UNIVERSIDAD CATÓLICA DE TEMUCO
INGENIERÍA CIVIL EN INFORMÁTICA
FACULTAD DE INGENIERÍA

Programación I

PEP8 y Archivos

Profesor: Ignacio Lincolao Venegas

Ingeniería Civil Informática, UCT

PEP8

La guia PEP8 posee un conjunto de recomendaciones que facilitan y ayudan a escribir código más legible, donde se abarcan desde los nombres de variables hasta el máximo número de caracteres que debería poseer una línea. Si desean consultarla puede acceder a: https://www.python.org/dev/peps/pep-0008/

Algunos puntos a destacar que destaca el PEP8:

- Siempre preferir espacios en vez de tabs
- Usar 4 espacios en la identación
- Las líneas deben tener como máximo 79 caracteres
- Las líneas que pasen la longitud hay que dividirla en dos líneas y la línea resulten de la división debe ser indentada
- En una fila, las funciones y las clases deben ser separadas por dos líneas en blanco
- No colocar espacios alrededor de índices de lista, llamadas de funciones o argumentos

fuente: https://dev.to/viktorvillalobos/que-es-el-pep-8-y-porque-deberia-implementarlo-54bh

PEP8

NOMBRES

- Las funciones deben estar declaradas en minúscula y las palabras separadas por guiones bajos **`def aplica_suma()`**
- Las clases y excepciones deben ser capitalizadas por palabra **`class DogClass`**
- Las constantes deben estar en mayúsculas separadas por guiones bajos **`NUMERO_CONSTANTE = 10`**

EXPRESIONES

- Usar negación en linea **`(if a is not b)`** en vez de negar una expresión positiva **`(if not a is b)`**
- No validar valores vacíos usando if(len(lista)==0), usar if not lista
- Siempre color los imports al inicio del archivo
- Siempre importar funciones y clases usando **from modulo import funcion_importada** en vez de usar el modulo completo **import modulo**
- Si aún se necesita hacer import con ruta relativa ocupar **from . import modulo**
- Las importaciones deben estar en el orden
    - Modulos de libreria standar (random, match, etc..)
    - Modulos externos (Como por ejemplo una libreria de obtenida de un repo de github)
    - Modulos del proyectos (Importaciones de otros archivos propios)

fuente: https://dev.to/viktorvillalobos/que-es-el-pep-8-y-porque-deberia-implementarlo-54bh

PEP8

COMENTARIOS

- Si actualizamos el codigo tambien es necesario actualizar los comentarios para tener no inconsistencias
- Los comentarios deben ser frases completas, con la primera letra en mayúsculas.
- Evitar comentarios pocos descriptivos que no aporten mas aya de lo que se ve.

Libreria autopep8

Existen librería que ayudan a formatear el archivo a PEP8, se llama autopep8, se puede instalar con pip

```
pip install autopep8
```

fuente: https://dev.to/viktorvillalobos/que-es-el-pep-8-y-porque-deberia-implementarlo-54bh

PEP 20 Zen de python

El zen de python es un conjunto de principios para el diseño de python, se pueden ver escribiendo **`import this`**

1. Bello es mejor que feo.
2. Explícito es mejor que implícito.
3. Simple es mejor que complejo.
4. Complejo es mejor que complicado.
5. Plano es mejor que anidado.
6. Espaciado es mejor que denso.
7. La legibilidad es importante.
8. Los casos especiales no son lo suficientemente especiales como para romper las reglas.
9. Sin embargo la practicidad le gana a la pureza.
10. Los errores nunca deberían pasar silenciosamente.
11. A menos que se silencien explícitamente.
12. Frente a la ambigüedad, evitar la tentación de adivinar.
13. Debería haber una, y preferiblemente sólo una, manera obvia de hacerlo.
14. A pesar de que eso no sea obvio al principio a menos que seas Holandés.
15. Ahora es mejor que nunca.
16. A pesar de que nunca es muchas veces mejor que *ahora* mismo.
17. Si la implementación es difícil de explicar, es una mala idea.
18. Si la implementación es fácil de explicar, puede que sea una buena idea.
19. Los espacios de nombres son una gran idea, ¡tengamos más de esos!

fuente: https://www.python.org/dev/peps/pep-0020/

Lista de comprensiones

Una lista de comprensión permite la generación de una lista en una sola linea de codigo, combinando el bucle for y la creación de nuevos elementos en la misma línea, agregando automáticamente cada elemento nuevo. Para realizarla se debe crear la lista abriendo los corchetes y definir la expresión de valor que uno quiere agregar ejemplo valor*3, esta expresión devolverá múltiplos de 3. Luego escribir el bucle para generar los números que se desea aplicar a la expresión. Ejemplo:

```
lista =  [ valor * 3 for valor in range(10) ]
print(lista)
>[0, 3, 6, 9, 12, 15, 18, 21, 24, 27]
```

Fuente: Matthes, Eric. Python Crash Course: A Hands-on, Project-Based Introduction to Programming. 2016.

Listas de comprensión

`[value`	`for value in itera`	`if condition]`
Valor que se insertará a la nueva lista.	Ciclo en donde se extraerán elementos de la lista o objeto iterable.	Condición opcional que permite realizar filtros en los ciclos.

Múltiples Inputs con lista de comprensión

```
# 2 inputs
x, y = [int(n) for n in input("Ingresa dos valores separados por espacios: ").split()]
print("El valor de x es: ", x)
print("El valor de y es: ", y)

# 3 inputs
x, y, z = [int(n) for n in input("Ingresa tres valores separados por espacio: ").split()]
print("El valor de x es: ", x)
print("El valor de y es: ", y)
print("El valor de z es: ", z)

# múltiples valores en un array
x = [int(n) for n in input("Ingrese múltiples valores separados por espacios: ").split()]
print("El arreglo es ", x)


# En split() pueden agregar otros caracteres ejemplo si quieren que el usuario ingrese los datos separados
# por comas .split(“,”), el usuario tendrá que ingresar los números como: 15,7,8,3
```

Fuente: https://www.geeksforgeeks.org/taking-multiple-inputs-from-user-in-python/

Archivos en python

Los objetos *File* en Python representan ficheros en los sistemas operativos, en el se pueden guardar datos para poder utilizarlos posteriormente. Los ficheros pueden ser de tipo texto o ficheros binarios, donde el fichero de texto contiene caracteres que son legibles por el ser humano y estos se guardan con una codificación (ASCII, UTF-8, …). En el caso del fichero binario este está compuesto por el flujo de bytes y sólo tiene sentido para los programas o aplicaciones.

Cuando se trabajan los ficheros de textos, se tiene en cuenta que posee una estructura como una secuencia de líneas. Cada línea acaba con un carácter especial conocido como EOL (fin de línea), dependiendo si es windows o UNIX el sistema operativo puede variar, en UNIX es \n y en windows es \r\n, pero en python cuando ingresamos \n en un fichero el automaticamente lo encarga de convertir el caracter al caracter correspondiente del sistema operativo.

fuente: https://j2logo.com/como-escribir-en-un-fichero-en-python/

Archivos en python

Para escribir en un fichero o leer el fichero se utiliza la función open() al invocar la función estamos generando un objeto tipo File. La función file posee dos parámetros de entrada el primero es la ruta donde se va a crear o acceder al archivo y el segundo parámetro es el modo en el que se abrirá el archivo donde estos pueden ser lectura, escritura, etc.., por defecto se abre en modo texto pero en los parámetro de modo si agregamos la letra b podemos abrir de forma binaria. Los siguientes modos son

r	Lectura
r+	Lectura/Escritura
w	Sobreescritura. Si no existe archivo se creará
a	Añadir. Escribe al final del archivo
b	Binario

+	Permite lectura/escritura simultánea
U	Salto de línea universal
rb	Lectura binaria
wb	Sobreescritura binaria
r+b	Lectura/Escritura Binaria

fuente: https://j2logo.com/como-escribir-en-un-fichero-en-python/

Archivos en python

La estructura básica para abrir crear uno nuevo es:

```
1  f = open('nuevo_archivo.txt', 'w')
2  try:
3      # Procesamiento para escribir en el fichero
4      print()
5  finally:
6      f.close()
```

Aunque existe la sentencia with que permite cerrar y liberar los recursos independiente si ocurre un error o no

```
9  with open('nuevo_archivo.txt', 'w') as f:
10     # Procesamiento del fichero
```

Para escribir un archivo se puede hacer de la siguiente manera

```
9  with open('nuevo_archivo.txt', 'w') as f:
10     f.write('Hola mundo\n')
```

fuente: https://j2logo.com/como-escribir-en-un-fichero-en-python/

Archivos en python

Para leer un archivo se debe abrir normalmente con el open y indicando la ruta, luego para acceder a los datos pueden hacerlo con read():

```
archivo = open('archivo.txt', 'r')
# Lee los primeros 10 bytes
cadena1 = archivo.read(10)
#lee el resto de los bytes
cadena2 = archivo.read()
# Para leer por linea completa
while True:
    linea = archivo.readline() # Lee hasta un \n
    if not linea: # Si no quedan mas lineas
        break
    print(linea)

archivo.readlines() # Leerá todas las líneas y las almacenará en un array

# Cierra el archivo
archivo.close()
## Nota: Ojo readline() y readlines() son distintos
```

fuente: https://j2logo.com/como-escribir-en-un-fichero-en-python/

Archivos en python

Para escribir un archivo se debe abrir normalmente con el open y indicando la ruta, luego para escribir datos pueden hacerlo con write():

```
archivo = open(‘archivo.txt’, ‘w’)
# Escribe una cadena añadiendo un salto de línea
palabra = “casa”
archivo.write(palabra+ ‘\n’)

lista = [“casa”,”perro”,”miércoles”]
archivo.writelines(lista) # Escribe la lista en el archivo

# Cierra el archivo
archivo.close()
```

fuente: https://j2logo.com/como-escribir-en-un-fichero-en-python/

Archivos en python

Existe otra forma de abrir un archivo donde se utiliza la sentencia **with**, esta sentencia se usa para el manejo de errores, ayudando a reducir la cantidad de código, haciendo automatica sentencia finally y la limpieza manual.

```
with open('archivo.txt', 'w') as archivo:
    archivo.write('Texto agregado')

print("El código continua aqui y ya termino de usar el archivo")
# Cuando finaliza with solo genera el close
```

fuente: https://j2logo.com/como-escribir-en-un-fichero-en-python/

Archivos en python

Tarea(15 min)

- Crear un archivo donde escribirás nombres de frutas o verduras usando python cada fruta o verdura será una línea en el archivo, por lo que la siguiente fruta o verdura se encontrara en la siguiente línea.
- Luego abre el archivo y extrae la información almacenando en una lista
- Crea una función donde abres el archivo y almacenar nueva información sin borrar la antigua. La nueva información se ira solicitando al usuario.
- Crea un archivo que contenga la información de 5 personas, donde cada línea contiene un nombre, edad, fecha de nacimiento, numero de telefono. El archivo donde se almacenarán se llamará contactos.txt
- En otro archivo de python abre el archivo contacto.txt extrae la información y muestra el nombre de las personas mayores de 18 años.

fuente: https://j2logo.com/como-escribir-en-un-fichero-en-python/