Olvera Ibarra Sharon Michelle

Actividad II Programación Orientada a Objetos con Python

Módulo 1.1.1.11

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1, Ejercicio 1
#Descripción: Quieres invocar la función make_money() contenida en el módulo
#llamado mint. Tu código comienza con la siguiente línea:

import mint

# Luego puedes llamar a la función make_money()
mint.make_money()
```

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1, Ejercicio 2
#Descripción: Quieres invocar la función make_money()
#contenida en el módulo llamado mint. Tu código comienza con la siguiente línea:

from mint import make_money

#¿Cuál es la forma adecuada de invocar a la función?
make_money()
```

Ejercicio 3

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1, Ejercicio 3
#Descripción: Has escrito una función llamada
#make_money por tu cuenta. Necesitas importar una función
#con el mismo nombre del módulo mint y no deseas cambiar el nombre
#de ninguno de tus nombres previamente definidos. ¿Qué variante de la
#sentencia import puede ayudarte con el problema?

from mint import make_money as mint_make_money
```

Ejercicio 4

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1, Ejercicio 4
#Descripción: ¿Qué forma de invocación de la función
#make_money es válida si tu código comienza con la siguiente línea?

from mint import *
make_money()
```

Módulo 1.2.1.17

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1.2.1.17, Ejercicio 1
#Descripción: ¿Cuál es el valor esperado de la
#variable result después de que se ejecuta el siguiente código?

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

```
True
```

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1.2.1.17, Ejercicio 2
#Descripción: (Completa el enunciado) Establecer
#la semilla del generador con el mismo valor cada vez
#que se ejecuta tu programa garantiza que ...

#... los valores pseudoaleatorios emitidos desde el
#módulo random serán exactamente los mismos.
```

Ejercicio 3

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1.2.1.17, Ejercicio 3
#Descripción: ¿Cuál de las funciones del módulo
#platform utilizarías para determinar el nombre del
#CPU que corre dentro de tu computadora?
```

```
La función processor()
```

Olvera Ibarra Sharon Michelle

Ejercicio 4

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1.2.1.17, Ejercicio 4
#Descripción: ¿Cuál es el resultado esperado del siguiente fragmento de código?

import platform

print(len(platform.python_version_tuple()))

```

Console >_

```
3
```

Módulo 1.3.1.11

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1.3.1.11, Ejercicio 1
#Descripción: Deseas evitar que el usuario
#de tu módulo ejecute tu código como un script
#ordinario. ¿Cómo lograrías tal efecto?

import sys

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

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1.3.1.11, Ejercicio 2
#Descripción: Algunos paquetes adicionales
#y necesarios se almacenan dentro del directorio
#D:\Python\Project\Modules. Escribe un código asegurándote
#de que Python recorra el directorio para encontrar todos los módulos solicitados.

import sys
sys.path.append("D:\\Python\\Project\\Modules")
```

Ejercicio 3

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1.3.1.11, Ejercicio 3
#Descripción: El directorio mencionado
#en el ejercicio anterior contiene un subárbol con la siguiente estructura:

abc
 |__ def
      |__ mymodule.py

#Asumiendo que D:\Python\Project\Modules se ha adjuntado con éxito a la lista
#sys.path, escribe una directiva de importación que te permita usar todas las
#entidades de mymodule.

import abc.def.mymodule
```

Módulo 1.4.1.18

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1.4.1.18, Ejercicio 1
#Descripción: ¿De donde proviene el nombre The Cheese Shop?

Es una referencia a un viejo sketch de Monty Python que lleva el mismo nombre.
```

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1.4.1.18, Ejercicio 2
#Descripción: ¿Por qué deberías asegurarte de cuál pip o pip3 es el correcto para ti?

Cuando Python 2 y Python 3 coexisten en el sistema operativo, es probable que pip
identifique la instancia de pip que trabaja solo con paquetes de Python 2.
```

Ejercicio 3

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1.4.1.18, Ejercicio 3
#Descripción: ¿Cómo puedes determinar si tu pip funciona con Python 2 o Python 3?

pip --version te lo dirá.
```

Ejercicio 4

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 1.4.1.18, Ejercicio 4
#Descripción: Desafortunadamente, no tienes
#privilegios de administrador. ¿Qué debes hacer
#para instalar un paquete en todo el sistema?

Tienes que consultar a tu administrador del sistema â··
¡no intentes hackear tu sistema operativo!
```

Módulo 2.1.1.4

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 2.1.1.4 , Ejercicio 1
#Descripción:¿Qué es BOM?

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

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 2.1.1.4 , Ejercicio 1
#Descripción:¿Está Python 3 internacionalizado?

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

Módulo 2.2.1.15

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 2.2.1.15, Ejercicio 1
#Descripción:¿Cuál es la longitud de la siguiente
#cadena asumiendo que no hay espacios en blanco entre las comillas?

"""
"""
1
```

Olvera Ibarra Sharon Michelle

Ejercicio 2

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 2.2.1.15, Ejercicio 2
3  #Descripción:¿Cuál es el resultado esperado del siguiente código?
4
5  s = 'yesteryears'
6  the_list = list(s)
7  print(the_list[3:6])
8
9
10
```

Console >_

```
['t', 'e', 'r']
```

Ejercicio 3

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 2.2.1.15, Ejercicio 3
3  #Descripción:¿Cuál es el resultado esperado del siguiente código?
4
5  for ch in "abc":
6      print(chr(ord(ch) + 1), end='')
7
8
9
10
```

Console >_

```
bcd
```

Módulo 2.3.1.17

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 2.3.1.17 , Ejercicio 1
#Descripción: ¿Cuál es el resultado esperado del siguiente código?

 for ch in "abc123XYX":
    if ch.isupper():
        print(ch.lower(), end='')
    elif ch.islower():
        print(ch.upper(), end='')
    else:
        print(ch, end='')


ABC123xyz
```

Olvera Ibarra Sharon Michelle

Ejercicio 2

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 2.3.1.17 , Ejercicio 2
3  #Descripción:¿Cuál es el resultado esperado del siguiente código?
4
5  s1 = '¿Dónde están las nevadas de antaño?'
6  s2 = s1.split()
7  print(s2[-2])
8
9
```

Console >_

```
de
```

Ejercicio 3

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 2.3.1.17 , Ejercicio 3
3  #Descripción: ¿Cuál es el resultado esperado del siguiente código?
4
5  the_list = ['¿Dónde', 'están', 'las', 'nevadas?']
6  s = '*'.join(the_list)
7  print(s)
8
```

Console >_

```
¿Dónde*están*las*nevadas?
```

Ejercicio 4

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 2.3.1.17 , Ejercicio 4
3  #Descripción: ¿Cuál es el resultado esperado del siguiente código?
4
5  s = 'Es fácil o imposible'
6  s = s.replace('fácil', 'difícil').replace('im', '')
7  print(s)
8
```

Console >_

```
Es difícil o posible
```

Módulo 2.4.1.5

Olvera Ibarra Sharon Michelle

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 2.4.1.5 , Ejercicio 1
#Descripción: ¿Cuál de las siguientes líneas describe una condición verdadera?

'smith' > 'Smith'

'Smiths' < 'Smith'

'Smith' > '1000'

'11' < '8'

# 1, 3 y 4
```

Ejercicio 2

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 2.4.1.5 , Ejercicio 2
3  #Descripción: ¿Cuál es el resultado esperado del siguiente código?
4
5  s1 = '¿Dónde están las nevadas de antaño?'
6  s2 = s1.split()
7  s3 = sorted(s2)
8  print(s3[1])
9
```

Console >_

```
de
```

Ejercicio 3

Olvera Ibarra Sharon Michelle

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 2.4.1.5 , Ejercicio 3
#Descripción: ¿Cuál es el resultado esperado del siguiente código?

s1 = '12.8'
i = int(s1)
s2 = str(i)
f = float(s2)
print(s1 == s2)


El código genera una excepción ValueError
```

Console >_

```
Traceback (most recent call last):
  File "main.py", line 6, in <module>
    i = int(s1)
ValueError: invalid literal for int() with base 10: '12.8'
```

Módulo 2.4.1.5

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**.

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. Otra solución del mismo tipo, pero basada en un supuesto diferente, es la **Distancia Levenshtein** descrita aquí: https://en.wikipedia.org/wiki/Levenshtein_distance.

3. Otra forma de comparar cadenas es encontrar su similitud *acústica*, lo que significa un proceso que lleva a determinar si dos cadenas suenan similares (como "echo" y "hecho"). Esa similitud debe establecerse para cada idioma (o incluso dialecto) por separado.

Un algoritmo utilizado para realizar una comparación de este tipo para el idioma Inglés se llama **Soundex** y se inventó, no lo creerás, en 1918. Puedes encontrar más información al respecto aquí: https://en.wikipedia.org/wiki/Soundex.

4. Debido a la precisión limitada de los datos enteros y flotantes nativos, a veces es razonable almacenar y procesar valores numéricos enormes como cadenas. Esta es la técnica que usa Python cuando se le fuerza a operar con un número entero que consta de una gran cantidad de dígitos.

Módulo 2.6.1.12

Ejercicio 1

```python
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 2.6.1.12, Ejercicio 1
#Descripción: ¿Cuál es el resultado esperado del siguiente código?

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

Console >_

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

Ejercicio 2

```python
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 2.6.1.12, Ejercicio 2
#Descripción: ¿Cuál es el resultado esperado del siguiente código?

try:
    print("alpha"[1/0])
except ZeroDivisionError:
    print("cero")
except IndexingError:
    print("índice")
except:
    print("algo")
```

Console >_

```
cero
```

Módulo 2.7.1.8

Ejercicio 1

Olvera Ibarra Sharon Michelle

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 2.7.1.8 , Ejercicio 1
#Descripción: ¿Cuál es la salida esperada del siguiente código?

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

Console >_

```
cero
```

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 2.7.1.8 , Ejercicio 2
#Descripción: ¿Cuál es la salida esperada del siguiente código?
try:
    print(1/0)
except ArithmeticError:
    print("arit")
except ZeroDivisionError:
    print("cero")
except:
    print("algo")
```

Console >_

```
arit
```

Ejercicio 3

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 2.7.1.8 , Ejercicio 3
#Descripción: ¿Cuál es la salida esperada del siguiente código?
def foo(x):
    assert x
    return 1/x


try:
    print(foo(0))
except ZeroDivisionError:
    print("cero")
except:
    print("algo")
```

Console >_

```
algo
```

Módulo 2.8.1.5

Olvera Ibarra Sharon Michelle

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 2.8.1.5 , Ejercicio 1
#Descripción: ¿Cuál de las excepciones se
#utilizará para proteger al código de ser interrumpido por el uso del teclado?

KeyboardInterrupt
```

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 2.8.1.5 , Ejercicio 2
#Descripción: ¿Cuál es el nombre de la más
#general de todas las excepciones de Python?

BaseException
```

Ejercicio 3

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 2.8.1.5 , Ejercicio 3
3  #Descripción: ¿Cuál de las excepciones será
4  #generada a través de la siguiente evaluación fallida?
5
6  huge_value = 1E250 ** 2
7
8  OverflowError
```

Console >_

```
Traceback (most recent call last):
  File "main.py", line 6, in <module>
    huge_value = 1E250 ** 2
OverflowError: (34, 'Result not representable')
```

Olvera Ibarra Sharon Michelle

Módulo 3.1.1.8

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.1.1.8 , Ejercicio 1
#Descripción: Si asumimos que pitones,
#víboras y cobras son subclases de la
#misma superclase, ¿cómo la llamarías?

Serpiente, reptil, vertebrado, animal:
todas estas respuestas son aceptables.
```

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.1.1.8 , Ejercicio 2
#Descripción: Intenta nombrar algunas
#subclases de las clase Pitón.

Pitón india, Pitón de Roca Sfricana, Pitón Bola,
Pitón Birmana: la lista es larga.
```

Ejercicio 3

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.1.1.8 , Ejercicio 3
#Descripción: ¿Puedes usar la palabra "class" para darle
#nombre a alguna de tus clases?
¡No, no puedes, class es una palabra clave reservada!
```

Módulo 3.2.1.13

Ejercicio 1

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 3.2.1.13 , Ejercicio 1
3  #Descripción: Suponiendo que hay una
4  #clase llamada Snakes, escribe la primera
5  #línea de la declaración de clase Python,
6  #expresando el hecho de que la nueva clase es
7  #en realidad una subclase de Snake.
8
9  class Python(Snakes):
```

Olvera Ibarra Sharon Michelle

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.2.1.13 , Ejercicio 2
#Descripción: 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).
```

Ejercicio 3

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 3.2.1.13 , Ejercicio 3
3  #Descripción: Modifica el código para garantizar
4  #que la propiedad venomous sea privada.
5  class Snakes
6      def __init__(self):
7          self.venomous = True
8  ##################################################
9  class Snakes
10     def __init__(self):
11         self.__venomous = True
```

Módulo 3.3.1.9

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.3.1.9 , Ejercicio 1
#Descripción:¿Cuáles de las propiedades
#de la clase Python son variables de instancia
#y cuáles son variables de clase? ¿Cuáles de ellos son privados?
class Python:
    population = 1
    victims = 0
    def __init__(self):
        self.length_ft = 3
        self.__venomous = False
#population y victims son variables de clase, mientras que length y
#__venomous son variables de instancia (esta última también es privada).
```

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.3.1.9 , Ejercicio 2
#Descripción: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()
version_2._Python__venomous = not version_2._Python__venomous
```

Ejercicio 3

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.3.1.9 , Ejercicio 3
#Descripción: Escribe una expresión que
#compruebe si el objeto version_2 contiene
#una propiedad de instancia denominada constrictor (¡si, constrictor!).

hasattr(version_2, 'constrictor')
```

Módulo 3.4.1.11

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.4.1.11 , Ejercicio 1
#Descripción: La declaración de la clase
#Snake se muestra a continuación. Enriquece
#la clase con un método llamado increment(),
#el cual incrementa en 1 la propiedad victims.
class Snake:
    def __init__(self):
        self.victims = 0

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

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

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.4.1.11 , Ejercicio 2
#Descripción: Redefine el constructur
#de la clase Snake para que tenga un
#parámetro que inicialice el campo victims
#con un valor pasado al objeto durante la construcción.

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

Ejercicio 3

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.4.1.11 , Ejercicio 3
#Descripción: ¿Puedes predecir el resultado del siguiente código?

class Snake:
    pass


class Python(Snake):
    pass


print(Python.__name__, 'es una', Snake.__name__)
print(Python.__bases__[0].__name__, 'puede ser una', Python.__name__)

#Python es una Snake
#Snake puede ser una Python
```

Olvera Ibarra Sharon Michelle

Módulo 3.4.1.11

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.5.1.22 , Ejercicio 1
#Descripción: ¿Cuál es el resultado esperado del siguiente código?
class Dog:
    kennel = 0
    def __init__(self, breed):
        self.breed = breed
        Dog.kennel += 1
    def __str__(self):
        return self.breed + " dice: ¡Guau!"


class SheepDog(Dog):
    def __str__(self):
        return super().__str__() + " ¡No huyas, corderito!"


class GuardDog(Dog):
    def __str__(self):
        return super().__str__() + " ¡Quédese donde está, intruso!"


rocky = SheepDog("Collie")
luna = GuardDog("Dobermann")

print(rocky)
print(luna)
```

Console >_

```
Collie dice: ¡Guau! ¡No huyas, corderito!
Dobermann dice: ¡Guau! ¡Quédese donde está, intruso!
```

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.5.1.22 , Ejercicio 2
#Descripción: ¿Cuál es el resultado esperado del siguiente código?
class Dog:
    kennel = 0
    def __init__(self, breed):
        self.breed = breed
        Dog.kennel += 1
    def __str__(self):
        return self.breed + " dice: ¡Guau!"


class SheepDog(Dog):
    def __str__(self):
        return super().__str__() + " ¡No huyas, corderito!"


class GuardDog(Dog):
    def __str__(self):
        return super().__str__() + " ¡Quédese donde está, intruso!"


rocky = SheepDog("Collie")
luna = GuardDog("Dobermann")

print(issubclass(SheepDog, Dog), issubclass(SheepDog, GuardDog))
print(isinstance(rocky, GuardDog), isinstance(luna, GuardDog))
```

Console >_

```
True False
False True
```

Olvera Ibarra Sharon Michelle

Ejercicio 3

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.5.1.22 , Ejercicio 3
#Descripción: ¿Cuál es el resultado esperado del siguiente código?
class Dog:
    kennel = 0
    def __init__(self, breed):
        self.breed = breed
        Dog.kennel += 1
    def __str__(self):
        return self.breed + " dice: ¡Guau!"


class SheepDog(Dog):
    def __str__(self):
        return super().__str__() + " ¡No huyas, corderito!"


class GuardDog(Dog):
    def __str__(self):
        return super().__str__() + " ¡Quédese donde está, intruso!"


rocky = SheepDog("Collie")
luna = GuardDog("Dobermann")

print(luna is luna, rocky is luna)
print(rocky.kennel)
```

Console >_

```
True False
2
```

Ejercicio 4

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.5.1.22 , Ejercicio 4
#Descripción: Define una subclase de
#SheepDog llamada LowlandDog, y equipala
#con un método __str__() que anule un método
#heredado del mismo nombre. El nuevo método __str__()
#debe retornar la cadena "¡Guau! ¡No me gustan las montañas!".
class LowlandDog(SheepDog):
    def __str__(self):
        return Dog.__str__(self) + " ¡No me gustan las montañas"
```

Olvera Ibarra Sharon Michelle

Módulo 3.6.1.9

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.6.1.9  , Ejercicio 1
#Descripción: ¿Cuál es el resultado esperado del siguiente código?
import math

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

Console >_

```
3.0
ok
```

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.6.1.9  , Ejercicio 1
#Descripción: ¿Cuál es el resultado esperado del siguiente código?
import math

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

Console >_

```
inf
fin
```

Ejercicio 3

Olvera Ibarra Sharon Michelle

```python
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 3.6.1.9  , Ejercicio 1
#Descripción: ¿Cuál es el resultado esperado del siguiente código?
import math

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

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

Console >_

```
Advertencia enemiga! Alerta roja! Alta disponibilidad!
```

Módulo 4.1.1.15

Ejercicio 1

```python
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 4.1.1.15, Ejercicio 1
#Descripción: ¿Cuál es el resultado esperado del siguiente código?
class Vowels:
    def __init__(self):
        self.vow = "aeiouy "  # Sí, sabemos que y no siempre se considera una vocal.
        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=' ')
```

Console >_

```
a e i o u y
```

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 4.1.1.15, Ejercicio 2
#Descripción: Escribe una función lambda,
#estableciendo a 1 su argumento entero, y
#aplícalo a la función map() para producir
#la cadena 1 3 3 5 en la consola.
any_list = [1, 2, 3, 4]
even_list = # Completar las líneas aquí.
print(even_list)
##############################################
any_list = [1, 2, 3, 4]
even_list = list(map(lambda n: n | 1, any_list))
print(even_list)
```

Ejercicio 3

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 4.1.1.15, Ejercicio 3
3  #Descripción: ¿Cuál es el resultado esperado del siguiente código?
4  def replace_spaces(replacement='*'):
5      def new_replacement(text):
6          return text.replace(' ', replacement)
7      return new_replacement
8
9
10 stars = replace_spaces()
11 print(stars("And Now for Something Completely Different"))
```

```
Console >_

And*Now*for*Something*Completely*Different
```

Módulo 4.2.1.12

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 4.2.1.12, Ejercicio 1
#Descripción: ¿Cómo se codifica el valor
#del argumento modo de la función open()si se va a crear un nuevo archivo de texto?

"wt"oh"w"
```

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 4.2.1.12, Ejercicio 2
#Descripción: ¿Cuál es el significado del
#valor representado por errno.EACESS?

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

Ejercicio 3

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 4.2.1.12, Ejercicio 3
#Descripción: ¿Cuál es la salida esperada del
#siguiente código, asumiendo que el archivo llamado file no existe?
import errno

try:
    stream = open("file", "rb")
    print("existe")
    stream.close()
except IOError as error:
    if error.errno == errno.ENOENT:
        print("ausente")
    else:
        print("desconocido"

ausente
```

Módulo 4.3.1.18

Ejercicio 1

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 4.3.1.18, Ejercicio 1
#Descripción: ¿Qué se espera del método
#readlines() cuando el stream está asociado con un archivo vacío?

Una lista vacía (una lista de longitud cero).
```

Olvera Ibarra Sharon Michelle

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 4.3.1.18, Ejercicio 2
#Descripción: ¿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='')
```

Copia el contenido del archivo file hacia la consola, ignorando las vocales.

Ejercicio 3

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 4.3.1.18, Ejercicio 3
#Descripción: Vas a procesar un mapa de
#bits almacenado en un archivo llamado
#image.png y quieres leer su contenido
#como un todo en una variable bytearray
#llamada image. Agrega una línea al siguiente
#código para lograr este objetivo.
try:
    stream = open("image.png", "rb")
    # Inserta una línea aquí.
    stream.close()
except IOError:
    print("fallido")
else:
    print("exitoso")

#Respuesta:
image = bytearray(stream.read())
```

Módulo 4.4.1.9

Ejercicio 1

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 4.4.1.9, Ejercicio 1
3  #Descripción: ¿Cuál es el resultado del siguiente fragmento si se ejecuta en Unix?
4  import os
5  print(os.name)
6
```

Console >_

```
posix
```

Ejercicio 2

```
#Autor: Sharon Michelle Olvera Ibarra
#Modulo 4.4.1.9, Ejercicio 2
#Descripción: ¿Cuál es el resultado del siguiente fragmento de código?

import os

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

#Respuesta:
['hello']
```

Módulo 4.5.1.23

Ejercicio 1

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 4.5.1.23, Ejercicio 1
3  #Descripción: ¿Cuál es el resultado del siguiente fragmento de código?
4  from datetime import time
5
6  t = time(14, 39)
7  print(t.strftime("%H:%M:%S"))
```

Console >_

```
14:39:00
```

Ejercicio 2

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 4.5.1.23, Ejercicio 2
3  #Descripción: ¿Cuál es el resultado del siguiente fragmento de código?
4  from datetime import datetime
5
6  dt1 = datetime(2020, 9, 29, 14, 41, 0)
7  dt2 = datetime(2020, 9, 28, 14, 41, 0)
8
9  print(dt1 - dt2)
```

Console >_

```
1 day, 0:00:00
```

Módulo 4.6.1.14

Ejercicio 1

Olvera Ibarra Sharon Michelle

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 4.6.1.14, Ejercicio 1
3  #Descripción: ¿Cuál es el resultado del siguiente fragmento de código?
4
5  import calendar
6  print(calendar.weekheader(1))
```

Console >_

```
M T W T F S S
```

Ejercicio 2

```
1  #Autor: Sharon Michelle Olvera Ibarra
2  #Modulo 4.6.1.14, Ejercicio 2
3  #Descripción: ¿Cuál es el resultado del siguiente fragmento de código?
4
5  import calendar
6
7  c = calendar.Calendar()
8
9  for weekday in c.iterweekdays():
10     print(weekday, end=" ")
```

Console >_

```
0 1 2 3 4 5 6
```