OOP Cheat Sheet

Object Oriented Programming (OOP)

```
# Defining a new class
class Robot:
  """
   This class implements a Robot.
  """
  population = 0  # class attribute

  # this is the constructor
  # it gets executed automatically each time an object is created
  def __init__(self, name=None, year=None):
     self.name = name           # instance attribute, default None
     self.year = year           # instance attribute, default None
     Robot.population += 1     # incrementing the class attribute when creating a new object

  # this is de destructor, it's automatically called then the object gets out of scope
  # MOST of the time it's not recommended to implement it (Python has a garbage collector)
  def __del__(self):
     # print('Robot died')
     pass

  def set_energy(self, energy):
     self.energy = energy    # instance attribute, set in methods

  def __str__(self):    # magic method, called automatically when printing the object
     return f'Name: {self.name}, Built Year: {self.year}'

  def __add__(self, other):    # magic method, called automatically when adding 2 objects
     return self.energy + other.energy


r0 = Robot()    # creating an instance with attributes set to default None
print(r0)          # => Name: None, Built Year: None

r1 = Robot('R2D2', 2030)    # creating an instance (object)
print(r1.__doc__)               # => This class implements a Robot. -> class docstring
print('Robot name:', r1.name)   # => R2D2 -> accessing an instance attribute
print(r1.__dict__)              # => {'name': 'R2D2', 'year': 2030} -> dictionary with instance attributes

r1.set_energy(555)        # creating the "energy" attribute
print(r1.energy)          # => 555

print(Robot.population)    # => 2
print(r1.population)       # => 2

r1.population += 10        # creating an instance attribute (doesn't modify the class attribute)
print(Robot.population)    # => 2
print(r1.population)       # => 12
```