

Clase Número: 12

Biblioteca estándar de Python

Todos los conceptos que hemos visto hasta ahora los hemos resuelto utilizando solo la sintaxis que nos provee Python y un conjunto de funciones básicas que se incluyen automáticamente como por ejemplo son print, range, len etc.

En Python se incluye una biblioteca extra de funciones, variables, clases etc. que nos facilitan la resolución de problemas en una gran diversidad de áreas como matemáticas, estadísticas, compresión de datos, internet, interfaces visuales etc.

Veremos en este concepto como se importa un módulo de la biblioteca estándar y como se accede a su funcionalidad.

Problema 1:

Confeccionar un programa que simule tirar dos dados y luego muestre los valores que salieron. Imprimir un mensaje que ganó si la suma de los mismos es igual a 7.

Para resolver este problema requerimos un algoritmo para que se genere un valor aleatorio entre 1 y 6. Como la generación de valores aleatorios es un tema muy frecuente la biblioteca estándar de Python incluye un módulo que nos resuelve la generación de valores aleatorios.

Programa:

```
import random

dadol=random.randint(1,6)
dado2=random.randint(1,6)
print("Primer dado:",dadol)
print("Segundo dado:",dado2)
suma=dadol+dado2
if suma==7:
    print("Gano")
else:
    print("Perdio")
```

Para importar un módulo de la biblioteca estándar de Python utilizamos la palabra clave import seguida por el nombre del módulo que necesitamos importar:

```
import random
```

Como dijimos la biblioteca estándar de Python se instala junto con Python.

Si disponemos un nombre de módulo inexistente aparecerá un error:

```
Traceback (most recent call last):
  File "C:/programaspython/ejercicio179.py", line 1, in
    import ran
```

```
ImportError: No module named 'ran'
```

Para acceder a todas las funciones contenidas en el módulo random es necesario primero importar el módulo y luego dentro del algoritmo de nuestro programa anteceder primero el nombre del módulo y seguidamente la función que queremos acceder:

```
dadol=random.randint(1,6)
```

Si tratamos de acceder directamente al nombre de la función sin disponer el nombre del módulo se produce un error:

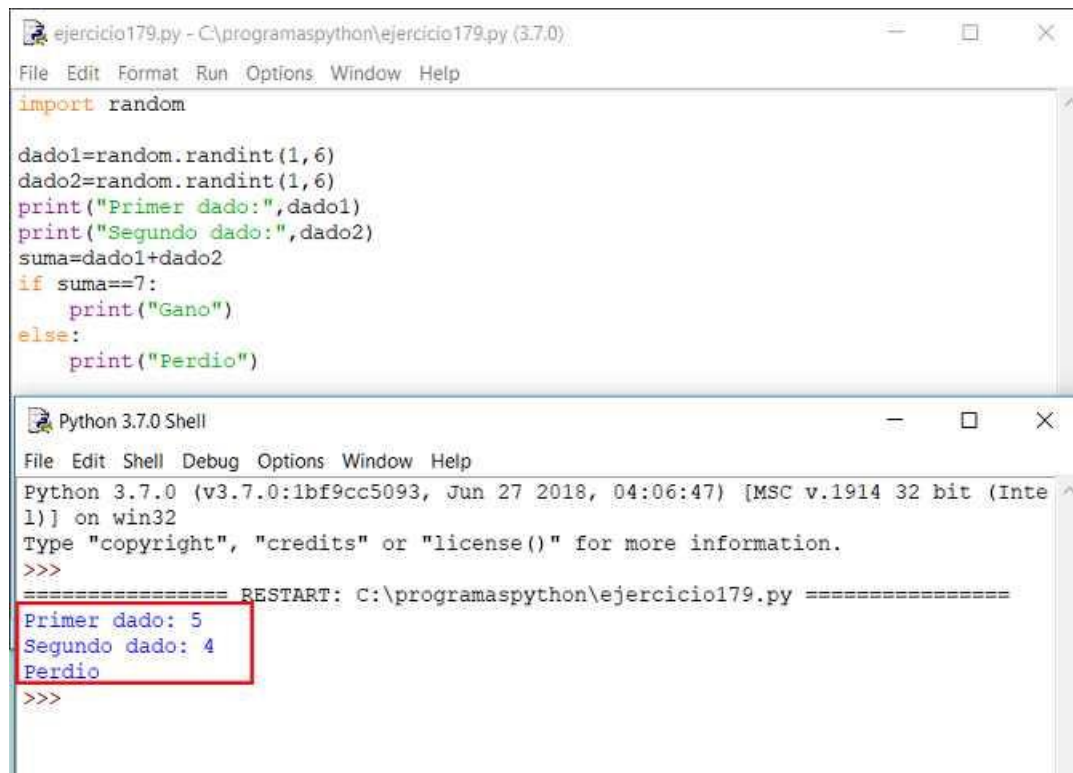
```
Traceback (most recent call last):
  File "C:/programaspython/ejercicio179.py", line 3, in
    dadol=randint(1,6)
NameError: name 'randint' is not defined
```

Este error se produce porque la función randint no es una función integrada en Python como print, range, len etc.

Entonces la sintaxis para acceder a la funcionalidad de un módulo requiere que dispongamos primero el nombre del módulo y seguidamente el nombre de la función.

Como podemos imaginar la función randint retorna un valor aleatorio comprendido entre los dos valores indicados en los parámetros.

La ejecución del programa tiene una salida similar a esta:



The screenshot shows a Python IDE window titled 'ejercicio179.py - C:\programaspython\ejercicio179.py (3.7.0)'. The code in the editor is as follows:

```
import random

dadol=random.randint(1,6)
dado2=random.randint(1,6)
print("Primer dado:",dadol)
print("Segundo dado:",dado2)
suma=dadol+dado2
if suma==7:
    print("Gano")
else:
    print("Perdio")
```

Below the editor is the 'Python 3.7.0 Shell' window. It shows the execution output:

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\programaspython\ejercicio179.py =====
Primer dado: 5
Segundo dado: 4
Perdio
>>>
```

The output lines 'Primer dado: 5', 'Segundo dado: 4', and 'Perdio' are highlighted with a red box.

Problema 2:

Desarrollar un programa que cargue una lista con 10 enteros.

Cargar los valores aleatorios con números enteros comprendidos entre 0 y 1000.

Mostrar la lista por pantalla.

Luego mezclar los elementos de la lista y volver a mostrarlo.

Programa:

```
import random

def cargar():
    lista=[]
    for x in range(10):
        lista.append(random.randint(0,1000))
    return lista

def imprimir(lista):
    print(lista)

def mezclar(lista):
    random.shuffle(lista)

# bloque principal

lista=cargar()
print("Lista generada aleatoriamente")
imprimir(lista)
mezclar(lista)
print("La misma lista luego de mezclar")
imprimir(lista)
```

No hay ningún problema de llamar a las funciones de un módulo dentro de una función propia siempre y cuando indiquemos el import respectivo:

```
import random

def cargar():
    lista=[]
    for x in range(10):
        lista.append(random.randint(0,1000))
    return lista
```

El módulo random cuenta con otra función llamada shuffle que le pasamos como parámetro una lista y nos la devuelve con los elementos mezclados (pensemos esto nos podría servir si estamos desarrollando un juego de naipes y necesitamos mezclarlos):

```
def mezclar(lista):
    random.shuffle(lista)
```

Desde el bloque principal procedemos a llamar a las funciones que hemos codificado:

```
# bloque principal

lista=cargar()
print("Lista generada aleatoriamente")
imprimir(lista)
```

```
mezclar(lista)
print("La misma lista luego de mezclar")
imprimir(lista)
```

Problemas propuestos

1 - Confeccionar un programa que genere un número aleatorio entre 1 y 100 y no se muestre.

El operador debe tratar de adivinar el número ingresado.

Cada vez que ingrese un número mostrar un mensaje "Gano" si es igual al generado o "El número aleatorio es mayor" o "El número aleatorio es menor".

Mostrar cuando gana el jugador cuantos intentos necesitó.

2 - Confeccionar una programa con las siguientes funciones:

- a) Generar una lista con 5 elementos enteros aleatorios comprendidos entre 1 y 3.
- b) Controlar que el primer elemento de la lista sea un 1, en el caso que haya un 2 o mezclar la lista y volver a controlar hasta que haya un 1.
- c) Imprimir la lista.

Solución al problema

```
import random

intentos=0
aleatorio=random.randint(1,100)
elegido=-1
print("Intenta adivinar el numero que pense entre 1 y 100")
while (elegido!=aleatorio):
    elegido=int(input("Cual numero elige?"))
    if aleatorio>elegido:
        print("Pense un valor mayor")
    else:
        if aleatorio<elegido:
            print("Pense un valor menor")
        intentos=intentos+1

print("Ganaste en",intentos,"intentos")
```

ejercicio182.py

```
import random

def cargar():
    lista=[]
    for x in range(5):
        lista.append(random.randint(1,3))
    return lista

def controlar_primer(lista):
    while lista[0]==1:
        random.shuffle(lista)

def imprimir(lista):
    print(lista)

# bloque principal

lista=cargar()
imprimir(lista)
controlar_primer(lista)
imprimir(lista)
```

Importar algunas funcionalidades de un módulo de la biblioteca estándar de Python

Hemos visto que para importar toda la funcionalidad de un módulo de la Biblioteca estándar de Python utilizamos la palabra clave `import` y seguidamente el nombre del módulo:

```
import random
```

Con esa sintaxis todas las funcionalidades del módulo "random" pueden ser accedidas desde nuestro módulo.

Ahora veremos que en Python tenemos otra sintaxis para las situaciones que queremos acceder a una o pocas funcionalidades de un módulo.

Por ejemplo si queremos acceder solo a la función `randint` del módulo `random` en Python lo podemos expresar con la siguiente sintaxis:

```
from random import randint
```

Utilizamos la palabra clave `from` y seguidamente el nombre del módulo de donde queremos importar funcionalidades del mismo. Luego indicamos la palabra clave `import` y la funcionalidad que queremos importar, en nuestro ejemplo la función `randint`.

También cambia como utilizamos la función `randint` dentro de nuestro módulo:

```
valor=randint(1,10)
print(valor)
```

Como vemos no le antecedemos ningún nombre de módulo y hacemos referencia directamente a la función importada.

Si necesitamos importar más de una funcionalidad de un módulo debemos separar por comas las funcionalidades importadas:

```
from random import randint,shuffle
```

Problema 1:

Confeccionar un programa que solicite la carga de un valor entero por teclado y luego nos muestre la raíz cuadrada del número y el valor elevado al cubo.

Para resolver este problema utilizaremos dos funcionalidades que nos provee el módulo `math` de la biblioteca estándar de Python. Podemos consultar el módulo `math`

Programa:

```
from math import sqrt, pow

valor=int(input("Ingresa un valor entero:"))
```

```
r1=sqrt(valor)
r2=pow(valor,3)
print("La raiz cuadrada es",r1)
print("El cubo es",r2)
```

El módulo math tiene dos funciones llamadas sqrt (para obtener la raíz cuadrada) y la función pow para elevar un valor a cierta potencia.

Utilizamos la sintaxis para importar solo dichas dos funcionalidades del módulo math:

```
from math import sqrt, pow
```

Una vez importadas las dos funciones podemos hacer uso de las mismas en nuestro programa indicando directamente su nombre:

```
r1=sqrt(valor)
```

Lo mismo para llamar la función pow:

```
r2=pow(valor,3)
```

Definición de alias para una funcionalidad

Podemos definir un nombre distinto para una funcionalidad que importamos de otro módulo. Esto puede tener como objetivo que nuestro programa sea más legible o evitar que un nombre de función que importamos colisione con un nombre de función de nuestro propio módulo.

Resolveremos el mismo problema anterior pero definiendo dos alias para las funciones sqrt y pow del módulo math.

Programa:

```
from math import sqrt as raiz, pow as elevar

valor=int(input("Ingrese un valor entero:"))
r1=raiz(valor)
r2=elevar(valor,3)
print("La raiz cuadrada es",r1)
print("El cubo es",r2)
```

Como vemos para definir un alias a una funcionalidad que importamos de un módulo debemos disponer la palabra clave as seguida del nuevo nombre:

```
from math import sqrt as raiz, pow as elevar
```

Luego para utilizar la funcionalidad que importamos debemos hacerlo mediante el alias y no con el nombre definido en el módulo que importamos:

```
r1=raiz(valor)
r2=elevar(valor,3)
```

Problema propuesto

1 -Calcular el factorial de un número ingresado por teclado.

El factorial de un número es la cantidad que resulta de la multiplicación de determinado número natural por todos los números naturales que le anteceden excluyendo el cero. Por ejemplo el factorial de 4 es 24, que resulta de multiplicar $4*3*2*1$.

No hay que implementar el algoritmo para calcular el factorial sino hay que importar dicha funcionalidad del módulo math.

El módulo math tiene una función llamada factorial que recibe como parámetro un entero del que necesitamos que nos retorne el factorial.

Solo importar la funcionalidad factorial del módulo math de la biblioteca estándar de Python.

Solución al problema

```
from math import factorial

valor=int(input("Ingrese un valor:"))
resu=factorial(valor)
print("El factorial de",valor,"es",resu)
```