


```
1
2 Programming 'Language' = {
3
4     Introducción = [Python,
5                     Micropython]
6
7
8
9
10
11 }
12
13
14
```



```
1
2
3 Clase Extra 01 = {
4
5     Presentación = [Les
6                     damos la bienvenida al
7                     curso]
8
9
10
11
12 }
13
14
```



```
1 Clases = {  
2
```

```
3  
4  
5 Clase Extra 01 Bibliotecas Estándar y Externas. Introducción a las  
6  bibliotecas estándar de Python. Uso de bibliotecas  
7 populares (NumPy, matplotlib).  
8
```

```
9 Clase  
10 Extra 02 Trabajando en equipo. Git y Github.  
11  
12  
13  
14
```



# Paquetes {

```
# Un paquete o librería es una colección de módulos que contienen
funciones, clases y variables que pueden ser reutilizadas en diferentes
programas. Los paquetes permiten a los desarrolladores aprovechar el
trabajo ya realizado por otros, facilitando la implementación de
funcionalidades complejas sin tener que escribir todo el código desde
cero.

# Como dijimos en clases previas, hay mucha referencia en internet (no
oficial) que hace alusión a los módulos como si fueran librerías y son
llamados de esta manera ya que es un concepto de otros lenguajes. Es
importante entender entonces cual es la diferencia entre los módulos y las
librerías o paquetes como se los nombra en la documentación oficial de
Python.
```

}



# Standard o Third Pary {

```
# Vamos a diferencias dos grandes grupos de paquetes:
```

```
# La librería estándar de Python es un conjunto de módulos y paquetes que vienen incluidos con la instalación de Python. Estos módulos proporcionan una amplia gama de funcionalidades que permiten realizar tareas comunes sin necesidad de instalar software adicional. La librería estándar cubre áreas como manipulación de archivos, operaciones matemáticas, manejo de fechas y horas, acceso a la red, y mucho más
```

```
# Las librerías de terceros son módulos y paquetes que no están incluidos en la librería estándar de Python, pero que pueden ser instalados y utilizados para ampliar las capacidades del lenguaje. Estas librerías son desarrolladas y mantenidas por la comunidad de desarrolladores y pueden ser instaladas utilizando herramientas como pip.
```

```
}
```



# Standard {

```
# Python viene con una biblioteca estándar que incluye muchas librerías  
útiles. Aquí hay algunos ejemplos:  
  
# math: Proporciona funciones matemáticas.  
# datetime: Permite trabajar con fechas y horas.  
# os: Interactúa con el sistema operativo.  
# sys: Proporciona acceso a variables y funciones del intérprete de  
Python.  
# random: Genera números pseudoaleatorios y realiza operaciones  
relacionadas.  
# json: Proporciona funciones para manipular archivos y datos en formato  
JSON.  
# re: Proporciona funciones para trabajar con expresiones regulares.  
# subprocess: Permite ejecutar comandos del sistema desde un script de  
Python.  
# threading: Permite realizar multitarea mediante el uso de hilos.  
# time: Proporciona funciones relacionadas con el manejo del tiempo.
```



# datetime {

# La librería **datetime** en Python proporciona clases para manipular fechas y horas. Es parte de la librería estándar de Python y permite realizar una amplia variedad de operaciones relacionadas con la fecha y la hora, como obtener la fecha y hora actual, calcular diferencias entre fechas, formatear fechas y horas, y mucho más.

# Funcionalidades Principales de datetime:

# Obtener la fecha y hora actual: Utilizando la clase datetime.

# Manipulación de fechas y horas: Sumar o restar días, horas, minutos, etc.

# Formateo de fechas y horas: Convertir objetos de fecha y hora a cadenas con un formato específico.

} # Cálculo de diferencias entre fechas: Utilizando la clase timedelta.



# Ejemplos {

```
from datetime import datetime

fecha = datetime.now() # Fecha y hora actual

print(fecha) # Imprime fecha y hora actual
print("Año:", fecha.year) # Año
print("Mes:", fecha.month) # Mes
print("Dia:", fecha.day) # Dia
print("Hora:", fecha.hour) # Hora
print("Minuto:", fecha.minute) # Minuto
print("Segundo:", fecha.second) # Segundo
print("Microsegundo:", fecha.microsecond) #
Microsegundo
print(fecha.strftime("%Y-%m-%d --
%H:%M:%S"))
```

}





# math {

# La librería **math** en Python proporciona funciones matemáticas básicas y avanzadas. Es parte de la librería estándar de Python y ofrece una amplia gama de funciones para realizar operaciones matemáticas, como trigonometría, logaritmos, potencias, y más.

# Funcionalidades Principales de la Librería **math**

# **Constantes Matemáticas:** `math.pi`, `math.e`, etc.

# **Funciones Trigonométricas:** `math.sin()`, `math.cos()`, `math.tan()`, etc.

# **Funciones Exponenciales y Logarítmicas:** `math.exp()`, `math.log()`, `math.log10()`, etc.

# **Funciones de Redondeo y Modificación:** `math.ceil()`, `math.floor()`, `math.trunc()`, etc.

# **Funciones de Potencia y Raíz:** `math.pow()`, `math.sqrt()`, etc.

# **Funciones de Combinatoria:** `math.factorial()`, `math.comb()`, `math.perm()`, etc.

}



# Ejemplos {

```
import math

print(math.pi)    # 3.141592653589793
print(math.e)     # 2.718281828459045
print(math.log(100, 10)) # 2.0 (logaritmo base 10)
print(math.pow(2, 3))  # 8.0 (2 elevado a la 3)
print(math.sqrt(16))   # 4.0 (raíz cuadrada)
print(math.ceil(4.2))  # 5 (redondeo hacia arriba)
print(math.floor(4.8)) # 4 (redondeo hacia abajo)
print(math.fabs(-4.8)) # 4.8 (valor absoluto)
print(math.factorial(5)) # 120 (5!)
print(math.gcd(12, 34)) # Máximo común divisor
print(math.lcm(12, 34)) # Mínimo común múltiplo
print((3+4j) * (7+10j)) # (-19+58j) (producto de dos números complejos)
```



# random {

# La librería **random** en Python proporciona funciones para generar números aleatorios y realizar operaciones aleatorias. Es parte de la librería estándar de Python y ofrece una amplia gama de funcionalidades para trabajar con aleatoriedad, como generar números enteros y flotantes aleatorios, seleccionar elementos aleatorios de una secuencia, y más.

# Funcionalidades Principales de la Librería random

# **Generación de Números Aleatorios:** random.randint(), random.random(), random.uniform(), etc.

# **Selección Aleatoria:** random.choice(), random.choices(), random.sample(), etc.

# **Reordenamiento Aleatorio:** random.shuffle()

# **Semillas para Reproducibilidad:** random.seed()

}



# Ejemplos {

```
import random
```

```
# Generar un número flotante aleatorio entre 0 y 1  
print(random.random())
```

```
# Generar un número flotante aleatorio entre 1 y 10  
print(random.uniform(1, 10))
```

```
# Generar un número entero aleatorio entre 1 y 10  
print(random.randint(1, 10))
```

```
# Seleccionar un elemento aleatorio de una cadena  
print(random.choice("Introduccion a Python y Micropython"))
```

```
}
```



```
1  os {
```

```
2  
3      # La librería os en Python proporciona una forma de interactuar con el  
4      sistema operativo. Es parte de la librería estándar de Python y ofrece una  
5      amplia gama de funcionalidades para realizar operaciones relacionadas con  
6      el sistema de archivos, la gestión de procesos, y la obtención de  
7      información del entorno del sistema.
```

```
8      # Funcionalidades Principales de la Librería os
```

```
9      # Operaciones con Archivos y Directorios: os.listdir(), os.mkdir(),  
10     os.remove(), os.rename(), etc.
```

```
11     # Información del Sistema: os.name, os.environ, os.getlogin(), etc.
```

```
12     # Gestión de Procesos: os.system(), os.popen(), os.kill(), etc.
```

```
13     # Rutas de Archivos: os.path.join(), os.path.exists(), os.path.isfile(),  
14     os.path.isdir(), etc.
```

```
}  
14
```



# Ejemplos {

```
1
2
3     import os
4
5     # Listar archivos y directorios en el directorio actual
6     print(os.listdir('.'))
7     # Crear un nuevo directorio
8     os.mkdir('nuevo_directorio')
9     # Renombrar un archivo o directorio
10    os.rename('nuevo_directorio', 'directorio_renombrado')
11    # Eliminar un directorio
12    os.rmdir('directorio_renombrado')
13    # Obtener el nombre del sistema operativo
14    print(os.name) # 'posix' en Unix, 'nt' en Windows
15    # Obtener el nombre del usuario actual
16    print(os.getlogin())
17 }
```



# 1 Librerías de terceros {

2  
3  
4 # Utilizar librerías de terceros en Python ofrece varias ventajas  
5 significativas que pueden mejorar la eficiencia, la productividad y la  
6 calidad del desarrollo de software. A continuación se detallan algunas de  
7 las principales ventajas:.

8 # Ahorro de Tiempo y Esfuerzo  
9 # Funcionalidades Avanzadas  
10 # Comunidad y Soporte  
11 # Optimización y Rendimiento  
12 # Mantenimiento y Actualizaciones  
13 # Facilidad de Uso  
14 # Compatibilidad y Estándares

}



## Ejemplo {

```
import time
import numpy as np

inicio_lista = time.time()
lista = list(range(1, 100000000))
lista_doble = [x * 2 for x in lista]
fin_lista = time.time()
inicio_array = time.time()
array = np.arange(1, 100000000)
array_doble = array * 2
fin_array = time.time()
print(f"Tiempo con lista de Python: {fin_lista - inicio_lista} segundos. Tiempo con array de NumPy: {fin_array - inicio_array} segundos.")
```

}





# numpy {

# La librería **matplotlib** es una de las librerías más populares en Python para la creación de gráficos y visualizaciones. Es altamente configurable y permite crear una amplia variedad de gráficos, desde simples gráficos de líneas hasta complejas visualizaciones en 3D.

Funcionalidades Principales de la Librería matplotlib

- # Gráficos de Líneas: plt.plot()
- # Gráficos de Barras: plt.bar()
- # Gráficos de Dispersión: plt.scatter()
- # Histogramas: plt.hist()
- # Gráficos de Tarta: plt.pie()
- # Gráficos de Caja: plt.boxplot()
- # Gráficos de Calor: plt.imshow()
- # Gráficos en 3D: Axes3D.plot()

}



# Ejemplo {

```
import numpy as np

array_1d = np.array([1, 2, 3, 4, 5])
print("Array 1D:", array_1d)
array_2d = np.array([[1, 2, 3], [4, 5, 6]])
print("Array 2D:\n", array_2d)
array_ceros = np.zeros((3, 3))
print("Array de ceros:\n", array_ceros)
array_unos = np.ones((2, 4))
print("Array de unos:\n", array_unos)
array_aleatorio = np.random.random((2, 2))
print("Array aleatorio:\n", array_aleatorio)
```

}



# matplotlib {

# La librería **datetime** en Python proporciona clases para manipular fechas y horas. Es parte de la librería estándar de Python y permite realizar una amplia variedad de operaciones relacionadas con la fecha y la hora, como obtener la fecha y hora actual, calcular diferencias entre fechas, formatear fechas y horas, y mucho más.

# Funcionalidades Principales de datetime:

# Obtener la fecha y hora actual: Utilizando la clase datetime.

# Manipulación de fechas y horas: Sumar o restar días, horas, minutos, etc.

# Formateo de fechas y horas: Convertir objetos de fecha y hora a cadenas con un formato específico.

# Cálculo de diferencias entre fechas: Utilizando la clase timedelta.

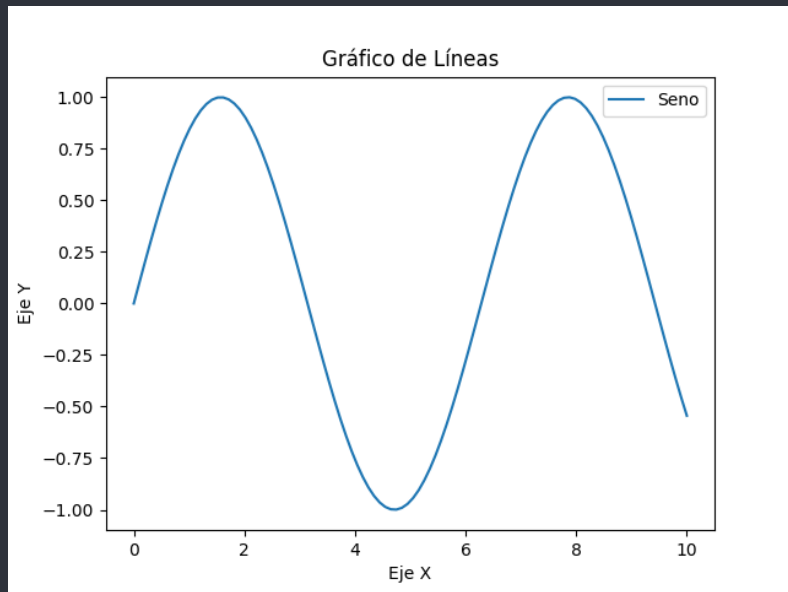


# Ejemplo {

```
import matplotlib.pyplot as plt
import numpy as np

# Datos
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Crear el gráfico de líneas
plt.plot(x, y, label='Seno')
plt.xlabel('Eje X')
plt.ylabel('Eje Y')
plt.title('Gráfico de Líneas')
plt.legend()
plt.show()
```



```
1
2
3 Aprender a programar
4
5 es aprender a pensar.
6
7
8
9
10
```

```
11 { Steve Jobs; }
12
13
14
```



```
1
2
3
4
5 { Nos vemos en la
6 proxima clase }
7
8
9
10
11
12
13
14
```

