Capítulo 1. Introducción a Python.

1.1. Asignación de variables.

Reglas para asignar nombres a variables en Python:

a) Se pueden utilizar letras mayúsculas y minúsculas, números y el guion bajo (_).
b) El nombre de la variable debe comenzar con una letra.
c) El lenguaje distingue entre mayúsculas y minúsculas, tal que las variables "`datos_personales`" y "`datos_Personales`" se consideran distintas, por ejemplo.
d) No se puede emplear espacios en los nombres de las variables.
e) Se recomienda utilizar nombres descriptivos que reflejen el contenido o propósito de la variable.

Además, existen algunos nombres de variables que no se pueden utilizan, pues están reservados para otras funciones. Estos son los siguientes:

```
and       as        assert    async     await     break
class     continue  def       del       elif      else
except    finally   for       from      global    If
import    in        is        lambda    nonlocal  not
or        pass      raise     return    try       While
with      yield     False     True      None
```

1.2. Tipos de datos.

Tipo de dato	*Descripción*	*Ejemplo*
`int`	Números enteros	`1, -1, 10`
`float`	Números en coma flotante	`1.2, 2.0, -3.1`
`str`	Cadena de caracteres	`'Hola', "Python", "1,2,3"`
`bool`	Valores booleanos	`True, False`
`list`	Lista ordenada de elementos	`[1,2,3], ["1","2","3"]`
`tuple`	Lista ordenada de elemento inmutables	`(1,2,3), ("1","2","3")`
`dict`	Diccionario de pares clave-valor	`{"nombre":"Álvaro", "edad":18}`
`set`	Conjunto no ordenado de elementos únicos	`{1,2,3}, {"1","2","3"}`

Es importante tener en cuenta que, al escribir **cadenas de caracteres** se pueden utilizar indistintamente comillas dobles o simples. Además, es conveniente diferenciar correctamente entre listas, tuplas y sets.

Una **lista** y una tupla comparten similitudes, pero presentan diferencias significativas. En una lista, es posible interactuar con los elementos de manera dinámica, permitiendo eliminar o añadir elementos cuando sea necesario, por ejemplo. En cambio, una vez que creamos una tupla esta no puede ser modificada; es inmutable.

Por otro lado, un **set** es un conjunto no ordenado de elementos únicos. Por ejemplo, si agregamos el elemento "1" al set varias veces, solo se almacenará una única instancia en dicho elemento, y no necesariamente en el orden en el que se agregaron.

1.3. Operaciones aritméticas básicas.

Operación	*Descripción*	*Ejemplo*
Suma (+)	Adición de dos valores	`1+1=2`
Resta (-)	Sustracción de dos valores	`1-1=0`
Multiplicación (*)	Multiplicación de dos valores	`2*3=6`
División (/)	División entre dos valores	`10/2=5.0`
División entera (//)	Divide entre dos valores y elimina los decimales	`10//3=3`
Módulo (%)	Devuelve el resto de la división entre dos valores	`10%3=1`
Potencia (**)	Eleva un número a una potencia	`2**3=8`

1.4. Funciones elementales.

A. Funciones integradas en Python de forma nativa

Función	*Descripción*	*Ejemplo*
`int()`	Convierte un valor en entero truncando la parte decimal	`int(3.14)=3`
`float()`	Convierte un valor en un número en coma flotante	`float(3)=3.0`
`abs()`	Devuelve el valor absoluto de un número real o el módulo de un número complejo	`abs(-4)=4` `abs(3+4j)=5.0`
`round()`	Redondea un número al entero más cercano	`round(3.6)=4` `round(2.3)=2`

B. Funciones integradas en el módulo math[1]

Función	*Descripción*	*Ejemplo*
`sqrt()`	Raíz cuadrada de un número	`math.sqrt(16) = 4.0`
`exp()`	Exponencial de un número	`math.exp(2) = 7.389`
`log()`	Logaritmo neperiano de un número	`math.log(10) = 2.30`
`log10()`	Logaritmo en base 10 de un numero	`math.log10(10) = 1`
`sin(), cos(), tan()`	Seno, coseno y tangente de un ángulo en radianes	`math.sin(math.pi/2) = 1`
`asin(), acos(), atan()`	Arcoseno, arcocoseno y acotangente de un valor	`math.asin(0.5) = 0.52`
`pi`	Número pi	`math.pi = 3.14159`

[1] Para acceder al módulo math es necesario importarlo a partir de la instrucción `import math`

1.5. Comentarios

El siguiente aparatado no se encuentra sujeto a evaluación en el examen. Sin embargo, resulta conveniente llevar a cabo una buena praxis y documentar el código con el objetivo de obtener una comprensión más precisa de la tarea que estamos llevando a cabo, dado que es frecuente revisar códigos que hemos creado hace mucho tiempo y no entenderlos dado que no recordamos con exactitud el propósito que estamos prestigiante.

Existen dos formas de escribir comentarios en Python:

a) **Comentarios de una línea:** se utilizan para agregar comentarios breves a partir del símbolo de corchete (#). Estos pueden ir al lado de una variable para añadir una descripción.

b) **Comentarios de varias líneas:** se utilizan para comentarios más extensos utilizando tres comillas simples (‘’’) o tres comillas dobles (“””).

```
# No son nuestras habilidades las que muestran lo que realmente somos, son nuestras elecciones.

'''
"La felicidad se puede encontrar hasta en los momentos más oscuros,
si uno recuerda encender la luz." - Albus Dumbledore, en Harry Potter y el prisionero de Azkaban.
'''

var1 = 10 # vida inicial del personaje

```

1.6. Interacción con el usuario.

Existen dos funciones principalmente que nos permiten interactuar con el usuario a través de la consola: `input()` y `print()`.

La función `input()` se utiliza para recibir datos introducidos por el usuario desde la consola, que podemos almacenar en una variable en forma de string (str). Si queremos recibir como dato un número entero o en coma flotante deberemos convertirlo posteriormente a través de las funciones elementales.

```
nombre = input('Introduzca, a continuación, su nombre: ')       # dtype: str
edad = int(input('Introduzca su edad: '))                       # dtype: int
numero = float(input('Introduce un número aleatorio: '))        # dtype: float
```

La función `print()` se utiliza para mostrar información en la consola. Esta puede recibir uno o varios argumentos separados por coma que imprime como una cadena de texto.

```
edad = 25
print("Tu edad es:", edad)
print("El resultado de la suma es:", 10 + 5)
```

Existen diversos caracteres que nos permiten hacer más “estético” o legible el mensaje que aparece por pantalla. Los más comunes son salto de línea (`\n`) y el tabulado horizontal (`\t`).

```
print("Texto en una\nnueva línea")
print("Texto con\ttabulación horizontal")
```

Un enfoque distinto para imprimir mensajes de texto que contienen diversos valores de variables es utilizar el formato f-string. Este es una forma (a gusto del escritor) más legible de combinar texto y representar resultados. Se utiliza la función `print(f'')`.

```
animal = 'gato'
raza = 'Scottish Fold'
print(f'Mi animal favorito es un {animal}, un {raza}')
```

1.7. Estructuras básicas de un condicional.

Los condicionales se utilizan para tomar decisiones en función del valor de una expresión booleana (verdadero o falso) para ejecutar diferentes bloques de código en función de que condición se cumple.

El condicional se inicializa según la instrucción `'if'`, que puede ser complementada por `'elif'` y `'else'` de forma opcional. La instrucción `'elif'` se ejecuta si la condición anterior no se cumple; mientas que `'else'` se ejecuta si ninguna de las condiciones anteriores se cumple.

```
edad = 20
if edad == 1:
    # Bloque de código que se ejecuta si la edad es igual a 1
    pass
elif edad < 10:
    # Bloque de código que se ejecuta si la condicion1 es falsa (la edad no es igual a 1), pero
    # la edad si es menor que 10.
    pass
else:
    # Bloque de código que se ejecuta si ninguna de las condiciones anteriores es verdadera
    pass
```

Se pueden formular condiciones más complejas utilizando operadores y conectores lógicos.

Operador	***Descripción***	***Ejemplo***
`==`	Comprueba si dos valores son iguales	`var == 1`
`¡=`	Comprueba si dos valores son diferentes	`var ¡= 1`
`>` o `<`	Comprueba si un valor es mayor o menor que otro	`var < 1` `var > 1`
`>=` o `<=`	Comprueba si un valor es mayor o menor que otro	`var <= 1` `var >= 1`
Conectores lógicos	***Descripción***	***Ejemplo***
`and`	Comprueba si ambas condiciones son verdaderas	`var>1 and var<1`
`or`	Comprueba si al menos una condición es verdadera	`var=0 or var>1`
`not`	Comprueba si la afirmación no es verdadera	`not var > 5`

1.8. Bucle FOR.

El bucle `for` se utiliza para ejecutar un bloque de código varias veces iterando sobre una secuencia de elementos (una lista, una tupla, una cadena de texto o un rango).

```
for i in [0, 1, 2, 3, 4]:              # lista
    print(i)

for i in (0, 1, 2, 3, 4):              # tupla
    print(i)

for i in 'estar':                      # cadena de texto
    print(i)

for i in range(5):                     # rango
    print(i)
```

La función `range()` genera una secuencia de números de forma ordenada respetando la siguiente sintaxis: `range(inicio, fin, paso)`, donde el paso es el incremento entre los elementos de la secuencia.

Por ejemplo, `range(1, 5, 2)` generará la sucesión 1,3. Si no se especifica el inicio y el paso el valor predeterminado es 0 y 1, respectivamente.

1.9. Bucle WHILE.

El bucle `while` permite ejecutar un bloque de código siempre que se verifique una condición. Una vez iniciado el bloque, si la condición sigue siendo verdadera, el bloque se ejecuta nuevamente.

```
num = 1
while num < 10:
    num = num + 1

print(num)
```

En el escenario anterior, se incrementa la variable `'num'` una unidad, siempre y cuando `'num'` sea menor que 10. Sin embargo, una vez que `'num'` alcanza el valor de 10, la condición evaluada deja de ser verdadera, lo que da lugar a la finalización del bucle.

1.10. Listas.

Las listas son estructuras de datos que nos permiten almacenar y organizar múltiples elementos en un solo objeto. Se trata de una secuencia ordenada y mutable, lo que significa que podemos realizar diversas operaciones para añadir elementos, eliminarlos, etc. Para crear una lista en Python se utilizan los corchetes '[]' y se separan los elementos por comas. Por ejemplo:

```
lista1 = [1, 2, '3', 'Marcos', 5.0]
```

Podemos acceder a los elementos de una lista indicando su índice. Debemos tener en cuenta que el elemento inicial de una lista tiene índice 0. Así pues,

```
>>> lista1[0]
1
>>> lista1[3]
'Marcos'
```

También, podemos visualizar una porción de una lista utilizando la sintaxis de los 'slicing', tal y como se muestra a continuación:

```
>>> # lista[inicio:fin:paso]
...
... lista = [1, 2, 3, 4, 5]
>>> lista[1:3]
[2, 3]
```

Algunas operaciones utilizadas en listas son

- `'append(elemento)'`: agrega un elemento al final de la lista.
- `'insert(índice, elemento)'`: agrega un elemento en una posición particular de la lista.
- `'remove(elemento)'`: elimina un elemento de una lista la primera vez que aparece. Por lo tanto, si un elemento aparece varias veces solo lo eliminará la primera.
- `'pop(indice)'`: elimina el elemento que se encuentra en la posición indicada.
- `'reverse()'`: invierte el orden de los elementos de la lista.
- `'clear()'`: elimina todos los elementos que contiene una lista
- `'index(elemento)'`: devuelve la primera posición en la que aparece el elemento seleccionado.
- `'len(lista)'`: devuelve la longitud de la lista (determinada por el número de elementos).

```
lista1.append('hello')
lista1.insert(1,'hello')
lista1.remove(1)
lista1.pop(2)
lista1.reverse()
lista1.clear()
lista1.index('Marcos')
len(lista1)
```

ACTIVIDADES SEMANA 2

1. Del siguiente listado de nombres indicar que nombres son válidos y cuáles no.

- `var`
- `var2`
- `mi_variable`
- `otra variable`
- `concentración`
- `2conc`
- `conc_ion`
- `conc-ion`
- `ConcEstaño`

2. Identificar el tipo de variable tras cada asignación.

```
var = 34
var = 3.4
var = 34 + 2.3
var = 8/3
var = 8/2
var = 0.3 + 0.6 + 0.1
var = 3/3
var = 3 * 1/3
var = 0.5 + 0.5
```

3. De la siguiente lista de operaciones (donde previamente se ha asignado a = 0.5, b = 0.3 y c = 0.1) identificar que operaciones son correctas y cuales no:

```
a + b
a * b * c ** 2
a + + b
(a + b) (a - b)
a / b - c
a** -2
-a * b
a ** 2 / b
```

4. Se quiere implementar la siguiente operación en Python

$$\frac{(a+b)(1-2b)+a^2}{(b+a)(b^2-1)}+\frac{1}{b-a}$$

Identificar cuál de las siguientes operaciones es la implementación correcta

(a + b)(1 - 2b) + a**2 / (b + a) / (b**2 -1) + 1/(b-a)

((a + b)(1 - 2*b) + a**2) / (b + a) / (b**2 -1) + 1/(b-a)

(a + b) * (1 - 2*b) + a**2 / (b + a) / (b**2 -1) + 1/(b-a)

(a + b) * (1 - 2*b) + a**2) / (b + a) / (b**2 -1) + 1/(b-a)

(a + b)(1 - 2b) + a**2 / ((b + a) * (b**2 -1)) + 1/(b-a)

((a + b)(1 - 2b) + a**2) / (b + a) / (b**2 -1) + 1/(b-a)

5. Habiendo asignado x = 2.3 calcular los siguientes valores mediante las correspondientes funciones de Python

a) $3x^3 + 5x^2 - 6x + 2$
b) $\frac{3x^3+5x^2-6x+2}{x^2+6x+1}$
c) $sen(4x)$
d) $\sqrt{2x^2 + 1}$
e) e^{x+3}
f) $\ln\left(x^2 + \frac{2}{3}\right)$
g) $\log(5x + 2)$
h) $senh(1 + cos(x^2 - 3))$

```
import cmath as math
x = 2.3

a = 3*x**3 + 5*x**2 -6*x +2

b = ( 3*x**3 + 5*x**2 -6*x +2 ) / (x**2 + 6*x + 1)

c = math.sin(4*x)

d = math.sqrt(2*(x**2) +1)

e = math.e(x+3)

f = math.log(x**2 + 2/3)

g = math.log10(5*x + 2)

h = math.sinh( 1 + math.cos(x**2 - 3))
```

6. Calcular en Python el valor de la $\sqrt{x}$ (tomando x = 6.2) de tres formas distintas:

- Empleando la función intrínseca para la raíz cuadrada.
- Empleando potencias.
- Empleando las funciones intrínsecas para la exponencial y el logaritmo.

```
'''
Para descomponer un número en centenas, decenas y unidades podemos dividirlo entre 10 y truncar
la parte decimal.
'''

num = 321

centenas = num/100
centenas = int(centenas)            # Truncamos la parte decimal

decenas = num /10
decenas = int(decenas)              # Truncamos la parte decimal
decenas = decenas - centenas*10

unidad = num - centenas*100 - decenas*10
```

7. Dado un número natural de 3 cifras (almacenado en una variable num) obtener mediante operaciones de Python descomposición en centenas, decenas y unidades, almacenando los correspondientes valores en las variables centenas, decenas y unidades. Hacer la descomposición para los casos num = 321, num 701 y num = 140.

```
x = 6.2

# a)  FORMA 1: Empleando la función intrínseca (del módulo math)

import cmath as math

solucion1 = math.sqrt(x)

# b)  FORMA 2: Empleando potencias

solución2 = x**(1/2)

# c)  FORMA 3: Empleando las funciones intrínsecas y el logaaritmo

'''
Se tiene que y = x**1/2 --> ln(y) = ln(x)/2 --> y = e**(ln(x)/2)
'''

z = math.log(x)/2           # Recordar que math.log hace referencia al log. natural
z = abs(z)                  # Nos quedamos solo con la parte real del valor obtenido

solucion3 = math.e(z)
```

8. Se hace la asignación num = 26 para almacenar en la variable num la expresión decimal para la cual se desea obtener su expresión binaria.
   Se sabe que con m dígitos binarios se pueden representar todos los números naturales comprendidos entre 0 y $2^m - 1$ (ambos incluidos).
   Se pide entonces:
   a) Comprobar que el número cuya expresión decimal se almacena en la variable num se puede representar con 5 dígitos binarios.
   b) Considerando que la expresión binaria de este número es $(b4,b3,b2,b1,b0)_2$ calcular los 5 dígitos binarios y guardarlos en las variables b0, b1, b2, b3 y b4
   c) Comprobar (transformando ahora de binario a decimal) que la descomposición calculada en (b) es correcta.

```
# a) Convertir un número a binario.

num = 26

binario = []
divisor = 2
cociente = 0                # Asignamos un valor inicial cualquiera para que se inicie el bucle
while cociente != 1:
    cociente = int( num/divisor )      # Truncamos el cociente
    residuo = num - cociente*divisor
    binario.append(residuo)

# b) Guardar los 5 dígitos binarios en las variables b.

b0 = binario[0]
b1 = binario[1]
b2 = binario[2]
b3 = binario[3]
b4 = binario[4]

# c) Convertir el número binario obtenido en decimal nuevamente.

num = b0*2**4 + b1*2**3 + b2*2**2 + b3*2**1 + b4*2**0
```

7. Dado un número natural de 3 cifras (almacenado en una variable num) obtener mediante operaciones de Python descomposición en centenas, decenas y unidades, almacenando los correspondientes valores en las variables centenas, decenas y unidades. Hacer la descomposición para los casos num = 321, num 701 y num = 140.

```
x = 6.2

# a)  FORMA 1: Empleando la función intrínseca (del módulo math)

import cmath as math

solucion1 = math.sqrt(x)

# b)  FORMA 2: Empleando potencias

solución2 = x**(1/2)

# c)  FORMA 3: Empleando las funciones intrínsecas y el logoaritmo

'''
Se tiene que y = x**1/2 --> ln(y) = ln(x)/2 --> y = e**(ln(x)/2)
'''

z = math.log(x)/2           # Recordar que math.log hace referencia al log. natural
z = abs(z)                  # Nos quedamos solo con la parte real del valor obtenido

solucion3 = math.e(z)
```

8. Se hace la asignación num = 26 para almacenar en la variable num la expresión decimal para la cual se desea obtener su expresión binaria.
Se sabe que con m dígitos binarios se pueden representar todos los números naturales comprendidos entre 0 y $2^m - 1$ (ambos incluidos).
Se pide entonces:
   a) Comprobar que el número cuya expresión decimal se almacena en la variable num se puede representar con 5 dígitos binarios.
   b) Considerando que la expresión binaria de este número es $(b4,b3,b2,b1,b0)_2$ calcular los 5 dígitos binarios y guardarlos en las variables b0, b1, b2, b3 y b4
   c) Comprobar (transformando ahora de binario a decimal) que la descomposición calculada en (b) es correcta.

```
# a) Convertir un número a binario.

num = 26

binario = []
divisor = 2
cociente = 0                # Asignamos un valor inicial cualquiera para que se inicie el bucle
while cociente != 1:
    cociente = int( num/divisor )      # Truncamos el cociente
    residuo = num - cociente*divisor
    binario.append(residuo)

# b) Guardar los 5 dígitos binarios en las variables b.

b0 = binario[0]
b1 = binario[1]
b2 = binario[2]
b3 = binario[3]
b4 = binario[4]

# c) Convertir el número binario obtenido en decimal nuevamente.

num = b0*2**4 + b1*2**3 + b2*2**2 + b3*2**1 + b4*2**0
```

ACTIVIDADES SEMANA 3

1. Escribir un programa en Python que pida por teclado, primero, el nombre y primer apellido del estudiante (separado por un espacio) y, después, su grupo para mostrar por pantalla un mensaje de la forma: El estudiante ... pertenece al grupo ... del Grado de Ing. Aeroespacial.

```
# EJERCICIO 1: Escribir un programa en Python que pida por teclado, primero,
# el nombre y primer apellido del estudiante (separado por un espacio) y, después,
# su grupo para mostrar por pantalla un mensaje de la forma:
# El estudiante … pertenece al grupo … del Grado de Ing. Aeroespacial.

nombre_appellido = input('Introduce tu nombre y appellido: ')
grupo = input('Introduce el grupo al que perteneces:')

print(f'\nEl estudiante {nombre_appellido} pertenece al grupo {grupo}')
```

2. Escribir un programa de Python que pida por teclado el nombre, primer apellido y grupo del estudiante (las tres cadenas en la misma fila y separadas por espacios en blanco) para mostrar por pantalla el mismo mensaje que en el ejercicio anterior.

```
datos = input('Introduce tu nombre, primer apellido y grupo')

datos_list = datos.split()  # Crea una lista separando los caracteres de un str
                            # a través de los espacios si no se especifica un separador

print(f'\nEl estudiante {datos_list[0]} {datos_list[1]} pertenece al grupo {datos_list[2]}')
```

3. Escribir un programa de Python que pida por teclado el nombre y primer apellido de un estudiante y los escriba por pantalla de modo que, independientemente de cómo se hayan tecleado la primera letra del nombre y el apellido aparezcan en mayúsculas y el resto en minúsculas.

```
nombre_appellido = input('Introduce tu nombre y appellido: ')
grupo = input('Introduce el grupo al que perteneces:')

datos = nombre_appellido.split()
datos.append(grupo)

for i in range(len(datos)):
    datos[i] = (datos[i].lower()).capitalize()
```

4. Escribir una expresión de Python que cuya evaluación muestre por pantalla True o False dependiendo de si es cierta o falsa, respectivamente, la siguiente relación de pertenecía.

```
import math

x = math.sqrt(13)

if 3.5 <= x  and x <= 3.7:
    print('True')
else:
    print('False')
```

5. Predecir el resultado (True o False) de las siguientes expresiones de Python:

- `(6 > 2) and (1 < 1)`
- `not (5 < 10)`
- `5.0 < 1 or 2 > - 1.0 or 3 > 5`
- `5.0 < 1 or 2 > -1.0 and not 3 > 5`
- `5.0 < 1 and 2 > -1.0 and not 3 > 5`
- `5.0 < 1 and 2 > -1.0 or not 3 > 5`
- `Math.sqrt(3.2 + 3.91**2 - 3 - 0.2) == 3.91`

6. Completar el código en este programa para intercambiar los contenidos de las variables a y b de modo que tras su ejecución aparezca en pantalla a = 5.1 y b = 3.4.

```
a = 3.4
b = 5.1

# COMPLETAR EL CÓDIGO

print('a = ',a)
print('b = ',b)
```

```
a = 3.4
b = 5.1

c = b       # c guarda el valor en b para que
b = a       # b adquiera el valor de a y,
a = c       # finalmente a toma el valor de b
            # almacenado previamente en c

print('a = ',a)
print('b = ',b)
```

7. Rehacer el ejercicio 6 empleando una sola línea de código.

```
a = 3.4
b = 5.1

a, b = b, a

print('a = ',a)
print('b = ',b)
```

8. Escribir un programa de Python que pida por teclado los elementos de una matriz A 4x4. Seguidamente, calcula el determinante de la matriz A y muéstralo por pantalla.

```
A = []
for i in range(4):
    a = float(input(f'Introduce el elemento {i} de la matriz A: '))
    A.append(a)

det = A[0]*A[3] - A[1]*A[2]
```

9. Escribir un programa de Python que pida (por teclado) la masa de un planeta (en kg) y su diámetro (en km), calcule la velocidad de escape de dicho planeta y la muestre (en km/s) por pantalla. Validar este programa calculando la velocidad de escape de varios planetas y comparándolas con resultados encontrados en diversas referencias

```
import math

mass = input('Introduce la masa del planeta: ')
diameter = input('Introduce el diámetro del planeta: ')

radio = diameter/2
G = 6.674e-11

v_esc = math.sqrt(2*G*mass/radio)
```

ACTIVIDADES SEMANA 4

1. Escribir un programa que pida convertir entre representaciones de ángulos de radianes y en grados. El programa deberá:

- Pedir por teclado un número entero n
- Si el número es igual a 0, el programa pide la representación en radianes (como una variable real), la convierte en grados y la muestra por pantalla (con un mensaje explicativo).
- Si el número es distinto de 0, el programa pide la representación en grados (como una variable real), la convierte en radianes y la muestra por pantalla (con un mensaje explicativo).

```
import math

key = input('Introduce 0 si decias convertir un número de radianes a grados u otro número si
num = input('Introduce el valor en radianes: ')

if key == 0:
    num = input('Introduce el valor en radianes: ')
    sol = num * ( 180 / (2*math.pi) )
else:
    num = input('Introduce el valor en radianes: ')
    sol = num * ( (2*math.pi) / 180 )
```

2. Escribir un programa que pida convertir entre escalas de temperatura: Celsius, Kelvin y Fahrenheit. Considérese las siguientes relaciones:

$$\mathrm{Tc} = \mathrm{Tk} - 273.15, \qquad \mathrm{Tf} = \frac{9}{5} * \mathrm{Tk} - 459.67$$

```
flag = input('''
· Introduce 1 si desea convertir la temperatura de Celsius a Kelvin y Fahrenheit.
· Introduce 2 si desea convertir la temperatura de Kelvin a Celsius y Fahrenheit.
· Introduce 3 si desea convertir la temperatura de Fahrenheit a Celsius y Kelvin.\n
''')

if flag == 1:
    C = input('Introduce la temperatura inicial en Celsius: ')
    K = C + 273.15
    F = 9*K/5 - 459.67
elif flag == 2:
    K = input('Introduce la temperatura inicial en Kelvin: ')
    C = K - 273.15
    F = 9*K/5 - 459.67
elif flag == 3:
    F = input('Introduce la temperatura inicial en Fahrenheit: ')
    K = 5*(F + 459.67)/9
else:
    print('No has seleccionado un número entre las tres opciones anteriores')
```

3. Escribir un programa que pida un número natural inferior a 1000. Si el número introducido no cumple ese requisito el programa terminará con este mensaje por pantalla El número no es correcto. Si se cumple la condición el programa devolverá la descomposición del número en centenas, decenas y unidades

```
num = float(input('Introduce un número menor que 1000: '))

if num < 1000 and num >= 0:
    C = int(num/100)                # centenas
    D = int(num/10) - C*10          # decenas
    U = int(num) - C*100 - D*10     # unidad
else:
    print('El número no es correcto')
```

4. Escribir un programa que pida (por teclado) una contraseña y compruebe que es válida al cumplir los siguientes requisitos:

- Tiene entre 6 y 16 caracteres (ambos valores incluidos)
- Incluye algún número

El programa deberá escribir por pantalla, según corresponda la clave es válida o la clave no es válida.

```
password = input('Introduce una clave de seguridad: ')
frag = False

if 6 <= len(password) and len(password) <= 16:
    for i in password:
        if i.isdigit():
            flag = True
            break

if flag == True:
    print('La contraseña es válida')
else:
    print('La contraseña no es válida')
```

5. Escribir un programa que pida por teclado el nombre de un mes (en mayúsculas) y el año y devuelva, por pantalla, el número de días de ese mes.

```
month = input('Introduce el nombre del més: ').upper()
year = input('Introduce el año en el que desea obtener los días del més: ')

month_31_days = ['ENERO','MARZO','MAYO', 'JULIO', 'AGOSTO', 'OCTUBRE', 'DICIEMBRE']

if month in month_31_days:
    print(f'En {month} hay 31 días en {year}.')
elif month == 'FEBRERO':
    # Verificamos si el año es bisiesto
    if (year % 4 == 0 and year % 100 != 0) or year % 400 == 0:
        print("El mes de FEBRERO en el año", year, "tiene 29 días (año bisiesto).")
    else:
        print("El mes de FEBRERO en el año", year, "tiene 28 días (no es un año bisiesto).")
else:
    print(f'En {month} hay 30 días en {year}.')
```

6. Una universidad establece en una de sus titulaciones las siguientes calificaciones mínimas en función del tipo de acceso:

- Pruebas de acceso a universidad o equivalentes: 12.223.
- Titulados o equivalentes: 6.660.
- Bachillerato homologado: no ser ofertan plazas.

Escribir un programa que pida al usuario el tipo de acceso y su calificación, mostrando por pantalla si cumple o no los requisitos fijados para acceder a la titulación.

```
flag = input('''
Introduce el número correspondiente a la vía por la que accede a la titulación:
1. Pruebas de acceso a universidad o equivalente.
2. Titulados o equivalentes.
3. Bachillerato homologado.
''')

if flag == 1:
    nota = int('Introduce la calificación obtenida: ')
    if nota >= 12.223:
        print('Cumples los requisitos.')
    else:
        print('No cumples los requisitos.')
elif flag == 2:
    nota = int('Introduce la calificación obtenida: ')
    if nota >= 6.660:
        print('Cumples los requisitos.')
    else:
        print('No cumples los requisitos.')
elif flag == 3:
    print('Lo sentimos. No estamos ofertando plazas para esta vía de acceso.')
else:
    print('No ha introducido correctamente el número que indica su vía de acceso.')
```

7. A continuación se reproduce la tabla de tributación del impuesto de la renta de las personas físicas (IRPF) en España para 2022.

Base imponible desde	(en euros) hasta	tipo aplicable
0	12450	19%
12450	20200	24%
20200	35200	30%
35200	60000	37%
60000	300000	45%
Más de 300.000		47%

Escribir un programa que solicite la base imponible y devuelva la cuota correspondiente del impuesto.

```python
base = float(input('Introduce tu base impositiva: '))

if 0 <= base < 12450:
    print('La cuota es del 19%')
elif 12450 <= base < 20200:
    print('La cuota es del 24%')
elif 20200 <= base < 35200:
    print('La cuota es del 30%')
elif 35200 <= base < 60000:
    print('La cuota es del 37%')
elif 60000 <= base < 300000:
    print('La cuota es del 45%')
else:
    print('La cuota es del 47%')
```

8. Escribir un programa que pida dos números naturales y devuelva un mensaje indicando si alguno de ellos es divisor del otro.

```python
numero1 = int(input("Ingrese el primer número natural: "))
numero2 = int(input("Ingrese el segundo número natural: "))

# Verificar si el primer número es divisor del segundo
if numero2 % numero1 == 0:
    print(numero1, "es divisor de", numero2)

# Verificar si el segundo número es divisor del primero
if numero1 % numero2 == 0:
    print(numero2, "es divisor de", numero1)

# Si ninguno de los números es divisor del otro, mostrar un mensaje indicando esto
if numero2 % numero1 != 0 and numero1 % numero2 != 0:
    print("Ninguno de los números es divisor del otro.")
```

9. Se considera la función f(x) definida del modo siguiente:

$$f(x) = \begin{cases} 1 \quad si\ x < 0 \\ \log(1+x^3)\cos(x)\ si\ 0 \le x \le \dfrac{\pi}{2} \\ \dfrac{x-5}{1+x^2}\ si\ x > \dfrac{\pi}{2} \end{cases}$$

```python
# EJERCICIO 9:
import math
x = input('Introduce un valor para evaluar: ')

if x < 0:
    fx = math.e(-x)*math.sin(x**2)
elif 0 <= x <= math.pi/2:
    fx = math.log10(1 + x**3)*math.cos(x)
else:
    fx = (x - 5) / (1 + x**2)

print(x)
```

10. Escribir un programa que pida dos números (reales) por teclado y muestre el máximo y el mínimo de estos números.

```
201  # EJERCICIO 10:
202
203  numero1 = float(input('Introduce el número 1: '))
204  numero2 = float(input('Introduce el número 2: '))
205
206  list = [numero1, numero2]
207  MAX = max(list)
208  MIN = min(list)
```

ACTIVIDADES SEMANA 5

1. Escribir un programa que pida un número natural y muestre por pantalla si se trata de un número primo o no. Para detectar si se trata o no de un número primo se deberá usar un bucle.

```
num = int(input("Ingrese un número natural: "))

if num < 2:
    print("No es un número primo")
else:
    flag = True
    for i in range(2, num):
        if num % i == 0:
            flag = False
            break

    if flag:
        print("Es un número primo")
    else:
        print("No es un número primo")
```

2. Escribir un programa que pida un número natural y muestre por pantalla el número de dígitos que contiene.

```
num = int(input('Introduce un número natural: '))

digitos = list(str(num))

print(f'El número {num} tiene {len(digitos)} dígitos')
```

3. Escribir un programa que pida un número natural y muestre por pantalla el primer y último dígito de ese número.

```
num = int(input('Introduce un número natural: '))

digitos = list(str(num))
end_index = len(digitos) - 1

print(f'El primer dígito es {digitos[0]} y el último es {digitos[end_index]}')
```

4. Escribir un programa que pida 10 números reales y muestre por pantalla la suma de todos ellos.

```
num = []
for i in range(11):
    x = float(input('Introduce un número: '))
    num.append(x)

suma = sum(num)
```

5. Escribir un programa que pida 10 números naturales y muestre por pantalla cuántos números son pares y cuántos impares.

```
num = []
for i in range(11):
    x = int(input('Introduce un número: '))     # Solo acepta números enteros
    num.append(x)

# Discriminamos entre si es un número par o impar:
par = []
impar = []
for j in num:
    if j % 2 == 0:
        par.append(j)
    elif j % 2 == 1:
        impar.append(j)
```

6. Escribir un programa que pida un número natural y devuelva el valor del factorial de dicho número.

```
import math
num = int(input('Introduce un número natural: '))

sol1 = math.factorial(num)         # forma 1

def factorial(n):            # forma 2
    resultado = 1
    for i in range(1, n + 1):
        resultado *= i
    return resultado

sol2 = factorial(num)
```

7. Se sabe que la sucesión $\{a_n\}_{n=1}^{inf}$ definida mediante $a_n = \left(1 + \frac{1}{n}\right)^n$ tiene como límite el número e. Se quiere escribir un programa que permita aproximar este número devolviendo el término a_m que corresponda al primer término de la sucesión (a_2, a_3, a_4,…) que verifica la siguiente propiedad $|a_m - a_{m-1}| < 10^{-3}$ mostrando también el valor de m y el error $|e - a_m|$

```
n = 1

a = ( 1 + 1/n )**n
a_ans = 0

while ( a - a_ans ) > 10e-3:
    n += 1
    a_ans = a
    a = (1 + 1 / n) ** n

print('El valor de a es: ',a)
```

8. Escribir un programa para calcular la suma parcial de los m primeros términos de la serie geométrica: a + ar + ar^2 + ar^3 + … . Los valores de m, a y r deberán proporcionarse por teclado.

```
m = int(input('Introduce el grado del polinómio: '))
a = float(input('Introduce el valor del coeficiente: '))
r = float(input('Introduce el valor de la incógnita: '))

suma = []
for i in range(m+1):
    s = a*(r**i)
    suma.append(s)

sol = sum(suma)
```

9. Se considera una sucesión de números reales generada mediante la siguiente fórmula de recurrencia: $\boldsymbol{a_{n+1} = \frac{1}{2}(a_n + \frac{2}{a_n})}$, con $a_0 > 0$. Escribir un programa que genere los m primeros términos de esta sucesión pidiendo el término inicial a_0 y el número m.

```
m = int(input('Introduce el número de términos que desea obtener de la sucesión: '))
a = float(input('Introduce el valor de a: '))

sol = []
for i in range(m):
    sol.append(a)
    a = (a + 2/a)/2

print(sol)
```

10. Escribir un programa que genere los m primeros términos de la sucesión de Fibonacci pidiendo el valor de m por teclado.
Nota: los m primeros términos son 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …. Además, dicha sucesión está generada por las formulas:

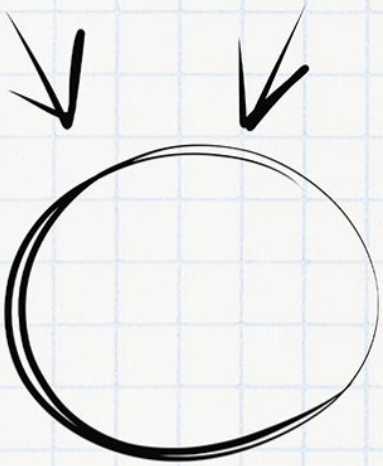
```
# EJERCICIO 10:

m = int(input('Introduce el número de elementos que desea visualizar de la serie de Fibo
l = []

for i in range(0,m):
    if i == 0 or i == 1:
        l.append(i)
        print(i)
    else:
        a = l[i-2] + l[i-1]
        l.append(a)
        print(a)
```

Imagínate aprobando el examen

Necesitas tiempo y concentración

Planes	PLAN TURBO	PLAN PRO	PLAN PRO+
Descargas sin publi al mes	10	40	80
Elimina el video entre descargas	✓	✓	✓
Descarga carpetas	✗	✓	✓
Descarga archivos grandes	✗	✓	✓
Visualiza apuntes online sin publi	✗	✓	✓
Elimina toda la publi web	✗	✗	✓
Precios Anual	0,99 € / mes	3,99 € / mes	7,99 € / mes

Ahora que puedes conseguirlo, ¿Qué nota vas a sacar?



WUOLAH