```python
class Neuron:
    def __init__(self, num_inputs):
        self.output = 0
        self.weights = []
        for i in range(num_inputs):
            # Initialize a neuron with random weights
            self.weights.append(randint(1, 10))

    # Calculate the weighted sum of inputs
    def adder(self, inputs):
        weighted_sum = 0
        for x, w in zip(inputs, self.weights):
            weighted_sum += x * w
        return weighted_sum

    # Apply ReLU activation function
    def relu_act(self, value):
        if value > 0:
            return value
        else:
            return 0

    # Activate the neuron with input values (for printing)
    def activate(self, inputs):
        sum_inputs = self.adder(inputs)
        self.output = self.relu_act(sum_inputs)
        return self.output

    # Print the output value of the neuron
    def print_output(self):
        print("Neuron Output:", self.output)
```

```python
from random import randint

# Generate a list of random values within a specified range
def generate_values(length, low, high):
    list_values = []
    for i in range(length):
        list_values.append(randint(low, high))
    return list_values

input_size = 5

# Generate input values
input_values = generate_values(input_size, 1, 10)
input_values
```

```
[5, 3, 6, 4, 9]
```

```python
# Create a list of neurons with specified number of neurons and inputs
def create_neurons(num_neurons, num_inputs):
    list_neurons = []
    for i in range(num_neurons):
        list_neurons.append(Neuron(num_inputs))
    return list_neurons

num_neurons_layer1 = 5

# Create neurons for the first hidden layer
neurons_HidLayer1 = create_neurons(num_neurons_layer1, input_size)

# Activate neurons in the first hidden layer and print their output
for neuron in neurons_HidLayer1:
    neuron.activate(input_values)
    neuron.print_output()
```

```
Neuron Output: 165
Neuron Output: 147
Neuron Output: 68
Neuron Output: 145
Neuron Output: 119
```

```python
num_neurons_layer2 = 2

# Create neurons for the second hidden layer
neurons_HidLayer2 = create_neurons(num_neurons_layer2, input_size)

# Create the output neuron
output_Neuron = Neuron(num_neurons_layer2)

class Graph:
    def __init__(self):
        self.graph = {}

    def add_edge(self, source, destination):
        if source not in self.graph:
            self.graph[source] = []
        else:
            self.graph[source].append(destination)

    def print_graph(self):
        for source, destinations in self.graph.items():
            print(f"{source} --> {destinations}")

myG = Graph()
```

```
# for neuron in neurons_HidLayer1:
#     myG.add_edge(input_values, neuron)
#
----------------------------------------------------------------------
-----
# TypeError                                 Traceback (most recent
call last)
# <ipython-input-127-39eaf16cd19e> in <cell line: 1>()
#       1 for neuron in neurons_HidLayer1:
# ----> 2     myG.add_edge(input_values, neuron)

# <ipython-input-121-d6c92d1fde42> in add_edge(self, source,
destination)
#       4
#       5     def add_edge(self, source, destination):
# ----> 6         if source not in self.graph:
#       7             self.graph[source] = []
#       8         else:

# TypeError: unhashable type: 'list'

# Add edges from input values to neurons in the first hidden layer
for neuron in neurons_HidLayer1:
    myG.add_edge(tuple(input_values), neuron)

# Add edges from neurons in the first hidden layer to neurons in the
second hidden layer
for neuron in neurons_HidLayer2:
    myG.add_edge(tuple(neurons_HidLayer1), neuron)

# Add edge from neurons in the second hidden layer to the output
neuron
myG.add_edge(tuple(neurons_HidLayer2), output_Neuron)

myG.print_graph()
# WHY IS LIFE?

(5, 3, 6, 4, 9) --> [<__main__.Neuron object at 0x7da5fce1be50>,
<__main__.Neuron object at 0x7da5fce18df0>, <__main__.Neuron object at
0x7da5fce1a5c0>, <__main__.Neuron object at 0x7da5fce19630>,
<__main__.Neuron object at 0x7da5fce1b400>, <__main__.Neuron object at
0x7da5fce1be50>, <__main__.Neuron object at 0x7da5fce18df0>,
<__main__.Neuron object at 0x7da5fce1a5c0>, <__main__.Neuron object at
0x7da5fce19630>]
(<__main__.Neuron object at 0x7da5fce1b400>, <__main__.Neuron object
at 0x7da5fce1be50>, <__main__.Neuron object at 0x7da5fce18df0>,
<__main__.Neuron object at 0x7da5fce1a5c0>, <__main__.Neuron object at
0x7da5fce19630>) --> [<__main__.Neuron object at 0x7da5fce1ac20>,
<__main__.Neuron object at 0x7da5fce1a080>, <__main__.Neuron object at
0x7da5fce1ac20>]
```

```
(<__main__.Neuron object at 0x7da5fce1a080>, <__main__.Neuron object
at 0x7da5fce1ac20>) --> [<__main__.Neuron object at 0x7da5fce1b2b0>]
```