DACD-E1

Total de puntos 24/50

24 de 25 puntos

✓ ¿Qué símbolos deben ser antecedidos al nombre de un atributo privado? 1/1

- ++
- ##
- __ ✓
- &&

✓ ¿Qué se requiere para crear un objeto? 1/1

- Un constructor ✓
- Un método
- Una función
- Una clase

✓ Considere a = 4*4-4+4/4. ¿Cuál es el valor de a? 1/1

- 5
- 19
- 4
- 13 ✓

✓ ¿Cuál será la salida del siguiente código? 1/1

```
def total(initial = 5, *num, **key):
    count = initial
    for n in num:
        count+=n
    for k in key:
        count+=key[k]
    return count

print(total(100,2,3, clouds=50, stars=100))
```

- 255 ✓
- 155
- 160
- 260

✓ ¿Cuál es la forma correcta de crear un objeto del tipo Dog? 1/1

```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

- ○ Dog.create("Rufus", 3)
- ○ Dog()
- ◉ Dog("Rufus", 3) ✓
- ○ Dog.__init__("Rufus", 3)

✓ ¿Cuál opción es correcta para convertir en una lista las llaves de un diccionario llamada “d”? 1/1

- ○ d.list(keys)
- ○ d(keys())
- ○ d(list(keys()))
- ◉ list(d.keys()) ✓

✓ ¿Qué opción es correcta para crear una tupla de un elemento? 1/1

- (●) a=(1,) ✓
- ( ) a=(1)
- ( ) a=tuple(1,)
- ( ) a=tuple(1)

✓ Considera un objeto “O” que es una instancia de la Clase “B”. ¿Cómo se puede validar si el objeto es una instancia de la clase B? 1/1

- ( ) isinstance(B,O)
- ( ) O.isinstance(B)
- ( ) B.isinstance(O)
- (●) isinstance(O,B) ✓

✓ ¿Cuál de los siguientes métodos es ejecutado inmediatamente al instanciar un objeto? 1/1

- ( ) __object__( )
- ( ) __del__( )
- (●) __init__( ) ✓
- ( ) __func__( )

✓ ¿Cuál será la salida del siguiente código? 1/1

```
for i in ['t', 'n', 'i ', 'o', 'p'][::-1]:
   print(i)
```

- ○ point00-1
- ○ tniop
- ◉ point ✓
- ○ tniop00-1

✓ ¿Cuál de las siguientes declaraciones de clase es correcta? 1/1

- ○ def class_name:
- ○ def class_name(self)
- ○ class class_name(self):
- ◉ class class_name: ✓

✗ ¿Cuál será la salida del siguiente código? 0/1

```
i = 1
   while True:
      if(1%2==0):
         break
      print(i)
       i += 2
```

- ○ 13579
- ◉ 1 ✗
- ○ 135
- ○ 12

✓ ¿Cuál de las siguientes funciones de un diccionario devuelve todas sus llaves y respectivos valores? 1/1

- ○ Ninguna
- ○ values()
- ◉ items() ✓
- ○ keys()

✓ ¿Cuál será la salida del siguiente código? 1/1

```
class Dog:
     def walk(self):
          return "*walking*"

     def speak(self):
          return "Woof!"

class StrayDog(Dog):
     def speak(self):
          return "Arff!"

toto = StrayDog()
toto.walk()
```

- ○ Arff!
- ◉ *walking* ✓
- ○ Woof!
- ○ AttributeError: 'StrayDog' object has no 'walk' attribute

✓ ¿Cuál sería la salida del comando list[:3:] si list = ['abcd', 786 , 2.23, 'john', 70.2]? 1/1

- ○ abcd
- ◉ Ninguna ✓
- ○ ['abcd', 786 , 2.23, 'john']
- ○ [786 , 2.23, 'john']

✓ ¿Cuál será la salida del siguiente código? 1/1

```
b = [11,13,15,17,19,21]
print(b[::2])
```

- ○ [13,17,21]
- ○ [11,15]
- ○ [13,17]
- ◉ [11,15,19] ✓

✓ ¿Qué símbolo se utiliza para acceder a los atributos o métodos de un objeto? 1/1

- ○ &
- ◉ . ✓
- ○ #
- ○ *

✓ Qué se obtendría al ejecutar '19' == 19 1/1

- ○ True
- ○ '19'
- ○ Error
- ◉ False ✓

✓ ¿Cuál es la salida de L[-2] si L = [1,2,3]? 1/1

- 3
- Ninguna
- 2 ✓
- 1

✓ ¿Cuál será la salida del siguiente código? 1/1

```
def f(x = 100, y = 100):
        return(x+y, x-y)

x, y = f(y = 200, x = 100)
print(x, y)
```

- 300 100
- 0 200
- 200 0
- 300 -100 ✓

✓ ¿Cuáles son los pilares de la programación orientada a objetos? 1/1

- (●) Herencia, Encapsulación, Abstracción y Polimorfismo ✓
- ( ) Constructor and Destructor
- ( ) Constructor y Clases
- ( ) Constructor, Clases y Objetos

✓ ¿Cuál será la salida del siguiente código? 1/1

```
minidict = { 'name': 'Tutorials', 'name': 'Website'}

print(minidict['name'])
```

- ( ) Error
- (●) Website ✓
- ( ) Tutorials
- ( ) Tutorials, Website

✓ ¿Cuál será la salida del siguiente código? 1/1

```
class Dog:
      def walk(self):
            return "*walking*"

      def speak(self):
            return "Woof!"

class StrayDog(Dog):
      def talk(self):
            return super().speak()

toto = StrayDog()
toto.speak()
```

- ○ AttributeError: 'StrayDog' object has no 'speak' attribute
- ○ *walking*
- ◉ Woof! ✓
- ○ Arff

✓ ¿Cuál será la salida del siguiente código? 1/1

```
def func(x, ans):
    if(x==0):
        return 0
    else:
        return func(x-1, x+ans)

print(func(2,0))
```

- ○ 3
- ○ 2
- ◉ 0 ✓
- ○ 1

✓ ¿Cuál será la salida del siguiente código? 1/1

```
class Dog:
      def walk(self):
            return "*walking*"

      def speak(self):
            return "Woof!"

class StrayDog(Dog):
      def speak(self):
            return "Arff!"

toto = StrayDog()
toto.speak()
```

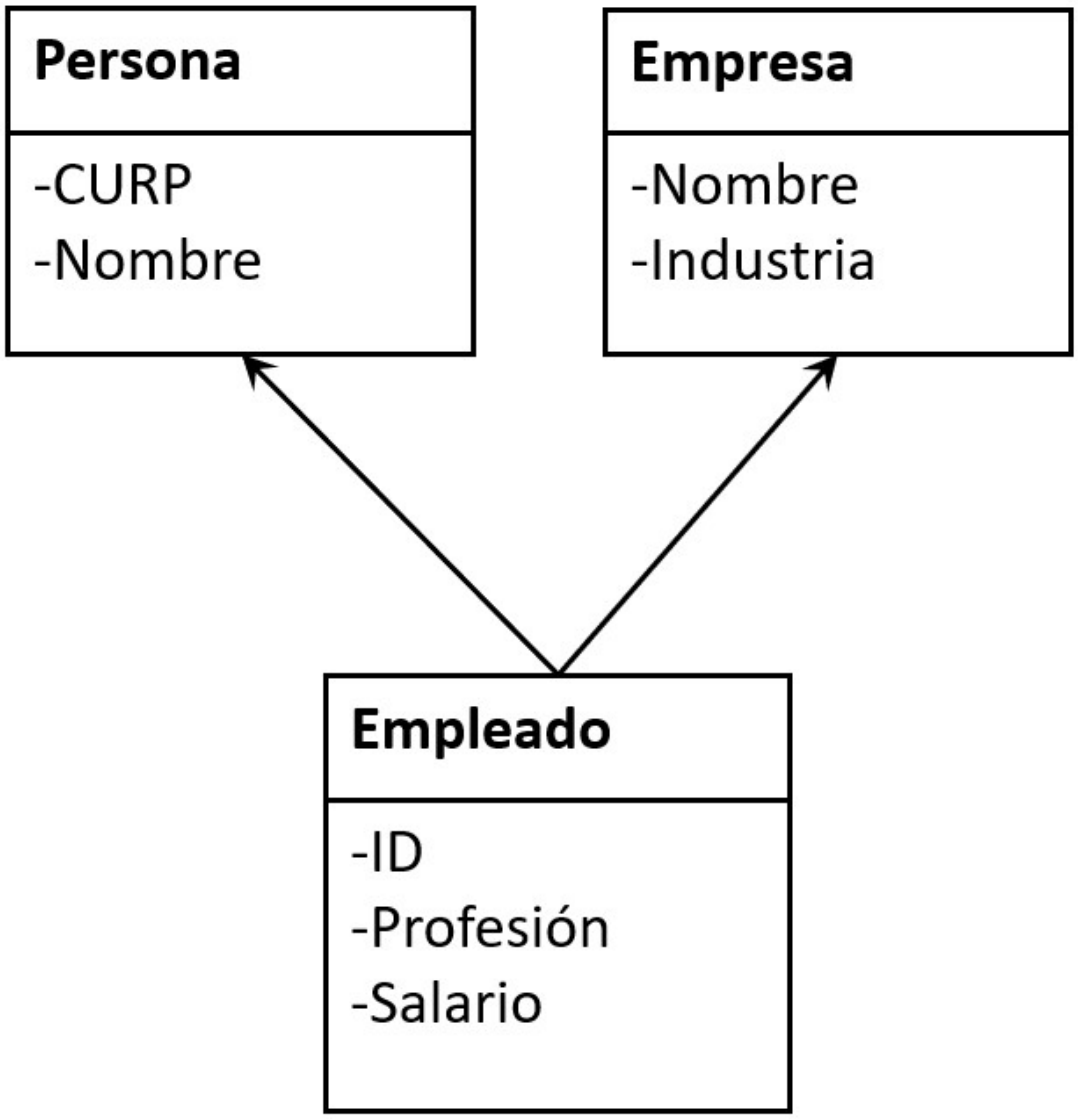
- ○ AttributeError: 'StrayDog' object has two 'speak' attribute
- ○ Woof!
- ◉ Arff! ✓
- ○ *walking*

Desarrolle el código para implementar las clases mostrada en el siguiente diagrama UML parcial. 0 de 25 puntos

Detalles adicionales:

1. Los atributos de los objetos deben ser capturadores en el respectivo constructor. Ejemplo: input("Proporciona el ID de empleado: ").
2. Deben existir dos tipos de empleados: Sustancial y de Apoyo. Los primeros deben tener un salario de $30,000 y los segundos de $20,000.

…/25




```
class Empleado():

    def __init__(self):

        self.id = input("Proporciona el ID del empleado: ")
        self.profesion = input("Proporciona la profesión del empleado: ")
        self.salario = 30000
        self.tipo = input("Qué tipo de empleado eres: ")
        if self.tipo == "Sustancial":
            self.salario
        else:
            self.salario - 10000
```

```
class Persona(Empleado):

    def __init__(self):
        Empleado.__init__(self)
        self.curp = input("Proporcione su CURP: ")
        self.nombre = input("Proporciona su nombre: ")


class Empresa(Empleado):

    def __init__(self):
        Empleado.__init__(self)
        self.nombreEm = input("Proporcione el nombre de la empresa: ")
        self.industria = input("Proporcione la industria: ")


emp = Empleado()
per = Persona()
fab = Empresa()
```