

# GeeksforGeeks

A computer science portal for geeks

[Practice](#)[GATE CS](#)[Placements](#)[Videos](#)[Contribute](#)[Login/Register](#)

## Quick Links for Python

[Recent Articles](#)[MCQ / Quizzes](#)[Practice Problems](#)

## Basics

[Introduction](#)[New Generation Language](#)[Keywords, Set 1 Set 2](#)[Explore More...](#)

## Variables

[Variables, Expressions & Functions](#)[Global and Local Variables](#)[Type Conversion](#)[Explore More...](#)


## Operators

[Increment and Decrement Operator](#)[Teranry Operator & Divison Operator](#)[Logical and Bitwise Not Operators on Boolean](#)[Any & ALL](#)

Operator Functions Set 1 & Set 2
<b>Data Types</b>
Introduction
Arrays Set 1, Set 2
String Methods Set 1, Set 2, Set 3
String Template Class & String Formatting using %
List Methods Set 1, Set 2, Set 3
Tuples & Sets
Dictionary Methods Set 1, Set 2
ChainMap
Explore More...
<b>Control Flow</b>
Loops and Control Statements
Counters & Accessing Counters
Iterators & Iterator Functions Set 1, Set 2
Generators
Explore More...
<b>Functions</b>
Function Decorators
Returning Multiple Values
Yield instead of Return
Python Closures & Coroutine
Explore More...
<b>Modules</b>
Introduction



Numeric Functions & Logarithmic and Power functions	
Calender Functions Set 1, Set 2	
Complex Numbers Introduction & Important functions	
Explore More...	
<b>Object Oriented Concepts</b>	
Class, Object and Members	
Data Hiding and Object Printing	
Inheritance, Subclass and super	
Class method vs static method & Class or Static Variables	
Explore More...	
<b>Exception Handling</b>	
Exception Handling	
User-Defined Exceptions	
Built-in Exceptions	
<b>Libraries and Functions</b>	
Timeit	
Numpy Set 1, Set 2	
Get and Post	
import module & reload module	
Collection Modules Deque, Namedtuple & Heap	
Explore More...	
<b>Machine Learning with Python</b>	
Classifying data using Support Vector Machines(SVMs) in Python	



K means Clustering
How to get synonyms/antonyms from NLTK WordNet in Python?
Explore More...
<b>Misc</b>
Sql using Python & MongoDB and Python
Json formatting & Python Virtual environment
Metaprogramming with Metaclasses in Python
Python Input Methods for Competitive Programming
Explore More...
<b>Applications and Projects</b>
Creating a proxy webserver Set 1, Set 2
Send Message to FB friend
Twitter Sentiment Analysis & Whatsapp using Python
Desktop Notifier & Junk File Organizer
Explore More...

## Generate a graph using Dictionary in Python

Prerequisite – [Graphs](#)

To draw graph using in built libraries – [Graph plotting in Python](#)

In this article, we will see how to implement graph in python using [dictionary](#) data structure in python.

The keys of the dictionary used are the nodes of our graph and the corresponding values are lists with each nodes, which are connecting by an edge.

This simple graph has six nodes (a-f) and five arcs:

```
a -> c
```

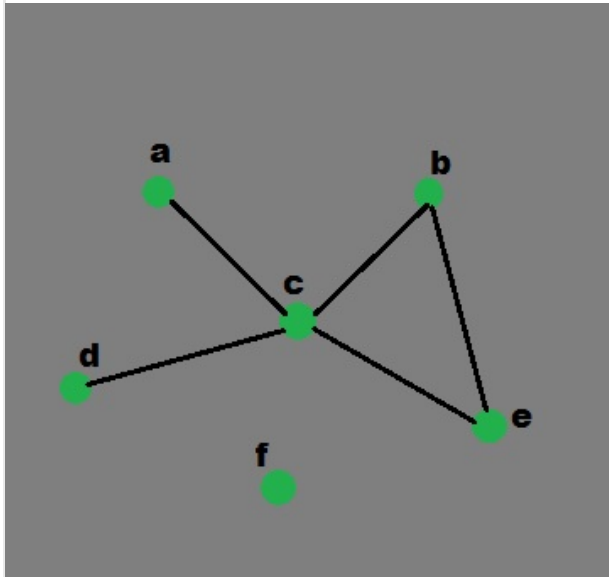


```
b -> c
b -> e
c -> a
c -> b
c -> d
c -> e
d -> c
e -> c
e -> b
```

It can be represented by the following Python data structure. This is a dictionary whose keys are the nodes of the graph. For each key, the corresponding value is a list containing the nodes that are connected by a direct arc from this node.

```
graph = { "a" : ["c"],
          "b" : ["c", "e"],
          "c" : ["a", "b", "d", "e"],
          "d" : ["c"],
          "e" : ["c", "b"],
          "f" : []
        }
```

**Graphical representation of above example:**



**defaultdict:** Usually, a Python dictionary throws a `KeyError` if you try to get an item with a key that is not currently in the dictionary. `defaultdict` allows that if a key is not found in the dictionary, then instead of a `KeyError` being thrown, a new entry is created. The type of this new entry is given by the argument of `defaultdict`.

**Python Function to generate graph:**

```
# definition of function
def generate_edges(graph):
    edges = []

    # for each node in graph
```



```

for node in graph:

    # for each neighbour node of a single node
    for neighbour in graph[node]:
        # if edge exists then append
        edges.append((node, neighbour))
return edges

```

**Recommended: Please try your approach on {IDE} first, before moving on to the solution.**

```

# Python program for
# validation of a graph

# import dictionary for graph
from collections import defaultdict

# function for adding edge to graph
graph = defaultdict(list)
def addEdge(graph,u,v):
    graph[u].append(v)

# definition of function
def generate_edges(graph):
    edges = []

    # for each node in graph
    for node in graph:

        # for each neighbour node of a single node
        for neighbour in graph[node]:

            # if edge exists then append
            edges.append((node, neighbour))
    return edges

# declaration of graph as dictionary
addEdge(graph,'a','c')
addEdge(graph,'b','c')
addEdge(graph,'b','e')
addEdge(graph,'c','d')
addEdge(graph,'c','e')
addEdge(graph,'c','a')
addEdge(graph,'c','b')
addEdge(graph,'e','b')
addEdge(graph,'d','c')
addEdge(graph,'e','c')

# Driver Function call
# to print generated graph
print(generate_edges(graph))

```

[Run on IDE](#)

Output:

```

[('a', 'c'), ('c', 'd'), ('c', 'e'), ('c', 'a'), ('c', 'b'),
 ('b', 'c'), ('b', 'e'), ('e', 'b'), ('e', 'c'), ('d', 'c')]

```

As we have taken example of undirected graph, so we have print same edge twice say as ('a','c') and ('c','a'). We can overcome this with use of directed graph.

Below are some more programs on graphs in python:

### 1. To generate the path from one node to the other node:

Using Python dictionary, we can find the path from one node to the other in a Graph. The idea is similar to DFS in graphs.

In the function, initially, the path is an empty list. In the starting, if the start node matches with the end node, the function will return the path. Otherwise the code goes forward and hits all the values of the starting node and searches for the path using recursion. If there is no such path, it returns None.

```
# Python program to generate the first
# path of the graph from the nodes provided

graph = {
    'a': ['c'],
    'b': ['d'],
    'c': ['e'],
    'd': ['a', 'd'],
    'e': ['b', 'c']
}

# function to find path
def find_path(graph, start, end, path = []):
    path = path + [start]
    if start == end:
        return path
    for node in graph[start]:
        if node not in path:
            newpath = find_path(graph, node, end, path)
            if newpath:
                return newpath
    return None

# Driver function call to print the path
print(find_path(graph, 'd', 'c'))
```

[Run on IDE](#)

Output:

```
['d', 'a', 'c']
```

### 2. Program to generate all the possible paths from one node to the other.:

In the above discussed program, we generated the first possible path. Now, let us generate all the possible paths from the start node to the end node. The basic functioning works same as the functioning of the above code. The place where the difference comes is instead of instantly returning the first path, it saves that path in a list named as 'paths' in the example given below. Finally, after iterating over all the possible ways, it returns the list of paths. If there is no path from the starting node to the ending node, it returns None.

```
# Python program to generate the all possible
# path of the graph from the nodes provided
graph = {
```



```

'a':['c'],
'b':['d'],
'c':['e'],
'd':['a', 'd'],
'e':['b', 'c']
}

# function to generate all possible paths
def find_all_paths(graph, start, end, path =[]):
    path = path + [start]
    if start == end:
        return [path]
    paths = []
    for node in graph[start]:
        if node not in path:
            newpaths = find_all_paths(graph, node, end, path)
            for newpath in newpaths:
                paths.append(newpath)
    return paths

# Driver function call to print all
# generated paths
print(find_all_paths(graph, 'd', 'c'))

```

[Run on IDE](#)

Output:

```
[[ 'd', 'a', 'c'], [ 'd', 'a', 'c']]
```

### 3. Program to generate the shortest path.:

To get to the shortest from all the paths, we use a little different approach as shown below. In this, as we get the path from the start node to the end node, we compare the length of the path with a variable named as shortest which is initialized with the None value. If the length of generated path is less than the length of shortest, if shortest is not None, the newly generated path is set as the value of shortest. Again, if there is no path, it returns None

# Python program to generate shortest path

```

graph = {
'a':['c'],
'b':['d'],
'c':['e'],
'd':['a', 'd'],
'e':['b', 'c']
}

# function to find the shortest path
def find_shortest_path(graph, start, end, path =[]):
    path = path + [start]
    if start == end:
        return path
    shortest = None
    for node in graph[start]:
        if node not in path:
            newpath = find_shortest_path(graph, node, end, path)

```





```
        if newpath:
            if not shortest or len(newpath) < len(shortest):
                shortest = newpath
    return shortest

# Driver function call to print
# the shortest path
print(find_shortest_path(graph, 'd', 'c'))
```

[Run on IDE](#)

Output:

```
['d', 'a', 'c']
```

This article is contributed by **Shivam Pradhan (anuj\_charm)** and **Rishabh Bansal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](https://contribute.geeksforgeeks.org) or mail your article to [contribute@geeksforgeeks.org](mailto:contribute@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

## GATE CS Corner Company Wise Coding Practice

[Graph](#) [Python](#)

### Recommended Posts:

Number of pair of positions in matrix which are not accessible  
Junk File Organizer in Python  
[Graph Plotting in Python | Set 1](#)



Graph and its representations  
Depth First Traversal or DFS for a Graph

([Login](#) to Rate and Mark)

**3**

Average Difficulty : **3/5.0**  
Based on **1** vote(s)



Add to TODO List



Mark as DONE

Writing code in comment? Please use [ide.geeksforgeeks.org](http://ide.geeksforgeeks.org), generate link and share the link here.

Load Comments

Share this post!

@geeksforgeeks, Some rights reserved

[Contact Us!](#)

[About Us!](#)

[Careers!](#)

[Privacy Policy](#)

