CMPUT 274

Classes

Topics Covered:

- Procedural vs. Object-oriented programming
- Define new class
- Instantiate new object
- Encapsulation

Procedural vs Object-Oriented

- **Procedural programming:**
  - Emphasis on actions (verb)
  - e.g. roll dice *n* times, build a table of data

- **Object-oriented programming:**
  - Emphasis on objects (noun) with properties and behaviours
  - Allows us to model real-world objects
  - e.g. car, dog, student

Example: Dogs

- Unique property values define each dog
  e.g. age, colour, size

- Common behaviours
  e.g. bark, wag tail

Python Class

- Class is template/blueprint
- Defines all the attributes (properties) and methods (behaviours) that an object will have

Attributes:
age
size
colour

Behaviours:
bark
wag_tail

- Object is an instance of a class
- Gives values to all the attributes
- The attributes values of one object differentiates it from other objects that are instances of the same class

Example: Define New Class

```python
# dice.py
import random

class Dice:
  def __init__(self):
    self.sides = 6

  def roll(self):
    return random.randint(1,self.sides)

  def __str__(self):
    return 'Die has ' + str(self.sides) + ' sides.'
```

Class name (convention: capitalized)

method definition

attribute

No need to pass attribute to method inside class definition

Example: Instantiate Object

```python
# use_dice.py
from dice import Dice

def play():

    # create new dice object
    my_die = Dice()

    # roll my dice three times
    print('Roll 1:', my_die.roll())
    print('Roll 2:', my_die.roll())
    print('Roll 3:', my_die.roll())

    # display object
    print(my_die)

if __name__ == "__main__":
    play()
```

Calls __init__ method

Use **dot operator** on object to invoke method

Calls __str__ method

```
Roll 1: 3
Roll 2: 4
Roll 3: 1
Die has 6 sides.
```

`__init__()`

- Special method; typically used to initialize attributes for the new object that is created
- Automatically called when an object is instantiated

  → i.e. when name of class is called
- May also be known as *constructor* method

  → not quite accurate:

  https://www.programiz.com/article/python-self-why

`__str__()` and `__repr__()`

- Both are used to represent an object
- Good idea to define at least one

- __str__ returns the informal string representation of an instance
- __str__ is called by the built-in functions str() and print()

- __repr__ returns an official string representation of an instance
- __repr__ is called by the built-in function repr()

`self` Parameter

- First parameter in every class method
- Refers to the object itself
- Don’t include as argument when invoking method of object

  → self is passed implicitly when using the dot operator on the object

Example: Make Dice Class More General

```python
# dice2.py
import random

class Dice:
  def __init__(self, howMany):
    self.sides = howMany

  def roll(self):
    return random.randint(1,self.sides)

  def __str__(self):
    return 'Die has ' + str(self.sides) + ' sides.'
```

Pass in additional value(s) to initialize attribute(s)

Example continued…

```
# use_dice2.py
from dice2 import Dice

def play():

    # create new dice objects
    cube_die = Dice(6)
    icosahedron_die = Dice(20)

    # roll dice
    print('Cube roll:', cube_die.roll())
    print('Icosahedron roll:', icosahedron_die.roll())

    # display objects
    print(cube_die)
    print(icosahedron_die)

if __name__ == "__main__":
    play()
```

```
Cube roll: 1
Icosahedron roll: 11
Die has 6 sides.
Die has 20 sides.
```

Encapsulation

- A class wraps up or encapsulates its attributes and methods
  - Ensures that all data related to an object is contained in a single structure
- Attributes can be made private to prevent them being accessed directly by outside programs
  - Define attribute name with 2 underscores at beginning
  - e.g. `self.__sides`
- Implement setter and getter methods to change and access attributes

  → control HOW attribute values can be changed and seen

  → form public interface between program and object

Encapsulation Example

- Traffic Light
- Properties:
  - current colour
- Behaviours:
  - change colour