数据结构与算法（Python版）

图抽象数据类型的Python实现

陈斌 北京大学 gischen@pku.edu.cn

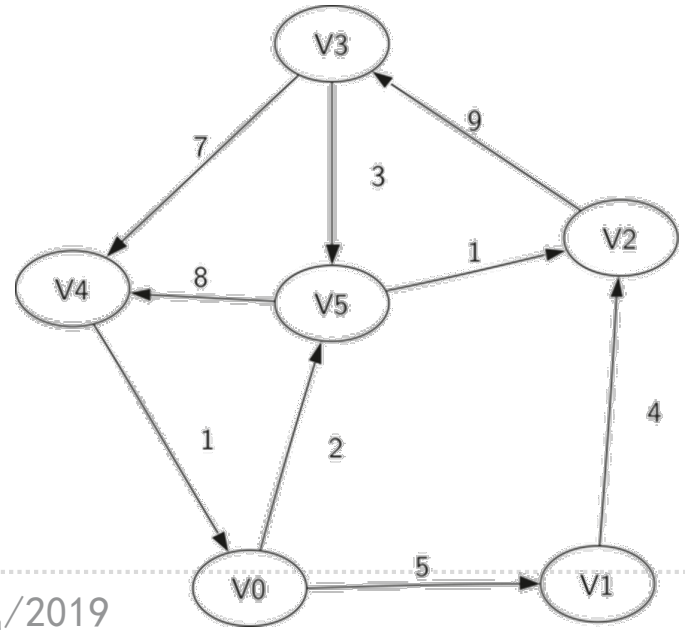
ADT Graph的实现：实例


```
>>> g= Graph()
>>> for i in range(6):
        g.addVertex(i)


0 connectedTo: []
1 connectedTo: []
2 connectedTo: []
3 connectedTo: []
4 connectedTo: []
5 connectedTo: []
>>> print g.vertList
{0: 0 connectedTo: [], 1: 1 connectedTo: [], 2: 2 conne
dTo: [], 4: 4 connectedTo: [], 5: 5 connectedTo: []}
```

ADT Graph的实现：实例

```
>>> g.addEdge(0,1,5)
>>> g.addEdge(0,5,2)
>>> g.addEdge(1,2,4)
>>> g.addEdge(2,3,9)
>>> g.addEdge(3,4,7)
>>> g.addEdge(3,5,3)
>>> g.addEdge(4,0,1)
>>> g.addEdge(5,4,8)
>>> g.addEdge(5,2,1)
>>> for v in g:
        for w in v.getConnections():
                print "(%s, %s)" % (v.getId(), w.getId())
```

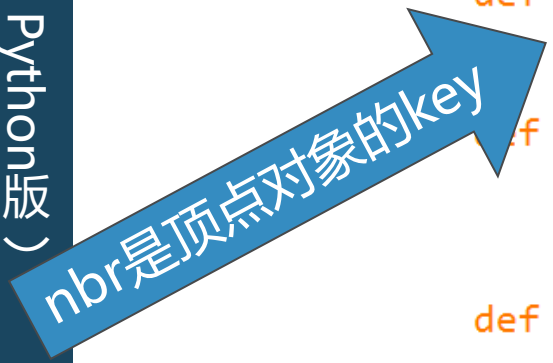
```
(0, 5)
(0, 1)
(1, 2)
(2, 3)
(3, 4)
(3, 5)
(4, 0)
(5, 4)
(5, 2)
>>>
```



ADT Graph的实现：顶点Vertex类

- Vertex包含了顶点信息，以及顶点连接边信息

```
class Vertex:
    def __init__(self,key):
        self.id = key
        self.connectedTo = {}

    def addNeighbor(self,nbr,weight=0):
        self.connectedTo[nbr] = weight

    def __str__(self):
        return str(self.id) + ' connectedTo: ' \
               + str([x.id for x in self.connectedTo])

    def getConnections(self):
        return self.connectedTo.keys()

    def getId(self):
        return self.id

    def getWeight(self,nbr):
        return self.connectedTo[nbr]
```



ADT Graph的实现：图Graph类

❖ Graph保存了包含所有顶点的主表

新加顶点

通过key查找顶点

```
class Graph:
    def __init__(self):
        self.vertList = {}
        self.numVertices = 0

    def addVertex(self,key):
        self.numVertices = self.numVertices + 1
        newVertex = Vertex(key)
        self.vertList[key] = newVertex
        return newVertex

    def getVertex(self,n):
        if n in self.vertList:
            return self.vertList[n]
        else:
            return None

    def __contains__(self,n):
        return n in self.vertList
```

ADT Graph的实现：图Graph类


```
def __contains__(self,n):
    return n in self.vertList

def addEdge(self,f,t,cost=0):
    if f not in self.vertList:
        nv = self.addVertex(f)
    if t not in self.vertList:
        nv = self.addVertex(t)
    self.vertList[f].addNeighbor(self.vertList[t], cost)

def getVertices(self):
    return self.vertList.keys()

def __iter__(self):
    return iter(self.vertList.values())
```

不存在的顶点先添加

调用起始顶点的方法添加邻接边