Clase 3_1_modulos

April 1, 2022

0.0.1 Seminario de Lenguajes - Python

0.1 Cursada 2022

0.1.1 Módulos en Python

1 Un módulo es un archivo (con extensión .py) que contiene sentencias y definiciones.

1.1 ¿Algo nuevo?

2 Analicemos un ejemplo en el IDE

- ¿Cuántos archivos forman mi programa?
- ¿Cómo se relacionan?

3 La sentencia import

- Permite acceder a funciones y variables definidas en otro módulo.

[ ]:
```
import random
```

- Sólo se importa las definiciones de las funciones. No las ejecuta.
- Para ejecutar una función **debo invocarla en forma explícita**.

[ ]:
```
random.randrange(10)
```

- ¿Qué son los archivos .pyc?
- +Info en el sitio oficial
- O podemos leer la PEP 3147

4 Espacios de nombres

- Un espacio de nombres **relaciona nombres con objetos**.
- En principio podemos pensar en tres espacios de nombres:
  - Locales
  - Globales
  - ___builtins___

- Los espacios de nombres **se crean en diferentes momentos y tienen distintos tiempos de vida**.
- El espacio de nombres que contiene los nombres **___builtins___** se crea al iniciar el intérprete y nunca se elimina.
- El espacio de nombres **global** de un módulo se crea cuando se lee la definición de un módulo y normalmente también dura hasta que el intérprete finaliza.
- El espacio de nombres **local** a una función se crea cuando la función es llamada y se elimina cuando la función retorna.

5 Espacios de nombres y módulos

- Cada módulo tiene su propio espacio de nombres.

```
#funciones
var_x = 10
def print_var_x():
    print(var_x)

#uso_funciones
import funciones

var_x = 20
funciones.print_var_x()
```

- Entonces, ¿qué valor imprime este código?

6 Volvemos al import

- Cuando usamos la sentencia **import** se actualizan los espacios de nombres.

```
import mi_modulo
```

- En este caso, todos los ítems definidos dentro de **mi_modulo** serán locales a **mi_modulo**.
- Lo que se agrega al espacio de nombres es el nombre del módulo (**mi_modulo**).
- Para usarlo debo hacerlo con notación puntual.

```
import mi_modulo as m
```

- ¿Y acá?¿Qué nombre se agrega al espacio de nombres?

7 Importando módulos

```
import funciones

funciones.uno()
```

- La importación se realización **sólo una vez por sesión del intérprete**.
- Veamos sobre el ejemplo anterior.

- Si necesito volver a importar: usar **reload()** incluido en el **módulo importlib**.

```
import importlib
importlib.reload(funciones)
```

- +Info en la documentación oficial

8 Otra forma de importar

```
from mi_modulo import una_funcion
```

- Sólo se importa **una_funcion** de **mi_modulo** (no el nombre del módulo).
- El único nombre que se agrega al espacio de nombres es **una_funcion**.

```
from mi_modulo import *
```

- En este caso, **todos los ítems** definidos en **mi_modulo** formarán parte del espacio de nombres actual.
- **Esta forma no está recomendada**: podrían existir conflictos de nombres.

8.0.1 Veamos qué pasa cuando queremos importar un módulo que no existe:

[ ]:
```
import pp
```

8.0.2 ¿Dónde se busca los módulos?

- Directorio actual + otros directorios definidos en la variable de ambiente PYTHONPATH

[ ]:
```
import sys
sys.path
```

9 Biblioteca de módulos estándar

- Existe un conjunto de módulos que vienen incluidos con el intérprete.
- Para usarlos se los debe importar en forma explícita (usando **sentencia import**).
- Ya usamos algunos, ¿cuáles?
- random, sys, string
- Hay muchos otros que nos ofrecen distinta funcionalidad:
  - Ejemplos: **math**, **random**, **time**, **sys**, **collections**, etc.

[ ]:
```
import math
import random

print(math.gcd(12,16))
print(math.sqrt(81))
print(math.pi)

lista = [1, 2, 3, 4]
```

```
print(random.choice(lista))
```

10 El módulo collections

- Define tipos de datos alternativos a los predefinidos dict, list, set, y tuple.
- **Counter**: es una colección desordenada de pares claves-valor, donde las claves son los elementos de la colección y los valores son la cantidad de ocurrencias de los mismos.

[ ]:
```
from collections import Counter
cnt = Counter(['red', 'blue', 'red', 'green', 'blue', 'blue'])
print(cnt)
```

[ ]:
```
print(Counter('abracadabra').most_common(1))
```

11 Para probar luego

- El módulo: **deque** ("double-ended queue")
  - Permite implementar pilas y colas.

[ ]:
```
from collections import deque
d = deque('abcd')
# agrega al final
d.append("final")
# agrega al principio
d.appendleft("ppio")
# eliminar último
elem = d.pop()
# elimina primero
elem1=d.popleft()
d
```

12 El módulo sys

- Entre otras cosas, define:
  - **exit([arg]):** sale del programa actual;
  - **path:** las rutas donde buscar los módulos a cargar;
  - **platform:** contiene información sobre la plataforma.

[ ]:
```
import sys
#sys.path
sys.platform
```

13 Los nombres de los módulos

- Es posible acceder al nombre de un módulo a través de **___name___**

[ ]:
```
print(__name__)
```

14 Tarea

- Averiguar cuándo un módulo se denomina **___main___**,