



PYTHON

Fundamentos de Python 2
Módulos, paquetes y PIP

MÓDULOS, PAQUETES Y PIP

MÓDULOS

MÓDULOS, PAQUETES Y PIP

- ¿Qué es un módulo?
 - “Un archivo que contiene definiciones y sentencias de Python” – *Tutorial de Python*.
- Roles:
 - Rol de usuario de módulo
 - Rol de proveedor de módulo
- Un módulo tiene:
 - Nombre:
 - Entidades (funciones, variables, constantes, clases y objetos).
- Biblioteca estándar de Python:
<https://docs.python.org/3/library/index.html>



MÓDULOS, PAQUETES Y PIP

- Importación de módulos:
 - Instrucción import.
 - Siempre antes del primer uso.
 - Se puede incluir en cualquier parte del código.
 - Formas básicas:
 - `import modulo1`
 - `import modulo1, modulo2`

MÓDULOS, PAQUETES Y PIP

- Namespace:

- Un espacio en el que cohabitan nombres sin conflicto.
- La referencia a una entidad se hace a través de su módulo:
 - modulo.entidad
 - Ejemplo: math.pi
- Coexistencia de namespaces sin conflicto:
 - En nuestro script existe una variable pi
 - En math existe la constante pi
 - En el código las referencias a pi son a nuestro namespace; las referencias a math.pi son al namespace de math.
- Conflicto: si no se referencia al namespace, se toma el propio por defecto.

MÓDULOS, PAQUETES Y PIP

- Importación de módulos:
 - `from modulo import entidad`
 - `from modulo import entidad1, entidad2`
 - `from modulo import *`
 - `import modulo as alias` → “aliasing” o renombrado
 - Las referencias al módulo se hacen a través del alias.
 - `from modulo import entidad as alias`
 - `from modulo import entidad1 as alias1, entidad2 as alias2`

MÓDULOS, PAQUETES Y PIP

- La función `dir` aplicada a módulos
 - Si se usa alias, utilizarlo en la función

```
>>> import math
```

```
>>> dir(math)
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin',  
'asinh', 'atan', 'atan2', 'atanh', 'cbrt', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees',  
'dist', 'e', 'erf', 'erfc', 'exp', 'exp2', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',  
'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp',  
'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter', 'perm', 'pi', 'pow', 'prod',  
'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc', 'ulp']
```

MÓDULOS, PAQUETES Y PIP

- Funciones trigonométricas del módulo math:
 - $\sin(x) \rightarrow$ seno
 - $\cos(x) \rightarrow$ coseno
 - $\tan(x) \rightarrow$ tangente
 - $\text{asin}(x) \rightarrow$ arcoseno
 - $\text{acos}(x) \rightarrow$ arcocoseno
 - $\text{atan}(x) \rightarrow$ arcotangente
 - $\text{pi} \rightarrow$ Constante aproximada a π .
 - $\text{radians}(x) \rightarrow$ Convierte grados a radianes
 - $\text{degrees}(x) \rightarrow$ Convierte radianes a grados

MÓDULOS, PAQUETES Y PIP

- Funciones trigonométricas del módulo math:
 - $\sinh(x)$ \rightarrow seno hiperbólico
 - $\cosh(x)$ \rightarrow coseno hiperbólico
 - $\tanh(x)$ \rightarrow tangente hiperbólico
 - $\operatorname{asinh}(x)$ \rightarrow arcoseno hiperbólico
 - $\operatorname{acosh}(x)$ \rightarrow arcocoseno hiperbólico
 - $\operatorname{atanh}(x)$ \rightarrow arcotangente hiperbólico

MÓDULOS, PAQUETES Y PIP

- Funciones relacionadas con la exponenciación del módulo `math`:
 - `e` → constante aproximada al número de Euler (e)
 - `exp(x)` → e elevado a x
 - `log(x)` → logaritmo natural o neperiano de x
 - `log(x, y)` → logaritmo de x con base y
 - `log10(x)` → logaritmo decimal de x : más preciso que `log(x, 10)`
 - `log2(x)` → logaritmo binario de x : más preciso que `log(x, 2)`
- `pow(x, y)` → No pertenece al módulo `math`, es una función *built-in*.

MÓDULOS, PAQUETES Y PIP

- Funciones de propósito general del módulo math:
 - `ceil(x)` → Entero más pequeño mayor o igual que `x`.
 - `floor(x)` → Entero más grande mayor o igual que `x`.
 - `trunc(x)` → Valor de `x` truncado a entero.
 - `factorial(x)` → Devuelve el factorial de `x` (`x!`), debiendo `x` ser entero y no negativo.
 - `hypot(x, y)` → Obtiene el valor de la hipotenusa de un triángulo rectángulo cuyos catetos miden `x` e `y`.

MÓDULOS, PAQUETES Y PIP

- Módulo **random**:

- Números pseudoaleatorios vs aleatorios.
- Semillas.
- Función `random()` → Genera un número entre ≥ 0.0 y < 1.0
- Función `seed()` → Genera una semilla con la hora actual.
- Función `seed(x)` → Siendo `x` un valor entero, establece la semilla a dicho valor.
- Misma semilla genera misma secuencia de números.

MÓDULOS, PAQUETES Y PIP

- Módulo **random**:

- Función **randrange** toma el valor de un **range**.

- Función `randrange(fin)` → genera un entero ≥ 0 y $< fin$.
 - Función `randrange(ini, fin)` → genera un entero $\geq ini$ y $< fin$.
 - Función `randrange(ini, fin, inc)` → genera un entero $\geq ini$ y $< fin$ de un rango que crece *inc* unidades.

- Función **randint**(izquierda, derecha) toma el valor del entre los valores indicados, incluyendo ambos. Equivale a **randrange(ini, fin+1)**

MÓDULOS, PAQUETES Y PIP

- Módulo **random**:

- Función **choice**(secuencia) → Elige un elemento de la secuencia.
- Función **sample**(secuencia, n) → Devuelve un subconjunto de n elementos no repetidos de la secuencia. Por defecto el valor de n es 1.

MÓDULOS, PAQUETES Y PIP

- Módulo **platform**:

- Función **platform**(aliased = False, terse = False) → Muestra información sobre el S.O.
 - aliased → Muestra nombre alternativo (si es posible)
 - terse → Muestra una versión más breve (si es posible)
- Función **machine**() → Muestra el nombre genérico del procesador.
- Función **processor**() → Muestra el nombre real del procesador.

MÓDULOS, PAQUETES Y PIP

- Módulo **platform**:

- Función **system()** → Muestra el nombre genérico del sistema operativo.
- Función **versión()** → Muestra la versión del sistema operativo.
- Función **python_implementation()** → Muestra la implementación de Python.
- Función **python_version_tuple()** → Muestra la versión de Python (mayor, menor, parche)

MÓDULOS, PAQUETES Y PIP

PAQUETES

MÓDULOS, PAQUETES Y PIP

- Paquetes:
 - Un paquete es un agrupador lógico de módulos.



MÓDULOS, PAQUETES Y PIP

■ Paquetes:

- Al importar un módulo se genera la carpeta `__pycache__`.
 - Contiene ficheros con extensión **pyc** (compilados).
 - El nombre contiene información sobre el módulo importado, la implementación de Python y la versión. Ejemplo: se importa *modulo2* desde *modulo1*, generándose el fichero *modulo2.cpython-311.pyc* en la carpeta `__pycache__`.
 - La importación implica la ejecución del código “libre” → El uso de la variable `__name__` y el valor “`__main__`”
 - ¿Qué ocurre con `__name__` cuando no es el módulo principal? → Toma el valor del módulo.
 - Varios import iguales no generan varias importaciones.

MÓDULOS, PAQUETES Y PIP

- Paquetes:
 - Las variables definidas fuera de las funciones de un módulo son visibles desde los módulos que las utilizan.
 - Ocultar anteponiendo un guion bajo _ o dos __ al nombre de la variable → Es una convención, no impide el acceso.

MÓDULOS, PAQUETES Y PIP

- Paquetes:
 - Búsqueda de módulos → variable **path** del módulo **sys**.
 - Proporciona una lista.
 - La búsqueda se realiza en orden.
 - **Puede incluir archivos .zip.**
 - Se puede modificar agregando ubicaciones (con el método **append** de las listas o con el método **insert** si se quiere modificar el orden)

MÓDULOS, PAQUETES Y PIP

- Paquetes:

- Construcción de paquetes:

- Crear una estructura de directorios con los diferentes módulos.
 - Las referencias se harán como *paquete1.subpaquete2.modulo.función()*
 - Dentro de cada paquete debe hacer un fichero `__init__.py`
 - Al importar un módulo del paquete se ejecuta el contenido. Útil para inicialización. Puede estar sólo en el paquete raíz o en cualquier subpaquete de la jerarquía.
 - Puede estar vacío.

MÓDULOS, PAQUETES Y PIP

- Paquetes:

- Nota:

- Un módulo puede incluir esta sentencia
 - `#!/usr/bin/env python3`
 - Denominado shabang, shebang, hashbang, poundbang o hashpling.
 - Permite indicar a sistemas basados en Unix ó Linux como ejecutar un archivo Python.

MÓDULOS, PAQUETES Y PIP

PIP

MÓDULOS, PAQUETES Y PIP

■ PIP:

- PyPI (*Python Package Index*) es el repositorio de paquetes de Python. Se le conoce como “*The Cheese Shop*”.
- Es gratuito.
- Es mantenido por el *Packaging Working Group* dependiente de la *Python Software Foundation*.
- Sitio web: <https://pypi.org/>
- Los paquetes se instalan mediante la herramienta *pip* (pip instala paquetes).
- Existen versiones distintas para Python2 y Python3.

MÓDULOS, PAQUETES Y PIP

■ PIP:

- `pip --version` → Muestra la versión de pip.
- `pip help` → Ayuda.
- `pip help comando` → Ayuda.
- `pip list` → Muestra el listado de paquetes instalados.
- `pip show paquete` → Muestra información sobre el paquete indicado (nombre, versión, autor, sitio web, dependencias en las dos direcciones (*Requires* y *Required-by...*)).
 - NOTA: pip resuelve de forma automática las dependencias.

MÓDULOS, PAQUETES Y PIP

■ PIP:

- `pip search paquete` → No soportado. Alternativamente utilizar <https://pypi.org/search> a través de un navegador.
- `pip install paquete` → Instala el paquete para todos los usuarios.
- `pip install --user paquete` → Instala el paquete para el usuario activo en el sistema operativo.
- `pip install -U paquete` → Actualiza (update) el paquete.
- `pip install paquete==versión` → Instala una versión concreta.
- `pip uninstall paquete` → Desinstala un paquete.