Librerias en Python

Módulos estándar

Fuente:hektorprofe.net

A continuación os resumo los que son para mí algunos de los módulos esenciales de Python, luego profundizaremos en algunos de ellos:

- **copy**: Se utiliza para crear copias de variables referenciadas en memoria, como colecciones y objetos.
- **collections**: Cuenta con diferentes estructuras de datos.
- **datetime**: Maneja tipos de datos referidos a las fechas/horas.
- **html**, **xml** y **json**: También quiero comentar estos tres módulos, que aunque no los vamos a trabajar, permiten manejar cómodamente estructuras de datos html, xml y json. Son muy utilizados en el desarrollo web.
- **math**: Posiblemente uno de los módulos más importantes de cualquier lenguaje, ya que incluye un montón de funciones para trabajar matemáticamente. Lo veremos más a fondo en esta misma unidad.
- **random**: Este es el cuarto y último módulo que veremos en esta unidad, y sirve para generar contenidos aleatorios, escoger aleatoriamente valores y este tipo de cosas que hacen que un programa tenga comportamientos al azar. Es muy útil en el desarrollo de videojuegos y en la creación de pruebas.
- **sys**: Nos permite conseguir información del entorno del sistema operativo o manejarlo en algunas ocasiones, se considera un módulo avanzado e incluso puede ser peligroso utilizarlo sin conocimiento.
- **threading**: Se trata de otro módulo avanzado que sirve para dividir procesos en subprocesos gracias a distintos hilos de ejecución paralelos. La programación de hilos es compleja y he considerado que es demasiado para un curso básico-medio como éste.
- **tkinter**: Finalmente, y posiblemente el que me hace más ilusión de todos. Tkinter es el módulo de interfaz gráfica de facto en Python. Le dedicaré una unidad entera

bastante extensa en la que aprenderemos a crear formularios con botones, campos de texto y otros componentes.

Módulo collections

El módulo integrado de colecciones nos provee otros tipos o mejoras de las colecciones clásicas.

Contadores

La clase **Counter** es una subclase de diccionario utilizada para realizar cuentas:

Ejemplo

```
from collections import Counter

lista = [1,2,3,4,1,2,3,1,2,1]
Counter(lista)
```

Resultado

```
Counter({1: 4, 2: 3, 3: 2, 4: 1})
```

Ejemplo

```
from collections import Counter

Counter("palabra")
```

Resultado

```
Counter({'a': 3, 'b': 1, 'l': 1, 'p': 1, 'r': 1})
```

Ejemplo

```
from collections import Counter
animales = "gato perro canario perro canario perro"
c = Counter(animales.split())
print(c)

print(c.most_common(1))  # Primer elemento más repetido
print(c.most_common(2))  # Primeros dos elementos más repetidos
```

Resultado

```
Counter({'canario': 2, 'gato': 1, 'perro': 3})
[('perro', 3)]
[('perro', 3), ('canario', 2)]
```

Módulo datetime

Este módulo contiene las clases **time** y **datetime** esenciales para manejar tiempo, horas y fechas.

Clase datetime: Esta clase permite crear objetos para manejar fechas y horas:

```
from datetime import datetime

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

print(dt)
print(dt.year)         # año
print(dt.month)        # mes
print(dt.day)          # día

print(dt.hour)         # hora
print(dt.minute)       # minutos
print(dt.second)       # segundos
print(dt.microsecond)  # microsegundos

print("{}:{}:{}".format(dt.hour, dt.minute, dt.second))
print("{}/{}/{}".format(dt.day, dt.month, dt.year))
```

Resultado

```
datetime.datetime(2016, 6, 18, 21, 29, 28, 607208)
2016
6
18
21
29
28
607208
21:29:28
18/6/2016
```

Es posible crear un datetime manualmente pasando los parámetros (year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None). Sólo son **obligatorios** el **año**, el **mes** y el **día**.

```
from datetime import datetime

dt = datetime(2000,1,1)
print(dt)
```

Resultado

```
datetime.datetime(2000, 1, 1, 0, 0)
```

No se puede cambiar un atributo al vuelo. Hay que utilizar el método **replace**:

```
dt = dt.replace(year=3000)
print(dt)
```

Resultado

```
datetime.datetime(3000, 1, 1, 0, 0)
```

Módulo math

Este módulo contiene un buen puñado de funciones para manejar números, hacer redondeos, sumatorios precisos, truncamientos... además de constantes.

Redondeos

```
import math

print(math.floor(3.99))  # Redondeo a la baja (suelo)
print(math.ceil(3.01))   # Redondeo al alta (techo)
```

Sumatoria mejorada

```
numeros = [0.9999999, 1, 2, 3])
math.fsum(numeros)
```

Resultado

```
6.9999999
```

Truncamiento

```
math.trunc(123.45)
```

Resultado

```
123
```

Potencias y raíces

```
math.pow(2, 3)  # Potencia
math.sqrt(9)    # Raíz cuadrada (square root)
```

Constantes

```
print(math.pi)  # Constante pi
print(math.e)   # Constante e
```

Módulo random

Aleatoriedad

Este módulo contiene funciones para generar números aleatorios:

```
import random

# Flotante aleatorio >= 0 y < 1.0
print(random.random())

# Flotante aleatorio >= 1 y <10.0
print(random.uniform(1,10))

# Entero aleatorio de 0 a 9, 10 excluído
print(random.randrange(10))

# Entero aleatorio de 0 a 100
print(random.randrange(0,101))

# Entero aleatorio de 0 a 100 cada 2 números, múltiplos de 2
print(random.randrange(0,101,2))

# Entero aleatorio de 0 a 100 cada 5 números, múltiplos de 5
print(random.randrange(0,101,5))
```

Resultado

```
0.12539542779843138
6.272300429556777
7
14
68
25
```

Muestras

También tiene funciones para tomar muestras:

```
# Letra aleatoria
print(random.choice('Hola mundo'))

# Elemento aleatorio
random.choice([1,2,3,4,5])

# Dos elementos aleatorios
random.sample([1,2,3,4,5], 2)
```

Resultado

```
o
3
[3, 4]
```

Mezclas

Y para mezclar colecciones:

```
# Barajar una lista, queda guardado
lista = [1,2,3,4,5]
random.shuffle(lista)
print(lista)
```

Resultado

```
[3, 4, 2, 5, 1]
```