Bangladesh University of Business and Technology

Final Lab Report

Course : Artificial Intelligence Lab (CSE 352)

Problem Name:

1. Write a program on how to Find Breadth First Search (BFS)Algorithm in python .
2. Write a program on how to find the Depth-First Search (DFS) Algorithm in python .
3. Write a program on how to find the generate and test random variable in python.
4. Write a program in python to find perception (OR).

Submitted by:

Name : Md. Rashidul Haq
Id : 19202103356
Intake : 44
Sec : 4
Email:19202103356@cse.bubt.edu.bd

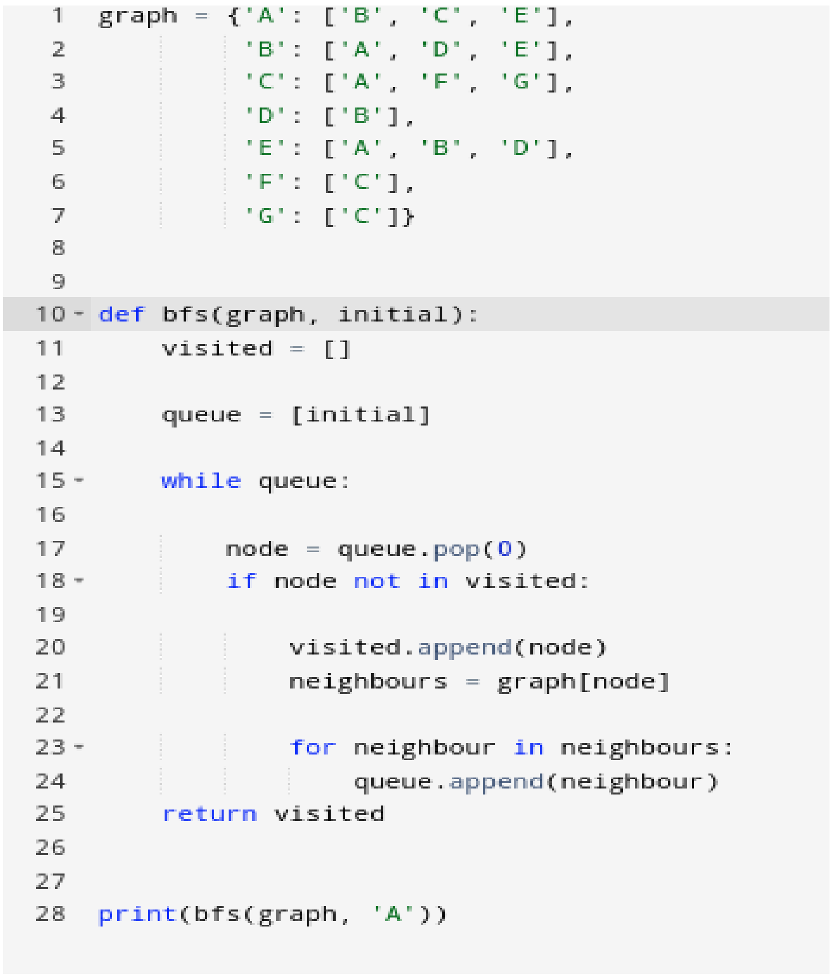
Submitted To:

Humayra Ahmed
(Assistant Professor)

Bangladesh University of Business & Technology (BUBT)

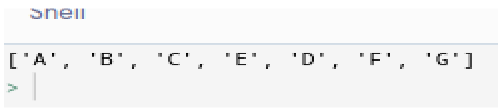
1. Write a program on how to Find Breadth First Search (BFS)Algorithm in python .

Code:



```python
graph = {'A': ['B', 'C', 'E'],
         'B': ['A', 'D', 'E'],
         'C': ['A', 'F', 'G'],
         'D': ['B'],
         'E': ['A', 'B', 'D'],
         'F': ['C'],
         'G': ['C']}


def bfs(graph, initial):
    visited = []

    queue = [initial]

    while queue:

        node = queue.pop(0)
        if node not in visited:

            visited.append(node)
            neighbours = graph[node]

            for neighbour in neighbours:
                queue.append(neighbour)
    return visited


print(bfs(graph, 'A'))
```

Output:

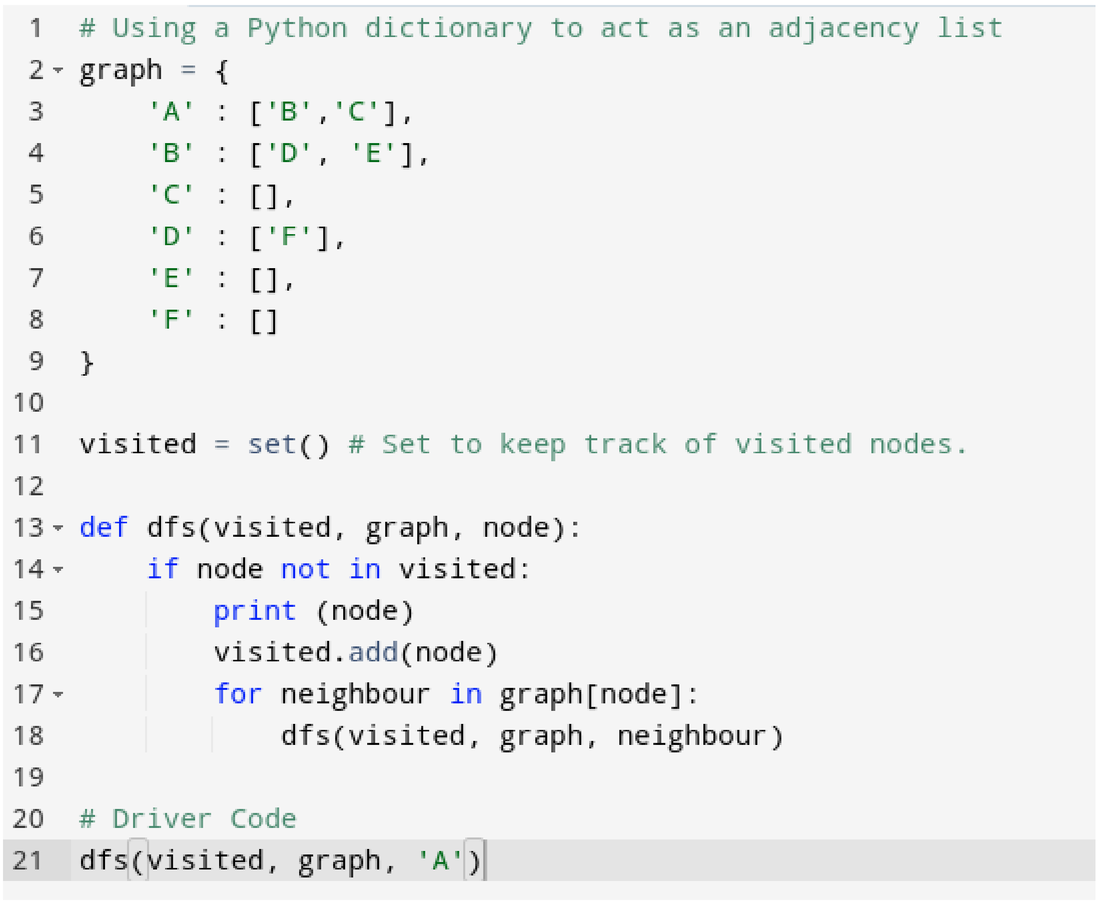


```
['A', 'B', 'C', 'E', 'D', 'F', 'G']
>
```

Discussion:

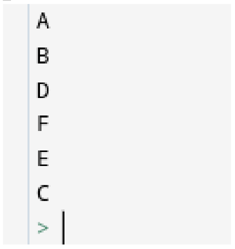
★ Create a graph.Initialize a starting node.Send the graph and initial node as parameters to the function.
★ Mark the initial node as visited and push it into the queue.
★ Explore the initial node and add its neighbors to the queue and remove the initial node from the queue.
★ Check if the neighbors node of a neighboring node is already visited. If not, visit the neighboring node neighbors and mark them as visited.
★ Repeat this process until all the nodes in a graph are visited and the queue becomes empty.

2. Write a program on how to find the Depth-First Search (DFS) Algorithm in python .
Code:



```python
# Using a Python dictionary to act as an adjacency list
graph = {
      'A' : ['B','C'],
      'B' : ['D', 'E'],
      'C' : [],
      'D' : ['F'],
      'E' : [],
      'F' : []
}

visited = set() # Set to keep track of visited nodes.

def dfs(visited, graph, node):
    if node not in visited:
        print (node)
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour)

# Driver Code
dfs(visited, graph, 'A')
```

Output:



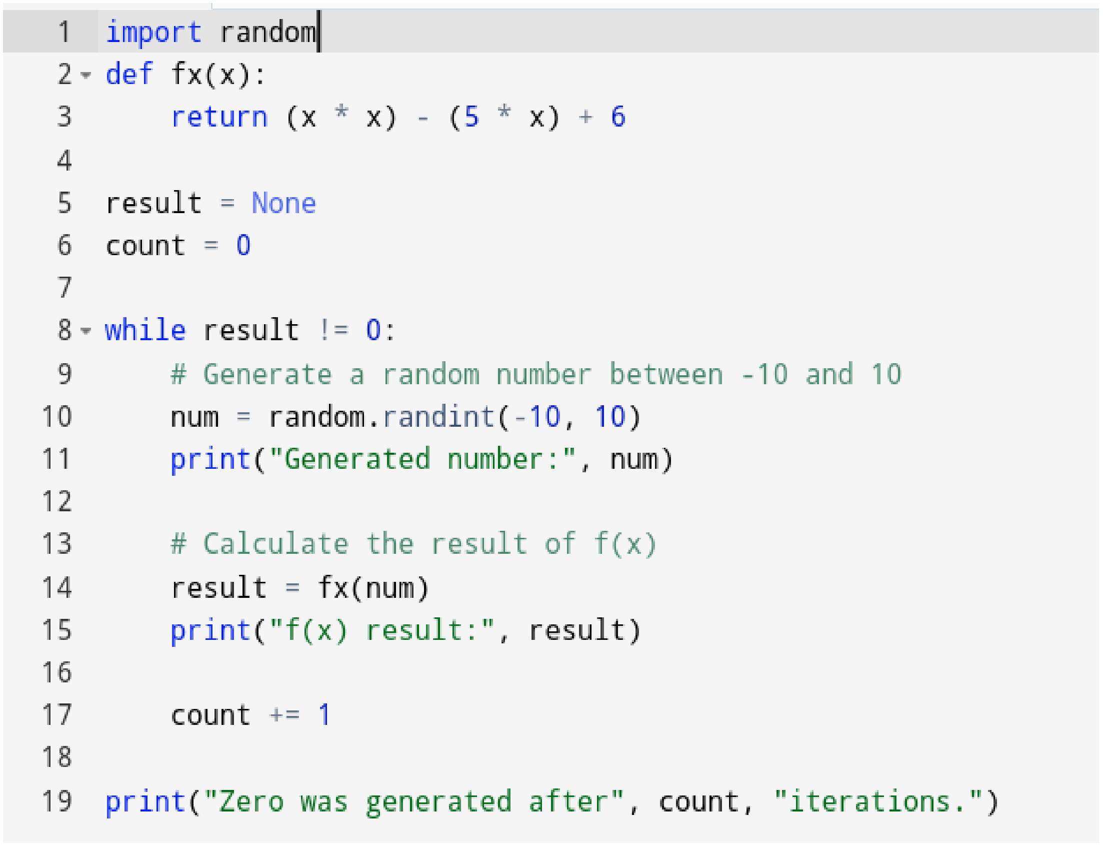
```
A
B
D
F
E
C
>
```

Discussion:

★ The illustrated graph is represented using an adjacency list - an easy way to do it in Python is to use a dictionary data structure.
★ Each vertex has a list of its adjacent nodes stored. visited is a set that is used to keep track of visited nodes.
★ The dfs function is called and is passed the visited set, the graph in the form of a dictionary, and A, which is the starting node.
★ dfs follows the algorithm described above: It first checks if the current node is unvisited - if yes, it is appended in the visited set.
★ Then for each neighbor of the current node, the dfs function is invoked again. The base case is invoked when all the nodes are visited. The function then returns.

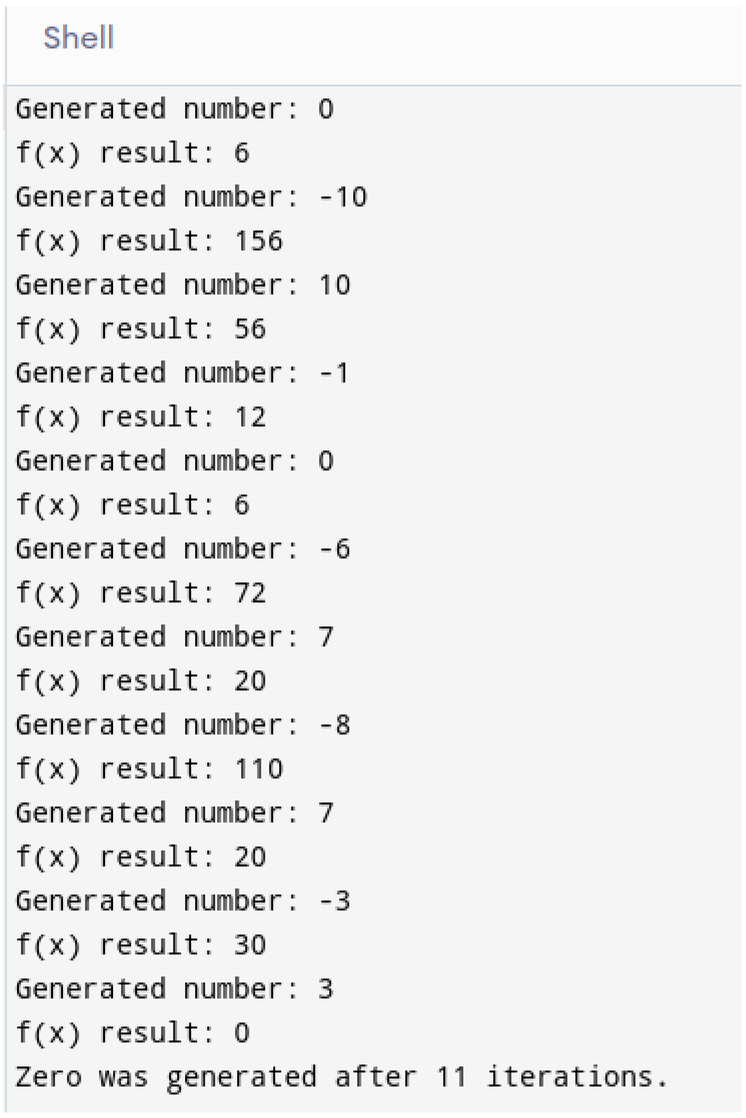
3. Write a program on how to find the generate and test random variable in python.

Code:



```python
import random
def fx(x):
    return (x * x) - (5 * x) + 6

result = None
count = 0

while result != 0:
    # Generate a random number between -10 and 10
    num = random.randint(-10, 10)
    print("Generated number:", num)

    # Calculate the result of f(x)
    result = fx(num)
    print("f(x) result:", result)

    count += 1

print("Zero was generated after", count, "iterations.")
```

Output:

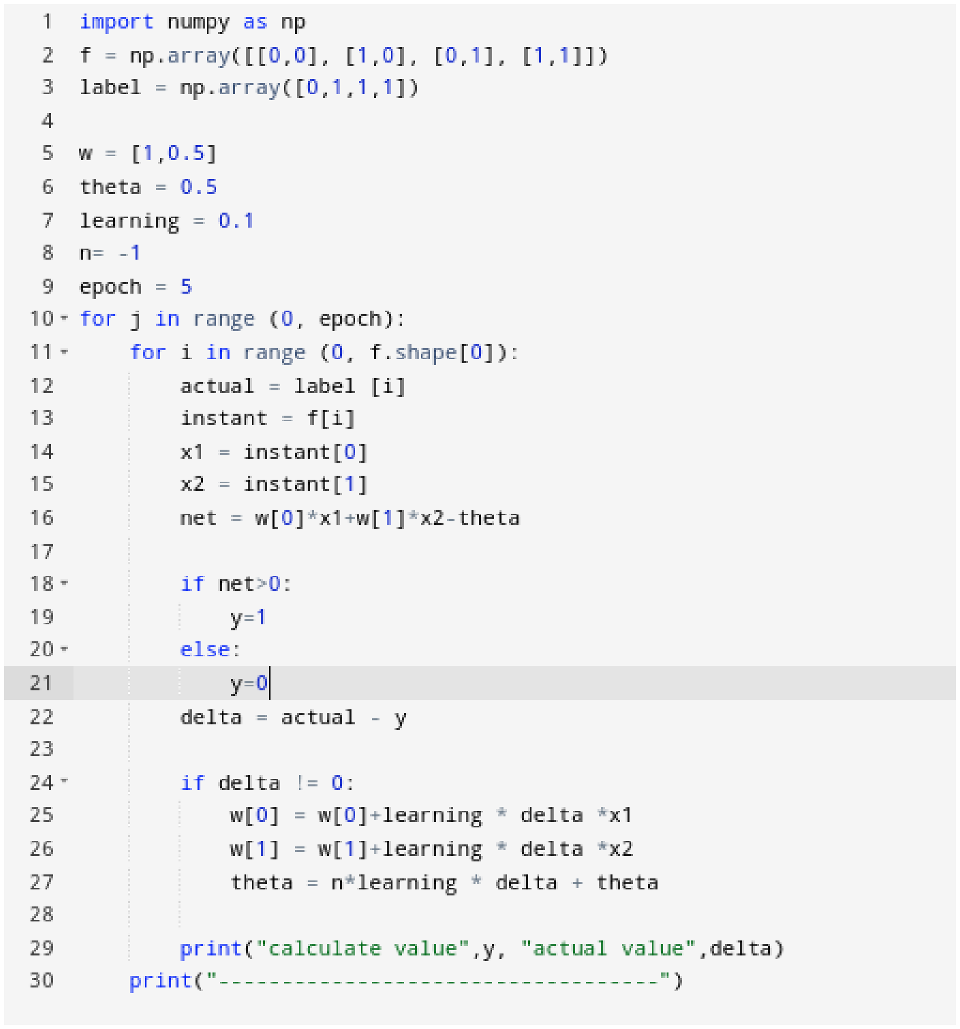

Shell

```
Generated number: 0
f(x) result: 6
Generated number: -10
f(x) result: 156
Generated number: 10
f(x) result: 56
Generated number: -1
f(x) result: 12
Generated number: 0
f(x) result: 6
Generated number: -6
f(x) result: 72
Generated number: 7
f(x) result: 20
Generated number: -8
f(x) result: 110
Generated number: 7
f(x) result: 20
Generated number: -3
f(x) result: 30
Generated number: 3
f(x) result: 0
Zero was generated after 11 iterations.
```

Discussion:

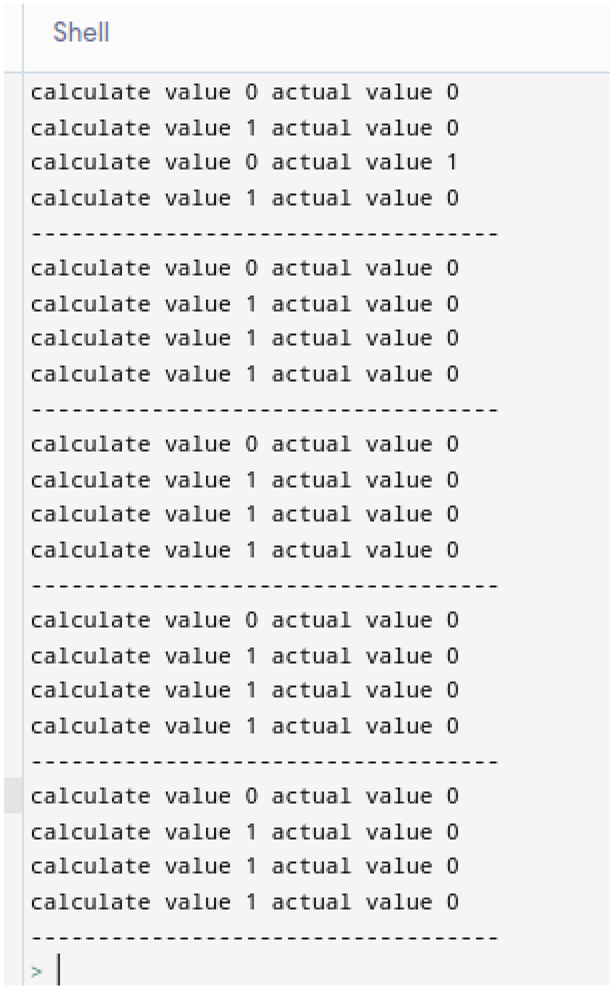
- Let the random variable X assume the values x1, x2, ...with correspondingprobability P (x1), P (x2),... then the expected value of the random variable isgiven by: Expectation of X, E (x) = ∑ x P (x).
- Returns a random float number between 0 and 1.
- Returns a random float number between two given parameters.Create an empty list which is the resultant random numbers list.
- Use the for loop, to traverse the loop 15 times. Use the randint() function(Returns a random number within the specified range) of the random
module, to generate a random number in the range in specified range from 1 to 100.

4. Write a program in python to find perception (OR).
Code:



```python
import numpy as np
f = np.array([[0,0], [1,0], [0,1], [1,1]])
label = np.array([0,1,1,1])

w = [1,0.5]
theta = 0.5
learning = 0.1
n= -1
epoch = 5
for j in range (0, epoch):
    for i in range (0, f.shape[0]):
        actual = label [i]
        instant = f[i]
        x1 = instant[0]
        x2 = instant[1]
        net = w[0]*x1+w[1]*x2-theta

        if net>0:
            y=1
        else:
            y=0
        delta = actual - y

        if delta != 0:
            w[0] = w[0]+learning * delta *x1
            w[1] = w[1]+learning * delta *x2
            theta = n*learning * delta + theta

        print("calculate value",y, "actual value",delta)
    print("-----------------------------------")
```

Output:



```
Shell
calculate value 0 actual value 0
calculate value 1 actual value 0
calculate value 0 actual value 1
calculate value 1 actual value 0
-----------------------------------
calculate value 0 actual value 0
calculate value 1 actual value 0
calculate value 1 actual value 0
calculate value 1 actual value 0
-----------------------------------
calculate value 0 actual value 0
calculate value 1 actual value 0
calculate value 1 actual value 0
calculate value 1 actual value 0
-----------------------------------
calculate value 0 actual value 0
calculate value 1 actual value 0
calculate value 1 actual value 0
calculate value 1 actual value 0
-----------------------------------
calculate value 0 actual value 0
calculate value 1 actual value 0
calculate value 1 actual value 0
calculate value 1 actual value 0
-----------------------------------
> |
```

Discussion:

★ First, we will look at the Unit Step Function and see how the Perceptron Algorithm classifies and then have a look at the perceptron update rule.

★ Finally, we will plot the decision boundary for our data.

★ We will use the data with only two features, and there will be two classes since Perceptron is a binary classifier.

★ Another feature of Python is that it is intuitive and is perfect for a collaborative deployment