde la última visita al médico → **Entero**
- Si el paciente se ha enfermado de gripa o no → **Booleano**
- Número de hijos del paciente → **Entero**

Conversión explícita entre tipos

```
num1 = 5
num2 = 2
num3 = 1.9
```

- Usa **`float(`*value*`)`** para crear un nuevo número flotante

  `float(num1)` => **`5.0`** **`(float)`**

  – Nota que **`num1`** no se cambia. Se crea un nuevo valor.

  `num1 + float(num2)` => **`7.0`** **`(float)`**

  `num1 + num2` => **`7`** **`(int)`**

- Usa **`int(`*value*`)`** para crear un nuevo número entero (se trunca toda la parte decimal)

  `int(num3)` => **`1`** **`(int)`**

  `int(-2.7)` => **`-2`** **`(int)`**

Conversión explícita entre tipos

```
num1 = 5
num2 = 2
num3 = 1.9
```

- Usa **`str(`*value*`)`** para crear un texto a partir de un número

```
str(num1)        => '5'      (String)
str(num2)        => '2'      (String)
str(num3)        => '1.9'    (String)
```

¿Cómo el computador procesa la información almacenada?

Expresiones

Recorderis: expresiones

- En Karel, solo vimos “expresiones booleanas” que producen verdadero/falso
- En Python, las expresiones producen cualquier tipo de dato
- El computador produce un único valor por cada expresión
- Usaremos operadores para combinar variables y literales en una expresión

Objetos Python escritos directamente en el código, ej., 37

```
temperatura = 37
```

Haciendo cálculos

- **Expresiones matemáticas: hacen un cálculo y devuelven un valor**
  - Operadores matemáticos: herramientas integradas para ejecutar cálculos
  - Operandos: valores que rodean al operador
    - Las variables pueden ser usadas como operandos
  - Producen un valor que típicamente es asignado a una variable

Haciendo cálculos

Operadores aritméticos

*	Multiplicación
/	División
//	División entera
%	Módulo (resto)
+	Suma
-	Resta
**	Exponenciación

- **Dos tipos de división:**
  - `Operador /` ejecuta una división flotante
  - `Operador //` ejecuta una división entera

Haciendo cálculos

```
num1 = 5
num2 = 2
```

- Operaciones sobre tipos numéricos (**`int`** and **`float`**)
- Operadores

				num3
`+`	"suma"	Ex.:	`num3 = num1 + num2`	7
`-`	"resta"	Ex.:	`num3 = num1 - num2`	3
`*`	"multiplicación"	Ex.:	`num3 = num1 * num2`	10
`/`	"división"	Ex.:	`num3 = num1 / num2`	2.5
`//`	"división entera"	Ex.:	`num3 = num1 // num2`	2
`%`	"módulo/resto"	Ex.:	`num3 = num1 % num2`	1
`**`	"exponenciación"	Ex.:	`num3 = num1 ** num2`	25
`-`	"negativo"	Ex.:	`num3 = -num1`	-5

Haciendo cálculos

Operadores aritméticos

*	Multiplicación
/	División
//	División entera
%	Módulo (resto)
+	Suma
-	Resta
**	Exponenciación

Operador	Precedencia
()	1
*, /, //, %	2
+, -	3

Sirve para asociar operandos y operadores de izq. a der.

Haciendo cálculos

Hagamos ejercicios...

- 4 + 2 * 3
- 5 + 1 / 2 - 4
- 15 / 2.0 + 6
- 5 + 1 / (2 - 4)
- 5 + 1 // (2 - 4)
- 1 * 2 + 3 * 5 % 4

Operador	Precedencia
()	1
*, /, //, %	2
+, -	3

¡Pensemos todos en esto!

Haciendo cálculos

Hagamos ejercicios…

- 4 + 2 * 3
- 5 + 1 / 2 - 4
- 15 / 2.0 + 6
- 5 + 1 / (2 - 4)
- 5 + 1 // (2 - 4)
- 1 * 2 + 3 * 5 % 4

NOTA: Cualquiera de los literales puede remplazarse con variables que tengan asociado el mismo valor

Haciendo cálculos

Hagamos ejercicios...

- 4 + 2 * 3
- 5 + 1 / 2 - 4
- 15 / 2.0 + 6
- 5 + 1 / (2 - 4)
- 5 + 1 // (2 - 4)
- 1 * 2 + 3 * 5 % 4

Por ejemplo:

```
x = 2
4 + x * 3
```

El resultado es 10, como la nuestra 1° expresión

Conversión implícita de tipos

```
num1 = 5
num2 = 2
num3 = 1.9
```

- Operaciones entre dos **`int`**s (excepto **`/`**) devolverán un valor de tipo **`int`**

  `num1 + 7  => 12  (int)`

  – La división (**`/`**) de dos **`int`**s devuelve un **`float`**, aún si el resultado está redondeado (Ej.: **`6 / 2`** = **`3.0`**)

- Si cualquiera de los operandos es **`float`**, el resultado es **`float`**

  `num3 + 1  => 2.9  (float)`

- La exponenciación depende del resultado:

  `num2 ** 3  => 8  (int)`

  `2 ** -1  => 0.5  (float)`

Your job: Play with variables!

Expression Shorthands

```
num1 = 5
num2 = 2
num3 = 1.9
```

num1 = num1 + 1	same as	**num1 += 1**
num2 = num2 – 4	same as	**num2 –= 4**
num3 = num3 * 2	same as	**num3 *= 2**
num1 = num1 / 2	same as	**num1 /= 2**

- Generally:

  variable* = *variable operator (***expression***)

  is same as:

  variable operator**=** ***expression***

How should we store information if it is known and never changes?

How should we store information if it is known and never changes?

Constants!

Constants

Constants are like variables that don't change

- Constants give descriptive names to literals

Style note

constants
Use constants with descriptive names instead of literals directly in your code.

```
d = 299792458 * 3
```

```
SPEED_OF_LIGHT = 299792458
d = SPEED_OF_LIGHT * 3
```

Constants

Constants are like variables that don't change

- Constants give descriptive names to literals

- Use all capital letters and snake_case when naming constants

Style note

constant names
Use all capital letters and snake_case, for example `MY_CONSTANT = 500`.

Constants

Constants are like variables that don't change

- Constants give descriptive names to literals
- Use all capital letters and snake_case when naming constants
- Constants are usually assigned outside functions and at the top of your program file (underneath the imports)

Example of Using Constants

```
"""
File: constants.py
------------------
An example program with constants
"""

INCHES_IN_FOOT = 12

def main():
    feet = float(input("Enter number of feet: "))
    inches = feet * INCHES_IN_FOOT
    print("That is " + str(inches) + " inches!")

# This provided line is required at the end of a Python file
# to call the main() function.
if __name__ == '__main__':
    main()
```