Programación con Phyton nivel avanzado

Módulos y paquetes

Introducción

La filosofía de todo lenguaje de programación consiste en manejar una estructura y una organización, esto con la finalidad de ayudar a tener un código ordenado y sistematizado. Se busca modularizar y segmentar el código en varias carpetas y archivos de trabajo, gracias a esto surgen las librerías que se pueden conceptualizar como un conjunto de archivos ubicados en el directorio del proyecto con un orden especial y alimentan al proyecto.

Python es un lenguaje de programación versátil y a lo largo de su existencia la comunidad desarrolladora ha creado librerías (paquetes) para que el uso de este lenguaje se vuelva cada vez más sencillo y rápido de utilizar y con la finalidad de no tener que crear todas las funcionalidades desde cero. No es necesario descubrir el hilo negro si este ya ha sido descubierto. Esto hace que se optimice la productividad y se minimice el tiempo invertido en el desarrollo, sin embargo, en muchas ocasiones, las funciones a implementar desde una librería importada se deben adaptar (modificar) al contexto del proyecto. En esta práctica es importante nunca perder el orden dentro del proyecto y llevar un control dentro de la codificación con ***docstrings,*** que son documentos cuya finalidad es la de proveer un medio de información que ayuda a identificar los procesos, las instrucciones, los argumentos, y el paso a paso que se realiza.

A continuación, aprenderás a agregar librerías y módulos existentes, o bien realizarlos por ti mismo conservado la armonía y la correcta estructura en tus proyectos de desarrollo.

Explicación

Identificar los módulos existentes en Python

Una **librería** es un conjunto de archivos con funcionalidades que permiten al usuario llevar a cabo tareas que antes no se podían realizar o que requieren líneas muy extensas de código para obtener un resultado.

Las **librerías de Python** responden al conjunto de archivos a los cuales se les llamará **módulos,** cuya finalidad es evitar el tener que escribir el mismo código de nuevo y en los diferentes programas o proyectos de desarrollo. Los archivos suelen tener una extensión **.py,** o bien, en el caso de las plataformas Windows, anexa la extensión **DLL,** *Dynamic Load Libraries,* otra extensión necesaria para su implementación. Estos archivos se alojan dentro del directorio del proyecto, con el objetivo de crear una interfaz independiente. Las **librerías,** de acuerdo con su **uso, y aplicación** se pueden agrupar en:

1. *Deep learning:* se enfocan en la predicción de datos a través del *Big Data.*
2. *Machine learning:* mejoran el proceso de información y la resolución de problemas de clasificación, además del análisis de regresión de datos.
3. **Cálculo numérico:** preparan los datos para su cálculo, ofreciendo atributos importantes para estas tareas.
4. **Visualización:** sirven para el entendimiento de datos en gráficas más intuitivas.
5. **Inteligencia Artificial:** buscan resultados óptimos en las decisiones inteligentes del programa, llevando adelante diversas metodologías tecnológicas.
6. **Procesamiento de lenguaje natural:** calculando las frecuencias normalizadas, se construyen los modelos con datos de texto, de acuerdo con Geek for Geeks, (2021).

Para poder instalar un paquete de Python basta con identificar qué paquetería es la de tu interés y puedes conocer acerca de las librerías y paquetes disponibles en el *Python Package Index* en:

Python Packaging. (2022). *Find, install and publish Python packages with the Python Package Index*. Recuperado de https://pypi.org/

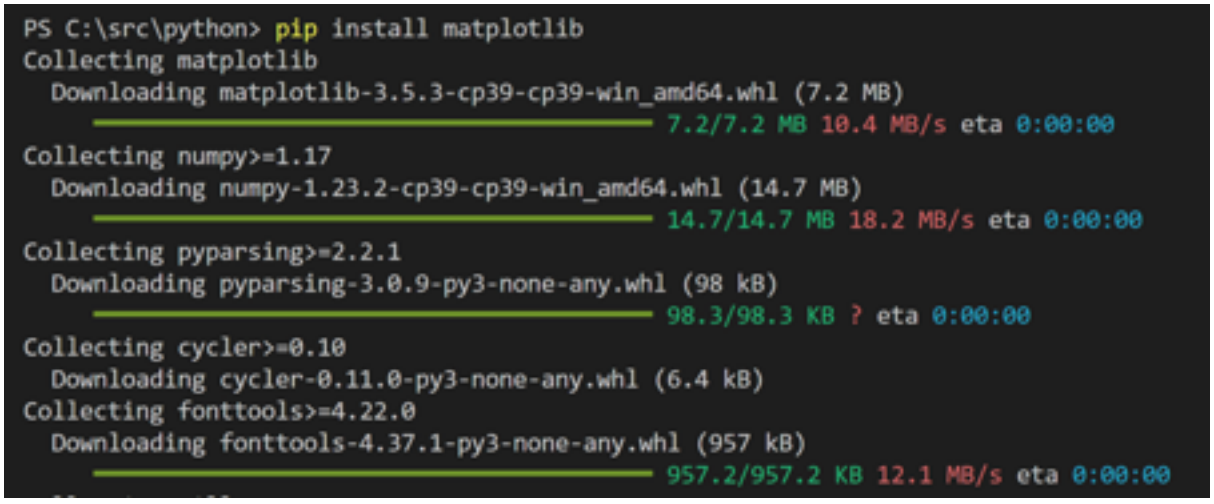
Explicación

Para **instalar un paquete o librería** puedes seguir las siguientes instrucciones (Python Software Foundation, 2022):

1. Verificar que Python está bien instalado, para ello, abre el comando **cmd** en tu ordenador y coloca el comando **"Python –version.**

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

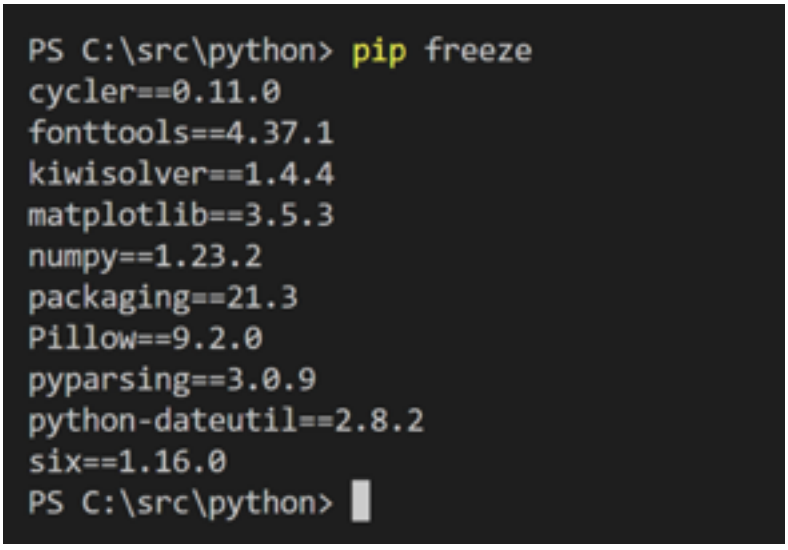
2. Después de investigar el nombre de la paquetería y bajo el comando **"pip install + nombre_de_la_paquetería"** se instalará la paquetería en tu entorno de trabajo.



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

3. Introduce en la terminal el comando **“pip freeze”** para tener una vista de todos los paquetes instalados.



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

En Python existen una gran cantidad de paquetes ya que constantemente la comunidad desarrolladora va agregando nuevas librerías, esto con la finalidad de facilitar el desarrollo de proyectos. Puedes armar tus propios paquetes y también puedes utilizar paquetes creados por otros autores los cuales están autorizados para su uso libre.

Dentro de los **paquetes de librerías más populares en Python** se tienen:

- **Matplotlib:** se utiliza para gráficos matemáticos, estadísticos y de algebra lineal tal como diagramas de barras, histogramas, series temporales y espectros de potencia.
- ***Tensorflow:*** desarrollado por Google y es clave en el cálculo numérico. Es una librería muy utilizada en el Deep Learning; su estructura permite crear redes neuronales.
- ***PyTorch:*** librería creada por Meta (Facebook) utilizando en cálculo numérico.
- **Keras:** su aplicación se da en una rama de análisis de datos, la de aprendizaje profundo, permite desarrollar prototipos de redes neuronales de una forma rápida y profunda.
- ***Scikit-learn:*** paquetes para machine learning con análisis de datos. Su interfaz es sencilla, ya que solo se requiere líneas de código.
- **Pandas:** librería de manejo de datos por excelencia, se destaca por un método y funciones fáciles de trabajar, genera estructuras para las operaciones con datos basados en *data frames.*
- **Numpy:** su principal funcionalidad está en el manejo de arreglos y matrices de datos para hacer un intercambio de datos entre estructuras o algoritmos de datos.

Explicación

Importar y hacer uso de funciones y estructuras existentes en módulos

Una paquetería está compuesta por varios módulos **.py**, estos módulos están creados justo para optimizar el tiempo de codificación entregando funciones precargadas para utilizar. Para poder utilizar estos módulos basta con ir al entorno de trabajo dentro del editor de texto y dentro del archivo (módulo) pon el comando **"import nombre_del_paquete"**.

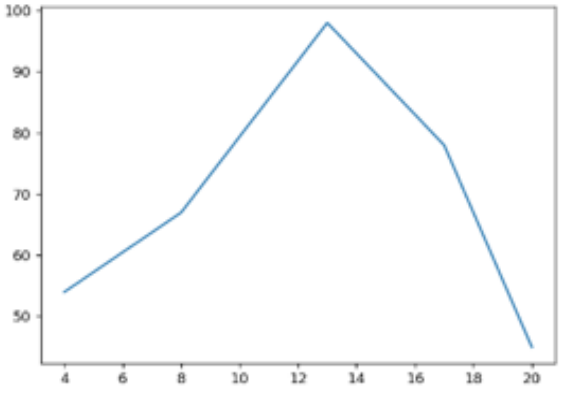
```
import matplotlib
```

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Para saber qué funciones están creadas dentro los módulos es imprescindible estudiar cada paquetería y encontrar el módulo que más convenga, en este ejemplo se utiliza Matplotlib como paquetería para poder visualizar lo siguiente:

```
nuevo.py
1
2   import matplotlib
3
4   import matplotlib.pyplot as plt
5   plt.plot([4,8,13,17,20],[54, 67, 98, 78, 45])
6   plt.show()

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER
```



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

En el ejemplo se observa cómo se importa la paquetería por el comando **"import"** de **"matplotlib.pyplot as plt"**, para que pueda tomar las funciones correspondientes a ese módulo.

Explicación

Crear módulos y paquetes propios

Un módulo prácticamente sirve para organizar lógicamente el código de Python, su objetivo es crear estructuras con atributos, con nombres arbitrarios a los que se pueda enlazar y hacer referencia. En pocas palabras es todo archivo de funcionalidad con código Python y es almacenado con la extensión **.py**. En un módulo se puede definir **clases, funciones, variables y código ejecutable** (El Libro de Python, s.f).

El código Python para un módulo nombrado **funciones**, normalmente, reside un archivo llamado **modulo.py**.

```python
# modulo.py
def suma(a, b):
    return a + b

def resta(a, b):
    return a - b
```

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Una vez creado el módulo se puede importar a otros archivos para que se pueda utilizar sin necesidad de copiarlo y hacer otro de nuevo. Lo único que debes hacer es generar la sentencia por el comando **import**.

```python
import modulo

def suma(a, b):
    return a + b

def resta(a, b):
    return a - b
```

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Un **módulo** puede contener tanto código, que declare **ejecuciones** en el programa como definiciones de **funciones.** Estas declaraciones están pensadas para inicializar el archivo, se ejecutan únicamente la primera vez que el módulo se encuentra en una declaración **import.**

También se puede **importar** únicamente los **componentes** que sean de interés como se muestra a continuación.

```
from modulo import suma, resta

print(suma(4, 3))   # 7
print(resta(10, 9)) # 1
```

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

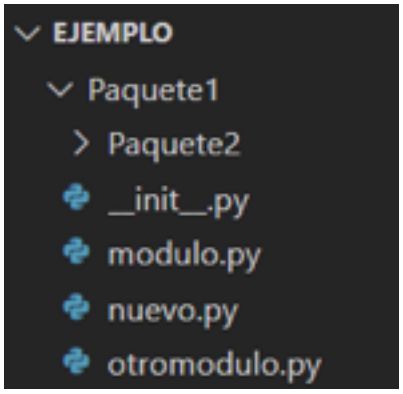
Por último, se puede **importar todo el módulo** haciendo uso de *, sin necesidad de usar **modulo.***.

```
from modulo import *

print(suma(4, 3))   # 7
print(resta(10, 9)) # 1
```

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Para **crear un paquete** tienes que imaginar un árbol de carpetas, donde dentro de la carpeta principal tengas más carpetas de archivos para cualquier proyecto que realices:



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Una **paquetería se conforma por varios archivos (módulos) y diferentes carpetas** dentro de la misma, lo interesante aquí es que para que se pueda utilizar la **carpeta principal** como paquetería, cada directorio dentro de esta debe tener un archivo llamado **__init__.py** como se ve en el ejemplo.

Entonces las **formas de importar** esta paquetería sería la siguiente:

```
import Paquete1.Paquete2
from Paquete1 import Paquete2
from Paquete1.Paquete2 import *
```

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Así puedes adquirir los módulos que vienen dentro de estas paqueterías y utilizar las funciones creadas dentro de estos.

Incluir docstrings dentro de los programas creados

Muchos programadores principiantes cuando inician en el mundo del desarrollo y comienzan a crear sus nuevos proyectos, piensan que es sencillo identificar qué hicieron en cada paso del código que están creando, y es verdad que cuando creas los primeros programas, quizá las líneas de código sean muy pocas y es fácil identificar procedimientos, la lógica del proyecto. Sin embargo, todo comienza a ponerse más difícil cuando estos proyectos comienzan a hacerse mucho más extensos y se agrega complejidad cuando hay que compartirlos con otros desarrolladores para que puedan trabajar en conjunto, entonces tanto tus colegas como tú pueden llegar a confundirse y no saber a ciencia cierta cuál es la funcionalidad de un método o variable creada en el código. Para evitar estas situaciones se crean los llamados **docstrings,** que se pueden entender como **cadenas de texto** a modo de documentación de código.

De acuerdo con DelfStack (2022), un **docstring** se considera como la primera cadena de texto que se coloca inmediatamente después de definir una función o clase con la finalidad de proveer al desarrollador o al equipo de desarrollo, conocimiento sobre la función o clase (su objetivo, funcionalidad, atributos, etc.)

Estos **docstrings** se colocan en el código en cada paso que se genera para realizar una tarea y se puede identificar porque normalmente son solo texto **(“Función”, #función, “””Función”””)**. Se visualiza de esta forma:

Explicación

```
""" Módulo para cálculos diversos """

def suma_total(monto=0):
    """ Calcula la suma total """
    calculo_suma = 20
    calculo_suma += monto

    """Retorna el calculo"""
    return calculo_suma
```

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Estas líneas de texto no afectan el funcionamiento del código, pero ayudan a identificar cada paso de código que se construye, es esencial llevar este control en la programación pues te puedes ahorrar muchísimo tiempo de búsqueda.

Cuando se crean proyectos grandes de programación y se comparten con varios desarrolladores es importante dejar una serie de pasos o instrucciones de lo que se trata tu código y cómo está funcionando, así que para ellos se crea un archivo *README* **(Docstring);** este archivo es de tipo **texto simple** y sólo sirve a modo de informar al lector la estructura y la funcionalidad del proyecto, se puede proveer de instrucciones, o bien, indicar lo que falta aún por concluir. Es común encontrar estos archivos en tutoriales o archivos creados por otros programadores a fin de utilizarlos para practicar o utilizar el código provisto.

Accede a la sección **Archivos del curso** del presente módulo para replicar la práctica.

Cierre

Gracias a la versatilidad de Python existen gran número de librerías de todo tipo para utilizar, pero es importante aprender cómo utilizarlas y agregarlas correctamente a los proyectos de desarrollo y no solo eso, sino también tener la habilidad de poder generar paqueterías y así compartirlas a otros desarrolladores que estén interesados en trabajar con las funcionalidades creadas por ti.

Recuerda que el orden dentro de estos proyectos son la base para no perder el tiempo replicando código y buscando líneas de código por todos lados.

Referencias bibliográficas

- DelfStack. (2022). *Accediendo a Docstrings en Python.* Recuperado de https://www.delftstack.com/es/howto/python/python-docstrings/
- Geeks for Geeks. (2021). *Libraries in Python.* Recuperado de https://www.geeksforgeeks.org/libraries-in-python/
- El libro de Python. (s.f.). *Módulos y paquetes.* Recuperado de https://ellibrodepython.com/modulos-python
- Python Software Foundation. (2022). *Instalando módulos en Python.* Recuperado de https://docs.python.org/es/3/installing/index.html

Para saber más

Los siguientes enlaces son externos a la Universidad Tecmilenio, al acceder a ellos considera que debes apegarte a sus términos y condiciones.

Lecturas

Para conocer más acerca de **sintaxis básica de Python,** te sugerimos leer lo siguiente:

- Programacion.net. (s.f.). *Cómo funcionan los módulos en Python.* Recuperado de https://programacion.net/articulo/como_funcionan_los_modulos_en_python_1510

Videos

Para conocer más acerca de **sintaxis básica de Python,** te sugerimos revisar lo siguiente:

- Farjan's CC. (2021, 2 de enero). *Instalar Librerías en Python (numpy, scipy, matplotlib, sympy...) + Solución a error FT2FONT [Archivo de video].* Recuperado de https://www.youtube.com/watch?v=e_OMEtUybjc
- nicosiored. (2020, 12 noviembre). *Documentar - 33 - Python Intermedio tutorial en español* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=4fkPKAnmUlQ

Checkpoints

Asegúrate de:

- Instalar paqueterías nuevas en Python desde el **cmd** y empezar a utilizarlas desde los módulos.
- Crear paqueterías entendiendo el uso de los archivos **.py**, para poder utilizar los directorios como paqueterías.
- Agregar paqueterías en los proyectos de programación.
- Agregar **docstrings** en la codificación para tener una mejor referencia en un futuro, o bien, para proveer de datos técnicos y de funcionalidad a otros desarrolladores.

Requerimientos técnicos

Python es capaz de ser utilizado para cualquier plataforma o sistema operativo existente. En el caso de Windows se sugiere contar con:

Hardware
- Se recomienda tener con al menos 500 MB de almacenamiento disponible.
- Procesador de por lo menos 1.6 GHz.
- 1 GB de RAM.

Plataformas
- OS X El Capitan (10.11+).
- Windows 10, 11 (32-bit and 64-bit).
- Linux (Debian): Ubuntu Desktop 16.04, Debian 9.
- Linux (Red Hat): Red Hat Enterprise Linux 7, CentOS 7, Fedora 34.

Prework

- Estudia el tema **Módulos y paquetes.**

- Estudia los recursos referentes a la creación de proyectos en Python y configuración a través de paqueterías con VS Code en las siguientes ligas, te servirán para prepararte en el tema:

Microsoft. (s.f.). *Creación y administración de proyectos en Python.* Recuperado de
https://docs.microsoft.com/es-es/training/modules/python-create-manage-projects/

Microsoft (s.f.). *Instalación de herramientas de codificación para el desarrollo de Python.* Recuperado de
https://docs.microsoft.com/es-es/training/modules/install-code-tools-python-nasa/

Los siguientes enlaces son externos a la Universidad Tecmilenio, al acceder a ellos considera que debes apegarte a sus términos y condiciones.