Entrada, salida y verificación

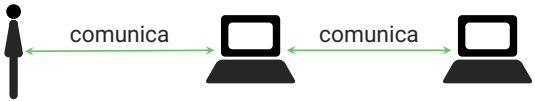
CI-0202 Principios de informática

Sivana Hamer - sivana.hamer@ucr.ac.cr
Escuela de Ciencias de la Computación e Informática, Universidad de Costa Rica

UNIVERSIDAD DE COSTA RICA

Entrada y salida

Se requiere **comunicar** con las computadoras para que podamos transmitir datos y recibir resultados de los programas.

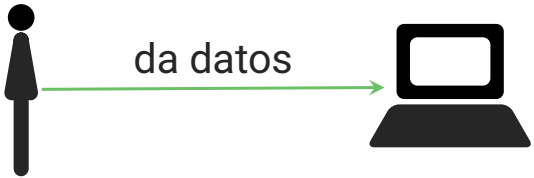


Nota

Vamos a ver como se transmite información entre humanos y computadoras.

Entrada

Manera en que se **proporciona al programa datos** para que utilice en su ejecución.



Nota

La entrada de datos nos permite resolver *familias de problemas* y no solo la *instancia*. Por ejemplo, si tenemos un problema que grafica ecuaciones lineales, la familia de problemas que resuelve es el universo de problemas de ecuaciones lineas. Si le decimos al programa que grafique $y = 2x + 10$, esa es la instancia del problema.

Entrada en Python (1)

Usa la instrucción **input()** que hace lo siguiente:

- Imprime el mensaje.
- Pausa el programa para que la persona escriba.
- Resume el programa cuando se presiona enter.
- Guarda lo que escribió la persona dentro de una variable.

Entrada en Python (2)

Ejemplo input

```
# Funcionamiento general de input
>>> entrada = input("Lo que se digita aqui se muestra. ")
Lo que se digita aqui se muestra. Hola
>>> entrada
Hola

#Ejemplo de pedir el nombre y mostrarlo
>>> nombre = input("Escriba su nombre: ")
Escriba su nombre: Sivana
>>> print("Hola ", nombre)
Hola Sivana
```

Entrada en Python (3)

Nota

Para el mensaje, input recibe strings.

```
 # Input con doble comilla
>>> nacionalidad = input("Escriba su nacionalidad: ")
Escriba su nacionalidad: tica
 # Input con una comilla
>>> comida_favorita = input('Escriba su comida favorita: ')
Escriba su comida favorita: pizza
# Input con tres doble comillas.
>>> color_favorito = input("""Escriba su color favorito:
""")
Escriba su color favorito:
naranja
# Input with a variable
>>> mensaje = "Escriba su provincia de residencia: "
provincia = input(mensaje)
Escriba su provincia de residencia: guanacaste
```

Entrada en Python (4)

Importante

Input devuelve strings por lo que hay que **convertir** la entrada si se quiere utilizar otros tipos de datos.

```
>>> edad = input("Escriba su edad: ")
Escriba su edad: 22
>>> type(edad)
<class 'str'>
```

Convertir tipos de datos

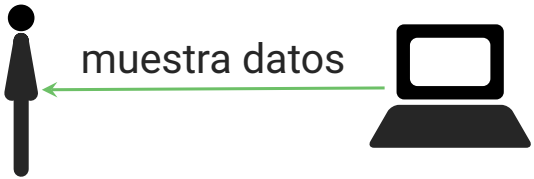
Convertir tipos de datos implica **transformar un tipo de dato a otro**. En python, se utilizan los nombres de los tipos de datos (int, float, str, y bool) para convertir a un tipo de dato respectivo.

Ejemplo conversión de input

```
#Programa que suma dos numeros
#Pide ambos numeros
>>> numero1 = float(input("Ingrese el primer numero: "))
Ingrese el primer numero: 5
>>> numero2 = float(input("Ingrese el primer numero: "))
Ingrese el primer numero: 5
#Suma los numeros
>>> total = numero1 + numero2
#Imprime el resultado
>>> print(f"{numero1} + {numero2} = {total}"")
5.0 + 5.0 = 10.0
```

Salida

Manera en que se **muestran los datos** resultados del programa.



Nota

Generalmente se muestran los resultados del programa como la salida.

Salida en Python

Usa la instrucción **print()** que imprime un mensaje.

Ejemplo print

```
>>> print("Esta es la salida del programa.")
Esta es la salida del programa
```

Nota

La salida del programa tambien se llama output.

Fun fact!

El **in**put **mete** y el **out**put **saca**.

Formato de entrada y salida en Python (1)

Es buena práctica **imprimir mensajes con formatos** que sean significativos.

Ejemplo formato

```
>>> #Programa que calcula area de un circulo
>>> from math import pi
>>> #Formato 1: Cuesta mas entender
>>> r = float(input("r = "))
r = 3
>>> a = r**2 * pi
>>> print(a)
28.274333882308138

>>> #Formato 2: Mas sencillo de entender
>>> r = float(input("Ingrese el radio: "))
Ingrese el radio: 3
>>> a = r**2 * pi
>>> print(f"Para un circulo de radio {r} su area es {a}.")
Para un circulo de radio 3.0 su area es 28.274333882308138.
```

Formato de entrada y salida en Python (2)

Varias maneras de utilizar el formato:

- Concatenación
- Sustitución

Nota

Los formatos que veremos se pueden utilizar para cualquier string, pero usualmente se utilizan en la salida. Por lo que, los ejemplos se realizan con las salidas.

Formato concatenación

Consiste que en **medio de crear el texto, se agregan las variables y se convierten a string**.

Ejemplo concatenación

```
>>> x = float(input("Ingrese un numero: "))
Ingrese un numero: 5
>>> print("El cubo de " + str(x) + " es " + str(x**3))
El cubo de 5.0 es 125.0
```

Nota

Funciona mediante la concatenación (“pegan”) de hileras.

Formato sustitución (1)

Consiste que al crear el texto se inserten comodines de las variables, que luego Python los sustituye por los valores.

Nota

Se puede realizar la sustitución en Python con *format* o *f-strings* (el que vamos a ver).

Ejemplo sustitución

```
>>> print(f"El cubo de {5} es {5**3}")
El cubo de 5 es 125
```

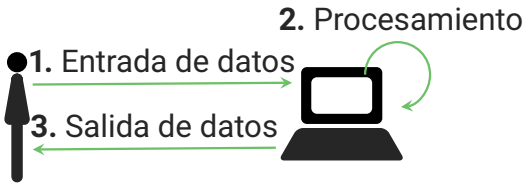
Formato sustitución (2)

Se puede formatear para mostrar una cantidad de digitos fijo para números.

Ejemplo sustitución

```
>>> from math import pi
>>> radio = float(input("Ingrese el radio: "))
Ingrese el radio: 6
>>> area = radio**2 * pi
>>> print(f"Para un circulo de radio {radio:010} su area es {area:.4f}")
Para un circulo de radio 00000006.0 su area es 113.0973
```

Proceso de los programas



Nota

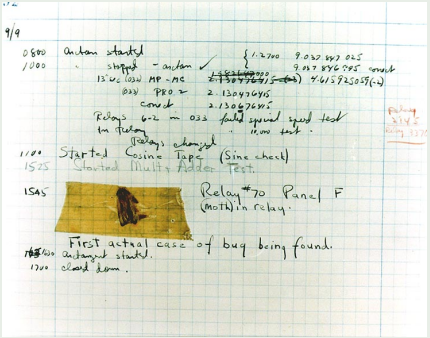
Todas las etapas pueden tener varios pasos.

Errores

Son **problemas** en el programa. También son llamados **bugs**.

Fun fact!

El primer bug de software fue una polilla que estaba en una computadora.



Errores sintácticos

Son errores cuando no se siguen la estructura que tiene que tener un programa. Significan que las **instrucciones no tienen sentido gramatical para el lenguaje de programación**. Provocan que se detenga la ejecución del programa.

Ejemplo error sintactico

```
>>> print("Hola mundo')
File "<stdin>", line 1
    print("Hola mundo')
                      ^
SyntaxError: EOL while scanning string literal
```

Errores de tiempo de ejecución (1)

Son errores que ocurren **cuando se ejecuta el programa**. También son llamados **excepciones** ya que sucenden en eventos excepcionales. Existen muchos errores de ejecución.

Ejemplo excepción de valor

```
>>> int('h0l4')
Traceback (most recent call last):
  File ".../program.py", line 1, in <module>
    int('h0l4')
ValueError: invalid literal for int() with base 10: 'h0l4'
```

Errores de tiempo de ejecución (2)

Ejemplo excepción de tipo de dato

```
>>> '2' + 2
Traceback (most recent call last):
  File ".../program.py", line 1, in <module>
    '2' + 2
TypeError: can only concatenate str (not "int") to str
```

Ejemplo excepción de división de cero

```
>>> 1 / 0
Traceback (most recent call last):
  File ".../program.py", line 1, in <module>
    1 / 0
ZeroDivisionError: division by zero
```

Errores de tiempo de ejecución (3)

Ejemplo excepción de nombre

```
>>> variable + 1
Traceback (most recent call last):
  File ".../program.py", line 1, in <module>
    variable + 1
NameError: name 'variable' is not defined
```

Manejo de errores de tiempo de ejecución

Se puede manejar las excepciones con las instrucciones *try* y *exception*

Ejemplo manejo de excepciones

```
>>> #Bloque que todo funciona
>>> try:
>>>     edad = int(input("Escriba su edad: "))
>>>     print(f"Su edad es {edad}")
>>> #Bloque de excepcion/error de tiempo de ejecucion
>>> except:
>>>     print("No se dio un numero como la edad.")
>>> print("Fin del programa")
```

Indentación

Campos antes de las instrucciones. Para indentar se utiliza la tecla de tabulador. Todas las instrucciones dentro del mismo nivel de indentación conforman un **bloque de instrucciones**. Se puede tener indentación de varios niveles.

Errores semanticos

Son errores donde **el resultado que se da no es el deseado**. Pasa cuando hay un error en la lógica del programa. No provoca problemas de ejecución.

Nota

Para resolver estos errores se requiere revisar el código del programa mediante pruebas y cambiar la lógica del programa.

Referencias I

N. S. W. Center. (2012) First computer bug, 1945. [Image]. [Online]. Available: https://commons.wikimedia.org/wiki/File:First_Computer_Bug,_1945.jpg

L. Villalobos, “Entrada y salida,” Material del curso CI-0202, Universidad de Costa Rica, 2019.

C. Swaroop, *A Byte of Python*. Independent, 2020.

J. Elkner, A. B. Downey, and C. Meyers, “How to think like a computer scientist: Learning with python,” 2012.