Clase 16 – Repaso POO

IIC1103 Sección 9 – 2019-2

Profesor: Felipe López

21-11-2019

Resumen

- Las clase en son representaciones abstractas en Python.
- Una instancia de una clase con atributos específicos es un objeto.
- Se puede interactuar con una clase mediante sus métodos. Estos también permiten la interacción entre distintas clases.

- En Python se definen de la siguiente manera:

```
class Nombre_clase:
  def __init__(self,…):
        self.atributo1
        self.atributo2
                …
  def metodo1(self,…):

  def metodo2(self,…):
```

Resumen

- Es posible “sobrecargar” el método (str) de un objeto para que imprima lo que nosotros queramos.
- En este caso, definimos el siguiente método de una clase:

```
def __str__(self):
        …
        return (string)
```

- Esto permitirá poder ocupar `print(objeto)` de esa clase y obtener lo que retorne este método. También permitirá obtener información cada vez que hagamos `str()` a este objeto.
- No obstante, el método __str__, no funciona al imprimir objetos dentro de una lista.

- Para eso podemos ocupar el método __repr__

```
def __repr__(self):
        …
        return (string)
```

- Que cumple las mismas funciones de __str__, además de permitir imprimir objetos dentro de una lista.

Ejercicio (I2 2018-2)

Pregunta 3

¡Bienvenido a la Feria de Investigación del DCC! El siguiente código usa programación orientada a objetos, incluyendo las clases de objetos `Stand` y `Feria` para gestionar la asignación de stands durante la feria. En concreto, primero se crea la feria con `ns` stands vacíos. A continuación, se escoge entre tres opciones:

1. **Inscribir un stand**. Si quedan stands vacíos, se solicita el título del stand y el número de participantes, de lo contrario, se imprime que no quedan stands vacíos.
2. **Información**. Dado un número de stand, en caso de estar vacío, se indica que está vacío. En caso contrario, se indica el título del stand y el número de participantes.
3. **Cerrar**. Cierra la feria.

```
ns = int(input("Número de stands en la Feria: "))
f = Feria(ns)

op = int(input("1-Inscribir 2-Info 3-Cerrar: "))
while op != 3:

    if op == 1:
        if f.quedan_stands_vacios():
            t = input("Título del stand: ")
            p = int(input("Número de participantes del stand: "))
            f.inscribir(t,p)
        else:
            print("No quedan stands vacíos")

    elif op == 2:
        n = int(input("Número de stand: "))
        s = f.obtener_stand(n) #s de tipo Stand
        if s.vacio():
            print("Stand vacío")
        else:
            print("Título =>", s.titulo())
            print("Participantes =>", s.participantes())

    op = int(input("1-Inscribir 2-Info 3-Cerrar: "))

print("Feria Cerrada")
print("Quedaron", f.numero_stands_vacios(), "stands por llenar")
```

Se te pide que implementes las clases `Stand` y `Feria` con los siguientes métodos con el fin de **que sea compatible con el código principal de ejemplo**:

1) `Stand`:

- **(6 puntos)** `__init__`: No recibe parámetros. Inicializa un stand vacío con sus atributos que corresponda.
- **(3 puntos)** `titulo`: No recibe parámetros. Retorna un *string* con el título del stand.
- **(3 puntos)** `participantes`: No recibe parámetros. Retorna un *int* con la cantidad de participantes del stand.
- **(3 puntos)** `vacio`: No recibe parámetros. Retorna `True` si el stand está vacío; y `False`, en caso contrario.

2) `Feria`:

- **(10 puntos)** `__init__`: Recibe como parámetro un *int* con el número de stands vacíos (de la clase `Stand`) que tiene la feria al crearse. Se inicializa la feria con los stands vacíos creados. Los stands están ordenados, y se pueden identificar con un número que va de `0` al número de stands -1.
- **(15 puntos)** `inscribir`: Recibe un *string* con el título del stand y un *int* con la cantidad de participantes de ese stand (debe ser mayor a 0 y puedes asumir que el input para este atributo siempre cumplirá dicha restricción). Se inscribe el primer stand vacío con dichos parámetros. Puedes suponer que al inscribirse, ya se verificó antes si hay stands vacíos. No retorna nada.
- **(5 puntos)** `obtener_stand`: Recibe un *int* con el índice del stand (va de 0 al número de stands -1) Retorna un objeto de tipo `Stand` con el stand solicitado.
- **(10 puntos)** `numero_stands_vacios`: No recibe parámetros. Retorna un *int* con el número de stands vacíos de la feria.
- **(5 puntos)** `quedan_stands_vacios`: No recibe parámetros. Retorna `True` si quedan stands vacíos o `False` en caso contrario. **Este método debe usar el método `numero_stands_vacios` anterior.**

```python
class Stand:
    def __init__(self):
        self.titulo_stand = ""
        self.participantes_stand = 0

    def titulo(self):
        return self.titulo_stand

    def participantes(self):
        return self.participantes_stand

    def vacio(self):
        if self.participantes_stand <= 0:
            return True
        else:
            return False
```

```python
class Feria:
    def __init__(self, cant_stands_vacios):
        self.stands = []
        for i in range(cant_stands_vacios):
            s = Stand()
            self.stands.append(s)

    def inscribir(self, titulo, participantes):
        for i in self.stands:
            if i.vacio():
                i.titulo_stand = titulo
                i.participantes_stand = participantes
                break

    def obtener_stand(self, indice):
        return self.stands[indice]

    def numero_stands_vacios(self):
        counter = 0
        for i in self.stands:
            if i.vacio():
                counter+=1
        return counter

    def quedan_stands_vacios(self):
        if self.numero_stands_vacios()>0:
            return True
        else:
            return False
```

Ejercicio (P2 I2 2019-1)

Pregunta 2

Te han pedido que implementes un programa para permitir a una cadena hotelera administrar sus distintos hoteles. Para esto, te dan las siguientes especificaciones.

La cadena posee varios hoteles, y cada hotel posee una cierta cantidad de pisos (la cantidad de pisos varía en cada hotel), y cada piso contiene piezas que se pueden arrendar a los clientes. Además, cada pieza tiene un precio por noche. Desde el punto de vista de la administración, las personas solo poseen nombre y edad. Lo que se te ha pedido a ti, es que implementes un programa que entregue las siguientes funcionalidades:

- Imprimir los nombres y edades de los ocupantes de una pieza en específico en este momento.
- La ganancia de un piso en específico en este momento.
- Desocupar una pieza.
- Ocupar una pieza. Para esto, debes registrar todos los nombres y edades de los nuevos huéspedes.
- Imprimir todas las piezas que están habitadas actualmente

Para facilitar el desarrollo, se te entregó un método y algunas de las clases necesarias para el programa, por lo que **sólo debes implementar el código principal y la(s) clase(s) que no esté(n) en el código que te entregan** (**Hint:** No está la clase `Pieza`)

Ejercicio (P2 I2 2019-1)

```
class Huesped:
    def __init__(self, n, e):
        self.nombre = n
        self.edad = e

class Piso:
    def __init__(self, numero):
        self.numero = numero
        self.piezas = []

class Hotel:
    def __init__(self, nombre):
        self.nombre = nombre
        self.pisos = []

def get_menu():
    menu = "¿Qué deseas hacer?\n[1] Mostrar ocupantes de una pieza\n[2] Mostrar ganancias de un piso\n"
    menu += "[3] Desocupar pieza\n[4] Ingresar nuevos huéspedes\n"
    menu += "[5] Mostrar todas las piezas ocupadas\n[6] Salir"
    return menu
```

Ejercicio (P2 I2 2019-1)

Si lo estimas conveniente, puedes implementar nuevas clases y agregar métodos a las clases ya implementadas. Para desarrollar el programa, puedes asumir que ya existe un hotel creado con todos sus pisos y piezas correspondientes. La instancia de este hotel se encuentra en una variable que se llama `h`. Además de esto, al querer ocupar una pieza para atender nuevos clientes, la asignación debe ser automática dando prioridad a las habitaciones que se encuentran en los primeros pisos (ej: Si llega un cliente a una pieza, y está desocupada la 103, la 205 y la 301, se le asigna la 103 porque pertenece al piso de más abajo). A continuación, se muestra un diálogo de ejemplo. Los caracteres en negrita son los datos que son ingresados por el usuario

```
¿Qué deseas hacer?
[1] Mostrar ocupantes de una pieza
[2] Mostrar ganancias de un piso
[3] Desocupar pieza
[4] Ingresar nuevos huéspedes
[5] Mostrar todas las piezas ocupadas
[6] Salir
1

¿Qué pieza quiere mostrar?
203
Pedro, 43 años
Francisca, 42 años
Juan, 10 años

¿Qué deseas hacer?
[1] Mostrar ocupantes de una pieza
[2] Mostrar ganancias de un piso
[3] Desocupar pieza
[4] Ingresar nuevos huéspedes
[5] Mostrar todas las piezas ocupadas
[6] Salir
2

¿Las ganancias de qué piso quiere mostrar?
2
El piso 2 está ganando $350000 pesos por noche

¿Qué deseas hacer?
[1] Mostrar ocupantes de una pieza
[2] Mostrar ganancias de un piso
[3] Desocupar pieza
[4] Ingresar nuevos huéspedes
[5] Mostrar todas las piezas ocupadas
[6] Salir
3

¿Qué pieza quiere desocupar?
103
Pieza 103 desocupada

¿Qué deseas hacer?
[1] Mostrar ocupantes de una pieza
[2] Mostrar ganancias de un piso
[3] Desocupar pieza
[4] Ingresar nuevos huéspedes
[5] Mostrar todas las piezas ocupadas
[6] Salir
4
¿Cuántos huéspedes quiere agregar?
2
Nombre huésped 1: Juan
Edad Huésped 1: 26
Nombre huésped 2: Rosario
Edad Huésped 2: 23
Usuarios ingresados en pieza 103.

¿Qué deseas hacer?
[1] Mostrar ocupantes de una pieza
[2] Mostrar ganancias de un piso
[3] Desocupar pieza
[4] Ingresar nuevos huéspedes
[5] Mostrar todas las piezas ocupadas
[6] Salir
5

Las piezas ocupadas son 103, 104, 105, 201, 302, 303

¿Qué deseas hacer?
[1] Mostrar ocupantes de una pieza
[2] Mostrar ganancias de un piso
[3] Desocupar pieza
[4] Ingresar nuevos huéspedes
[5] Mostrar todas las piezas ocupadas
[6] Salir
6
```

Ejercicio (P2 I2 2019-1)

Considera que las ganancias se generan solamente por las piezas que están siendo ocupadas por algún huésped. Por simplicidad, puedes asumir que siempre habrá al menos una habitación ocupada, y que siempre habrá una habitación vacía para los nuevos huéspedes ingresados.

Ejercicio (P2 I2 2019-1)

```
class Huesped:
  def __init__(self, n, e):
    self.nombre = n
    self.edad = e
  def __repr__(self):
    return self.nombre+", "+str(self.edad)+" años"

class Piso:
  def __init__(self, numero):
    self.numero = numero
    self.piezas = []
  def __repr__(self):
    return "\n\t\tPiso num:"+str(self.numero)+" \n\t\t\tPiezas:"+str(self.piezas)
```

```
class Hotel:
  def __init__(self, nombre):
    self.nombre = nombre
    self.pisos = []

  def primera_desocupada(self):
    contador_piso = 1
    for piso in self.pisos:
      if piso.numero == contador_piso:
        for pieza in piso.piezas:
          if not pieza.ocupada():
            return pieza
      contador_piso += 1
    return "No hay piezas desocupadas"

  def __repr__(self):
    return "Hotel nombre: "+str(self.nombre)+" \n\tPisos:"+str(self.pisos)

class Pieza:
  def __init__(self, numero, precio):
    self.numero = numero
    self.huespedes = []
    self.precio = precio

  def ocupada(self):
    if len(self.huespedes)>0:
      return True
    else:
      return False

  def __repr__(self):
    return "\n\t\t\t\tPieza num:"+str(self.numero)+" \n\t\t\t\t\tHuespedes:"+str(self.huespedes)+"
\n\t\t\t\t\tPrecio:"+str(self.precio)
```

```
opcion = int(input(get_menu()))

while opcion != 6:
  if opcion == 1:
    num_pieza = int(input("\n¿Qué pieza quiere mostrar?\n"))
    for piso in h.pisos:
      for pieza in piso.piezas:
        if num_pieza == pieza.numero:
          for huesped in pieza.huespedes:
            print(huesped)

  elif opcion == 2:
    num_piso = int(input("\n¿Las ganancias de qué piso quiere mostrar?\n"))
    ganancias = 0
    for piso in h.pisos:
      if piso.numero == num_piso:
        for pieza in piso.piezas:
          if pieza.ocupada():
            ganancias += pieza.precio
    print("El piso "+str(num_piso)+" está ganando $"+str(ganancias)+" pesos por noche")
```

```
while opcion != 6:
 (…)
  elif opcion == 3:
    num_pieza = int(input("\n¿Qué pieza quieres desocupar?\n"))
    for piso in h.pisos:
      for pieza in piso.piezas:
        if num_pieza == pieza.numero:
          pieza.huespedes = []

    print("Pieza",num_pieza,"desocupada")
    print(h)

  elif opcion == 4:
    cant_huespedes = int(input("\n¿Cuántos huéspedes quiere agregar?\n"))
    pieza = h.primera_desocupada()
    for i in range(cant_huespedes):
      nombre = input("Nombre huésped "+str(i+1)+": ")
      edad = int(input("Edad huésped "+str(i+1)+": "))
      huesped = Huesped(nombre,edad)
      pieza.huespedes.append(huesped)
    print("Usuarios ingresados en pieza",pieza.numero)
    print(h)
```

```
while opcion != 6:
 (…)

  elif opcion == 5:
    s = "\nLas piezas ocupadas son "
    for piso in h.pisos:
      for pieza in piso.piezas:
        if pieza.ocupada():
          s+=str(pieza.numero)+", "

    s = s[:-2]
    print(s)

  opcion = int(input(get_menu()))
```

Bibliografía

- http://runest.ing.puc.cl/class.html
- A. B. Downey. Think Python: How to think like a computer scientist. Green Tea Press, 2013 -> Capítulo 15

Links

- https://repl.it/@FelipeLopez/IIC1103RepasoPOO que contiene todos los ejemplos de la clase.

Clase 16 – Repaso POO

IIC1103 Sección 9 – 2019-2

Profesor: Felipe López

21-11-2019