```python
import networkx as nx
import matplotlib.pyplot as plt
import random as rnd

G1 = nx.Graph()

## Add some nodes and some edges
## Note: nodes can be (almost) any object, including inages and graphs
for node_name in ['armin', 130, 20.5]:
    G1.add_node(node_name, color= 'red')

G1.add_edge('armin', 130)
G1.add_edge('armin', 20.5)

print(list(G1.nodes()))
print(list(G1.edges()))

## Add some integer nodes via edge adding
for x in range(10):
    for y in range(10):
        if x != y:
            prob = rnd.random()
            if prob < 0.6:
                G1.add_edge(x,y)

G1.add_edge('armin', 2)

print(list(G1.nodes()))
print(list(G1.edges()))

# remove some nodes
G1.remove_node(1)
G1.remove_nodes_from([3,4])

# use matplotlib to drow the network
color_list = ['r', 'g', 'b', 'y', 'c', 'orange', 'b', 'b', 'b', 'b']
size_list = [1000, 900, 800, 700, 600, 500, 400, 300, 200, 100]
nx.draw(G1, with_labels=True, node_color=color_list, node_size = size_list)
plt.show()

# analyze the graph
print("Degree of node 5:", G1.degree(5))
print("Degree of node armin:", G1.degree('armin'))
print(G1.degree([5,'armin']))

G1.remove_node(2)
print(list(nx.connected_components(G1)))
```

```python
class Dog:
    """A simple attempt to model a dog."""

    def __init__(self, name, age):
        """Initialize name and age attributes."""
        self.name = name
        self.age = age

    def sit(self):
        """Simulate a dog sitting on command"""
        print(f"{self.name} is now sitting.")

    def roll_over(self, n=1):
        """Simulate a dog rolling over n times"""
        if n==1:
            plural_marker = ""
        else:
            plural_marker = "s"
        print(f"{self.name} rolled over {n} time{plural_marker}.")

    def increase_age(self):
        """Increase the age by one"""
        print(f"Happy birthday, {self.name}!")
        self.age += 1


class RacingDog(Dog):
    """A simple attempt to model a racing dog."""

    def __init__(self, name, age):
        """Initialize attributes of parent class."""
        super().__init__(name, age)
        self.max_speed = 80

    def increase_speed(self, value):
        """Increase the dog's speed by value"""
        self.max_speed += value

    def roll_over(self, n=1):
        """Simulate a dog rolling over n times"""
        print(f"{self.name} is not allowed to roll over.")
```

```python
from ClassDog import *

# 1. Introducing my dog
print("\n*Introducing my dog*")
my_dog = Dog('Willie', 6)
print(f"My dog's name is {my_dog.name}.")
print(f"{my_dog.name} is {my_dog.age} years old.")
my_dog.sit()
my_dog.roll_over(2)

# 2. Introducing your dog
print("\n*Introducing your dog*")
your_dog = Dog('Lucy', 3)
print(f"Your dog's name is {your_dog.name}.")
print(f"{your_dog.name} is {your_dog.age} years old.")
your_dog.sit()
your_dog.roll_over()

# 3. Celebrating Lucy's birthday
print("\n*Celebrating Lucy's birthday*")
your_dog.increase_age()
print(f"{your_dog.name} is {your_dog.age} years old.")

# 4. Adding a racing dog
print("\n*Introducing her dog*")
her_dog = RacingDog('Speedy Gonzales', 4)
print(f"Her dog's name is {her_dog.name}.")
print(f"{her_dog.name} is {her_dog.age} years old.")
her_dog.sit()
print(f"{her_dog.name} has a maximal speed of {her_dog.max_speed} kmh.")

# 5. Increasing the racing speed
print("\n*Increasing the racing speed*")
her_dog.increase_speed(5)
print(f"{her_dog.name} has a maximal speed of {her_dog.max_speed} kmh.")

# 6. Let Speedy roll over
print("\n*Let Speedy roll over*")
her_dog.roll_over()

# 7. Try to apply RacingDog features on a Dog class object
#print(f"{my_dog.name} has a maximal speed of {my_dog.max_speed} kmh.")

```