Módulos

El código de un programa en Python puede reutilizarse en otro importándolo. Cualquier fichero con código de Python reutilizable se conoce como *módulo* o *librería*.

Los módulos suelen contener funciones reutilizables, pero también pueden definir variables con datos simples o compuestos (listas, diccionarios, etc), o cualquier otro código válido en Python.

Python permite importar un módulo completo o sólo algunas partes de él. Cuando se importa un módulo completo, el intérprete de Python ejecuta todo el código que contiene el módulo, mientras que si solo se importan algunas partes del módulo, solo se ejecutarán esas partes.

Importación completa de módulos (`import`)

- `import M` : Ejecuta el código que contiene `M` y crea una referencia a él, de manera que pueden invocarse un objeto o función `f` definida en él mediante la sintaxis `M.f`.
- `import M as N` : Ejecuta el código que contiene `M` y crea una referencia a él con el nombre `N`, de manera que pueden invocarse un objeto o función `f` definida en él mediante la sintaxis `N.f`. Esta forma es similar a la anterior, pero se suele usar cuando el nombre del módulo es muy largo para utilizar un alias más corto.

Importación parcial de módulos (`from import`)

- `from M import f, g, ...` : Ejecuta el código que contiene `M` y crea referencias a los objetos `f, g, ...`, de manera que pueden ser invocados por su nombre. De esta manera para invocar cualquiera de estos objetos no hace falta precederlos por el nombre del módulo, basta con escribir su nombre.
- `from M import *` : Ejecuta el código que contiene `M` y crea referencias a todos los objetos públicos (aquellos que no empiezan por el carácter _) definidos en el módulo, de manera que pueden ser invocados por su nombre.

Cuando se importen módulos de esta manera hay que tener cuidado de que no haya coincidencias en los nombres de funciones, variables u otros objetos.

```
>>> import calendar
>>> print(calendar.month(2019, 4))
April 2019
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30

>>> from math import *
>>> cos(pi)
-1.0
```

Módulos de la librería estándar más importantes

Python viene con una biblioteca de módulos predefinidos que no necesitan instalarse. Algunos de los más utilizados son:

- sys: Funciones y parámetros específicos del sistema operativo.
- os: Interfaz con el sistema operativo.
- os.path: Funciones de acceso a las rutas del sistema.
- io: Funciones para manejo de flujos de datos y ficheros.
- string: Funciones con cadenas de caracteres.
- datetime: Funciones para fechas y tiempos.
- math: Funciones y constantes matemáticas.
- statistics: Funciones estadísticas.
- random: Generación de números pseudo-aleatorios.

Otras librerías imprescindibles

Estas librerías no vienen en la distribución estándar de Python y necesitan instalarse.

- NumPy: Funciones matemáticas avanzadas y arrays.
- SciPy: Más funciones matemáticas para aplicaciones científicas.
- matplotlib: Análisis y representación gráfica de datos.
- Pandas: Funciones para el manejo y análisis de estructuras de datos.
- Request: Acceso a internet por http.

La librería Datetime

Para manejar fechas en Python se suele utilizar la librería `datetime` que incorpora los tipos de datos `date`, `time` y `datetime` para representar fechas y funciones para manejarlas. Algunas de las operaciones más habituales que permite son:

- Acceder a los distintos componentes de una fecha (año, mes, día, hora, minutos, segundos y microsegundos).
- Convertir cadenas con formato de fecha en los tipos `date`, `time` o `datetime`.
- Convertir fechas de los tipos `date`, `time` o `datetime` en cadenas formateadas de acuerdo a diferentes formatos de fechas.
- Hacer aritmética de fechas (sumar o restar fechas).
- Comparar fechas.

Los tipos de datos `date`, `time` y `datetime`

- `date(año, mes, dia)` : Devuelve un objeto de tipo `date` que representa la fecha con el `año`, `mes` y `dia` indicados.

- `time(hora, minutos, segundos, microsegundos)` : Devuelve un objeto de tipo `time` que representa un tiempo la `hora`, `minutos`, `segundos` y `microsegundos` indicados.
- `datetime(año, mes, dia, hora, minutos, segundos, microsegundos)` : Devuelve un objeto de tipo `datetime` que representa una fecha y hora con el `año`, `mes`, `dia`, `hora`, `minutos`, `segundos` y `microsegundos` indicados.

```
from datetime import date, time, datetime
>>> date(2020, 12, 25)
datetime.date(2020, 12, 25)
>>> time(13,30,5)
datetime.time(13, 30, 5)
>>> datetime(2020, 12, 25, 13, 30, 5)
datetime.datetime(2020, 12, 25, 13, 30, 5)
>>> print(datetime(2020, 12, 25, 13, 30, 5))
2020-12-25 13:30:05
```

_

Acceso a los componentes de una fecha

- `date.today()` : Devuelve un objeto del tipo `date` la fecha del sistema en el momento en el que se ejecuta.
- `datetime.now()`: Devuelve un objeto del tipo `datetime` con la fecha y la hora del sistema en el momento exacto en el que se ejecuta.
- `d.year` : Devuelve el año de la fecha `d`, puede ser del tipo `date` o `datetime`.
- `d.month` : Devuelve el mes de la fecha `d`, que puede ser del tipo `date` o `datetime`.
- `d.day` : Devuelve el día de la fecha `d`, que puede ser del tipo `date` o `datetime`.
- `d.weekday()` : Devuelve el día de la semana de la fecha `d`, que puede serpuede ser del tipo `date` o `datetime`.
- `t.hour` : Devuelve las horas del tiempo `t`, que puede ser del tipo `time` o `datetime`.
- `t.hour` : Devuelve los minutos del tiempo `t`, que puede ser del tipo `time` o `datetime`.
- `t.second` : Devuelve los segundos del tiempo `t`, que puede ser del tipo `time` o `datetime`.
- `t.microsecond` : Devuelve los microsegundos del tiempo `t`, que puede ser del tipo `time` o `datetime`.

```
>>> from datetime import date, time, datetime
>>> print(date.today())
2020-04-11
>>> dt = datetime.now()
>>> dt.year
2020
>>> dt.month
4
>>> dt.day
11
```

```
>>> dt.hour
22
>>> dt.minute
5
>>> dt.second
45
>>> dt.microsecond
1338
```

Conversión de fechas en cadenas con diferentes formatos

- `d.strftime(formato)` : Devuelve la cadena que resulta de transformar la fecha `d` con el formato indicado en la cadena `formato`. La cadena `formato` puede contener los siguientes marcadores de posición: `%Y` (año completo), `%y` (últimos dos dígitos del año), `%m` (mes en número), `%B` (mes en palabra), `%d` (día), `%A` (día de la semana), `%a` (día de la semana abrevidado), `%H` (hora en formato 24 horas), `%I` (hora en formato 12 horas), `%M` (minutos), `%S` (segundos), `%p` (AM o PM), `%C` (fecha y hora completas), `%x` (fecha completa), `%X` (hora completa).

```
>>> from datetime import date, time, datetime
>>> d = datetime.now()
>>> print(d.strftime('%d-%m-%Y'))
13-04-2020
>>> print(d.strftime('%A, %d %B, %y'))
Monday, 13 April, 20
>>> print(d.strftime('%H:%M:%S'))
20:55:53
>>> print(d.strftime('%H horas, %M minutos y %S segundos'))
20 horas, 55 minutos y 53 segundos
```

Conversión de cadenas en fechas

- `strptime(s, formato)` : Devuelve el objeto de tipo `date`, `time` o `datetime` que resulta de convertir la cadena `s` de acuerdo al formato indicado en la cadena `formato`. La cadena `formato` puede contener los siguientes marcadores de posición: `%Y` (año completo), `%y` (últimos dos dígitos del año), `%m` (mes en número), `%B` (mes en palabra), `%d` (día), `%A` (día de la semana), `%a` (día de la semana abrevidado), `%H` (hora en formato 24 horas), `%I` (hora en formato 12 horas), `%M` (minutos), `%S` (segundos), `%p` (AM o PM), `%C` (fecha y hora completas), `%x` (fecha completa), `%X` (hora completa).

```
>>> from datetime import date, time, datetime
>>> datetime.strptime('15/4/2020', '%d/%m/%Y')
datetime.datetime(2020, 4, 15, 0, 0)
>>> datetime.strptime('2020-4-15 20:50:30', '%Y-%m-%d %H:%M:%S')
datetime.datetime(2020, 4, 15, 20, 50, 30)
```

Aritmética de fechas

Para representar el tiempo transcurrido entre dos fechas se utiliza el tipo `timedelta`.

- `timedelta(dias, segundos, microsegundos)` : Devuelve un objeto del tipo `timedelta` que representa un intervalo de tiempo con los `dias`, `segundos` y `micorsegundos` indicados.
- `d1 - d2` : Devuelve un objeto del tipo `timedelta` que representa el tiempo transcurrido entre las fechas `d1` y `d2` del tipo `datetime`.
- `d + delta` : Devuelve la fecha del tipo `datetime` que resulta de sumar a la fecha `d` el intervalo de tiempo `delta`, donde `delta` es del tipo `timedelta`.

```
>>> from datetime import date, time, datetime, timedelta
>>> d1 = datetime(2020, 1, 1)
>>> d1 + timedelta(31, 3600)
datetime.datetime(2020, 2, 1, 1, 0)
>>> datetime.now() - d1
datetime.timedelta(days=132, seconds=1826, microseconds=895590)
```