IIC1103 - TAV

Clase 12 módulo 3: Interacción entre clases

Prof. Francisca Cattan

Universidad Católica de Chile

23 de enero 2023

Repaso: Metodo str

- El metodo __str__ nos ayuda a asociar la representación en string del objeto.
- Se llama cuando invocamos las funciones print() o str().
- Al definirlo y usarlo, nos ayuda a controlar cómo es la representación del objeto en el tipo string.

```
1  class Animal():
2      def __init__(self, animal, especie):
3          self.animal = animal
4          self.especie = especie
5
6      def __str__(self):
7          return f"Yo soy un {self.animal} chilena y mi
   especie es {self.especie}"
8
9  ani = Animal("ave", "Caiquen")
10 print(ani)
11 str(ani) #>>'Yo soy un ave chilena y mi especie es Caiquen'
```

Yo soy un ave chilena y mi especie es Caiquen

Repaso: Metodo float

- El metodo __float__ nos ayuda a asociar la representación en float del objeto.
- Se llama cuando invocamos a la función float().
- Al definirlo y usarlo, nos ayuda a controlar cómo es la representación del objeto en el tipo float.

```
1  class Fraccion:
2      def __init__(self, numerador, denominador):
3          self.numerador = numerador
4          self.denominador = denominador
5
6      def __float__(self):
7          return self.numerador / self.denominador
8
9  f1 = Fraccion(10, 5)
10 print('El valor de f1 es', float(f1))
11 f2 = Fraccion(22, 9)
12 print('El valor de f2 es', float(f2))
```

```
El valor de f1 es 2.0
El valor de f2 es 2.444444444446
```

Solucion ejercicio en Replit

```python
class Tren:
    def __init__(self, pos, estaciones, direccion):
    self.pos = pos # int
    self.estaciones = estaciones # lista
    self.direccion = direccion # str
    self.estacion_actual = self.estaciones[self.pos]

  def __str__(self):
    return f"Tren en {self.estacion_actual} que esta en la posicion {self.pos} y tiene direccion {self.direccion}"

  def avanzar(self):
    if self.direccion == "derecha":
      self.pos += 1
      if self.pos == len(self.estaciones) - 1:
        self.direccion = "izquierda"
    elif self.direccion == "izquierda":
      self.pos -= 1
      if self.pos == 0:
        self.direccion = "derecha"

    self.estacion_actual = self.estaciones[self.pos]
```

Interaccion entre tipos

¿Qué pasaría si yo quiero usar los datos de un objeto en otro objeto? Es posible acceder a los atributos y métodos de un objeto en cualquier momento. Eso si, requiere de mucho cuidado y conocimiento de tu código al momento de desarrollar.

Interaccion entre tipos

¿Qué pasaría si yo quiero usar los datos de un objeto en otro objeto? Es posible acceder a los atributos y métodos de un objeto en cualquier momento. Eso si, requiere de mucho cuidado y conocimiento de tu código al momento de desarrollar.

```
1  class Estudiante:
2    def __init__(self, nombre, numero_alumno):
3      self.nombre = nombre
4      self.numero_alumno = numero_alumno
5      self.cursos_tomados = []
6
7    def asignar(self, curso_actual):
8      self.cursos_tomados.append(curso_actual)
9      curso_actual.agregar_estudiante(self) # que pasa aqui?
10
11 class CursoActual:
12   def __init__(self, nombre, sigla):
13     self.nombre = nombre
14     self.sigla = sigla
15     self.estudiantes = []
16
17   def agregar_estudiante(self, estudiante):
18     self.estudiantes.append(estudiante)
```

Interaccion entre tipos

¿Qué pasaría si yo quiero usar los datos de un objeto en otro objeto? Es posible acceder a los atributos y métodos de un objeto en cualquier momento. Eso si, requiere de mucho cuidado y conocimiento de tu código al momento de desarrollar.

```
1  class Estudiante:
2    def __init__(self, nombre, numero_alumno):
3      self.nombre = nombre
4      self.numero_alumno = numero_alumno
5      self.cursos_tomados = []
6
7    def asignar(self, curso_actual):
8      self.cursos_tomados.append(curso_actual)
9      curso_actual.agregar_estudiante(self) # que pasa aqui?
10
11 class CursoActual:
12   def __init__(self, nombre, sigla):
13     self.nombre = nombre
14     self.sigla = sigla
15     self.estudiantes = []
16
17   def agregar_estudiante(self, estudiante):
18     self.estudiantes.append(estudiante)
```

Interaccion entre tipos

¿Qué pasaría si yo quiero usar los datos de un objeto en otro objeto? Es posible acceder a los atributos y métodos de un objeto en cualquier momento. Eso si, requiere de mucho cuidado y conocimiento de tu código al momento de desarrollar.

```
1  class Estudiante:
2    def __init__(self, nombre, numero_alumno):
3      self.nombre = nombre
4      self.numero_alumno = numero_alumno
5      self.cursos_tomados = []
6
7    def asignar(self, curso_actual):
8      self.cursos_tomados.append(curso_actual)
9      curso_actual.agregar_estudiante(self) # que pasa aqui?
10
11 class CursoActual:
12   def __init__(self, nombre, sigla):
13     self.nombre = nombre
14     self.sigla = sigla
15     self.estudiantes = []
16
17   def agregar_estudiante(self, estudiante):
18     self.estudiantes.append(estudiante)
```

Interaccion entre tipos

¿Qué pasaría si yo quiero usar los datos de un objeto en otro objeto? Es posible acceder a los atributos y métodos de un objeto en cualquier momento. Eso si, requiere de mucho cuidado y conocimiento de tu código al momento de desarrollar.

```
1  class Estudiante:
2    def __init__(self, nombre, numero_alumno):
3      self.nombre = nombre
4      self.numero_alumno = numero_alumno
5      self.cursos_tomados = []
6
7    def asignar(self, curso_actual):
8      self.cursos_tomados.append(curso_actual)
9      curso_actual.agregar_estudiante(self) # que pasa aqui?
10
11 class CursoActual:
12   def __init__(self, nombre, sigla):
13     self.nombre = nombre
14     self.sigla = sigla
15     self.estudiantes = []
16
17   def agregar_estudiante(self, estudiante):
18     self.estudiantes.append(estudiante)
```

Interaccion entre tipos

```
1  class Estudiante:
2    def __init__(self, nombre, numero_alumno):
3      self.nombre = nombre
4      self.numero_alumno = numero_alumno
5      self.cursos_tomados = []
6
7    def asignar(self, curso_actual):
8      self.cursos_tomados.append(curso_actual)
9      curso_actual.agregar_estudiante(self) # que pasa aqui?
10
11 class CursoActual:
12   def __init__(self, nombre, sigla):
13     self.nombre = nombre
14     self.sigla = sigla
15     self.estudiantes = []
16
17   def agregar_estudiante(self, estudiante):
18     self.estudiantes.append(estudiante)
```

```
1 curso = CursoActual("Intro a la Progra",'IIC1103')
2 bob = Estudiante("Bob Ross", 21035413)
3 bob.asignar(curso)
4 curso.estudiantes[0].nombre #>> 'Bob Ross'
```