

**vuela**

andaluciavuela.es

CONECTA CON ANDALUCÍA

**PLATAFORMA  
DIGITAL DE  
ANDALUCÍA**



Junta de Andalucía

# EXPERTO EN PYTHON. CASO PRÁCTICO.

# 1 Descripción del caso práctico.

En estas cinco sesiones de clases en vivo vamos a desarrollar distintos casos prácticos, consistentes en la construcción de programas en Python que permitan abordar en detalle los diferentes conceptos estudiados en los apuntes y en las diapositivas: En particular, se tratarán conjuntamente casos de:

- Programación básica en Python
- Biblioteca NumPy
- Biblioteca Pandas
- Visualización de datos

Con estos cinco puntos, tendremos la base para abordar los siguientes módulos del curso con éxito.

## 2 Ejercicios propuestos

- Ejercicio 1: Haciendo uso de sentencias condicionales, elabore un programa Python que pida por teclado 3 números y devuelva el máximo y el mínimo de ellos por pantalla..
- Ejercicio 2: Escribir un programa que lea dos enteros por pantalla y, en caso de que uno divida a otro (resto de dividir igual a 0), que indique cuál divide a cuál.
- Ejercicio 3: Realice un programa que muestre todos los términos **pares** de la siguiente sucesión, comenzando desde  $a_0$  hasta  $a_{1000}$  inclusive.

$$a_{ii} = \frac{(-1)^{ii}(ii^2 - 1)}{2ii}$$

- Ejercicio 4: Escribir un programa que lea números positivos por teclado hasta que o bien se introduzcan 10 números, o bien se

introduzca un 0. Se deberá mostrar la media de todos los números positivos introducidos.

- | Ejercicio 5: Escribir un programa que lea un entero positivo por teclado e indique si el número es primo o no.
- | Ejercicio 6: Escribir una función que tenga como entrada por parámetro un entero positivo y devuelva con return si es primo o no.
- | Ejercicio 7: Escriba una función que tenga como entrada dos números mínimo y máximo, y pida al usuario introducir un valor en ese rango. Deberá devolver el valor introducido (previa comprobación de que es válido).
- | Ejercicio 8: Escribir una función que tenga como entrada 3 valores y devuelva con return su media y varianza.
- | Ejercicio 9: Escribir una función que tenga como entrada por parámetro dos enteros positivos  $n, m$  y devuelva con return una lista de listas conteniendo una matriz de esa dimensión cuyas celdas son pedidas por teclado.
- | Ejercicio 10: Escriba una función que tenga como entrada dos listas de listas de tamaño  $n$  filas y  $m$  columnas y devuelva una matriz (lista de listas) con la suma de ambas, o None si no es posible.
- | Ejercicio 11: Escribir una función que tenga como entrada 2 matrices (listas de listas) y devuelva su multiplicación, si es posible, o None en otro caso.
- | Ejercicio 12: Escribir una función que tenga como entrada por parámetro dos números y que devuelva su multiplicación y su división (None si no es posible), como una tupla.
- | Ejercicio 13: Escriba una función que tenga como entrada una lista y devuelva una tupla con sus valores mínimo, máximo, media, varianza, desviación típica.

Ejercicio 14: Escribir un programa que cree un diccionario con la siguiente tabla de datos, donde la clave será la columna **caso**. Posteriormente, itere sobre el diccionario para mostrar las filas con clave par.

Caso	Tasación	Valoración	Entradas	Balance	Aprobar
1	+	-	No	+	No
2	-	-	Sí	-	No
3	+	+	No	+	Sí
4	-	+	Sí	+	Sí
5	-	+	Sí	-	No
6	+	+	Sí	-	Sí
7	+	+	Sí	+	Sí
8	+	-	Sí	-	No
9	+	+	No	-	No

Ejercicio 15: Escribir un programa que cree un diccionario para traducir letras del alfabeto a código Morse. Realice un programa que lea una cadena en mayúsculas desde teclado y muestre su correspondiente traducción en Morse.

CÓDIGO MORSE

A	.-	J	.-.-.-	S	...
B	-...	K	-.-	T	-
C	-.-.-	L	.-...	U	...-
D	-..	M	--	V	...-
E	.	N	-.	W	.-.-
F	...-	O	---	X	...-
G	--.	P	.-.-.	Y	.-.-.-
H	....	Q	-.-.-	Z	-....
I	..	R	.-.		

- Ejercicio 16: Escribir un fichero con diferentes funciones de operaciones con matrices (suma, resta, multiplicación). Llamar a este fichero **matrices.py**. Escribir un programa en otro fichero, que haga uso de las funciones escritas en **matrices.py**.
- Ejercicio 17: Escribir un fichero **ClaseMatriz.py**, que sirva para representar matrices con un número de filas y columnas dado por argumento al constructor. Por defecto, la matriz tendrá todas las celdas a 0. Deberá tener métodos para asignar valores a celdas, obtener valores de celdas, sumar, restar y multiplicar matrices. También sobrecarga de conversión a cadena (**str**).
- Ejercicio 18: Escribir un fichero **Vehiculos.py**, que sirva para representar una clase **Vehículo** con atributos **color** y **número de ruedas**. Herede las clases **Coche** (**cilindrada**, **número de puertas**) y **Bicicleta** (**tipo -montaña**, **ciudad...**). De bicicleta, herede también **BicicletaElectrica** (**potencia**, **autonomía**). Implemente métodos **mover** para todos, **abrir/cerrar** para coche, **BateriaRestante** para la bicicleta eléctrica.
- Ejercicio 19: Usando **np.arange** y **np.reshape**, genere una matriz de 3x3 con números del 0 al 8.

- | Ejercicio 20: Usando `np.eye`, genere una matriz identidad de 10x10
- | Ejercicio 21: Cree un array conteniendo 11 valores uniformemente espaciados entre los valores 0 y 100 (inclusive) usando `np.linspace`.
- | Ejercicio 22: Genere 10000 pares de números entre -1 y 1, mediante una distribución uniforme  $U(-1, 1)$ . Calcule la norma de dichos números usando `np.linalg.norm`. Descarte los de norma  $> 1$ . Aproxime  $\pi$  como 4 por el número de puntos restante dividido por 10000 (Ejercicio de Monte-Carlo del módulo 1).
- | Ejercicio 23: Genere un array de 100 elementos mayores que 1000 con `arange`. Utilice `random.permutation` para desordenar el array.
- | Ejercicio 24: Genere un array de 100000 valores con `random.randn`. Calcule la media y la desviación típica de los elementos generados.
- | Ejercicio 25: Genere un data frame conteniendo la información de la tabla del ejercicio 14, y visualice algunas de sus filas y características (`head`, `tail`, `sample`, `shape`, `info`...). Reordene la tabla por el atributo **Aprobar**.
- | Ejercicio 26: En el siguiente enlace se presentan datos de los pasajeros del Titanic:

<https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.csv>

Elimine valores perdidos y haga un estudio entre los elementos (cuántos hombre, mujeres, niños, cuántos se salvaron, ...). El estudio es libre, usando los conocimientos adquiridos. Puede completar este ejercicio con técnicas de Visualización de Datos.

### 3 Documentación a entregar

La solución a los ejercicios deberá realizarse en horario de clase y, excepcionalmente, en horario de trabajo autónomo. Se deberá entregar un documento correspondiente con la plantilla de la siguiente página, modificado con la solución. Dicha solución consiste en un conjunto de entradas y salidas realizadas por los programas desarrollados. En el caso del ejercicio 26, se requiere una descripción del análisis realizado y las conclusiones a las que se han llegado, apoyándose de análisis de datos sobre la tabla y/o gráficas (que se deben incluir también en el documento).

## 3.1 Plantilla para entrega

Nombre y apellidos: David Mateo Merino

Ejercicio	Entradas y salidas del programa
1	<pre> num1 = float(input("Introduce el primer número: ")) num2 = float(input("Introduce el segundo número: ")) num3 = float(input("Introduce el tercer número: "))  maximo = num1 minimo = num1  if num2 &gt; maximo:     maximo = num2 if num3 &gt; maximo:     maximo = num3  if num2 &lt; minimo:     minimo = num2 if num3 &lt; minimo:     minimo = num3  print(f"El máximo es {maximo}") print(f"El mínimo es {minimo}") </pre>
2	<pre> num1 = int(input("Introduce el primer número: ")) num2 = int(input("Introduce el segundo número: "))  if num1 % num2 == 0:     print(f"{num2} divide a {num1}") elif num2 % num1 == 0:     print(f"{num1} divide a {num2}") else:     print("Los números no son divisibles entre sí") </pre>
3	<pre> for i in range(1001):     ai = ((-1)**i * (i**2 - 1)) / (2**i)     if ai % 2 == 0:         print(f"a{i} = {ai}") </pre>



4

```

suma = 0
contador = 0
num = int(input("Introduce un número entero positivo (0
para terminar): "))
while num != 0 and contador < 10:
    if num > 0:
        suma += num
        contador += 1
        num = int(input("Introduce un número entero positivo
(0 para terminar): "))

if contador > 0:
    media = suma / contador
    print(f"La media de los números positivos introducidos
es {media:.2f}")
else:
    print("No se ha introducido ningún número positivo.")

```

5

```

def es_primo(numero):

    # Los números menores o iguales a 1 no son primos
    if numero <= 1:
        return False
    # Los números 2 y 3 son primos
    elif numero <= 3:
        return True
    # Los números pares mayores que 2 no son primos
    elif numero % 2 == 0:
        return False
    # Verificar si el número es divisible por algún impar
    mayor que 3
    i = 3
    while i <= int(numero**0.5) + 1:
        if numero % i == 0:
            return False
        i += 2
    return True

numero = int(input("Introduce un número entero positivo:
"))
if es_primo(numero):
    print(f"{numero} es un número primo.")
else:
    print(f"{numero} no es un número primo.")

```

6

```
def es_primo(numero):

    # Los números menores o iguales a 1 no son primos
    if numero <= 1:
        return False
    # Los números 2 y 3 son primos
    elif numero <= 3:
        return True
    # Los números pares mayores que 2 no son primos
    elif numero % 2 == 0:
        return False
    # Verificar si el número es divisible por algún impar
    mayor que 3
    i = 3
    while i <= int(numero**0.5) + 1:
        if numero % i == 0:
            return False
        i += 2
    return True
print(es_primo(7))
print(es_primo(15))
print(es_primo(2))
print(es_primo(1))
print(es_primo(73))
```

7

```
def pedir_valor_en_rango(minimo, maximo):

    # Pedir al usuario que introduzca un valor en el rango
    while True:
        valor = input(f"Introduzca un valor entre {minimo}
y {maximo}: ")
        try:
            valor = float(valor)
        except ValueError:
            print("Debe introducir un valor numérico.")
            continue
        if minimo <= valor <= maximo:
            return valor
        else:
            print(f"El valor debe estar entre {minimo} y
{maximo}.")
print(pedir_valor_en_rango(0, 10))
```

8

```
def calcular_media_y_varianza(a, b, c):

    # Calcular la media
    media = (a + b + c) / 3

    # Calcular la varianza
    varianza = ((a - media) ** 2 + (b - media) ** 2 + (c -
media) ** 2) / 3

    return media, varianza
print(calcular_media_y_varianza(2, 4, 6))
```

9

```
n=0
m=0
L=[]

def ejercicio9():
    n= int(input("introduce n: "))
    m=int(input("introduce m: "))
    for i in range(n):
        L.append([])
        for f in range(m):
            L[i].append(f)
    return L

ejercicio9()
print(L)
```

10

```
def sumar_matrices(matriz1, matriz2):

    # Verificar si es posible la suma de las matrices
    if len(matriz1) != len(matriz2) or len(matriz1[0]) !=
len(matriz2[0]):
        return None

    # Crear la matriz resultante con ceros
    resultado = [[0 for j in range(len(matriz1[0]))] for i
in range(len(matriz1))]

    # Realizar la suma de las matrices
    for i in range(len(matriz1)):
        for j in range(len(matriz1[0])):
            resultado[i][j] = matriz1[i][j] +
matriz2[i][j]
```

	<pre> return resultado  matriz1 = [[1, 2, 3], [4, 5, 6]] matriz2 = [[7, 8, 9], [10, 11, 12]] print(sumar_matrices(matriz1, matriz2)) </pre>
11	<pre> def multiplicar_matrices(matriz1, matriz2):      # Verificar si es posible la multiplicación de las     matrices     if len(matriz1[0]) != len(matriz2):         return None      # Crear la matriz resultante con ceros     resultado = [[0 for j in range(len(matriz2[0]))] for i in range(len(matriz1))]      # Realizar la multiplicación de las matrices     for i in range(len(matriz1)):         for j in range(len(matriz2[0])):             for k in range(len(matriz2)):                 resultado[i][j] += matriz1[i][k] * matriz2[k][j]      return resultado  matriz1 = [[1, 2, 3], [4, 5, 6]] matriz2 = [[7, 8], [9, 10], [11, 12]] print(multiplicar_matrices(matriz1, matriz2)) </pre>
12	<pre> def multiplicacion_division(num1, num2):      multiplicacion = num1 * num2     division = None      if num2 != 0:         division = num1 / num2      return (multiplicacion, division)  print(multiplicacion_division(5, 2)) print(multiplicacion_division(5, 0)) </pre>
13	<pre> import statistics </pre>

```
def estadisticas(lista):
    minimo = min(lista)
    maximo = max(lista)
    media = statistics.mean(lista)
    varianza = statistics.variance(lista)
    desviacion_tipica = statistics.stdev(lista)
    return (minimo, maximo, media, varianza,
desviacion_tipica)

lista = [1, 2, 3, 4, 5, 6]

print(estadisticas(lista))
```

14

```
tabla = {
    "Caso": [1,2,3,4,5,6,7,8,9],
    "Tasacion": ["+", "-", "+", "-", "-", "+", "+", "+", "+"],
    "Valoracion": ["-", "-", "+", "+", "+", "+", "+", "-", "+"],
    "Entradas": ["No", "Si", "No", "Si", "Si", "Si", "Si", "Si", "No"],
    "Balance": ["+", "-", "+", "+", "-", "-", "+", "-", "-"],
    "Aprobar": ["No", "No", "Si", "Si", "No", "Si", "Si", "No", "No"]
}

# Crear el diccionario con la clave "Caso"
diccionario = {}
for i in range(len(tabla["Caso"])):
    caso = tabla["Caso"][i]
    diccionario[caso] = {}
    for columna in tabla:
        diccionario[caso][columna] = tabla[columna][i]

# Iterar sobre las claves pares del diccionario y mostrar
las filas correspondientes
for caso in diccionario:
    if caso % 2 == 0:
        fila = diccionario[caso]
        print("Caso:", caso)
        print("Tasacion:", fila["Tasacion"])
        print("Valoracion:", fila["Valoracion"])
        print("Entradas:", fila["Entradas"])
        print("Balance:", fila["Balance"])
        print("Aprobar:", fila["Aprobar"])
        print("-----")
```

15

```
D={
    'A': '.- ',
    'B': '-... ',
    'C': '-.- ',
    'D': '-.. ',
    'F': '. ',
    'G': '...- ',
    'H': '--.. ',
    'I': '.... ',
    'J': '.. ',
    'K': '.--- ',
    'L': '-.- ',
    'M': '-- ',
    'N': '-. ',
    'O': '--- ',
    'P': '.--. ',
    'Q': '--.- ',
    'R': '.-.. ',
    'S': '... ',
    'T': '- ',
    'U': '..- ',
    'V': '...- ',
    'W': '.-- ',
    'X': '-.-. ',
    'Y': '-.- ',
    'Z': '--.. '
}

cadena=input()

salida=[]
for letra in cadena:
    salida.append(D[letra])

print('El codigo morse de la salida es ',salida)
```

16

```
import matrices

# Ejemplo de uso de las funciones
matriz1 = [[1, 2], [3, 4]]
matriz2 = [[5, 6], [7, 8]]
resultado = matrices.sumar_matrices(matriz1, matriz2)
if resultado is not None:
    print("Suma de matrices:")
    for fila in resultado:
        print(fila)
```

```
else:
    print("Las matrices no tienen dimensiones compatibles
para la suma.")

Clase matrices:
def sumar_matrices(matriz1, matriz2):
    """Suma dos matrices."""
    if len(matriz1) != len(matriz2) or len(matriz1[0]) !=
len(matriz2[0]):
        return None
    resultado = []
    for i in range(len(matriz1)):
        fila = []
        for j in range(len(matriz1[0])):
            fila.append(matriz1[i][j] + matriz2[i][j])
        resultado.append(fila)
    return resultado

def restar_matrices(matriz1, matriz2):
    """Resta dos matrices."""
    if len(matriz1) != len(matriz2) or len(matriz1[0]) !=
len(matriz2[0]):
        return None
    resultado = []
    for i in range(len(matriz1)):
        fila = []
        for j in range(len(matriz1[0])):
            fila.append(matriz1[i][j] - matriz2[i][j])
        resultado.append(fila)
    return resultado

def multiplicar_matrices(matriz1, matriz2):
    """Multiplica dos matrices."""
    if len(matriz1[0]) != len(matriz2):
        return None
    resultado = []
    for i in range(len(matriz1)):
        fila = []
        for j in range(len(matriz2[0])):
            suma = 0
            for k in range(len(matriz2)):
                suma += matriz1[i][k] * matriz2[k][j]
            fila.append(suma)
        resultado.append(fila)
```

17

```

return resultado

from ClaseMatriz import Matriz

matriz1 = Matriz(2, 3)
matriz1.asignar_valor(0, 0, 1)
matriz1.asignar_valor(0, 1, 2)
matriz1.asignar_valor(0, 2, 3)
matriz1.asignar_valor(1, 0, 4)
matriz1.asignar_valor(1, 1, 5)
matriz1.asignar_valor(1, 2, 6)

matriz2 = Matriz(2, 3)
matriz2.asignar_valor(0, 0, 7)
matriz2.asignar_valor(0, 1, 8)
matriz2.asignar_valor(0, 2, 9)
matriz2.asignar_valor(1, 0, 10)
matriz2.asignar_valor(1, 1, 11)
matriz2.asignar_valor(1, 2, 12)

resultado = matriz1.suma(matriz2)
print(resultado)

resultado = matriz1.resta

Clase ClaseMatriz:
class Matriz:
    def __init__(self, filas, columnas):
        self.filas = filas
        self.columnas = columnas
        self.matriz = [[0 for j in range(columnas)] for i
in range(filas)]

    def __str__(self):
        cadena = ""
        for i in range(self.filas):
            cadena += "| "
            for j in range(self.columnas):
                cadena += str(self.matriz[i][j]) + " "
            cadena += "|\n"
        return cadena

    def asignar_valor(self, fila, columna, valor):
        self.matriz[fila][columna] = valor

```



```
def obtener_valor(self, fila, columna):
    return self.matriz[fila][columna]

def suma(self, otra_matriz):
    if self.filas != otra_matriz.filas or
self.columnas != otra_matriz.columnas:
        return None
    resultado = Matriz(self.filas, self.columnas)
    for i in range(self.filas):
        for j in range(self.columnas):
            resultado.matriz[i][j] = self.matriz[i][j]
+ otra_matriz.matriz[i][j]
    return resultado

def resta(self, otra_matriz):
    if self.filas != otra_matriz.filas or
self.columnas != otra_matriz.columnas:
        return None
    resultado = Matriz(self.filas, self.columnas)
    for i in range(self.filas):
        for j in range(self.columnas):
            resultado.matriz[i][j] = self.matriz[i][j]
- otra_matriz.matriz[i][j]
    return resultado

def multiplicacion(self, otra_matriz):
    if self.columnas != otra_matriz.filas:
        return None
    resultado = Matriz(self.filas,
otra_matriz.columnas)
    for i in range(self.filas):
        for j in range(otra_matriz.columnas):
            for k in range(self.columnas):
                resultado.matriz[i][j] +=
self.matriz[i][k] * otra_matriz.matriz[k][j]
    return resultado
```

18

```
class Vehiculo:
    def __init__(self,color,nRuedas):
        self.color=color
        self.nRuedas=nRuedas
```

```
def __str__(self):
    return 'vehiculo '+self.color

def mover(self):
    raise Exception('Funcion no implementada')

class Coche(Vehiculo):
    def __init__(self,color,nRuedas,cilindrada,nPuertas):
        super().__init__(color,nRuedas)
        self.cilindrada=cilindrada
        self.nPuertas=nPuertas

    def __str__(self):
        return 'coche '+self.color

    def mover(self):
        print('Coche ',self.color,'se mueve y
tiene',self.cilindrada,'cilindradas')

    def abrirCerrar(self):
        print('El coche tiene',self.nPuertas,'puertas',)

class Bicicleta(Vehiculo):
    def __init__(self,color,nRuedas,tipo):
        super().__init__(color,nRuedas)
        self.tipo=tipo

    def __str__(self):
        return 'bici '+self.color

    def mover(self):
        print('Bici',self.color,'se mueve y
tiene',self.nRuedas,'ruedas')

class BicicletaElectrica(Bicicleta):
    def
__init__(self,color,nRuedas,tipo,potencia,autonomia):
        super().__init__(color,nRuedas,tipo)
        self.potencia=potencia
        self.autonomia=autonomia

    def __str__(self):
        return 'bici electrica '+self.color

    def mover(self):
```

	<pre>         print('Bici Electrica',self.color,'se mueve y tiene',self.nRuedas,'ruedas')          def bateriaRestante(self):             print('Bici Electrica',self.color,'se mueve y tiene',self.autonomia,'autonomia')  coche=Coche('Rojo',4,3000,5) bici=Bicicleta('Rojo',2,'Montana') biciElec=BicicletaElectrica('Rojo',2,'Montana',400,100)  print(coche) print(bici) print(biciElec)  coche.mover() coche.abrirCcerrar()  bici.mover()  biciElec.mover() biciElec.bateriaRestante() </pre>
19	<pre> import numpy as np  # Generar un array de números del 0 al 8 array = np.arange(9) # Reorganizar el array en una matriz de 3x3 matriz = np.reshape(array, (3, 3)) print(matriz) </pre>
20	<pre> import numpy as np  matriz_identidad = np.eye(10) print(matriz_identidad) </pre>
21	<pre> import numpy as np  array_uniforme = np.linspace(0, 100, 11)  print(array_uniforme) </pre>
22	<pre> import numpy as np </pre>

	<pre> # Generar 10000 pares de números aleatorios points= np.random.rand( 10000, 2 ) * 2 - 1 #U(-1, 1) # Calcular la norma de cada par de números norms = np.linalg.norm(points, axis=1)  # Filtrar los puntos cuya norma es mayor que 1 mask = norms &lt;= 1 filtered_points = np.sum(mask)  # Aproximar PI utilizando los puntos restantes pi_approx = 4 * filtered_points / 10000  print(f"PI se aproxima a: {pi_approx}") </pre>
23	<pre> import numpy as np  arr = np.arange(1001, 1101) arr = np.random.permutation(arr)  print(arr) </pre>
24	<pre> import numpy as np  arr = np.random.randn(100000) mean = np.mean(arr) std_dev = np.std(arr)  print("Media:", mean) print("Desviación estándar:", std_dev) </pre>
25	<pre> import pandas as pd  tabla={     "Caso": [1,2,3,4,5,6,7,8,9],     "Tasacion": ["+", "-", "+", "-", "-", "+", "+", "+", "+"],     "Valoracion": ["-", "-", "+", "+", "+", "+", "+", "-", "+"],     "Entradas": ["No", "Si", "No", "Si", "Si", "Si", "Si", "Si", "No"],     "Balance": ["+", "-", "+", "+", "-", "-", "+", "-", "-"],     "Aprobar": ["No", "No", "Si", "Si", "No", "Si", "Si", "No", "No"] } </pre>

	<pre>data=pd.DataFrame(tabla)  print(data.head) print(data.tail) print(data.sample) print(data.shape[0]) print(data.info)</pre>
26	<pre>import pandas as pd import matplotlib.pyplot as plt import seaborn as sns  # Importar datos del archivo CSV url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.csv" df = pd.read_csv(url) # Información general del DataFrame df.info()  # Contar La cantidad de hombres, mujeres y niños hombres = df[df['sex'] == 'male']['sex'].count() mujeres = df[df['sex'] == 'female']['sex'].count() ninos = df[df['age'] &lt; 18]['age'].count()  print("Cantidad de hombres: ", hombres) print("Cantidad de mujeres: ", mujeres) print("Cantidad de niños: ", ninos)</pre>

# vuela

**PLATAFORMA  
DIGITAL DE  
ANDALUCÍA**

