MÓDULOS

E

BIBLIOTECAS

Joan Clarke (1917-1996)

"A veces, la persona a la que nadie imagina capaz de nada, es la que hace cosas que nadie imagina"

Joan Clarke (1917)
Descrifó los mensajes alemanes de Enigma.

- Informática do exército británico especializada en criptografía
- Conseguiu descifrar o código Enigma dos alemáns durante a 2ª guerra mundial
- Nomeada cabaleira da Orde do imperio británico en 1947

OpenWebinars

Módulos e bibliotecas

- Un módulo (ou biblioteca) é un conxunto de definicións de variables, funcións e tipos de datos (entre outras cousas) que poden importarse para usalas nos nosos programas.
- Tipos de módulos:
  - Módulos da biblioteca estándar de Python (incorporados no intérprete). Podes consultar o listado coa axuda en: https://docs.python.org/3/library/
  - Módulos dispoñibles en internet que se poden instalar
  - Os nosos propios módulos.

Acceso a funcionalidade dun módulo

- Temos tres opcións:
  - Importar o módulo en cuestión e acceder á función utilizando o nome do módulo (recomendable cando usamos funcións co mesmo nome en distintos módulos):
    - `>>> import math` # importa modulo math
    - `>>> print math.sin(2.0)` # visualiza o seno(2.0)
  - Importar a función do módulo e utilizala directamente:
    - `>>> from math import sin`
    - `>>> print sin(2.0)`
  - Importar todas as funcións do módulo e utilizalas directamente (usaremos esta opción):
    - `>>> from math import *`
    - `>>> print sin(2.0)`

Acceso a funcionalidade dun módulo

- A última opción será a utilizada habitualmente neste curso pola súa simplicidade.
- Se un programa utiliza moitos módulos, pode ocorrer que o mesmo nome de función estea en varios módulos e o programa non sabe a qué función se refire (úsase a función do último módulo que se invoca no programa). Para desfacer esta ambigüedade será preferible a primeira opción.

Algunhas bibliotecas estándar

- **`math`** : con diversas funcións matemáticas para números reais. Para números complexos e cambio de sistemas de coordenadas (cartesianas e polares) usa a o paquete **`cmath`** (https://docs.python.org/3/library/cmath.html).
- **`random`** : xera números pseudo-aleatorios para distintas distribucións.
- **`os`** : funcionalidade dependente do sistema operativo (máis información en https://docs.python.org/3/library/os.html).
- **`sys`** : proporciona acceso a funcionalidade usada polo intérprete (máis información en: https://docs.python.org/3/library/sys.html).
- **`pdb`** : para depurar un programa (máis información en https://docs.python.org/3/library/pdb.html).

Algunhas bibliotecas estándar

- **`string`** : proporciona un conxunto de variables e clases para o manexo de cadeas de caracteres (https://docs.python.org/3/library/string.html). A funcionalidade aparece como métodos do obxeto string definido en Python (ver métodos en https://docs.python.org/3/library/stdtypes.html#string-methods ou na presentación de programación orientada a obxetos).
- **`time`** : funcións relacionadas coa medida do tempo (máis información en: https://docs.python.org/3/library/time.html).
- **`tkinter`** : é un paquete para o desenvolvemento de interfaces de usuario (GUI) para Unix and Windows. Executando o comando **`python -m Tkinter`** podes comprobar se Tkinter está ben instalado no ordenador (axuda en https://docs.python.org/3/library/tkinter.html).

Funcions da biblioteca **`math`**

- Todas as funcións en https://docs.python.org/2/library/math.html

`sin(x), cos(x), tan(x), asin(x), atan(x), acos(x)`	Seno, coseno, tanxente, arcoseno, arcocoseno, arcotanxente de x
`sinh(x), cosh(x), tanh(x), asinh(x), atanh(x), acosh(x)`	Igual que o anterior con seno hiperbólico de x ...
`degrees(x), radians(x)`	Convirte o ángulo x de graos a radiáns e vice versa.
Constantes **`pi`** e **`e`**	Constante matemática pi=3.14 e o número e=2.7
`exp(x), log(x), log10(x)`	Función exponencial (e`**`x), logaritmo neperiano e logaritmo en base 10 de x.
`sqrt(x)`	Raiz cadrada de x.

Funcións da biblioteca `math`

`floor(x)`	Enteiro máis grande que sexa menor ou igual que o real x
`round(x)`	Enteiro máis próximo o real x
`ceil(x)`	Enteiro máis pequeno maior o igual que o real x
`trunc(x)`	Devolve a parte enteira do real x
`fabs(x)`	Devolve valor absoluto de x
`factorial(x)`	Factorial do enteiro positivo x
`isnan(x)`	Comproba se x é NaN (Not a Number)
`isinf(x)`	Comproba se x é infinito
`max(x1,x2,x3,...)`	Devolve máximo dos argumentos
`min(x1,x2,x3,...)`	Devolve mínimo dos argumentos
`choice(secuencia)`	Devolve un elemento aleatorio da secuencia

Funcions da biblioteca `random`

- Hai funcións para xerar números aleatorios e para seleccinar aleatoriamente elementos en secuencias. Mais información en: https://docs.python.org/3/library/random.html
  - **`seed(a=None)`** : inicializa o estado interno do xerador de números aleatorios. Con None ou sen argumento, o xerador inicializase normalmente co reloxio do sistema operativo.
  - **`random()`** : devolve un número aleatorio no intervalo [0.0, 1.0).
  - **`uniform(a, b)`** : devolve un número aleatorio real N tal que a <= N <= b for a <= b and b <= N <= a for b < a. Equivalente a **`a+(b-a)*random()`**.
  - **`randint(a, b)`** : devolve un número aleatorio enteiro N tal que a <= N <= b.
  - **`randrange(start, stop[, step])`**: devolve un elemento aleatoriamente seleccionado da secuencia xerada con **`range(start, stop, step)`**. Equivalente a **`choice(range(start, stop, step))`**.

Funcions da biblioteca `random`

- **`choice(seq)`** : devolve un elemento aleatorio dunha secuencia que non está baleira. Se a secuencia está baleira eleva a excepción **`IndexError`**.
- **`shuffle(x[, random])`** : baralla a secuencia x.
- **`sample(population, k)`** : devolve unha lista de lonxitude k con elementos aleatorio da secuencia population.

```
from random import *
print('x= ', random())         # numero aleatorio x, 0.0 <= x < 1.0
print('y= ', uniform(1, 10))   # numero aleatorio y, 1.0 <= y < 10.0
print('z= ', randint(1, 10))   # enteiro entre 1 e 10, ambos inclusive
print('Enteiro par: ', randrange(0, 101, 2))  # enteiro par entre 0 e 100
print('Enteiro par: ', choice(range(0,101, 2)))
print('Elemento aleatorio: ', choice('abcdefghij'))  # elixe un elemento aleatorio nunha secuencia
elementos = [1, 2, 3, 4, 5, 6, 7] # define unha lista
print('Elemento aleatorio: ', choice(elementos))
print('Elementos: ', elementos)
shuffle(elementos)  # baralla os elementos dunha lista
print('Elementos barallados: ', elementos)
print('Elementos dunha lista: ', sample(elementos,  3))  # elixe 3 elementos na lista
```

Funcions da biblioteca **`sys`**

- **`sys.argv`** : a lista de argumentos da liña de comandos dun programa python. `argv[0]` é o nome do programa, `argv[1]` o primeiro argumento, ...
- **`sys.maxint`** devolve o enteiro máis grande que usa python.
- **`sys.maxsize`** devolve a lonxitude máxima dos contedores de python.
- **`sys.platform`** contén o tipo de plataforma usada.
- Función **`exit()`** finaliza un programa.

Funcions da biblioteca `os`

- Funcións para traballar de xeito portable con arquivos e directorios:
  - **`access(path, modo_de_acceso)`**: saber se se pode acceder ó arquivo ou directorio path.
  - **`getcwd()`**: devolve o directorio actual.
  - **`chdir(novo_path)`**: cambia de directorio de traballo a novo_path.
  - **`mkdir(path[, modo])`**: crea un directorio de nome path.
  - **`rmdir(path)`**: elimina o directorio path.
  - **`rename(actual, novo)`**: renomea un arquivo.
  - **`listdir(path)`**: contido do directorio path.

Funcions da biblioteca `os`

- Contén o diccionario **`environ`** coas variables de entorno relativas ó sistema:

```
import os
for variable, valor in os.environ.iteritems():
    print "%s: %s" % (variable, valor)
```

- O submódulo **`os.path`** (https://docs.python.org/3/library/os.path.html) permite acceder a funcionalidades relacionadas co nome dos arquivos e directorios:
  - **`exists(path)`**: devolve true se existe path e False en caso contrario.
  - **`isfile(path)`**: devolve true se path é un arquivo e false en caso contrario.
  - **`isdir(path)`**: devolve true se path é un directorio e false en caso contrario.

Funcions da biblioteca `os`

```
from os import *
actualdir= getcwd()
print('Directorio actual: ', actualdir)
print("Acceso: ", access(actualdir, R_OK)) # este directorio ten acceso de lectura?
chdir(actualdir) # cambia de directorio
print('Directorio actual: ', getcwd())
novodir='tmp'
mkdir(novodir) # mira que no directorio actual se creou o directorio
chdir(novodir) # cambia de directorio
print('Directorio: ', getcwd())
chdir(actualdir)
print('Directorio despois de cambiar dir: ', getcwd())
rename(novodir, 'tmp2') # cambia nome o directorio
print("Directorio despois de cambiar nome: ", listdir(getcwd()))
rmdir('tmp2') # elimina directorio

from os.path import *
print('Existe dir: ', exists(actualdir))
print('Existe dir: ', exists('tmp2'))
print('E un dir?: ', isdir(actualdir))
print('E un arquivo?: ', isfile(actualdir))
print('E un arquivo?: ', isfile('modulo os.py'))
```

Funcions da biblioteca `time`

- **`time()`**: devolve un número real co tempo en segundos transcurrido desde a época (o punto onde empeza o tempo e pode ser diferente para distintos sistemas).
- **`gmtime([secs])`**: convirte o tempo expresado en segundos a un formato de ano, mes, día. A sentencia gmtime(0) devolve a época.
- **`localtime([secs])`**: igual que **`gmtime()`**, pero se non se proporciona argumentos devolve o tempo actual.
- **`sleep(secs)`**: suspende a execución do programa durante secs segundos.
- **`process_time()`**: devolve o tempo de uso do procesador desde a última vez que se invocou a función **`process_time()`**.

Funcions da biblioteca `time`

```
from time import *
print('Tempo da epoca: ', gmtime(0))
print('Tempo actual: ', time(), ' segundos')
print('Tempo actual: ', localtime())
inicio = time()
sleep(2) # para o programa 2 segundos
final = time()
# asi contabiliza o tempo que o programa esta parado
print('Tempo transcurrido bloque de codigo: ', final-inicio)
# en unix
inicio = process_time()
sleep(1) # para o programa 1 segundos
final = process_time()
# Non contabiliza o tempo que temos parado o programa
# Esta opcion e a que se usa para medir o tempo de execucion dun algoritmo
print('Tempo de uso do procesador do bloque de codigo: ', final-inicio, ' segundos')
```

Módulos que non están na biblioteca estándar

- Paquetes externos a python:
  - **`Matplotlib`**: paquete de python para gráficos.
  - **`NumPy`**: paquete de cálculo científico en Python. Proporciona vectores e matrices con funcións para a súa manipulación, transformadas de Fourier, álxebra linear, operacións estatísticas básicas, etc.
  - **`Scipy`**: un Numpy avanzado (https://scipy.org/scipylib/).
  - **`Sympy`**: paquete para o cálculo simbólico.
  - **`PyGTK`**: paquete para o desenvolvemento avanzado de interfaces de usuario (GUI) (https://pygobject.readthedocs.io/en/latest/).

Creación de módulos

- Un módulo sinxelo é simplemente un arquivo **`.py`** (meuModulo.py) onde o nome do arquivo indica o nome do módulo.
- Para usar **`meuModulo.py`** dentro dun programa, o arquivo debe estar no mesmo directorio.
- O uso dun módulo definido polo usuario é igual ó dun módulo de python.