



**Universidad Tecnológica
del Norte de Guanajuato**

Organismo Público Descentralizado del Gobierno del Estado

“Educación y progreso para la vida”

Unidad 2

Recuperación 1

Zuñiga Olvera Jorge Daniel

Programación de redes

Evidencias

Base de datos

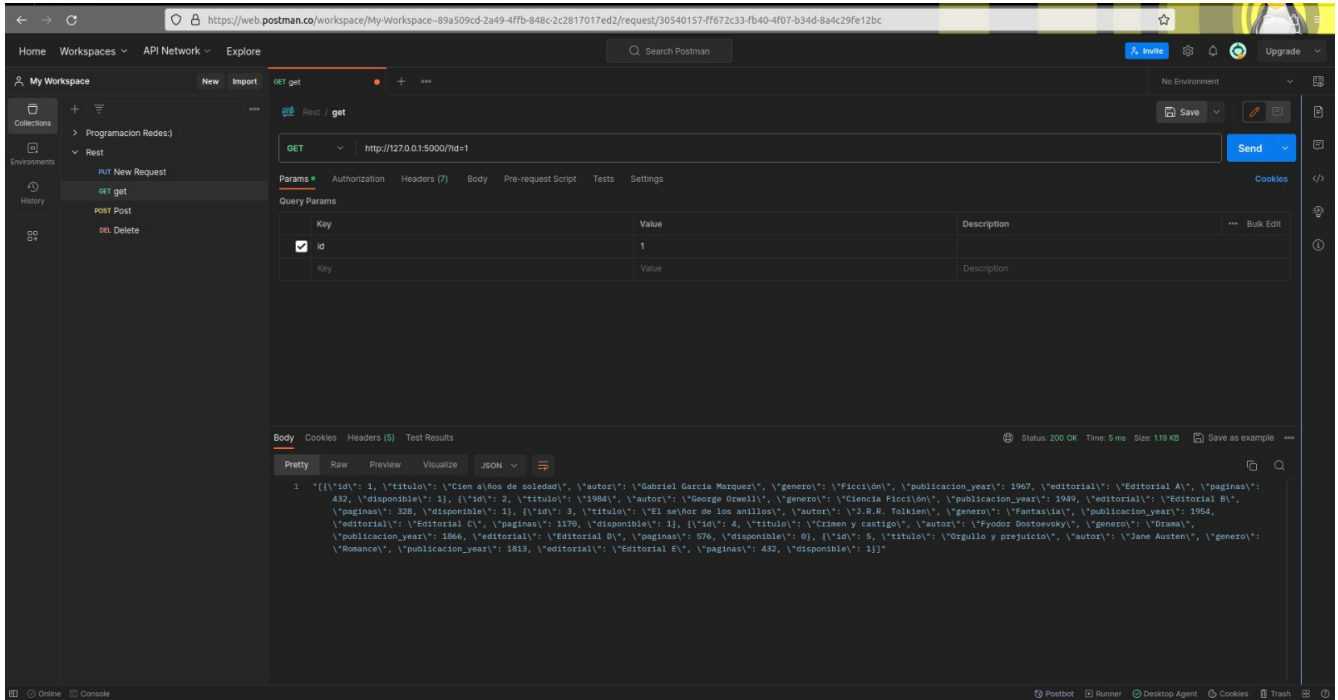
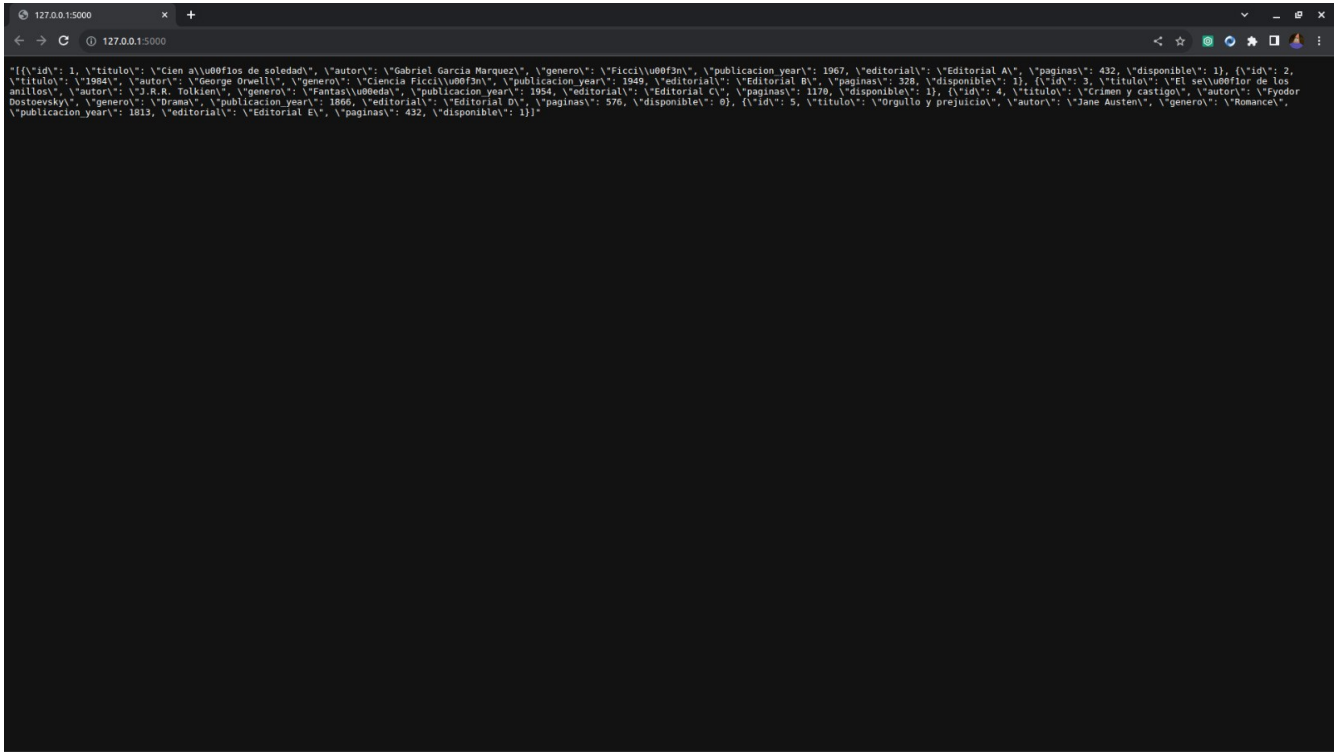
```
sqlite> INSERT INTO Libros (id, titulo, autor, genero, publicacion_year, editorial, paginas, disponible)
...> VALUES
...> (1, 'Cien años de soledad', 'Gabriel Garcia Marquez', 'Ficción', 1967, 'Editorial A', 432, 1),
...> (2, '1984', 'George Orwell', 'Ciencia Ficción', 1949, 'Editorial B', 328, 1);
sqlite> INSERT INTO Libros (id, titulo, autor, genero, publicacion_year, editorial, paginas, disponible)
...> VALUES
...> (3, 'El señor de los anillos', 'J.R.R. Tolkien', 'Fantasía', 1954, 'Editorial C', 1170, 1),
...> (4, 'Crimen y castigo', 'Fyodor Dostoevsky', 'Drama', 1866, 'Editorial D', 576, 0),
...> (5, 'Orgullo y prejuicio', 'Jane Austen', 'Romance', 1813, 'Editorial E', 432, 1);
sqlite>
```

```
sqlite> CREATE TABLE Libros (
...> id INTEGER PRIMARY KEY,
...> titulo TEXT NOT NULL,
...> autor TEXT NOT NULL,
...> genero TEXT,
...> publicacion_year INTEGER,
...> editorial TEXT,
...> paginas INTEGER,
...> disponible BOOLEAN NOT NULL CHECK (disponible IN (0, 1))
...> );
```


```
sqlite> CREATE TABLE Prestamos (
...> id INTEGER PRIMARY KEY,
...> libro_id INTEGER NOT NULL,
...> fecha_prestamo DATE NOT NULL,
...> fecha_devolucion DATE,
...> multa REAL DEFAULT 0,
...> FOREIGN KEY (libro_id) REFERENCES Libros(id),
...> CHECK (multa >= 0)
...> );
```

```
sqlite> INSERT INTO Prestamos (id, libro_id, fecha_prestamo, fecha_devolucion, multa)
...> VALUES
...> (1, 1, '2023-01-10', '2023-01-20', 0),
...> (2, 2, '2023-02-15', '2023-03-01', 0),
...> (3, 3, '2023-03-20', '2023-04-05', 0),
...> (4, 4, '2023-04-25', NULL, 0),
...> (5, 5, '2023-05-30', NULL, 0);
sqlite>
```

```
sqlite> SELECT * FROM Libros;
1|Cien años de soledad|Gabriel Garcia Marquez|Ficción|1967|Editorial A|432|1
2|1984|George Orwell|Ciencia Ficción|1949|Editorial B|328|1
3|El señor de los anillos|J.R.R. Tolkien|Fantasía|1954|Editorial C|1170|1
4|Crimen y castigo|Fyodor Dostoevsky|Drama|1866|Editorial D|576|0
5|Orgullo y prejuicio|Jane Austen|Romance|1813|Editorial E|432|1
sqlite> SELECT * FROM Prestamos;
1|1|2023-01-10|2023-01-20|0.0
2|2|2023-02-15|2023-03-01|0.0
3|3|2023-03-20|2023-04-05|0.0
4|4|2023-04-25||0.0
5|5|2023-05-30||0.0
```




Exámenes


**PYTHON INSTITUTE**
Open Education & Development Group

Examen Módulo 1

● Online

Progress (100%)





Your score: **18/18**

100%


Congratulations, you've passed the test!

SECTION ANALYSIS

PE2 -- Module 1 Test	100%
----------------------	------

[Retake Test](#)


[Review Test](#)


**PYTHON INSTITUTE**
Open Education & Development Group

Examen Módulo 2

● Online

Progress (100%)





Your score: **15/15**

100%



Congratulations, you've passed the test!

SECTION ANALYSIS

PE2 -- Module 2 Test	100%
----------------------	------

[Retake Test](#)

[Review Test](#)

22:23

Browser tabs: (2) Yoss Bones x Akapellah PLAYING, Course: Programación de R, Edube Interactive :: Examen, Plataforma Educativa Instituto X

Address bar: <https://edube.org/quiz/python-essentials-2-esp/pe2-module-3-test-4>

PYTHON INSTITUTE
Open Education & Development Group

Examen Módulo 3

Progress (100%)

Your score: 16/16

100%

Congratulations, you've passed the test!

SECTION ANALYSIS

PE2 -- Module 3 Test	100%
----------------------	------

[Retake Test](#) [Review Test](#)

System tray: 22:47

Browser tabs: (2) Aleman - Rolemos Otro, DH-PDR/TSU: Unidad 2 Rec, Course: Programación de R, Edube Interactive :: Examen, Plataforma Educativa Instituto X

Address bar: <https://edube.org/quiz/python-essentials-2-esp/pe2-module-4-test-4>

PYTHON INSTITUTE
Open Education & Development Group

Examen Módulo 4

Progress (100%)

Your score: 16/16

100%

Congratulations, you've passed the test!

SECTION ANALYSIS

PE2 -- Module 4 Test	100%
----------------------	------

[Retake Test](#) [Review Test](#)

System tray: 23:09

Resúmenes

1.1.1.11

Importar módulos en Python puede hacerse de varias formas. Puedes importar un módulo completo o entidades específicas de él. Al importar, es importante considerar el peligro de conflictos de nombres. Se puede cambiar el nombre de la entidad importada usando "as". Importar todas las entidades con "*" no se recomienda debido a posibles conflictos. Ejercicios prácticos incluyen invocar la función `make_money` desde el módulo `mint` de varias maneras, considerando la forma de importación utilizada.

Ejercicios:

1. `import mint` → `mint.make_money()`
2. `from mint import make_money` → `make_money()`
3. `from mint import make_money as my_make_money` → `my_make_money()`
4. `from mint import *` → `make_money()` (No recomendado debido a posibles conflictos de nombres).

1.2.1.17

La función `dir()` revela las entidades de un módulo, como en `import os; dir(os)`, proporcionando una visión general útil para su uso en el código. El módulo `math` ofrece más de 50 funciones y constantes, facilitando operaciones matemáticas y proporcionando valores clave como π y la constante de Euler, e . El módulo `random` ofrece más de 60 entidades para generar números pseudoaleatorios, destacando la naturaleza "pseudo" de la aleatoriedad en la computación. El módulo `platform` (con cerca de 70 funciones) permite explorar las capas subyacentes del sistema operativo y hardware, brindando información valiosa sobre el entorno de ejecución del código. El Índice de Módulos de Python es un recurso comunitario en <https://docs.python.org/3/py-modindex.html> para encontrar módulos adecuados a tus necesidades.

Ejercicios:

1. `result` será `True` ya que `math.e` es la base del logaritmo natural y es igual a `math.exp(1)`.
2. Establecer la semilla del generador con el mismo valor garantiza la reproducibilidad de resultados al generar números pseudoaleatorios.
3. Para determinar el nombre del CPU, podrías usar la función del módulo `platform`: `platform.processor()`.
4. El resultado esperado es 3, ya que `platform.python_version_tuple()` devuelve una tupla de tres elementos representando la versión de Python.

1.3.1.11

Un módulo agrupa entidades relacionadas, mientras que un paquete es un contenedor que organiza varios módulos bajo un nombre común, distribuyéndose como archivos o en un archivo zip. Python traduce el código fuente de un módulo a un formato semi-compilado durante la primera importación, almacenándolo en archivos `pyc` en el directorio **pycache**. Marcar entidades como privadas en un módulo se logra con prefijos `_` o `__`, una recomendación más que una orden. El shebang (`#!`) en el

inicio de archivos Python indica a sistemas Unix cómo ejecutar el script. No afecta a MS Windows. Para que Python reconozca un directorio de paquete no estándar, debes agregarlo a la variable path en el módulo sys. El archivo **init.py** se ejecuta al importar un paquete, inicializando el paquete y sus subpaquetes, si existen.

Ejercicios:

1. Evita que tu código sea ejecutado como un script ordinario marcando el código con:

```
python
```

```
if __name__ == "__main__":  
    # código solo para ejecución directa
```

2. Para recorrer el directorio D:\Python\Project\Modules en busca de módulos, puedes usar:

```
python
```

```
import os
```

```
module_directory = "D:\\Python\\Project\\Modules"  
module_files = [f for f in os.listdir(module_directory) if f.endswith(".py")]
```

3. Para importar todas las entidades de mymodule, después de agregar D:\Python\Project\Modules a sys.path:

```
python
```

```
from abc.def import mymodule
```

1.4.1.18

El repositorio para compartir código Python es conocido como Python Package Index (PyPI) o The Cheese Shop. Su sitio web es <https://pypi.org/>. La herramienta para gestionar paquetes en PyPI es pip. Debes instalarlo manualmente, ya que puede no ser parte de la instalación estándar de Python. Pip es una herramienta de consola. Para verificar la versión de pip, se utilizan los comandos **pip --version** o **pip3 --version**. La elección depende del entorno del sistema operativo. Las principales actividades de pip incluyen mostrar ayuda, listar paquetes instalados, mostrar información de un paquete, buscar paquetes en PyPI, instalar, actualizar y desinstalar paquetes. Para instalar un paquete en todo el sistema, se utiliza **pip install nombre** (puede requerir privilegios de administrador).

Ejercicios:

1. El nombre "The Cheese Shop" proviene de un sketch cómico de Monty Python en el que un cliente visita una tienda que no vende queso, a pesar de su nombre.
2. Deberías asegurarte de cuál es el pip correcto para ti porque algunos sistemas utilizan **pip** y otros **pip3** para Python 3. Verificar cuál es compatible con tu versión de Python es esencial.
3. Puedes determinar si tu pip funciona con Python 2 o Python 3 usando los comandos **pip --version** y **pip3 --version**. La versión asociada te indicará la compatibilidad.
4. Si no tienes privilegios de administrador, debes usar **pip install --user nombre** para instalar el paquete solo para ti, sin afectar a otros usuarios del sistema.

2.1.1.4

Las computadoras almacenan caracteres como números, y dos formas comunes de codificación son ASCII (para el alfabeto latino) y UNICODE (que abarca casi todos los alfabetos humanos). El número que representa un carácter se llama punto de código. UNICODE utiliza diversas formas de codificación, como UCS-4 y UTF-8, siendo esta última la más común por su eficiencia en el uso de espacio de memoria.

Ejercicios:

1. BOM (Byte Order Mark) es un marcador de orden de bytes, un conjunto de bytes al inicio de un archivo que indica el orden de bytes de su codificación, como UTF-8 o UTF-16.
2. Sí, Python 3 está internacionalizado y tiene soporte nativo para UNICODE, facilitando el trabajo con varios alfabetos y caracteres en distintos idiomas.

2.2.1.15

Las cadenas en Python son secuencias inmutables con dos tipos: de una línea ('cadena') y multilínea ('''cadena''' o """"cadena"""). La longitud de una cadena se obtiene con len(). El carácter de escape () no cuenta en la longitud. Concatenación y replicación de cadenas se logran con + y * respectivamente. Las funciones chr() y ord() crean caracteres y obtienen puntos de código, y cumplen con ciertas expresiones. Otras funciones útiles incluyen list(), max(), y min(). El método index() encuentra el índice de una subcadena.

Ejercicios:

Ejercicio 1: La longitud de """"cadena"""" es 0, ya que solo contiene tres comillas sin ningún otro carácter.

Ejercicio 2: El resultado esperado es ['t', 'e', 's'], ya que la rebanada [3:6] incluye caracteres en esos índices. Ejercicio 3: El resultado esperado es "bcd", ya que el código incrementa en 1 el punto de código de cada carácter en "abc".

2.3.1.17

Métodos útiles para cadenas incluyen capitalize(), center(), count(), join(), lower(), lstrip(), replace(), rfind(), rstrip(), split(), strip(), swapcase(), title(), y upper(). Métodos para determinar el contenido de cadenas incluyen endswith(), isalnum(), isalpha(), islower(), isspace(), isupper(), y startswith(), devolviendo valores booleanos.

Ejercicios:

1. Ejercicio 1: El resultado esperado es "ABC123xyx", ya que se invierten las mayúsculas y minúsculas.
2. Ejercicio 2: El resultado esperado es "las", ya que se imprime la palabra en la penúltima posición de la cadena dividida.
3. Ejercicio 3: El resultado esperado es "¿Dóndeestánlas*nevadas?", ya que se unen las palabras de la lista con asteriscos.
4. Ejercicio 4: El resultado esperado es "Es difícil o posible", ya que se reemplaza "fácil" con "difícil" y se elimina "im".

2.4.1.5

Las cadenas se pueden comparar con operadores de comparación, pero compararlas con números genera resultados no razonables. Por ejemplo, cadena == número siempre es False, cadena != número siempre es True y cadena >= número genera una excepción. El ordenamiento de listas de cadenas se puede lograr con sorted() para crear una nueva lista ordenada o con el método sort() para ordenar la lista existente. Se puede convertir un número en una cadena usando la función str(). Una cadena se

puede convertir en un número utilizando `int()` o `float()`. La conversión falla si la cadena no contiene un número válido, generando una excepción en ese caso.

Ejercicios:

1. Ejercicio 1: La condición verdadera es `'Smiths' < 'Smith'`.
2. Ejercicio 2: El resultado esperado es `"de"`, ya que `s3` contiene las palabras ordenadas alfabéticamente y se imprime el elemento en la posición 1.
3. Ejercicio 3: El resultado esperado es `False`, ya que `s1` es una cadena que representa el número 12.8, mientras que `s2` es la cadena `'12'` después de convertir y volver a convertir el número.

2.5.1.5

Las cadenas son fundamentales en el procesamiento de datos moderno, ya que la mayoría de los datos útiles son cadenas. Motores de búsqueda web, por ejemplo, utilizan procesamiento de cadenas complejo. Comparar cadenas de manera estricta puede ser insatisfactorio para búsquedas avanzadas. Algoritmos de comparación de cadenas difusos, como la Distancia Hamming y la Distancia Levenshtein, encuentran similitudes entre cadenas. La similitud acústica de cadenas, que determina si suenan parecidas (como `"echo"` y `"hecho"`), se puede lograr con algoritmos como Soundex, diseñado en 1918 y utilizado para el idioma inglés. Debido a la limitación de precisión en datos enteros y flotantes, a veces es razonable almacenar y procesar valores numéricos enormes como cadenas, técnica utilizada por Python en operaciones con números grandes.

2.6.1.12

Las excepciones son eventos durante la ejecución del programa causados por situaciones anormales. Para manejarlas, se utiliza el bloque `try-except`. El código riesgoso va dentro del bloque `try`, y si ocurre una excepción, la ejecución salta al bloque `except` donde se maneja la excepción. Puedes tener más de un bloque `except` para manejar diferentes excepciones. Cada bloque puede estar etiquetado con un nombre específico. Solo se ejecuta el bloque cuya excepción coincide con la generada. No se puede tener más de un bloque de excepción sin nombre después de los bloques con nombre. Un bloque `except` sin nombre se utiliza para capturar cualquier excepción no manejada por los bloques con nombre.

Ejercicios:

1. Ejercicio 1: El resultado esperado es:

```
Tratemos de hacer esto
Hemos fallado
Hemos terminado
```

2. Ejercicio 2: El resultado esperado es: `0`

2.7.1.8

No se puede agregar más de un bloque `except` sin nombre después de los bloques con nombre en la misma secuencia. Las excepciones en Python forman una jerarquía. Es importante colocar las excepciones más concretas antes de las más generales en bloques `except` para evitar que las excepciones más específicas no sean alcanzadas. La sentencia `raise ExceptionName` genera una excepción bajo demanda. Sin `ExceptionName`, se usa solo dentro del bloque `try` y genera la misma excepción que se está manejando. La sentencia `assert expression` evalúa la expresión y genera la excepción `AssertionError` si la expresión es igual a cero, una cadena vacía o `None`. Se utiliza para proteger partes críticas del código.

Ejercicios:

1. La salida esperada es "cero", ya que se maneja la excepción `ZeroDivisionError` antes de `ArithmeticError`.
2. La salida esperada es "arit", ya que se maneja la excepción `ArithmeticError` antes de `ZeroDivisionError`.
3. La salida esperada es "algo", ya que la expresión `assert x en foo` genera un `AssertionError` cuando `x` es igual a cero, y se maneja por el bloque `except` genérico.

2.8.1.5

Para proteger al código de ser interrumpido por el uso del teclado, se utiliza la excepción `KeyboardInterrupt`. La excepción más general de todas en Python es `BaseException`. La excepción generada por la evaluación fallida `huge_value = 1E250 ** 2` sería `OverflowError`.

3.1.1.8

Si asumimos que pitones, víboras y cobras son subclases de la misma superclase, podríamos llamar a la superclase "Serpiente" o "Reptil" dependiendo del contexto y la relación deseada. Algunas posibles subclases de la clase Pitón podrían ser "PitónReticulada", "PitónDeBola", "PitónVerde", etc., dependiendo de las características específicas que quieras representar. Sí, puedes usar la palabra clave "class" para darle nombre a tus clases en Python. En el ejemplo proporcionado, la clase se llama "This_Is_A_Class".

3.3.1.9

Ejercicio 1:

- `population` y `victims` son variables de clase.
- `length_ft` es una variable de instancia.
- `__venomous` es una variable de instancia privada.

Ejercicio 2: Para negar la propiedad `__venomous` del objeto `version_2`, puedes asignarle el valor opuesto (negar su valor actual). Por ejemplo:

```
python
```

```
version_2.__venomous = not version_2.__venomous
```

Sin embargo, ten en cuenta que esto no es la forma adecuada de acceder a una variable de instancia privada en Python. Lo recomendable es utilizar métodos de la clase para acceder y modificar variables privadas.

Ejercicio 3: Puedes usar la función `hasattr()` para verificar si el objeto `version_2` contiene una propiedad de instancia llamada `constrictor`. Por ejemplo:

```
python
```

```
if hasattr(version_2, 'constrictor'):
    print("El objeto version_2 tiene una propiedad de instancia llamada
constrictor.")
else:
```

```
print("El objeto version_2 no tiene una propiedad de instancia llamada constrictor.")
```

3.4.1.11

Ejercicio 1:

python

```
class Snake:
    def __init__(self):
        self.victims = 0

    def increment(self):
        self.victims += 1
```

Ejercicio 2:

python

```
class Snake:
    def __init__(self, initial_victims):
        self.victims = initial_victims
```

Ejercicio 3: El resultado del código será:

Python es una Snake
Snake puede ser una Python

En este caso, Python es una subclase de Snake, por lo que `Python.__bases__[0]` se refiere a la superclase, que es Snake.

3.5.1.22

1. **Método `__str__()`**: Es responsable de convertir el contenido de un objeto en una cadena legible. Puedes redefinirlo para personalizar la presentación de tu objeto.
2. **Función `issubclass(Class_1, Class_2)`**: Determina si `Class_1` es una subclase de `Class_2`.
3. **Función `isinstance(Object, Class)`**: Comprueba si un objeto proviene de una clase indicada.
4. **Operador `is`**: Comprueba si dos variables hacen referencia al mismo objeto.
5. **Función `super()`**: Retorna la referencia a la superclase más cercana de la clase actual.
6. **Herencia automática**: Los métodos, variables de instancia y de clase definidos en una superclase son heredados por sus subclases.
7. **Búsqueda de propiedades**: Python busca propiedades dentro del objeto, sus clases y las superclases siguiendo una jerarquía.
8. **Anulación de propiedades**: Si una subclase define una propiedad con el mismo nombre que existe en la superclase, la nueva propiedad anula la anterior.

Estos conceptos son fundamentales para comprender cómo funcionan las clases y la herencia en Python.

3.5.1.23

Ejercicio 1:

El resultado esperado del siguiente código sería:

```
yaml
```

```
Collie dice: ¡Guau!
```

```
Dobermann dice: ¡Guau! ¡Quédese donde está, intruso!
```

Ejercicio 2:

El resultado esperado del siguiente código sería:

```
graphql
```

```
True False
```

```
False True
```

Ejercicio 3:

El resultado esperado del siguiente código sería:

```
graphql
```

```
True False
```

```
2
```

Ejercicio 4:

```
python
```

```
class LowlandDog(SheepDog):  
    def __str__(self):  
        return super().__str__() + " ¡No me gustan las montañas!"
```

Con esta definición, puedes crear instancias de LowlandDog y utilizar el método `__str__()` personalizado:

```
python
```

```
lowland_dog = LowlandDog("SomeBreed")  
print(lowland_dog)
```

El resultado esperado sería:

```
vbnet
```

```
SomeBreed dice: ¡Guau! ¡No me gustan las montañas!
```

3.6.1.9

El bloque `else` de la sentencia `try` se ejecuta cuando no ha habido ninguna excepción durante la ejecución del `try`. El bloque `finally` de la sentencia `try` siempre se ejecuta, independientemente de si se produce una excepción o no. La sintaxis `except Exception_Name as exception_object` te permite interceptar un objeto que contiene información sobre una excepción pendiente. La propiedad del objeto llamada `args` (una tupla) almacena todos los argumentos pasados al constructor del objeto. Las clases de excepciones pueden extenderse para enriquecerlas con nuevas capacidades o para adoptar sus características a excepciones recién definidas.

Ejercicio 1:

```
python

import math

    print(math.sqrt(9))
except ValueError:
    print("inf")
else:
    print("ok")
```

Resultado esperado: ok

Ejercicio 2:

```
python

import math

    print(math.sqrt(-9))
except ValueError:
    print("inf")
else:
    print("ok")
finally:
    print("fin")
```

Resultado esperado: inf fin

Ejercicio 3:

```
python

import math

class NewValueError(ValueError):
    def __init__(self, name, color, state):
        self.data = (name, color, state)

    raise NewValueError("Advertencia enemiga", "Alerta roja", "Alta
disponibilidad")
except NewValueError as nve:
    for arg in nve.args:
        print(arg, end='! ')
```

Resultado esperado: Advertencia enemiga! Alerta roja! Alta disponibilidad!