

# Actividad II Programación Orientada a Objetos con Python

## 1.1.1.11 RESUMEN DE SECCIÓN.

**Puntos Claves**

1. Si deseas importar un módulo como un todo, puedes hacerlo usando la sentencia `import nombre_del_módulo`. Puedes importar más de un módulo a la vez utilizando una lista separada por comas. Por ejemplo:

```
import mod1
import mod2, mod3, mod4
```

Aunque la última forma no se recomienda por razones estilísticas, y es mejor y más bonito expresar la misma intención de una forma más detallada y explícita, como por ejemplo:

```
import mod2
import mod3
import mod4
```

2. Si un módulo se importa de la manera anterior y desea acceder a cualquiera de sus entidades, debes anteponer el nombre de la entidad empleando la **notación con punto**. Por ejemplo:

```
import my_module

result = my_module.my_function(my_module.my_data)
```

El fragmento de código utiliza dos entidades que provienen del módulo `my_module`: una función llamada `my_function()` y una variable con el nombre `my_data`. Ambos nombres **deben tener el prefijo** `my_module.`

**Ejercicio 1**

Quieres invocar la función `make_money()` (contenida en el módulo `mint`) en la siguiente línea:

```
import mint
```

¿Cuál es la forma adecuada de invocar a la función?

**Revisar**

```
mint.make_money()
```

**Ejercicio 2**

Quieres invocar la función `make_money()` (contenida en el módulo `mint`) en la siguiente línea:

```
from mint import make_money
```

¿Cuál es la forma adecuada de invocar a la función?

**Revisar**

**Otros perfiles**

- fenisco
- iren
- iren (íren)
- Neri (Trabajo)
- Invitado
- Agregar

## 1.2.1.17 RESUMEN DE LA SECCIÓN

**Puntos Clave**

1. Una función llamada `dir()` puede mostrarte una lista de las entidades contenidas dentro de un módulo importado. Por ejemplo:

```
import os
dir(os)
```

Imprime una lista de todo el contenido del módulo `os` (el cual, puedes usar en tu código).

2. El módulo `math` contiene más de 50 funciones y constantes que realizan operaciones matemáticas (como `sin()`, `pow()`, `factorial()`) o aportando valores importantes (como  $\pi$  y la constante de Euler  $e$ ).

3. El módulo `random` agrupa más de 60 entidades diseñadas para ayudarte a usar números pseudoaleatorios. No olvides el prefijo "pseudo", ya que no existe un número aleatorio real cuando se trata de generarlos utilizando los algoritmos de la computadora.

4. El módulo `platform` contiene alrededor de 70 funciones que te permiten sumergirte en las capas subyacentes del sistema operativo y el hardware. Usarlos te permite aprender más sobre el entorno en el que se ejecuta tu código.

5. El **Índice de Módulos de Python** (<https://docs.python.org/3/py-modindex.html>) es un directorio de módulos impulsado por la comunidad disponible en el universo de Python. Si deseas encontrar un módulo que se adapte a tus necesidades, puedes utilizarlo para determinar el nombre del CPE Linux como directorio de módulos.

**Ejercicio 1**

¿Cuál es el valor esperado de la variable `result` después de que se ejecute el siguiente código?

```
import math
result = math.e == math.exp(1)
```

**Revisar**

True

**Ejercicio 2**

(Completa el enunciado) Establecer la semilla del generador con `random.seed(1)` garantiza que...

**Revisar**

... los valores pseudoaleatorios emitidos desde el módulo `random` sean los mismos.

**Ejercicio 3**

¿Cuál es el valor esperado de la variable `result` después de que se ejecute el siguiente código?

```
import random
result = random.random()
```

**Revisar**

0.5

**Otros perfiles**

- fenisco
- iren
- iren (íren)
- Neri (Trabajo)
- Invitado
- Agregar

### 1.3.1.11 RESUMEN DE LA SECCIÓN

Edube Interactive: 1.3.1.11 RESUMEN DE LA SECCIÓN

edube.org/learn/python-essentials-2-esp/resumen-de-secci-oacute-n-14

Linux G1R0531: Exa... | portafolio de evade... | linux 9-16 exame... | WhatsApp | Gmail | YouTube | Maps | Traducir | Noticias

1. Mientras que un **módulo** está diseñado para acoplar algunas entidades relacionadas como funciones, variables o constantes, un **paquete** es un contenedor que permite el acoplamiento de varios módulos relacionados bajo un mismo nombre. Dicho contenedor se puede distribuir tal cual (como un lote de archivos implementados en un subárbol de directorio) o se puede empaquetar dentro de un archivo zip.

2. Durante la primera importación del módulo, Python traduce su código fuente a un formato **semi-compilado** almacenado dentro de los archivos `.pyc` y los implementa en el directorio `__pycache__` ubicado en el directorio de inicio del módulo.

3. Si deseas decirle al usuario del módulo que una entidad en particular debe tratarse como **privada** (es decir, no debe usarse explícitamente fuera del módulo), puedes marcar su nombre con el prefijo `_` (o `__`). No olvides que esto es solo una recomendación, no una orden.

4. Los nombres *shabang*, *shebang*, *hasbang*, *poundbang* y *hashpling* describen el dígrafo escrito como `#!`; se utiliza para instruir a los sistemas operativos similares a Unix sobre cómo se debe iniciar el archivo fuente de Python. Esta convención no tiene efecto en MS Windows.

5. Si deseas convencer a Python de que debe tomar en cuenta el directorio de un paquete no estándar, su nombre debe insertarse/agregarse en la lista de directorios de importación almacenada en la variable `sys.path` contenida en el módulo `sys`.

6. Un archivo de Python llamado `__init__.py` se ejecuta implícitamente cuando un paquete que lo contiene está sujeto a importación y se utiliza para inicializar un paquete y/o sus subpaquetes (si los hay). El archivo puede estar

Deseas evitar que el usuario de tu módulo ejecute tu código como administrador.

Revisar

```
import sys

if __name__ == "__main__":
    print("¡No hagas eso!")
    sys.exit()
```

Ejercicio 2

Algunos paquetes adicionales y necesarios se almacenan dentro de `sys.path`. Escribe un código asegurándote de que Python recorra el directorio de instalación de Python.

Revisar

```
import sys

# Toma en cuenta las diagonales invertidas dobles
sys.path.append("D:\\Python\\Proyecto\\Modules")
```

Prev | Next

Person 1

neri bueno  
buenon668@gmail.com

La sincronización está activada.

Administrar tu Cuenta de Google

Cerrar 2 ventanas

Otros perfiles

fenisco

iren

iren (iren)

Neri (Trabajo)

Invitado

Agregar

### 1.4.1.18 RESUMEN DE LA SECCIÓN

Edube Interactive: 1.4.1.18 RESUMEN DE LA SECCIÓN

edube.org/learn/python-essentials-2-esp/resumen-de-secci-oacute-n-15

Linux G1R0531: Exa... | portafolio de evade... | linux 9-16 exame... | WhatsApp | Gmail | YouTube | Maps | Traducir | Noticias

**Puntos Claves**

1. Un **repositorio** (o **repo** para abreviar) diseñado para recopilar y compartir código Python gratuito lleva por nombre **Python Package Index (PyPI)** aunque también es probable que te encuentres con el nombre de **The Cheese Shop (La Tienda de Queso)**. Su sitio web está disponible en <https://pypi.org/>.

2. Para hacer uso de The Cheese Shop, se ha creado una herramienta especializada y su nombre es **pip** (*pip instala paquetes* mientras que *pip* significa *... ok, no importa*). Como es posible que *pip* no se implemente como parte de la instalación estándar de Python, es posible que debas instalarlo manualmente. *Pip* es una herramienta de consola.

3. Para verificar la versión de *pip*, se deben emitir los siguientes comandos:

```
pip --version
```

o

```
pip3 --version
```

Comprueba tu mismo cuál de estos funciona en el entorno de tu sistema operativo.

4. La lista de las actividades principales de *pip* tiene el siguiente aspecto:

- `pip help operación_o_comando` muestra una breve descripción de *pip*.

Ejercicio 1

¿De dónde proviene el nombre *The Cheese Shop*?

Revisar

Es una referencia a un viejo sketch de *Monty Python* que lleva el nombre de *The Cheese Shop*.

Ejercicio 2

¿Por qué deberías asegurarte de cuál *pip* o *pip3* es el correcto para tu sistema operativo?

Revisar

Cuando Python 2 y Python 3 coexisten en el sistema operativo, *pip* trabaja solo con paquetes de Python 2.

Ejercicio 3

¿Cómo puedes determinar si tu *pip* funciona con Python 2 o Python 3?

Revisar

```
pip --version
```

 te lo dirá.

Prev | Next

Person 1

neri bueno  
buenon668@gmail.com

La sincronización está activada.

Administrar tu Cuenta de Google

Cerrar 2 ventanas

Otros perfiles

fenisco

iren

iren (iren)

Neri (Trabajo)

Invitado

Agregar

#### 2.1.1.4 RESUMEN DE LA SECCIÓN

Edube Interactivo: 2.1.1.4 RESUMEN DE SECCIÓN

edube.org/learn/python-essentials-2-esp/resumen-de-secci-oacute-n-16

Linux GIR0533: Exa... | portfolio de evide... | linux 9-16 examene... | WhatsApp | Gmail | YouTube | Maps | Traducir | Noticias

2.1.1.4 RESUMEN DE SECCIÓN

MODULO 17%

SECCIÓN 100%

☰

🔍

⚙️

🔗

### Puntos Clave

- Las computadoras almacenan caracteres como números. Hay más de una forma posible de codificar caracteres, pero solo algunas de ellas ganaron popularidad en todo el mundo y se usan comúnmente en TI: estas son **ASCII** (se emplea principalmente para codificar el alfabeto latino y algunos de sus derivados) y **UNICODE** (capaz de codificar prácticamente todos los alfabetos que utilizan los seres humanos).
- Un número correspondiente a un carácter en particular se llama **punto de código**.
- Los archivos de texto que se crean usando archivos o pérdida menos esp...

Curso: DH-Conexión de Rede...

← → ↺

elearning.utn...

☆ 📁 ⬇️ 📄 📱 🗎

Linux GIR0533: Exa... | portfolio de evide...

☰

FELEPE NERI FRANCISCO BUENO GONZALEZ

### Ejercicio 1

¿Qué es BOM?

Revisar

**BOM** (Byte Order Mark), Una Marca de Orden de Bytes es una combinación especial de bits que anuncia la codificación utilizada por el contenido de un archivo (por ejemplo, UCS-4 o UTF-B).

### Ejercicio 2

¿Está Python 3 internacionalizado?

Revisar

Sí, está completamente internacionalizado: podemos usar caracteres UNICODE dentro de nuestro código, leerlos desde la entrada y enviarlos a la salida.

Prev

Next

#### 2.2.1.15 RESUMEN DE LA SECCIÓN

The screenshot shows a web browser window with the address bar displaying "edub.org/learn/python-essentials-2-esp/resumen-de-secci-oacute-n-17". The page title is "2.2.1.15 RESUMEN DE SECCIÓN". Below the header, there's a progress bar indicating "MODULE (24%)". The main content area has a heading "Puntos Claves" (Key Points) followed by a numbered list:

- Las cadenas de Python son **secuencias inmutables** y se pueden indexar, dividir en rebanadas e iterar como cualquier otra secuencia, además de estar sujetas a los operadores `in` y `not in`. Existen dos tipos de cadenas en Python:
  - Cadenas de **una línea**, las cuales no pueden cruzar los límites de una línea, las denotamos usando apóstrofes (`'cadena'`) o comillas (`"cadena"`).
  - Cadenas **multilinea**, que ocupan más de una línea de código fuente, delimitadas por apóstrofes triples:

An inset window shows a video player titled "Curso: DH-Conexión de Redes..." with a name overlay: "FELIPE NERI FRANCISCO BUENO GONZALEZ".

### 2.3.1.17 RESUMEN DE LA SECCIÓN

Edube Interactive : 2.3.1.17 RESUMEN DE SECCIÓN

< 2.3.1.17 RESUMEN DE SECCIÓN >

PYTHON INSTITUTE

MÓDULO (80%)

SECCIÓN (80%)

Linux GIR053: Exa... portafolio de evide... linux 9-16 exame... WhatsApp Gmail YouTube Maps Traducir Noticias

≡ ⚙ 🔍 ↻

## Puntos Clave

1. Algunos de los métodos que ofrecen las cadenas son:

- `capitalize()` cambia todas las letras de la cadena a mayúsculas.
- `center()` centra la cadena dentro de una longitud conocida.
- `count()` cuenta las ocurrencias de un carácter dado.
- `join()` une todos los elementos de una tupla/lista en una cadena.
- `lower()` convierte todas las letras de la cadena en minúsculas.
- `lstrip()` elimina los caracteres en blanco al principio de la cadena.

Curso: DH-Conexión de Redes

elearning.utn...

Linux GIR053: Exa... portafolio de evide...

FELIPE NERI FRANCISCO BUENO GONZALEZ

## Ejercicio 1

¿Cuál es el resultado esperado del siguiente código?

```
for ch in "abc123XYZ":  
    if ch.isupper():  
        print(ch.lower(), end='')  
    elif ch.islower():  
        print(ch.upper(), end='')  
    else:  
        print(ch, end='')
```

Revisar

ABC123xyz

## Ejercicio 2

¿Cuál es el resultado esperado del siguiente código?

```
s1 = "¿Dónde están las nevadas de añoño?"  
s2 = s1.split()  
print(s2[-2])
```

Prev

Next

#### 2.4.1.5 RESUMEN DE LA SECCIÓN

The screenshot shows the Python Institute eLearning platform. The top navigation bar includes the Python logo and the text "PYTHON INSTITUTE". Below it, there's a progress bar indicating "MODULE (56%)". The main header displays the course title "2.4.1.5 RESUMEN DE SECCIÓN".

### Puntos Claves

- Las cadenas se pueden comparar con otras cadenas utilizando operadores de comparación generales, pero compararlos con números no da un resultado razonable, porque **ninguna cadena puede ser igual** a ningún otro número. Por ejemplo:
  - `cadena == número` es siempre `False` (falso).
  - `cadena != número` es siempre `True` (verdadero).
  - `cadena >= número` siempre genera una excepción.

### Ejercicio 1

¿Cuál de las siguientes líneas describe una condición verdadera?

```

1 'smith' > 'Smith'
2 'Smiths' < 'Smith'
3 'Smith' > '1000'
4 '11' < '8'
5

```

[Revisar]

1, 3 y 4

### Ejercicio 2

¿Cuál es el resultado esperado del siguiente código?

```

def funcion(int()):
    # ...

```

[Revisar]

## 2.5.1.5 RESUMEN DE LA SECCIÓN

**Puntos Claves**

1. Las cadenas son herramientas clave en el procesamiento de datos modernos, ya que la mayoría de los datos útiles son en realidad cadenas. Por ejemplo, el uso de un motor de búsqueda web (que parece bastante trivial en estos días) utiliza un procesamiento de cadenas extremadamente complejo, que involucra cantidades inimaginables de datos.
2. El comparar cadenas de forma estricta (como lo hace Python) puede ser muy insatisfactorio cuando se trata de búsquedas avanzadas (por ejemplo, durante consultas extensas a bases de datos). En respuesta a esta demanda, se han creado e implementado una serie de algoritmos de comparación de cadenas difusos. Estos algoritmos pueden encontrar cadenas que no son iguales en el sentido de Python, pero que son **similares**.
3. Otra forma de comparar cadenas es encontrar su similitud ac... determinar si dos cadenas suenan similares (como "echo" y "hecho" o incluso dialecto) por separado.

Un algoritmo utilizado para realizar una comparación de este tipo... no lo crearás, en 1918. Puedes encontrar más información al res...

4. Debido a la precisión limitada de los datos enteros y flotantes... valores numéricos enormes como cadenas. Esta es la técnica que... número entero que consta de una gran cantidad de dígitos.

Uno de esos conceptos es la **Distancia Hamming**, que se utiliza para determinar la similitud de dos cadenas. Si este tema te interesa, puedes encontrar más información al respecto aquí: [https://en.wikipedia.org/wiki/Hamming\\_distance](https://en.wikipedia.org/wiki/Hamming_distance). Otra solución del mismo tipo, pero basada en un supuesto diferente, es la **Distancia Levenshtein** descrita aquí: [https://en.wikipedia.org/wiki/Levenshtein\\_distance](https://en.wikipedia.org/wiki/Levenshtein_distance).

**Ejercicio 1**

¿Cuál es el resultado esperado del siguiente código?

```
try:
    print("Tratemos de hacer esto")
    print("a"[2])
    print("¡Tuvimos éxito!")
except:
    print("¡Hemos fallado!")
    print("¡Hemos terminado!")
```

**Ejercicio 2**

¿Cuál es el resultado esperado del siguiente código?

```
try:
    print("alpha"[1/0])
```

## 2.6.1.12 RESUMEN DE LA SECCIÓN

**Puntos Clave**

1. Una excepción es un evento durante la ejecución del programa causado por una situación anormal. La excepción debe manejarse para evitar la terminación del programa. La parte del código que se sospecha que es la fuente de la excepción debe colocarse dentro del bloque `try`.
2. Si necesitas manejar más de una excepción proveniente del mismo bloque `try`, puedes agregar más de un bloque `except`, pero debes etiquetarlos con diferentes nombres, así:

Cuando ocurre la excepción, la ejecución del código no se termina, sino que salta al bloque `except`. Este es el lugar donde debe llevarse a cabo el manejo de la excepción. El esquema general para tal construcción es el siguiente:

```
# El código que siempre corre suavemente.
:
try:
    # Código arriesgado.
:
except:
    # La gestión de la crisis se lleva a cabo aquí.
:
# De vuelta a la normalidad.
:
```

**Ejercicio 1**

¿Cuál es el resultado esperado del siguiente código?

```
try:
    print("Tratemos de hacer esto")
    print("a"[2])
    print("¡Tuvimos éxito!")
except:
    print("¡Hemos fallado!")
    print("¡Hemos terminado!")
```

**Ejercicio 2**

¿Cuál es el resultado esperado del siguiente código?

```
try:
    print("alpha"[1/0])
```

## 2.7.1.18 RESUMEN DE LA SECCIÓN

Edube Interactive :: 2.7.1.8 RESUMEN DE SECCIÓN

edube.org/learn/python-essentials-2-esp/resumen-de-secci-oacute-n-22

MODULE (91%) SECTION (100%)

### Puntos Clave

1. No se puede agregar más de un bloque `except` sin nombre después de los bloques con nombre.

```
# El código que siempre corre suavemente.
:
try:
    # Código arriesgado.
    :
except Except_1:
    # La gestión de la crisis se lleva a cabo aquí.
except Except_2:
    # Salvamos el mundo aquí.
except:
    # Todos los demás problemas caen aquí.
:
# De vuelta a la normalidad.
:
```

2. Todas las excepciones de Python predefinidas forman una jerarquía, es decir, algunas de ellas son más generales (le llamada `BaseException` es la más general) mientras que otras son más o menos concretas (por ejemplo, `IndexError` es más concreta que `LookupError`).

Debes evitar colocar excepciones generales antes de las más concretas dentro de la misma secuencia de bloques `except`. Por ejemplo, puedes hacer esto:

### Ejercicio 1

¿Cuál es la salida esperada del siguiente código?

```
try:
    print(1/0)
except ZeroDivisionError:
    print("cero")
except ArithmeticError:
    print("arit")
except:
    print("algo")
```

Revisar

cero

### Ejercicio 2

¿Cuál es la salida esperada del siguiente código?

```
try:
    print(1/0)
except ArithmeticError:
```

Prev Next

## 2.8.1.5 RESUMEN DE LA SECCIÓN

Edube Interactive :: 2.8.1.5 RESUMEN DE SECCIÓN

edube.org/learn/python-essentials-2-esp/resumen-de-secci-oacute-n-23

MODULE (97%) SECTION (93%)

### Puntos Clave

1. Algunas excepciones integradas abstractas de Python son:

- `ArithmeticError`
- `BaseException`
- `LookupError`

2. Algunas excepciones integradas concretas de Python son:

- `AssertionError`
- `ImportError`
- `IndexError`
- `KeyboardInterrupt`
- `KeyError`
- `MemoryError`
- `OverflowError`

### Ejercicio 1

¿Cuál de las excepciones se utilizará para proteger al código de se...

Revisar

KeyboardInterrupt

### Ejercicio 2

¿Cuál es el nombre de la más general de todas las excepciones de...

Revisar

BaseException

### Ejercicio 3

¿Cuál de las excepciones será generada a través de la siguiente evaluación?

```
huge_value = 1E250 ** 2
```

Prev Next

### 3.1.1.8 RESUMEN DE SECCIÓN.

The screenshot shows the Edube Interactive course page for '3.1.1.8 RESUMEN DE SECCIÓN'. The page is divided into two main sections: 'Puntos Clave' (Key Points) and 'Ejercicios' (Exercises). The 'Puntos Clave' section contains five numbered points explaining the concepts of classes, subclasses, inheritance, and objects in Python. The 'Ejercicios' section contains three exercises: 'Ejercicio 1' (Si asumimos que pitones, víboras y cobras son subclases de la m...), 'Ejercicio 2' (Intenta nombrar algunas subclases de la clase Pitón.), and 'Ejercicio 3' (¿Puedes usar la palabra "class" para darle nombre a alguna de tus...). A user profile overlay is visible on the right side of the page, showing the user's name 'neri bueno', email 'buenon668@gmail.com', and a list of other profiles.

**Puntos Clave**

1. Una **clase** es una idea (más o menos abstracta) que se puede utilizar para crear varias encarnaciones; una encarnación de este tipo se denomina **objeto**.
2. Cuando una clase se deriva de otra clase, su relación se denomina **herencia**. La clase que deriva de la otra clase se denomina **subclase**. El segundo lado de esta relación se denomina **superclase**. Una forma de presentar dicha relación es en un **diagrama de herencia**, donde:
  - Las superclases siempre se presentan **encima** de sus subclases.
  - Las relaciones entre clases se muestran como flechas dirigidas **desde la subclase hacia su superclase**.
3. Los objetos están equipados con:
  - Un **nombre** que los identifica y nos permite distinguirlos.
  - Un conjunto de **propiedades** (el conjunto puede estar vacío).
  - Un conjunto de **métodos** (también puede estar vacío).
4. Para definir una clase de Python, se necesita usar la palabra clave reservada `class`. Por ejemplo:

```
class This_Is_A_Class:
    pass
```
5. Para crear un objeto de la clase previamente definida, se necesita usar la clase como si fuera una función. Por

**Ejercicio 1**

Si asumimos que pitones, víboras y cobras son subclases de la m...

**Ejercicio 2**

Intenta nombrar algunas subclases de la clase Pitón.

**Ejercicio 3**

¿Puedes usar la palabra "class" para darle nombre a alguna de tus...

### 3.3.1.13 RESUMEN DE SECCIÓN.

The screenshot shows the Edube Interactive course page for '3.2.1.13 RESUMEN DE SECCIÓN'. The page is divided into two main sections: 'Puntos Clave' (Key Points) and 'Ejercicios' (Exercises). The 'Puntos Clave' section contains six numbered points explaining the concepts of stacks, POO, methods, constructors, and private components in Python. The 'Ejercicios' section contains two exercises: 'Ejercicio 1' (Suponiendo que hay una clase llamada 'Snakes', escribe la primera línea de código que exprese el hecho de que la nueva clase es en realidad una subclase de 'Snakes') and 'Ejercicio 2' (Algo falta en la siguiente declaración, ¿qué es?). A user profile overlay is visible on the right side of the page, showing the user's name 'neri bueno', email 'buenon668@gmail.com', and a list of other profiles.

**Puntos Clave**

1. Una **pila** es un objeto diseñado para almacenar datos utilizando el modelo **LIFO**. La pila normalmente realiza al menos dos operaciones, llamadas **push()** y **pop()**.
2. La implementación de la pila en un modelo procedimental plantea varios problemas que pueden resolverse con las técnicas ofrecidas por la **POO** (Programación Orientada a Objetos).
3. Un **método** de clase es en realidad una función declarada dentro de la clase y capaz de acceder a todos los componentes de la clase.
4. La parte de la clase en Python responsable de crear nuevos objetos se llama **constructor** y se implementa como un método de nombre `__init__`.
5. Cada declaración de método de clase debe contener al menos un parámetro (siempre el primero) generalmente denominado `self`, y es utilizado por los objetos para identificarse a sí mismos.
6. Si queremos ocultar alguno de los componentes de una clase del mundo exterior, debemos comenzar su nombre con `__`. Estos componentes se denominan **privados**.

**Ejercicio 1**

Suponiendo que hay una clase llamada 'Snakes', escribe la primera línea de código que exprese el hecho de que la nueva clase es en realidad una subclase de 'Snakes'.

**Ejercicio 2**

Algo falta en la siguiente declaración, ¿qué es?

```
class Snakes
def __init__():
    self.sound = 'Sssssss'
```

El constructor `__init__()` carece del parámetro obligatorio (deberíamos llamarlo `self` para cumplir con los estándares).

### 3.3.1.9 RESUMEN DE SECCIÓN.

**Puntos Clave**

- Una **variable de instancia** es una propiedad cuya existencia depende de la creación de un objeto. Cada objeto puede tener un conjunto diferente de variables de instancia. Además, se pueden agregar y quitar libremente de los objetos durante su vida útil. Todas las variables de instancia de un objeto se almacenan dentro de un diccionario dedicado llamado `__dict__`, contenido en cada objeto por separado.
- Una variable de instancia puede ser privada cuando su nombre comienza con `__`, pero no olvides que dicha propiedad aún es accesible desde fuera de la clase usando un **nombre modificado** construido como `<code>_ClassName__PrivatePropertyName</code>`.
- Una **variable de clase** es una propiedad que existe exactamente en una copia y no necesita ningún objeto creado para ser accesible. Estas variables no se muestran como contenido de `__dict__`. Todas las variables de clase de una clase se almacenan dentro de un diccionario dedicado llamado `__dict__`, contenido en cada clase por separado.
- Una función llamada `hasattr()` se puede utilizar para determinar si algún objeto o clase contiene cierta propiedad especificada. Por ejemplo:

**Ejercicio 1**

¿Cuáles de las propiedades de la clase `Python` son variables de instancia y cuáles son privadas?

```
class Python:
    population = 1
    victims = 0
    def __init__(self):
        self.length_ft = 3
        self.__venomous = False
```

Revisar

population y victims son variables de clase, mientras que \_\_venomous es una variable de instancia (esta última también es privada).

**Ejercicio 2**

Vas a negar la propiedad `__venomous` del objeto `version_2`, ignorando el hecho de que la propiedad es privada. ¿Cómo vas a hacer esto?

```
version_2 = Python()
```

Prev Next

**User Profile Overlay:**

Person 1  
neri bueno  
buenon668@gmail.com

La sincronización está activada.  
Administrar tu Cuenta de Google  
Cerrar 3 ventanas

Otros perfiles

- fenisco
- iren
- iren (iren)
- Neri (Trabajo)
- Invitado
- Agregar

### 3.4.1.11 RESUMEN DE SECCIÓN.

**Puntos Clave**

- Un método es una función dentro de una clase. El primer (o único) parámetro de cada método se suele llamar `self`, que está diseñado para identificar al objeto para el que se invoca el método con el fin de acceder a las propiedades del objeto o invocar sus métodos.
- Si una clase contiene un **constructor** (un método llamado `__init__`), este no puede devolver ningún valor y no se puede invocar directamente.
- Todas las clases (pero no los objetos) contienen una propiedad llamada `__name__`, que almacena el nombre de la clase. Además, una propiedad llamada `__module__` almacena el nombre del módulo en el que se ha declarado la clase, mientras que la propiedad llamada `__bases__` es una tupla que contiene las superclases de una clase. Por ejemplo:

```
1 class Sample:
2     def __init__(self):
3         self.name = Sample.__name__
4     def myself(self):
5         print("Mi nombre es " + self.name + " y vivo en " + Sample.__module__)
6
7
8 obj = Sample()
9 obj.myself()
10
```

**Ejercicio 1**

La declaración de la clase `Snake` se muestra a continuación. En `increment()`, el cual incrementa en 1 la propiedad `victims`.

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

Revisar

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

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

**Ejercicio 2**

Redefine el constructor de la clase `Snake` para que tenga un parámetro que inicialice el campo `victims` con un valor pasado al objeto durante la construcción.

Prev Next

**User Profile Overlay:**

Person 1  
neri bueno  
buenon668@gmail.com

La sincronización está activada.  
Administrar tu Cuenta de Google  
Cerrar 3 ventanas

Otros perfiles

- fenisco
- iren
- iren (iren)
- Neri (Trabajo)
- Invitado
- Agregar



### 3.5.1.22 RESUMEN DE SECCIÓN.

Edube Interactive - 3.5.1.22 RESUMEN DE SECCIÓN 1/2

edube.org/learn/python-essentials-2-esp/resumen-de-secci-oacutec-n-1-2

LINUX GIR053: Exa... | portfolio de evade... | linux 9-16 exame... | WhatsApp | Gmail | YouTube | Maps | Traducir | Noticias

### Puntos Clave

- Un método llamado `__str__()` es responsable de convertir el contenido de un objeto en una cadena (más o menos) legible. Puedes redefinirlo si deseas que tu objeto pueda presentarse de una forma más elegante. Por ejemplo:

```
1 class Mouse:
2     def __init__(self, name):
3         self.my_name = name
4
5     def __str__(self):
6         return self.my_name
7
8 tme_mouse = Mouse('micky')
9 print(tme_mouse) # Imprime "micky".
10
```

- Una función llamada `issubclass(Class_1, Class_2)` es capaz de determinar si `Class_1` es una subclase de `Class_2`. Por ejemplo:

```
1 class Mouse:
2     pass
3
4 class LabMouse(Mouse):
5     pass
```

- Una función sin parámetros llamada `super()` retorna la referencia al superclase. Por ejemplo:

```
1 class Mouse:
2     def __str__(self):
3         return "Mouse"
4
5 class LabMouse(Mouse):
6     def __str__(self):
7         return "Laboratory " + super().__str__()
8
9 doctor_mouse = LabMouse()
10 print(doctor_mouse) # Imprime "Laboratory Mouse"
```

- Los métodos, así como las variables de instancia y de clase se definen automáticamente por sus subclases. Por ejemplo:

```
1 class Mouse:
2     Population = 0
3     def __init__(self, name):
4         Mouse.Population += 1
5         self.name = name
6
7     def __str__(self):
8         return "Hola, mi nombre es " + self.name
```

### 3.5.1.23 RESUMEN DE SECCIÓN.

The screenshot shows a web browser window displaying a Python course page. The address bar shows the URL: `edube.org/learn/python-essentials-2-esp/resumen-de-secci-oacute-n-2-2`. The page title is "3.5.1.23 RESUMEN DE SECCIÓN 2/2". The main content area has a heading "Ejercicios" and a subheading "Escenario". Below it, there is a paragraph: "El siguiente fragmento de código se ha ejecutado con éxito:". A code editor displays the following Python code:

```
1 class Dog:
2     kennel = 0
3     def __init__(self, breed):
4         self.breed = breed
5         Dog.kennel += 1
6     def __str__(self):
7         return self.breed + " dice: ¡Gauau!"
8
9
10 class SheepDog(Dog):
11     def __str__(self):
12         return super().__str__() + " ¡No huyas, corderito!"
13
14
15 class GuardDog(Dog):
16     def __str__(self):
17         return super().__str__() + " ¡Quédese donde está, intruso!"
18
19
20 rocky = SheepDog("Collie")
21 luna = GuardDog("Doberman")
22
```

To the right of the code editor, there are two exercise questions:

**Ejercicio 1**  
¿Cuál es el resultado esperado del siguiente código?

```
print(rocky)
print(luna)
```

**Ejercicio 2**  
¿Cuál es el resultado esperado del siguiente código?

```
print(isinstance(SheepDog, Dog), isinstance(SheepDog, Sheep))
print(isinstance(rocky, GuardDog), isinstance(luna, Luna))
```

On the far right, there is a sidebar with a profile card for "Person 1" (User: neri bueno, buenon66@gmail.com). Below the profile card, there are links for "La sincronización está activada.", "Administrar tu Cuenta de Google", and "Cerrar 3 ventanas". At the bottom of the sidebar, there is a section titled "Otros perfiles" with a list of users: fenisico, iren, iren (irren), Neri (Trabajo), Invitado, and Agregar.

### 3.6.1.9 RESUMEN DE SECCIÓN.

The screenshot shows the Educe Interactive interface for the section '3.6.1.9 RESUMEN DE SECCIÓN'. The page is divided into three main columns. The left column, titled 'Puntos Clave', contains four key points about try-except-else and try-except-finally blocks, and the use of the 'except' keyword. The middle column, titled 'Ejercicio 1', contains a code snippet for a function that calculates the square root of a number and returns it, or raises a ValueError if the number is negative. The right column, titled 'Ejercicio 2', contains a code snippet for a function that calculates the square root of a number and returns it, or returns 'ok' if the number is negative. The bottom of the page has 'Prev' and 'Next' buttons. On the right side, there is a user profile card for 'neri bueno' with a Google account link and a list of other profiles.

**Puntos Clave**

1. El bloque `else:` de la sentencia `try` se ejecuta cuando no ha habido ninguna excepción durante la ejecución del `try:`.
2. El bloque `finally:` de la sentencia `try` es siempre ejecutado.
3. La sintaxis `except ExceptionName 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.
4. Las clases de excepciones pueden extenderse para enriquecerlas con nuevas capacidades o para adoptar sus características a excepciones recién definidas.

Por ejemplo:

```
try:
    assert __name__ == "__main__"
except:
    print("fallido", end=" ")
else:
    print("éxito", end=" ")
finally:
    print("terminado")
```

**Ejercicio 1**

¿Cuál es el resultado esperado del siguiente código?

```
import math

try:
    print(math.sqrt(5))
except ValueError:
    print("inf")
else:
    print("ok")
```

Revisar

3.0  
ok

**Ejercicio 2**

¿Cuál es el resultado esperado del siguiente código?

```
import math
```

### 4.1.1.15 RESUMEN DE SECCIÓN.

The screenshot shows the Educe Interactive interface for the section '4.1.1.15 RESUMEN DE SECCIÓN'. The page is divided into three main columns. The left column, titled 'Puntos Clave', contains four key points about iterators, generators, conditional expressions, and list comprehensions. The middle column, titled 'Ejercicio 1', contains a code snippet for a class 'Vowels' that implements an iterator and a generator. The right column, titled 'Ejercicio 2', contains a code snippet for a function that calculates the square root of a number and returns it, or returns 'ok' if the number is negative. The bottom of the page has 'Prev' and 'Next' buttons. On the right side, there is a user profile card for 'neri bueno' with a Google account link and a list of other profiles.

**Puntos Clave**

1. Un **iterador** es un objeto de una clase que proporciona al menos **dos** métodos (sin contar el constructor):
  - `__iter__()` se invoca una vez cuando se crea el iterador y devuelve el **propio** objeto del iterador.
  - `__next__()` se invoca para proporcionar el **valor de la siguiente iteración** y genera la excepción `StopIteration` cuando la iteración **llega a su fin**.
2. La sentencia `yield` solo puede ser utilizada dentro de funciones. La sentencia `yield` suspende la ejecución de la función y hace que la función regrese el argumento de `yield` como resultado. Esta función no puede invocarse de forma regular, su único propósito es ser utilizada como un **generador** (es decir, en un contexto que requiera una serie de valores, como un bucle `for`).
3. Una **expresión condicional** es una expresión construida usando el operador `if-else`. Por ejemplo:

```
print(True if 0 >= 0 else False)
```

Da como salida: `True`.
4. Una **lista por comprensión** se convierte en un **generador** cuando se emplea dentro de **paréntesis** (usado entre corchetes, produce una lista regular). Por ejemplo:

```
for x in (e1 + 2 for e1 in range(5)):
```

**Ejercicio 1**

¿Cuál es el resultado esperado del siguiente código?

```
class Vowels:
    def __init__(self):
        self.vow = "aeiouy" # Si, sabemos que y
        self.pos = 0

    def __iter__(self):
        return self

    def __next__(self):
        if self.pos == len(self.vow):
            raise StopIteration
        self.pos += 1
        return self.vow[self.pos - 1]

vowels = Vowels()
for v in vowels:
    print(v, end=" ")
```

Check

a e i o u y

#### 4.2.1.12 RESUMEN DE SECCIÓN.

Edu Interactive > 4.2.1.12 RESUMEN DE SECCIÓN

edube.org/learn/python-essentials-2-esp/resumen-de-secci-oacute-n-29

LINUX G1R053I: Exa... portafolio de evade... linux 9-16 examen... WhatsApp Gmail YouTube Maps Traducir Noticias

PYTHON INSTITUTE

<< 4.2.1.12 RESUMEN DE SECCIÓN >>

MODULE (50%) SECTION (100%)

## Puntos Clave

- Un archivo necesita ser **abierto** antes de que pueda ser procesado por un programa, y debe ser **cerrado** cuando el procesamiento termine.  
  
El abrir un archivo lo asocia con el **stream**, que es una representación abstracta de los datos físicos almacenados en los medios. La forma en que se procesa el stream se llama **modo de apertura**. Existen **tres** modos de apertura:
  - modo lectura:** solo se permiten operaciones de lectura.
  - modo escritura:** solo se permiten operaciones de escritura.
  - modo de actualización:** se permiten ambas, lectura y escritura.
- Dependiendo del contenido del archivo físico, se pueden usar diferentes clases de Python para procesar archivos. En general, `BufferedIOBase` es capaz de procesar cualquier archivo, mientras que `TextIOBase` es una clase especializada dedicada al procesamiento de archivos de texto (es decir, archivos que contienen textos visibles para humanos divididos en líneas usando marcadores de nueva línea). Por lo tanto, los streams se pueden dividir en **binarios** y de **texto**.
- Las siguientes sintaxis de la función `open()` se utilizan para abrir un archivo:  

```
open(nombre_archivo, modo=modo_apertura, codificación=codificación_de_texto)
```

  
La invocación crea un objeto stream y lo asocia con el archivo llamado `nombre_archivo`, utilizando el modo `modo_apertura` y configurando la especificada `codificación_de_texto`, o genera una excepción en caso de un error.

### Ejercicio 1

¿Cómo se codifica el valor del argumento modo de la función?

Rewear

```
"rt" | "w"
```

### Ejercicio 2

¿Cuál es el significado del valor representado por `errno.EACCESS`?

Rewear

**Permiso denegado:** no se permite acceder al contenido del archivo.

### Ejercicio 3

¿Cuál es la salida esperada del siguiente código, asumiendo que el sistema de archivos funciona correctamente?

```
import errno  
try:
```

Person 1

neri bueno  
buenon668@gmail.com

La sincronización está activada.

Administrar tu Cuenta de Google

Cerrar 3 ventanas

Otros perfiles

f fenico

w iren

N iren (iren)

N Neri (Trabajo)

@ Invitado

+ Agregar

#### 4.3.1.18 RESUMEN DE SECCIÓN.

Python Institute

4.3.1.18 RESUMEN DE SECCIÓN

MODULE (49%)

SECTION (100%)

Python 4.3.1.18 RESUMEN DE SECCIÓN

Person 1

neri bueno  
buenon668@gmail.com

La sincronización está activada.
 Administrar tu Cuenta de Google

Cerrar 3 ventanas

Otros perfiles

- fenisco
- iren
- iren (irren)
- Neri (Trabajo)
- Invitado
- Agregar

Ejercicio 1

¿Qué se espera del método `readlines()` cuando el stream es...

Revisar

Una lista vacía (una lista de longitud cero).

Ejercicio 2

¿Qué se pretende hacer con el siguiente código?

```
for line in open("file", "rt"):
    for char in line:
        if char.lower() not in "aeiouy ":
            print(char, end='')
```

Revisar

Copla el contenido del archivo `file` hacia la consola, ignorando las...

Ejercicio 3

Prev Next

#### 4.4.1.9 RESUMEN DE SECCIÓN.

Edube Interactive: 4.4.1.9 RESUMEN DE SECCIÓN

edube.org/learn/python-essentials-2-esp/resumen-de-secci-oacute-n-31

MODULES (100%) SECTION (100%)

4.4.1.9 RESUMEN DE SECCIÓN

1. La función `uname` devuelve un objeto que contiene información sobre el sistema operativo actual. El objeto tiene los siguientes atributos:

- `systemname` (almacena el nombre del sistema operativo)
- `nodename` (almacena el nombre de la máquina en la red)
- `release` (almacena el release (actualización) del sistema operativo)
- `version` (almacena la versión del sistema operativo)
- `machine` (almacena el identificador de hardware, por ejemplo, x86\_64)

2. El atributo `name` disponible en el módulo `os` te permite distinguir el sistema operativo. Devuelve uno de los siguientes tres valores:

- `posix` (obtendrás este nombre si usas Unix)
- `nt` (obtendrás este nombre si usas Windows)
- `java` (obtendrás este nombre si tu código está escrito en algo como Jython)

3. La función `mkdir` crea un directorio en la ruta pasada como argumento. La ruta puede ser relativa o absoluta, por ejemplo:

```
import os

os.mkdir("hello") # la ruta relativa
os.mkdir("/home/python/hello") # la ruta absoluta
```

**Nota:** Si el directorio existe, una excepción `FileExistsError` será generada. Además de la función `mkdir`, el módulo `os` proporciona la función `mkdirs`, que te permite crear recursivamente todos los directorios en una ruta.

¿Cuál es el resultado del siguiente fragmento si se ejecuta en Unix?

```
import os
print(os.name)
```

Revisar

posix

Ejercicio 2

¿Cuál es el resultado del siguiente fragmento de código?

```
import os
os.mkdir("hello")
print(os.listdir())
```

Revisar

['hello']

Person 1

neri bueno  
buenon668@gmail.com

La sincronización está activada.

Administrar tu Cuenta de Google

Cerrar 3 ventanas

Otros perfiles

- fenisco
- iren
- iren (iren)
- Neri (Trabajo)
- Invitado
- Agregar

Prev Next

#### 4.5.1.23 RESUMEN DE SECCIÓN.

Edube Interactive: 4.5.1.23 RESUMEN DE SECCIÓN

edube.org/learn/python-essentials-2-esp/resumen-de-secci-oacute-n-32

MODULES (100%) SECTION (100%)

4.5.1.23 RESUMEN DE SECCIÓN

```
my_date = date(2020, 9, 29)
print("Año:", my_date.year) # Año: 2020
print("Mes:", my_date.month) # Mes: 9
print("Día:", my_date.day) # Día: 29
```

El objeto `date` tiene tres atributos (de solo lectura): año, mes y día.

2. El método `today` devuelve un objeto de fecha que representa la fecha local actual:

```
from datetime import date
print("Hoy:", date.today()) # Muestra: Hoy: 2020-09-29
```

3. En Unix, la marca de tiempo expresa el número de segundos desde el 1 de Enero de 1970 a las 00:00:00 (UTC). Esta fecha se llama la "época de Unix", porque ahí comenzó el conteo del tiempo en los sistemas Unix. La marca de tiempo es en realidad la diferencia entre una fecha en particular (incluida la hora) y el 1 de Enero de 1970, 00:00:00 (UTC), expresada en segundos. Para crear un objeto de fecha a partir de una marca de tiempo, debemos pasar una marca de tiempo Unix al método `fromtimestamp`:

```
from datetime import date
import time

timestamp = time.time()
d = date.fromtimestamp(timestamp)
```

¿Cuál es el resultado del siguiente fragmento de código?

```
t = time(14, 39)
print(t.strftime("%H:%M:%S"))
```

Revisar

14:39:00

Ejercicio 1

¿Cuál es el resultado del siguiente fragmento de código?

```
from datetime import datetime
dt1 = datetime(2020, 9, 28, 14, 41, 0)
dt2 = datetime(2020, 9, 28, 14, 41, 0)
print(dt1 - dt2)
```

Revisar

1 day, 0:00:00

Person 1

neri bueno  
buenon668@gmail.com

La sincronización está activada.

Administrar tu Cuenta de Google

Cerrar 3 ventanas

Otros perfiles

- fenisco
- iren
- iren (iren)
- Neri (Trabajo)
- Invitado
- Agregar

Prev Next

4.6.1.14 RESUMEN DE SECCIÓN.

Edube Interactive - 4.6.1.14 RE

edube.org/learn/python-essentials-2-esp/resumen-de-secci-oacute-n-33

Linux G1R0531: Exa... | portafolio de evade... | linux 9-16 examene... | WhatsApp | Gmail | YouTube | Maps | Traducir | Noticias

Python Institute

4.6.1.14 RESUMEN DE SECCIÓN

MODULOS (17%)

SECTION (97%)

2. Para mostrar un calendario de cualquier año, se emplea la función `calendar` con el año pasado como argumento, por ejemplo:

```
import calendar
print(calendar.calendar(2020))
```

Nota: Una buena alternativa a la función anterior es la función llamada `prcal`, que también toma los mismos parámetros que la función `calendar`, pero no requiere el uso de la función `print` para mostrar el calendario.

3. Para mostrar un calendario de cualquier mes del año, se emplea la función `month`, pasándole el año y el mes. Por ejemplo:

```
import calendar
print(calendar.month(2020, 9))
```

Nota: También puedes usar la función `pmonth`, que tiene los mismos parámetros que la función `month`, pero no requiere el uso de la función `print` para mostrar el calendario.

4. La función `setfirstweekday` te permite cambiar el primer día de la semana. Toma un valor de 0 a 6, donde 0 es Domingo y 6 es Sábado.

```
import calendar
print(calendar.weekheader(1))
```

Revisar

M T W T F S S

Ejercicio 2

¿Cuál es el resultado del siguiente fragmento de código?

```
import calendar
c = calendar.Calendar()
for weekday in c.iterweekdays():
    print(weekday, end=" ")
```

Revisar

0 1 2 3 4 5 6

Person 1

neri bueno

buenon668@gmail.com

La sincronización está activada.

Administrar tu Cuenta de Google

Cerrar 3 ventanas

Otros perfiles

fenisco

iren

iren (iren)

Neri (Trabajo)

Invitado

Agregar

Alumno: Felipe Neri Francisco Bueno González