Práctica 3:
Introducción a Python

Índice

Objetivos y planificación

Objetivos

- Conocer la estructura básica de un programa Python
- Repasar los tipos de datos básicos y conversiones entre ellos
- Conocer la asignación de variables
- Diferenciar entre Expresiones y Sentencias
- Introducir mínimamente las Sentencias Condicionales
- Manejar funciones
- Conocer algunas funciones de Entrada y Salida
- Comprender el orden de ejecución cuando hay expresiones anidadas

Planificación

1. Comenzar con el programa más sencillo posible, el "Hola mundo".
2. Desarrollara una aplicación que pida datos médicos.
3. Desarrollar una aplicación que permita cifrar/descifrar mensajes usando el cifrado del César.

Hola mundo

Estructura básica

Esta es la estructura básica que usaremos en esta asignatura.

Todo el código debe ir dentro de la función main.

```
def main():
    # Aquí va el código del programa
    ...


if __name__ == "__main__":
    main()
```

Programa 1.0 - Hola mundo

Tradicionalmente este es el primer programa que se hace.

Crea un fichero main.py y copia el código.

```
def main():
    print("Hello, World!")


if __name__ == "__main__":
    main()
```

Indentación

En Python la **indentación** es **MUY IMPORTANTE**.

Sustituye a las llaves ({}) que se usan en Java para delimitar código.

Presta mucha atención a la indentación de los ejemplos o el código no funcionará.

Variables

Definición y uso de variables

Las variables en Python se definen así:

```
x = 2
```

Esto permite usar la variable x después de la definición:

```
x + 1
```

Programa 1.1 – Hola X

Modifica el programa para que pueda saludar a cualquiera usando el siguiente código como ejemplo:

```
name = "Anacleto Skywalker"
```

```
print("Hi " + name)
```

Orden de ejecución

Teniendo en cuenta la siguiente expresión:

¿Cuántas expresiones hay?

¿Qué orden de ejecución tienen?

```
print("Hi " + name)
```

Pedir datos

Programa 1.2 – Pedir nombre

Con la versión anterior, cada vez que queremos cambiar el nombre hay que modificar el archivo main.py.

Con la función input podemos pedir al usuario un valor.

Modifica el programa para pedir el nombre al usuario.

```
name = input("What is your name? ")
```

La cruda realidad sobre los usuarios

¿Qué pasa si el usuario pulsa *ENTER* sin escribir nada?

Para ser robustos, los programas tienen que ser capaces de continuar cuando el usuario hace cosas inesperadas (Cosa que ocurre muchas mas veces de lo que esperáis).

Validar la entrada

Para validar la entrada hacen falta 3 cosas:

1. La **condición** (expresada como una expresión booleana).
2. La **sentencia** que se ejecuta si la condición **se cumple.**
3. La **sentencia** que se ejecuta si la condición **no se cumple.**

```
if condition:
    # What to do if condition is True
else:
    # What to do otherwhise
```

Programa 1.3 – Pedir nombre

Modifica el programa para que muestre un mensaje de error si el nombre está vacío.

¿Cómo codificamos “el nombre está vacío”?

¿Cómo mostramos un “mensaje de error”?

Asignación vs Igualdad

¿Cuántos habéis escrito lo siguiente?

```
if name = "": # (Wrong) This is an assignment
```

Aunque en matemáticas, asignación y igualdad sean lo mismo, en programación no, y hay que saber cuando usarlos.

```
if name == "": # This is a comparison
```

Programa 1.4 – Pedir nombre

Modifica el programa para que además pida la edad del usuario y muestre un mensaje diferente para mayores y menores de edad.

Usa el operador >= para escribir la condición.

Trazas de error

Tras ejecutar el programa, **¿a cuantos os ha salido algo como esto?**

¿Qué significa?

¿Qué parte ha fallado?

¿Por qué?

```
Traceback (most recent call last):
  File "<python-input-1>", line 1, in <module>
    main()
    ~~~~^^
  File "<python-input-0>", line 19, in main
    if age >= 18:
    ^^^^^^^^^
TypeError: '>=' not supported between instances of 'str' and 'int'
```

Esto es una **traza de error** causada por una **excepción no controlada**.

Errores de programación vs de usuario

¿Este error es:

- **un error de programación (nosotros hemos escrito mal código),**
- **o un error del usuario (ha introducido un dato inválido)?**

Conversión de tipos

Para solucionar el error anterior, tenemos que convertir la edad a número.

No es lo mismo la cadena “123”, que el numero 123.

La conversión se hace usando **funciones de conversión:**

```
text = "123"
number = int(text)
```

Programa 1.5 – Pedir nombre

Modifica el programa para que convierta la edad de cadena de caracteres a numero antes de hacer la comparación.

Números negativos

¿Qué pasa si se introduce un numero negativo?

¿Tiene sentido a nivel de tipo de datos?

¿Tiene sentido para el programa?

(EXTRA*) Modifica el programa para mostrar un mensaje de error si la edad es negativa.

* No hagas estos ejercicios ahora, hazlos cuando llegues a la ultima sección.

Código corto ⇏ Buen código

No intentes escribir todo en una sola línea:

```
age = int(input("How old are you? "))
```

Es más legible hace cada cosa de forma separada:

```
age_text = input("How old are you? ")
age = int(age_text)
```

Código legible ⇒ (Potencialmente) Buen código

El código se escribe una vez, pero se lee muchas veces.

Por eso es importante que sea fácil de leer y entender.

Si no, al leer tu propio código pasado, a menudo te preguntarás: ¿qué hace este código?

Trazas de error

¿Qué pasa si introducimos texto no convertible a numero?

¿Qué significa?

¿Qué parte ha fallado?

¿Por qué?

```
>>> age_text = input("How old are you? ")
How old are you? A lot
>>> age = int(age_text)
Traceback (most recent call last):
  File "<python-input-1>", line 1, in <module>
    age = int(age_text)
ValueError: invalid literal for int() with base 10: 'A lot'
```

¿Es un error de programación o del usuario?

Programa 1.6 – Calcular IMC

Modifica el programa para pedir la altura (en metros) y el peso (en kilogramos) del usuario y calcular su IMC (BMI en inglés)

```
bmi = weight / (height * height)
```

Aunque IMC es un acrónimo, en Python, **las variables se escriben siempre en minúsculas**. Las **mayúsculas se reservan para CONSTANTES**.

Números negativos (otra vez)

¿Qué pasa si se introducen números negativos o 0?

¿Tiene sentido a nivel de tipo de datos?

¿Tiene sentido para el programa?

(EXTRA) Modifica el programa para que maneje estos casos y muestre un mensaje adecuado.

Trazas de error

¿A cuantos os ha salido este error?

¿Qué significa?

¿Qué parte ha fallado?

¿Por qué?

```
>>> print("Your BMI is: " + bmi)
Traceback (most recent call last):
  File "<python-input-1>", line 1, in <module>
    print("Your BMI is: " + bmi)
          ~~~~~~~~~~~~~~~^~~~~
TypeError: can only concatenate str (not "float") to str
```

¿Es un error de programación o del usuario?

Arregla este error en el programa

Trazas de error

Responde “1,8” cuando el programa te pida altura

¿Qué significa?

¿Qué ha fallado?

¿Por qué?

```
>>> height_text = input (
...     "How tall are you (in meters)? ")
How tall are you (in meters)? 1,8
>>> height = float(height_text)
Traceback (most recent call last):
  File "<python-input-9>", line 1, in <module>
    height = float(height_text)
ValueError: could not convert string to float: '1,8'
```

¿Es un error de programación o del usuario?

El 99,99% de los lenguajes de programación usan el Inglés como base.

Pedir datos sin conversión automática

Programa 1.7 – ¿Fumas?

Modifica el programa para preguntarle si fuma

```
smoker_text = input("Are you smoker? (True/False) ")
smoker = bool(smoker_text)
if smoker == True:
    print("Smoker")
else:
    print("Non-smoker")
```

¿Que tenemos que escribir para que smoker sea False?

Conversión (por defecto) de str a bool

La conversión por defecto no es muy útil:

- La cadena vacía (“”) se convierte a False
- Todo lo demás se convierte a True

Vamos a hacer otra cosa más útil:

- Solo un valor elegido será convertido a True
- Todo lo demás se convierte a False

Conversión (manual) de str a bool

Modifica el programa para cambiar la conversión

```
smoker_text = input(
    "Write 'Y' if you are a smoker, enter any other value if you are not: "
)
if smoker_text == "Y" or smoker_text == "y":
    smoker = True
else:
    smoker = False

if smoker == True:
    print("Smoker")
else:
    print("Non-smoker")
```

Código válido ⇏ Buen código

Vamos a analizar este código

```
if smoker_text == "Y" or smoker_text == "y":
    smoker = True
else:
    smoker = False
```

¿Que posibles valores puede tener la condición?

Mejorado el código

La condición ya tiene los valores que necesitamos.

No hace falta usar un IF para asignar valores a smoker.

Podemos almacenar el valor de la condición directamente.

```
smoker = smoker_text == "Y" or smoker_text == "y"
```

Código válido ⇏ Buen código

Vamos a analizar este código

```
smoker = smoker_text == "Y" or smoker_text == "y"
if smoker == True:
    print("Smoker")
else:
    print("Non-smoker")
```

¿Que posibles valores puede tener la condición?

Mejorado el código (más todavía)

La variable smoker ya es booleana

Se puede usar directamente como la condición.

```
if smoker:
    print("Smoker")
else:
    print("Non-smoker")
```

Programa 1.8 – ¿Fumas? (corregido)

Modifica el programa para eliminar el código innecesario que acabamos de ver.

Actualización de variables

Programa 1.9 – Pedir pulsaciones

Vamos a pedir al usuario las pulsaciones por minuto en 3 momentos:

1. En reposo
2. Tras la primera actividad
3. Tras la segunda actividad

Copia el código de la siguiente transparencia

Programa 1.9 – Pedir pulsaciones

```
pulse_rest_text = input(
   "How many beats per minute do you have at rest? ") # Input: "60"
pulse_rest = int(pulse_rest_text) # Value: 60

pulse_activity_text = input(
   "How many beats per minute do you have after the first activity? ")
# Input: "80"
pulse_activity = int(pulse_activity_text) # Value: 80

puse_range = pulse_activity - pulse_rest # Value: 20
print(f"Your heart rate range is {puse_range} beats per minute")
# Look at how we use the f before the strinng to allow interpolation

pulse_activity_text = input(
   "How many beats per minute do you have after the second activity? ")
# Input: "90"
pulse_activity = int(pulse_activity_text) # Value: 90
```

Programa 1.9 – Pedir pulsaciones

¿Cuál es el valor de pulse_range al final del programa?

Fíjate en como usamos la F antes del string para poder intercalar variables dentro de ella.

Programa 1.9 – Pedir pulsaciones (corregido)

Las variables no se actualizan automáticamente (como en Excel).

Para actualizar una variable, hay que recalcular el valor.

```
...
pulse_activity_text = input(
    "How many beats per minute do you have after the second activity? ")
# Input: "90"
pulse_activity = int(pulse_activity_text) # Value: 90
puse_range = pulse_activity - pulse_rest # Value: 30
```

Patrones

En ocasiones … veo patrones …

¿Te has fijado que siempre se hacen los mismos pasos para pedir valores al usuario?

1. Pedir un string con input
2. Convertirlo al tipo que necesitamos (si es distinto de string)
3. Comprobar si el valor es válido para la semántica del programa

En ocasiones … veo patrones …

En ocasiones … veo patrones …

Estamos escribiendo mucho código muy parecido.

La estructura es la misma, pero cambian algunos detalles.

De momento no podemos hacer nada, pero en la próxima práctica vamos a aprender a evitarlo.

En ocasiones … veo patrones …

Copiar, pegar y modificar (a.k.a. duplicar) código es una MUY MALA práctica que

- **ha dado muchos problemas en el pasado,**
- **y te dará muchos problemas en el futuro (si no lo evitas)**

¡NO DUPLIQUES CÓDIGO!

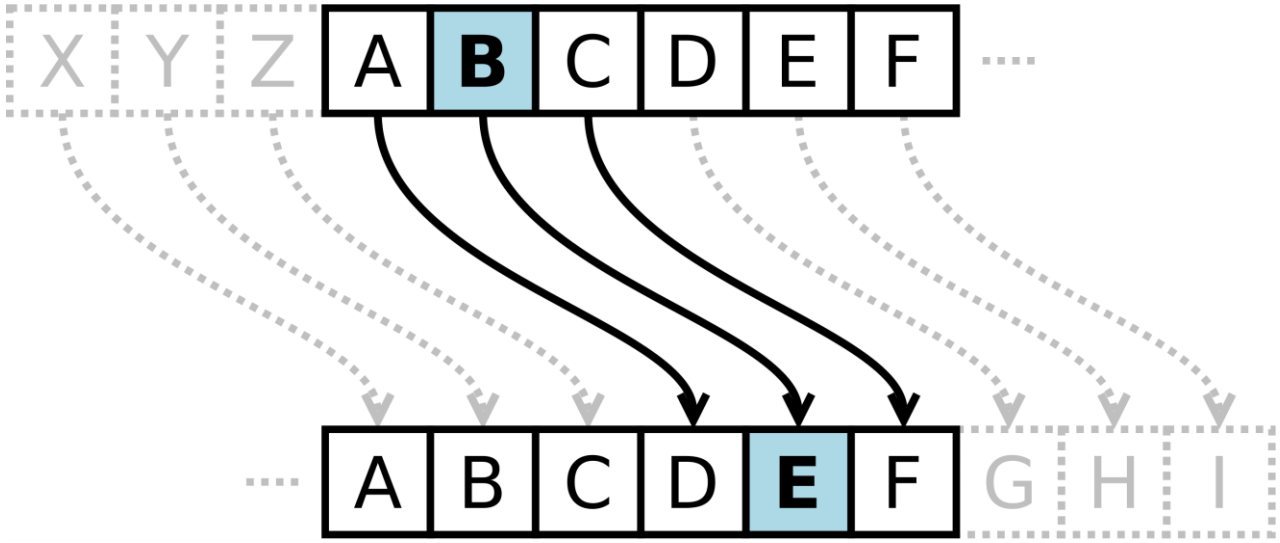
Cifrado del César

Ave César, *los que van a programar te saludan*

El cifrado del César es un método de cifrado en el que cada letra se sustitute por otra letra que se encuentra un número N de posiciones más adelante en el alfabeto.

Con desplazamiento N=3:

- A ⇔ D
- B ⇔ E
- …



Programa 2.0 - Esqueleto

Escribe el esqueleto del programa para que pida:

- un carácter (string de tamaño 1)
- y un desplazamiento,

y los muestre.

¿Cómo combinamos el carácter y el desplazamiento?

Trazas de error

¿Cuántos usasteis int para convertir el carácter a entero?

¿Qué significa?

¿Qué ha fallado?

¿Por qué?

```
>>> char = input("Enter a character (str of size 1): ")
Enter a character (str of size 1): K
>>> code = int(char)
Traceback (most recent call last):
  File "<python-input-1>", line 1, in <module>
    code = int(char)
ValueError: invalid literal for int() with base 10: 'K'
```

¿Es un error de programación o del usuario?

ASCII

Lo que queremos es ser capaces de relacionar un carácter con su código ASCII (los primeros 128 caracteres de Unicode).

```
>>> char = input("Enter a character (str of size 1): ")
Enter a character (str of size 1): A
>>> code = ord(char)          # ord: char -> ASCII code
>>> code
65
>>> new_char = chr(code + 3)   # chr: ASCII code -> char
>>> new_char
'D'
```

Programa 2.1 - Cifrado

Modifica el programa para que haga el cifrado usando las funciones que ya conoces.

¿Que pasa si el desplazamiento es negativo?

¿Tiene sentido?

Programa 2.1 - Descifrado

¿Ahora que sabes como se cifra, como se hace el descifrado?

(EXTRA) Modifica el programa para que descifre el numero que acaba de ser cifrado. Después comprueba que el valor obtenido es el mismo que el original. Es decir:

```
decypher(cypher(text)) == text
```

Trabajo en casa

Partes Extra

Si quieres hacer ejercicios extra, ahora es el momento de hacer las partes marcadas como EXTRA en las transparencias.

Número mayor y menor

Usando las funciones min y max, haz un programa que pida 3 números y muestre:

- El menor
- El mayor
- La diferencia entre ambos

Representación numérica

Usando las funciones bin, hex y len, haz un programa que un entero positivo y muestre:

- Su valor binario
- Su valor hexadecimal
- Cuantos dígitos tiene en base decimal

La cadena mas larga

Usando las funciones max y len, haz un programa que pida 3 cadenas de caracteres y muestre cual tiene mayor tamaño.