TEMARIO.

T1.- Introducción y Tipos de Datos Simples (3 semanas)

- Distribución Anaconda y Entorno Spyder
- Editor y Consola
  - Consola como herramienta: calculadora-editor ***
  - Editor. Primer programa 'Hello world'
  - Ejecutar programa desde consola y con botón
- Sintaxis de Python
  - Asignación de variables (reglas)
  - Entrada y salida (input/print)
  - Comentarios en Python (#) y (“””)
- Tipos de variables
  - Objetos y métodos

    dir(objeto) type(objeto)
  - Numéricas
    - Entero - Real (flotante) – Complejo
    - Módulo math

TEMARIO.

- Tipos de variables Numéricas
    - Entero - Real (flotante) – Complejo
    - Operadores
        - Aritméticos
            - Suma +
            - Resta -
            - Producto *
            - División /
            - División entera //
            - Módulo (resto) %
            - Exponente **
        - Asignación
            - Simple =
            - Suma y asignación +=
            - Resta y asignación -=
            - Producto y asigna. *=
            - Cociente y asigna. /=

TEMARIO.

- Tipos de variables
  - Numéricas
  - Booleanos
    - True False
    - Operadores
      - Lógicos
        - AND `True and False # Devolverá False`
        - OR `True or False  # Devolverá True`
        - NOT `not True  # Devolverá False`
      - Comparación
        - Menor <
        - Mayor >
        - Menor o igual <=
        - Mayor o igual >=
        - Igual ==
        - Distinto !=

TEMARIO.

- Tipos de variables
  - Numéricas
  - Booleanos
  - Cadenas de caracteres (string)

```
mensaje = 'Esto es un mensaje de texto'
mensaje = “Esto es un mensaje de texto”
```

    - Operadores
      - Concatenar +
      - Multiplicar *
  - Datos de tipo None

    variable = None (Variable declarada sin valor)

- Identificador de tipo y conversiones
  - Type()
  - Conversión
    - Str()
    - Int()
    - Float()
    - Bool()

TEMARIO.

T2.- Programación Básica. Control de Flujo (3 semanas)

- Sentencias condicionales
  - ```
    if <condición>:
          <sentencia1>
          <sentencia2>
          …
    ```
  - ```
    if <condición>:
          <sentencia1>
          <sentencia2>
          …
    else:
          <sentencia1>
          <sentencia2>
          …
    ```
  - ```
    if <condición>:
          <sentencia1>
          …
    elif <condición>:
          <sentencia1>
          …
    elif <condición>:
          <sentencia1>
          …
    …
    else:
          <sentencia1>
          <sentencia2>
          …
    ```
- Indentación
    ```

TEMARIO.

T2.- Programación Básica. Control de Flujo

- Ejecuciones iterativas
  - Bucles `for`

```
for var in range(n):
        print(var)
```

range – break – continue

  - Bucles `while`

➢
```
while <condición>:
    <sentencia1>
    <sentencia2>
    …
```

➢
```
while <condición>:
    <sentencia1>
    …
else:
    <sentencia1>
    …
```

```
letra = 0
while letra<len(palabra):
        print(palabra[letra])
        letra += 1
```

TEMARIO.

T3.- Estructuras de datos (2 semanas)

- Listas

```
lista = [3, 'Hola', True]
```

  - Operaciones con listas
    - Lista[0]
    - Lista[:1]
    - Lista[-1]
    - Lista[0:2]
    - Lista[0:2:2]

```
lista = [1,2,3,4,5]
print(lista*2)      # [1,2,3,4,5,1,2,3,4,5]
lista = lista + [6]
print(lista)        # [1,2,3,4,5,6]
```

  - Funciones para listas
    - Len()
    - Lista.index()
    - Lista.append()
    - Lista.extend()
    - Lista.insert()
    - Lista.remove()
    - Lista.count()

  - Strings como listas

```
mensaje = 'Esto es un mensaje de texto'
```

    - len(mensaje)
    - mensaje.find()
    - mensaje.upper()
    - mensaje.lower()

TEMARIO.

T3.- Estructuras de datos

- Listas
- Tuplas (inmutable)
- Diccionarios

```
diccionario = {
'fruta1': 'melocotón’,
'fruta2': 'piña’,
'fruta3': 'naranja clementina’
}
```

  - Operaciones con diccionarios
    - `Diccionario_nuevo = dict()`
    - `Diccionario[‘fruta1’]`
    - `diccionario[‘fruta4’] = ‘papaya’`
  - Funciones para diccionarios
    - del diccionario['fruta2’]
    - list(diccionario)
    - sorted(diccionario)
    - ‘sandia’ in diccionario (True or False)

TEMARIO.

T3.- Estructuras de datos

- Listas
- Tuplas (inmutable) mencionar

```
tupla = 3.2, 'Hola', True
```

  - Operaciones con tuplas
    - `tupla[0]`
    - `tupla[:1]`
    - …
  - Funciones para tuplas
    - `Len()`
    - `tupla.index()`
    - `tupla.count()`

- Diccionarios (mencionar)

```
diccionario = {
'fruta1': 'melocotón’,
'fruta2': 'piña’,
'fruta3': 'naranja clementina’
}
```

  - Operaciones con diccionarios
    - `Diccionario_nuevo = dict()`
    - `Diccionario[‘fruta1’]`
    - `diccionario[‘fruta4’] = ‘papaya’`

TEMARIO.

T3.- Estructuras de datos

- Listas
- Tuplas (inmutable)
- Diccionarios

- Bucles `for` sobre listas, tuplas, cadenas de caracteres

```
lista = [3, 'Hola', True]
for element in lista:
        print(element)

for element in lista[1]:
        print(element)

for element in diccionario:
        print(element)
        print(diccionario[element])

palabra = 'prueba de texto’
for letra in palabra:
        print(letra)
```

TEMARIO.

T4.- Funciones en Python(2 semanas):

- Definición de funciones
  - ```
    def Nombre_función(argumento1,argumento2,…):
        """
        Documentación función
        """
        <sentencia1>
        <sentencia2>
        …
        return(objeto_salida)
    ```

```
def funcion():
    print('Ejemplo de función sin argumentos')
    return None

def area_triangulo(base,altura):
    area = base*altura/2
    return(area)

area_triangulo(6,3)
```
```

TEMARIO.

T4.- Funciones en Python:

- Definición de funciones
- Argumentos

```
def area_triangulo(base,altura):
    area = base*altura/2
    return(area)
```

  - Por posición

```
area_triangulo(6,3)
```

  - Por nombre

```
area_triangulo(altura=3 ,base=6)
```

  - Por defecto

```
def area_triangulo(base=6,altura=3):
    area = base*altura/2
    return(area)
area_triangulo()
```

  - Indeterminados

```
def imprime_numeros(*args):
    for numero in args:
        print(numero)
imprime_numeros(1, 7, 89, 46, 9394)
```

TEMARIO.

T4.- Funciones en Python:

- Definición de funciones
- Argumentos
- Retorno de funciones
  - Devuelve el valor pedido

```
def suma(s1,s2):
    return s1+s2
suma(10,3)
```

  - O una lista de objetos

```
def funcion():
    return 'Hola', 3.4*4, True or False
```

  - O ‘None’ si no usamos return

```
def impar(i):
    i%2 == 0
print(impar(4))  # Devuelve None

def impar(i):
    return i%2 == 0
print(impar(4))  # Devuelve True
```

TEMARIO.

T4.- Funciones en Python:

- Definición de funciones
- Argumentos
- Retorno de funciones

- “Módulos” propios import alias

- ~~Variables globales/locales (ámbito/scope)~~
- Funciones recursivas
- Función lambda

TEMARIO.

T5.- Vectores y Matrices (numpy) (2 semanas + 1 semana después del examen):

- Listas como vectores

```
n = 5
dx = 1.0/(n-1)
xlist = [i*dx for i in range(n)]
ylist = [f(x) for x in xlist]

print(ylist)
```

- Uso de numpy

```
import numpy as np
```

- Arrays en numpy
  - Desde listas

```
vector1 = np.array([1,2,3,4,5])
vector2 = np.array(xlist)
matriz1 = np.array([[1,2,3],[4,5,6]])
matriz2 = np.array([xlist,ylist])
```

  - Inicialización

```
x = np.zeros(n)   # vector de variables tipo float tamaño n
y = np.zeros_like(x)   # vector de variables tipo float tamaño n
u = np.zeros(5,dtype=int)
M = np.zeros(3,2) # Matriz de 3x2 variables float
```

TEMARIO.

T5.- Vectores y Matrices (numpy):

- Listas como vectores
- Uso de numpy
- Arrays en numpy
  - Trabajando con vectores

```
vector1 = np.array([1,2,3,4,5])
vector1[0]
vector1[4]
vector1[1:3]
vector1[1:-1]
vector1[0:4:2]
```

  - Métodos en numpy para vectores

```
np.arange(0, 11)            #[ 0  1  2  3  4  5  6  7  8  9 10]
np.arange(0., 2., 0.5)      #[ 0. 0.5 1.  1.5]
np.linspace(0., 1., 11)     #[ 0. 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ]
np.ones(5)                  #[ 1. 1. 1. 1. 1.]
np.random.rand(5)           #[ 0.34042503 0.4401839  0.82451019 0.50338825 0.90251413]
np.random.randint(0,100,5)  #[ 37 50 43 64 37]
```

  - Operaciones en numpy para vectores

```
v1 = np.ones(5)
v2 = v1[:]   !!!!!!!!!!!!
v2 = np.zeros_like(v1)
V2[:] = 2
```

TEMARIO.

T5.- Vectores y Matrices (numpy):

- Arrays en numpy
  - Operaciones en numpy para vectores
    - Suma de elementos: `np.sum(v1)`
    - Suma: `np.add(v1,v2)` `v1+v2` `6+v1`
    - Resta: `np.substract(v1,v2)` `v1-v2` `6-v1`
    - Multiplicación(elemento a elemento): `np.multiply(v1,v2)` `v1*v2` `6*v1`
    - Division(elemento a elemento): `np.divide(v1,v2)` `v1/v2` `v1/2`
    - Raiz cuadrada: `np.sqrt(v1)`
    - Redondeo: `np.round(v1,2)` `# redondeo a dos decimales`
    - `np.exp(v1)` `np.sin(v1)` `np.cos(v1)` `np.log(v1)` `np.square(v1)`

    - Copiar arrays

```
v2 = v1
v2[3] = 6
print (v2)  # [1. 1. 1. 6. 1.]
print (v1)  # [1. 1. 1. 6. 1.]  !!!!!!!!!!!!
```

```
v2 = v1.copy() ¿deepcopy?
v2[3] = 6
print (v2)  # [1. 1. 1. 6. 1.]
print (v1)  # [1. 1. 1. 1. 1.]
```

TEMARIO.

T5.- Vectores y Matrices (numpy):

- Arrays en numpy
  - Operaciones en numpy para vectores
  - Operaciones estadísticas con vectores
    - `np.amax(v1)  np.amin(v1)`
    - `np.argmax(v1) np.argmin(v1)  #Primera posición donde está el max/min`
    - `np.mean(v1)  np.var(v1)  np.std(v1)  # Media, varianza y desviación estándar`

TEMARIO.

T5.- Vectores y Matrices (numpy):

- Arrays en numpy
  - Trabajando con matrices

```
matriz1 = np.array([[1,2,3],[4,5,6]])
matriz2 = np.array([xlist,ylist])
M = np.zeros(3,2) # Matriz de 3x2 variables float

matriz1[1][0] o matriz1[1,0]     # 4
matriz1[1,:]                     # [4,5,6]
matriz1[0:1,2]                   # [3]
matriz1[:,0:3:2]                 # [[1 3]
                                 #  [4 6]]
```

  - Métodos en numpy para matrices

```
np.shape(matriz1)                # (2,3 )
matriz1.shape[0]                 # 2
matriz1.shape[1]                 # 3
matriz2 = np.eye(5)              # Identidad 5x5 float

lista = [1,2,3,4,5,6]
matriz = np.array(lista).reshape(3,2)     # [[1 2]
                                          #  [3 4]
                                          #  [5 6]]
matriz = np.linspace(1,30,30).reshape(5,6)
```

TEMARIO.

T5.- Vectores y Matrices (numpy):

- Arrays en numpy
  - Operaciones con matrices

```
matriz1 = np.array([[1,2,3],[4,5,6]])
np.sum(matriz)                # 21
np.sum(matriz,axis = 0)       # [5 7 9]
np.sum(matriz,axis = 1)       # [6 15]

np.max(matriz)                # 6
np.max(matriz,axis = 0)       # [4 5 6] Fila con los máximos por columna
np.max(matriz,axis = 1)       # [3 6]   Columna con los máximos por fila

np.argmax(matriz)             # 5  Convierte la matriz en un vector (1,2,3,4,5,6) y devuelve la posición max
np.argmax(matriz,axis = 0)    # [1 1 1] Fila con la posición del máximo en cada columna
np.argmax(matriz,axis = 1)    # [2 2]   Columna con la posición del máximo en cada fila
```

  - Matriz transpuesta

```
matriz2 = matriz1.T
```

TEMARIO.

T6.- Entrada y salida. Datos. Matplotlib (1 semana)

Escritura en fichero
Lectura desde fichero (poner límites)

SEMANA	CONTENIDO
Semana 1 (5-9 Septiembre)	T1.- Introducción Anaconda Spyder/Uso de consola como calculadora
Semana 2 (5-9 Septiembre)	T1.- Sintaxis básica. Tipos de datos. Variables Numéricas
Semana 3 (5-9 Septiembre)	T1.- Booleanos. Strings. Operadores. Métodos
Semana 4 (5-9 Septiembre)	T2.- Programación Básica. Control de Flujo
Semana 5 (3-8 Octubre)	T2.- Programación Básica. Bucles
Semana 6 (10-15 Octubre)	T2.- Programación Básica.
Semana 7 (17-22 Octubre)	**PEI 1 T1 y T2**
	T3.- Estructuras de datos. Listas. Tuplas. Diccionarios
Semana 8 (24-28 Octubre)	T3.- Estructuras de datos. Bucles sobre listas
Semana 9 (2-4 Noviembre) (**No Martes**)	T4.- Funciones en Python I
Semana 10 (7-11 Noviembre) (**No Jueves**)	
Semana 11 (15-18 Noviembre)	T4.- Funciones en Python. II
Semana 12 (21-25 Noviembre)	T5.- Vectores y Matrices I
Semana 13 (28 Nov- 2 Diciembre) (5-9 Diciembre Puente)	**PEI 2 T1, T2, T3 y T4**
	T5.- Vectores y Matrices II
Semana 14 (12-16 Diciembre)	T5.- Vectores y Matrices III
Semana 15 (19 -22 Diciembre)	T6 Ficheros-Matplotlib