UNIVERSIDAD AUTONÓMA DE BAJA CALIFORNIA

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

PROGRAMACION CON PYTHON

ELABORADO POR:

KENIA PAOLA JIMÉNEZ MARTÍNEZ

GRUPO:

372

EVIDENCIA:

INVESTIGACIÓN

DOCENTE:

EDGAR IVAN AVILA GARRIDO

Tijuana B.C, 05 de noviembre del 2023

PEP 8 – GUÍA DE ESTILOS EN PYTHON: se define la guía de cómo escribir Python de forma correcta, a modo de guía de estilos del lenguaje.

DISEÑO DEL CÓDIGO: Se definen las claves de cómo debería de escribirse el código Pythónico en sí, definiendo conceptos como la indentación, la longitud de las líneas, las líneas en blanco y la guía de importación de otros paquetes entre otros conceptos. Su objetivo es establecer normas y recomendaciones que ayuden a que el código Python sea más legible y consistente. Esto facilita la colaboración en proyectos, la mantenibilidad del código y la prevención de errores comunes.

CADENAS DE CARACTERES: Python soporta diferentes tipos de caracteres para crear cadenas de caracteres siendo simples o triples, comillas simples o dobles.

- Comillas simples o dobles: para cadenas de caracteres de una línea o varias.
- Triples comillas dobles: para cadenas de documentación y bloques de cadenas de caracteres.

¿COMO DETECTAR ERRORES DE PEP 8 AUTOMÁTICAMENTE?:

Existen librerías que permite detectar errores/violaciones de reglas del PEP 8 denominadas linters y suelen estar incluidas en IDEs de Python (por ejemplo en PyCharm)

Algunos IDEs actuales pueden sugerir el cambio necesario automáticamente antes de realizar un commit en Git.

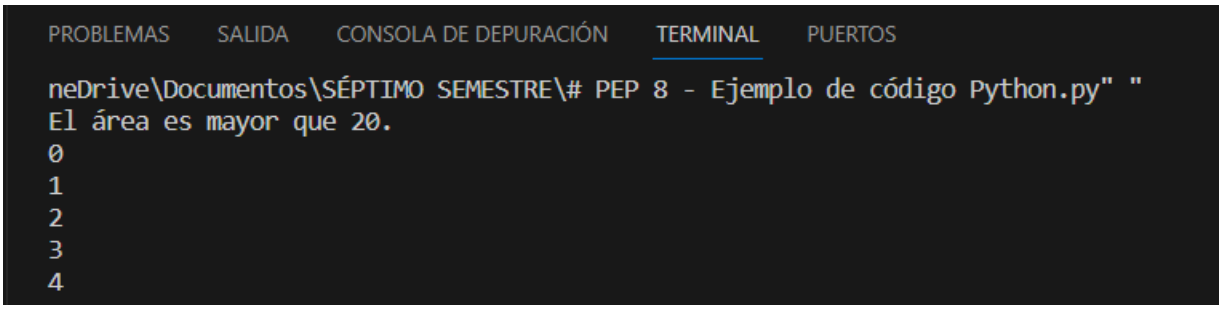
Se puede utilizar el programa pycodestyle (anteriormente llamado pep8), para comprobar si el código sigue las reglas de correctamente. Este programa se puede instalar usando pip y ejecutarse en consola sobre cualquier módulo python.

EJEMPLO:

```python
# PEP 8 - Ejemplo de código Python

# Importar módulos
import math
import os

# Definir una función con un docstring
def calcular_area_circulo(radio):
    """
    Calcula el área de un círculo.

    :param radio: El radio del círculo
    :type radio: float
    :return: El área del círculo
    :rtype: float
    """
    if radio <= 0:
        raise ValueError("El radio debe ser un valor positivo.")

    area = math.pi * radio**2
    return area

# Utilizar espacios en blanco de forma consistente
resultado = calcular_area_circulo(5)

# Evitar líneas de código demasiado largas
ruta_del_archivo = os.path.join(
    "carpeta",
    "subcarpeta",
    "archivo.txt"
)

# Seguir las convenciones de nombrado
nombre_del_estudiante = "Juan Perez"

# Evitar espacios en blanco al final de las líneas
if resultado > 20:
```

Lín. 27, col. 33 Espacios: 4

```python
if resultado > 20:
    print("El área es mayor que 20.")

# Utilizar sangrías con 4 espacios
for i in range(5):
    print(i)

```


PROBLEMAS
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL
PUERTOS
neDrive\Documentos\SÉPTIMO SEMESTRE\# PEP 8 - Ejemplo de código Python.py" "
El área es mayor que 20.
0
1
2
3
4

MÓDULOS Y PAQUETES

Módulos: Un módulo es un archivo de Python cuyos objetos (funciones, clases, excepciones, etc.) pueden ser accedidos desde otro archivo. Se trata simplemente de una forma de organizar grandes códigos.

Paquetes: Un paquete es una carpeta que contiene varios módulos.

Python incluye una inmensa cantidad de módulos y paquetes en su instalación (aún más grande es aquella desarrollada por la comunidad, de la que hablaremos más adelante), a los que se conoce como librería estándar.

ENTORNOS VIRTUALES EN PYTHON: Los entornos virtuales se pueden describir como directorios de instalación aislados. Este aislamiento te permite localizar la instalación de las dependencias de tu proyecto, sin obligarte a instalarlas en todo el sistema.

BENEFICIOS:

- Puedes tener varios entornos, con varios conjuntos de paquetes, sin conflictos entre ellos. De esta manera, los requisitos de diferentes proyectos se pueden satisfacer al mismo tiempo.
- Puedes lanzar fácilmente tu proyecto con sus propios módulos dependientes.

Aquí hay dos formas en las que puede crear entornos virtuales Python.

Virtualenv

virtualenv es una herramienta que se utiliza para crear entornos Python aislados. Crea una carpeta que contiene todos los ejecutables necesarios para usar los paquetes que necesitaría un proyecto de Python.

Puedes instalarlo con pip:

```
pip install virtualenv
```

Verifica la instalación con el siguiente comando:

```
virtualenv --version
```

Crear un entorno

Para crear un entorno virtual utiliza:

```
virtualenv --no-site-packages my-env
```

Esto crea una carpeta en el directorio actual con el nombre del entorno (my-env/). Esta carpeta contiene los directorios para instalar módulos y ejecutables de Python.

También puedes especificar la versión de Python con la que quieres trabajar. Simplemente usa el argumento --python=/ruta/a/la/version/de/python. Por ejemplo, python2.7:

virtualenv --python=/usr/bin/python2.7 my-env

Lista de entornos

Puedes enumerar los entornos disponibles con:

lsvirtualenv

Activar un entorno

Antes de utilizar el entorno, debes activarlo:

source my-env/bin/activate

Esto asegura que solo se usen los paquetes bajo my-env/.

Notarás que el nombre del entorno se muestra a la izquierda de la línea de comandos. De esta forma puedes ver cuál es el entorno activo.

Instalar paquetes

Puede instalar paquetes uno por uno o configurando un archivo requirements.txt para tu proyecto.

pip install algun-paquete

pip install -r requirements.txt

Si quieres crear un archivo requirements.txt a partir de los paquetes ya instalados, ejecuta el siguiente comando:

pip freeze > requirements.txt

El archivo contendrá la lista de todos los paquetes instalados en el entorno actual y sus respectivas versiones. Esto te ayudará a lanzar tu proyecto con sus propios módulos dependientes.

Desactivar un entorno

Si has terminado de trabajar con el entorno virtual, puedes desactivarlo con:

deactivate

Esto te devuelve al intérprete de Python predeterminado del sistema con todas sus bibliotecas instaladas.

Eliminar un entorno

Simplemente elimina la carpeta del entorno.

Conda

Conda es una gestión de paquetes, dependencias y entornos para muchos lenguajes, incluido Python.

Para instalar Conda, sigue estas instrucciones.

Crear un entorno

Para crear un entorno virtual, use:

conda create --name my-env

Conda creará la carpeta correspondiente dentro del directorio de instalación de Conda.

También puedes especificar con qué versión de Python quieres trabajar:

conda create --name my-env python=3.6

Lista de entornos

Puedes enumerar los entornos disponibles con:

conda info --envs

Activar un entorno

Antes de utilizar el entorno, debes activarlo:

source activate my-env

Instalar paquetes

Igual que con virtualenv.

Desactivar un entorno

Si has terminado de trabajar con el entorno virtual, puedes desactivarlo con:

source deactivate

Eliminar un entorno

Si quieres eliminar un entorno de Conda, utiliza:

conda remove --name my-env

¿QUÉ ES PYPI?

py.pi o the Python Package Index es una herramientas de Python que funciona como un repositorio de software para el lenguaje de programación. En esta plataforma se encuentran paquetes de dependencias y otras herramientas, cada vez más avanzadas, para el mismo lenguaje.

En esta plataforma, al igual que cualquier otro repositorio de código, como Github o Gitlabs, te permite crear una cuenta de usuario para poder agregar nuevos paquetes, buscarlos o guardarlos para utilizarlos después en la escritura de tus códigos.

REFERENCIAS BIBLIOGRÁFICAS

Tuxskar. (2021, 18 febrero). *PEP 8 – Guía de estilos en Python*. El Pythonista. https://elpythonista.com/pep-8

Bustamante, S. J. (2021, 9 febrero). *Entornos virtuales de Python explicados con ejemplos*. freeCodeCamp.org. https://www.freecodecamp.org/espanol/news/entornos-virtuales-de-python-explicados-con-ejemplos

}*Módulos y paquetes - tutorial de Python*. (s. f.). https://tutorial.recursospython.com/modulos-y-paquetes/

KeepCoding, R. (2023, 11 mayo). ¿Qué es PYPI? | KeepCoding Bootcamps. *KeepCoding Bootcamps*. https://keepcoding.io/blog/que-es-pypi/