In [18]:

```
# Program to check weather a number is prime or not.
x=int(input("Enter a number:"))
y=int(x/2)+1
for i in range (2,y):
    rem=x%i
    if rem==0:
        print("Number",x,"is not a prime number.")
        break
else:
    print("Number",x,"is a prime number.")
```

```
Enter a number:59
Number 59 is a prime number.
```

In [5]:

```
# Program to check a number is palindrome or not.
def cekpal(n):
    num=n
    d=0
    rev=0
    while n>0:
        d=n%10
        n=int(n/10)
        rev=int(rev*10+d)
    if(num==rev):
        return True
    else:
        return False
#__main__
n=int(input("Enter a number:"))
if cekpal(n):
    print(n,"is a palindrome")
else:
    print(n,"is a not palindrome")
```

```
Enter a number:123454321
123454321 is a palindrome
```

In [18]:

```
# Program to calculate compound intrest.
def compint(p,r,n):
    m=n
    x=p*(1+r/100)**(n)
    return x
p=float(input("Enter the principal amount:"))
r=float(input("Enter the rate of intrest:"))
n=float(input("Enter the number of years:"))
x=compint(p,r,n)
print("The amount after",n,"years is",x,"rupees.")
```

```
Enter the principal amount:6250
Enter the rate of intrest:8
Enter the number of years:2
The amount after 2.0 years is 7290.000000000001 rupees.
```

In [26]:

```
# Program to display ASCII code of a character and vice versa.
def asc(a):
    print([ord(c) for c in a])
    def asc_val(a):
    char=[chr(ascii) for ascii in a]
    ''.join(char)
    print(char)
#__main__
print("Enter '1' for converting string in ascii code")
print("Enter '2' for converting ascii code in string")
x=int(input("Enter your choice:"))
if x==1:
    n=input("Enter your character:")
    asc(n)
elif x==2:
    n=eval(input("Enter your ASCII value:"))
    asc_val(n)
else:
    print("You have entered a in valid choice")
```

```
Enter '1' for converting string in ascii code
Enter '2' for converting ascii code in string
Enter your choice:2
Enter your ASCII value:[11,22,33,44,55,66,77,8,8,8,9,9,0,128]
['\x0b', '\x16', '!', ',', '7', 'B', 'M', '\x08', '\x08', '\x08', '\t',
'\t', '\x00', '\x80']
```

In [41]:

```
# Program to input a character and print weather a given character is an alphabet, digit or
ch = input("Please Enter Your Own Character : ")
if((ch >= 'a' and ch <= 'z') or (ch >= 'A' and ch <= 'Z')):
    print("The Given Character ", ch, "is an Alphabet")
elif(ch >= '0' and ch <= '9'):
    print("The Given Character ", ch, "is a Digit")
else:
    print("The Given Character ", ch, "is Not an Alphabet or a Digit")
```

```
Please Enter Your Own Character : 55
The Given Character  55 is a Digit
```

In [42]:

```
# Program to calculate the factorial of an integer using recursion.
def factorial(n):
   if n == 1:
       return n
   else:
       return n*factorial(n-1)
num = int(input("Enter a number: "))
if num < 0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   print("The factorial of",num,"is",factorial(num))
```

```
Enter a number: 5
The factorial of 5 is 120
```

In [24]:

```
# Program to print fibonacci series using recursion.
def fib(n):
    if n==1:
        return 0
    elif n==2:
        return 1
    else:
        return fib(n-1)+fib(n-2)
#__main__
n=int(input("Enter the last term required:"))
for i in range(1,n+1):
    print(fib(i),end=',')
print ("...")
```

```
Enter the last term required:8
0,1,1,2,3,5,8,13,...
```

In [31]:

```
# Program for binary search.
def binarysearch(ar,key):
    low=0
    high=len(ar)-1
    while low<=high:
        mid=int((low+high)/2)
        if key==ar[mid]:
            return mid
        elif key<ar[mid]:
            high=mid-1
        else:
            low=mid+1
    else:
        return-999
#__main__
ar=eval(input("Enter a list of integers:" ))
item=int(input("Enter search item:"))
res=binarysearch(ar,item)
if res>=0:
    print(item,"found at index",res,".")
else:
    print("Sorry",item,"not found in array.")
```

```
Enter a list of integers:[12,15,21,25,28,32,33,36,43,45]
Enter search item:32
32 found at index 5 .
```

In [44]:

```
#Program to find weather a string is palindrome or not using recursion.
def par(s):
    if len(s) < 1:
        return True
    else:
        if s[0] == s[-1]:
            return par(s[1:-1])
        else:
            return False
a=str(input("Enter string:"))
if (par(a)==True):
    print("String is a palindrome")
else:
    print("String isn't a palindrome")
```

```
Enter string:mam
String is a palindrome
```

In [50]:

```
#Program to count the noumber of vowels present in a text file.
file1=open("D:\\mytext.txt","r")
str1=file1.read()
vowel_count=0
for i in str1:
    if(i=='A' or i=='a' or i=='E' or i=='e' or i=='I' or i=='i'or i=='O' or i=='o' or i=='U
        vowel_count+=1
print("The number of vowels in the text file :",vowel_count)
file1.close()
```

```
The number of vowels in the text file : 1635
```

In [83]:

```
#Program to write those lines which have the character 'p' from one text  file to another t
fn=open("D:\\mytext.txt","r")
fn1=open("D:\\mytext1.txt","w")
count=fn.read()
type(count)
for i in range (0,len(count)):
    if count[i]=='p':
        fn1.write(count[i])
    else:
        pass
fn1.close
fn1=open("D:\\mytext1.txt","r")
count1=fn1.read()
print(count1)
fn.close
fn1.close
```

```
pppppppppppppppppppppppppppppppp
```

Out[83]:

```
<function TextIOWrapper.close()>
```

In [19]:

```
#Program to count number of words in a file.
num_words=0
with open("D:\\mytext.txt","r") as f:
    for line in f:
        words=line.split()
        num_words+=len(words)
print("Number of words:",num_words)
```

```
Number of words: 11
```

In [24]:

```
#Program to calculate the value of sin(x) using its Taylor series expansion upto n terms.
import math
def cal_sin(n):
    accuracy=0.0001
    n=n*(3.142/180.0)
    x=n
    sinx=n
    sinval=math.sin(n)
    i=1
    while (True):
        denominator=2*i*(2*i+1)
        x=-x*n*n/denominator
        sinx=sinx+x
        i=i+1
        if(accuracy<= abs(sinval-sinx)):
            break
    print(sinx)
n=int(input("Enter the angle:"))
cal_sin(n)
```

```
Enter the angle:45
0.7047230747708333
```

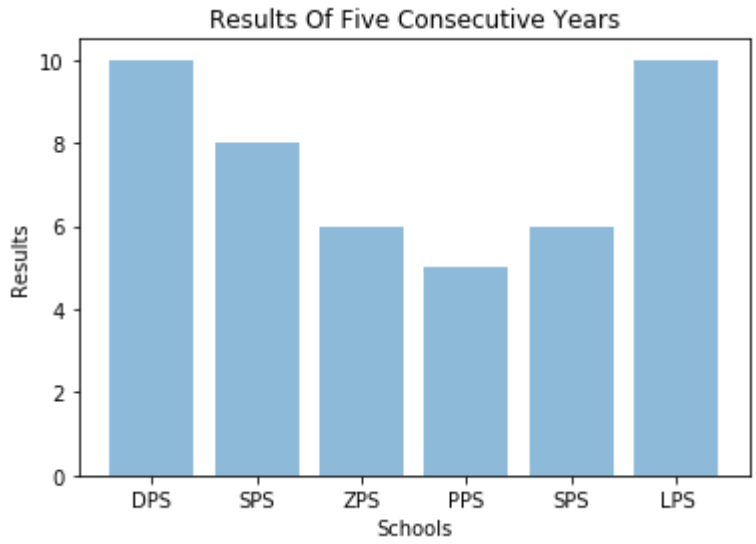
In [57]:

```
#Program to generate random numbers between 1 to 6 and check weather the user has won a lot
import random
x=int(input("Enter a number :"))
a=int(random.randint(0,6))
print("The lottery number is",a)
if(x==a):
    print ("You WON!!!!")
else:
    print("YOU LOSE")
```

```
Enter a number :5
The lottery number is 5
You WON!!!!
```

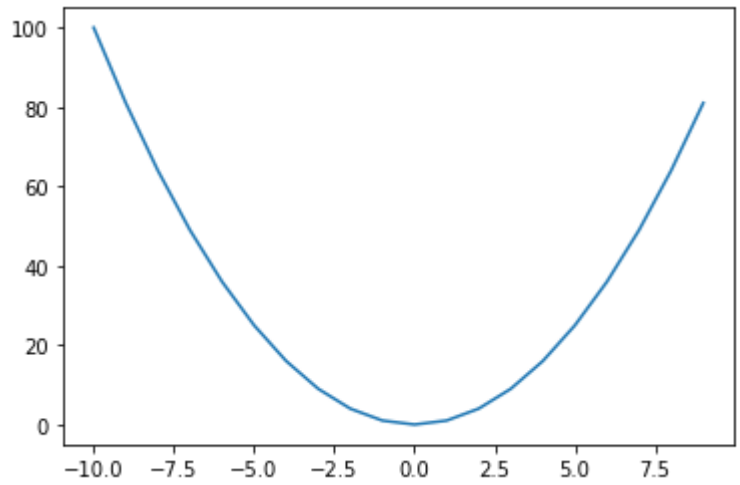
In [7]:

```
#Program to plot a bar chart on to display the result of a school for five consecutive year
import matplotlib.pyplot as plt
import numpy as np
objects = ('DPS', 'SPS', 'ZPS', 'PPS', 'SPS', 'LPS')
y = np.arange(len(objects))
performance = [10,8,6,5,6,10]
plt.bar(y, performance, align='center', alpha=0.5)
plt.xticks(y, objects)
plt.ylabel('Results')
plt.xlabel('Schools')
plt.title('Results Of Five Consecutive Years')
plt.show()
```



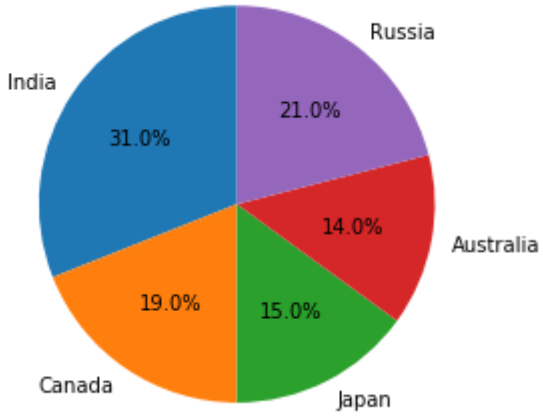
In [6]:

```
#Program to plot a graph for the function y=x^2.
import matplotlib.pyplot as plt
import numpy as np
x = np.arange( -10 , 10 )
y = np.square( x )
plt.plot( x , y )
plt.show()
```



In [7]:

```
#Program to plot a pie chart on consumption of water in daily life.
import matplotlib.pyplot as plt
labels = ['India', 'Canada', 'Japan', 'Australia', 'Russia']
sizes = [31, 19, 15, 14, 21]  # Add upto 100%
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.axis('equal')
plt.show()
```



In [16]:

```
#Program for linear search.
def linear_search(alist,key):
    for i in range (len(alist)):
        if alist==key:
            return True
    return False
alist=eval(input("enter the list of numbers:"))
key=str(input("The numbd=er to search for:"))
index=linear_search(alist,key)
if index<0:
    print(key,"was not found in the list")
else:
    print(key,"was found in the list")
```

```
enter the list of numbers:[15,6,13,22,3,52,2]
The numbd=er to search for:22
22 was found in the list
```

In [10]:

```
#Program for bubble sort search.
aList=eval(input("Enter Your List:"))
print("Orginal list is:",aList)
n=len(aList)
for i in range(n):
    for j in range (0,n-i-1):
        if aList[j]>aList[j+1]:
            aList[j],aList[j+1]=aList[j+1],aList[j]
print("List after sorting:",aList)
```

```
Enter Your List:[15,6,13,22,3,52,2]
Orginal list is: [15, 6, 13, 22, 3, 52, 2]
List after sorting: [2, 3, 6, 13, 15, 22, 52]
```

In [7]:

```
#Menu based proagram to perform the operation on stack in python.
def empty (stk):
    if stk==[]:
        return True
    else:
        return False
def push (stk,item):
    stk.append(item)
    top=len(stk)-1
def pop(stk):
    if empty(stk) is True:
        print("empty stack")
    else:
        item=stk.pop()
        if len(stk)==0:
            top=None
        else:
            top=len(stk)-1
            print(top,item)
        return item
def peek(stk):
    if empty:
        return "underflow"
    else:
        top=len(stk-1)
        return stk[top]
def display(stk):
    if empty(stk):
        print("empty stack")
        quit
    else :
        top=len(stk)-1
        print(stk[top],",top")
        for i in range(top,-1,-1):
            print(stk[i])
#main program
stk=[1,2,3,4,5,6,7,8,9]
top = None
while empty(stk) is not True:
    print("STACK OPERATION")
    print("1.PUSH")
    print("2.POP")
    print("3.PEEK")
    print("4.DISPLAY")
    print("5.EXIT")
    ch=int(input("ENTER YOUR CHOICE FROM 1 TO 5: "))
    if ch not in (1,2,3,4,5):
        print ("invalid choice")
        break
    elif ch == 1:
        item=int(input("enter item:"))
        push(stk,item)
    elif ch==2:
        item= pop(stk)
        if item=="underflow":
            print("underflow the stack is empty")
            print(stk)
        else:
            print("poped item is ",item)
```

```
    elif ch==3:
        item=peek(stk)
        if item=="underflow":
            print("underflow,stack is empty")
        else:
            print ("top most item is",item)
    elif ch==4:
        display(stk)
    elif ch==5:
        break
```

```
STACK OPERATION
1.PUSH
2.POP
3.PEEK
4.DISPLAY
5.EXIT
ENTER YOUR CHOICE FROM 1 TO 5: 1
enter item:0
STACK OPERATION
1.PUSH
2.POP
3.PEEK
4.DISPLAY
5.EXIT
ENTER YOUR CHOICE FROM 1 TO 5: 2
8 0
poped item is  0
STACK OPERATION
1.PUSH
2.POP
3.PEEK
4.DISPLAY
5.EXIT
ENTER YOUR CHOICE FROM 1 TO 5: 3
underflow,stack is empty
STACK OPERATION
1.PUSH
2.POP
3.PEEK
4.DISPLAY
5.EXIT
ENTER YOUR CHOICE FROM 1 TO 5: 4
9 ,top
9
8
7
6
5
4
3
2
1
STACK OPERATION
1.PUSH
2.POP
3.PEEK
4.DISPLAY
5.EXIT
ENTER YOUR CHOICE FROM 1 TO 5: 5
```

In [2]:

```
# Menu based proagram to perform the operation on queue in python.
class Queue:
    def __init__(self):
        self.items = []

    def is_empty(self):
        return self.items == []

    def enqueue(self, data):
        self.items.append(data)

    def dequeue(self):
        return self.items.pop(0)


q = Queue()
while True:
    print('enqueue <value>')
    print('dequeue')
    print('quit')
    do = input('What would you like to do? ').split()

    operation = do[0].strip().lower()
    if operation == 'enqueue':
        q.enqueue(int(do[1]))
    elif operation == 'dequeue':
        if q.is_empty():
            print('Queue is empty.')
        else:
            print('dqueued value: ',q.dequeue())
    elif operation=='quit':
        break
```

```
enqueue <value>
dequeue
quit
What would you like to do? 3
enqueue <value>
dequeue
quit
What would you like to do? quit
```

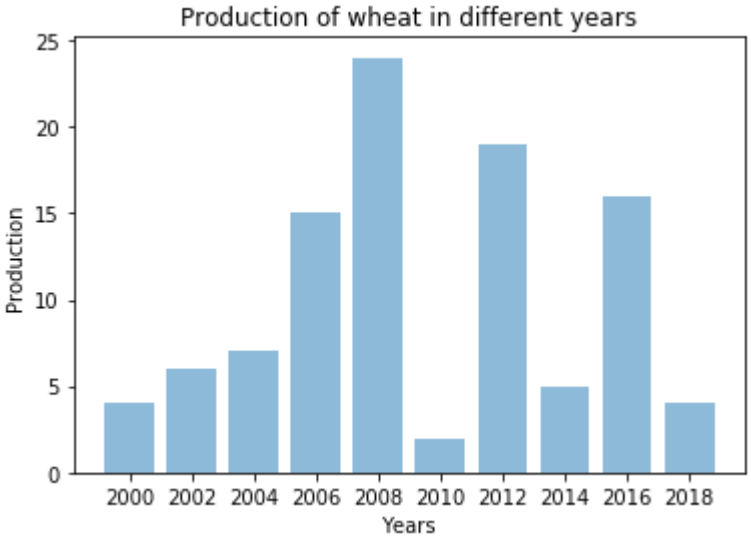
In [1]:

```
# Menu based proagram for circular queue in python.
class CircularQueue():
    def __init__(self, size): # initializing the class
        self.size = size
        self.queue = [None for i in range(size)]
        self.front = self.rear = -1
    def enqueue(self, data):
        if ((self.rear + 1) % self.size == self.front):
            print(" Queue is Full\n")
        elif (self.front == -1):
            self.front = 0
            self.rear = 0
            self.queue[self.rear] = data
        else:
            self.rear = (self.rear + 1) % self.size
            self.queue[self.rear] = data
    def dequeue(self):
        if (self.front == -1):
            print ("Queue is Empty\n")
        elif (self.front == self.rear):
            temp=self.queue[self.front]
            self.front = -1
            self.rear = -1
            return temp
        else:
            temp = self.queue[self.front]
            self.front = (self.front + 1) % self.size
            return temp
    def display(self):
        if(self.front == -1):
            print ("Queue is Empty")
        elif (self.rear >= self.front):
            print("Elements in the circular queue are:",end = " ")
            for i in range(self.front, self.rear + 1):
                print(self.queue[i], end = " ")
            print ()
        else:
            print ("Elements in Circular Queue are:",end = " ")
            for i in range(self.front, self.size):
                print(self.queue[i], end = " ")
            for i in range(0, self.rear + 1):
                print(self.queue[i], end = " ")
            print ()
        if ((self.rear + 1) % self.size == self.front):
            print("Queue is Full")
#__main__
ob = CircularQueue(5)
ob.enqueue(14)
ob.enqueue(22)
ob.enqueue(13)
ob.enqueue(-6)
ob.display()
print ("Deleted value = ", ob.dequeue())
print ("Deleted value = ", ob.dequeue())
ob.display()
ob.enqueue(9)
ob.enqueue(20)
ob.enqueue(5)
ob.display()
```

```
Elements in the circular queue are: 14 22 13 -6
Deleted value =  14
Deleted value =  22
Elements in the circular queue are: 13 -6
Elements in Circular Queue are: 13 -6 9 20 5
Queue is Full
```

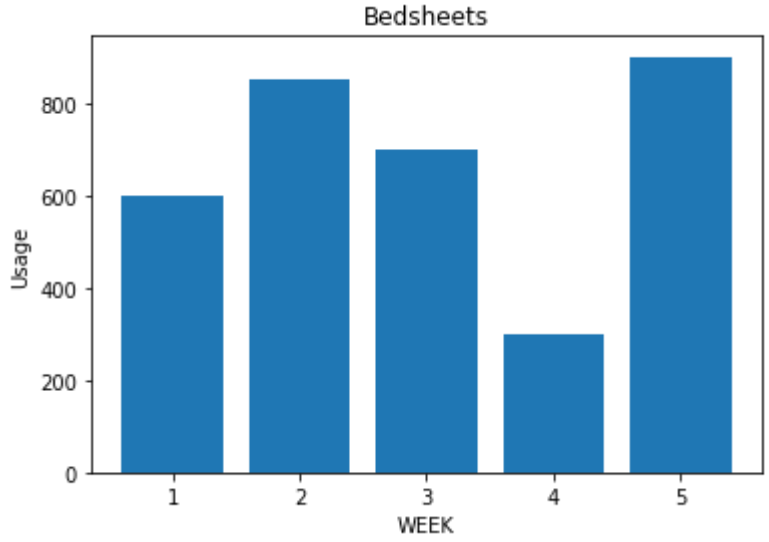
In [10]:

```
#Graoh based on the production of wheat in different years
import matplotlib.pyplot as plt
import numpy as np
objects = (2000,2002,2004,2006,2008,2010,2012,2014,2016,2018)
y = np.arange(len(objects))
performance = [4,6,7,15,24,2,19,5,16,4]
plt.bar(y, performance, align='center', alpha=0.5)
plt.xticks(y, objects)
plt.ylabel('Production')
plt.xlabel('Years')
plt.title('Production of wheat in different years')
plt.show()
```



In [12]:

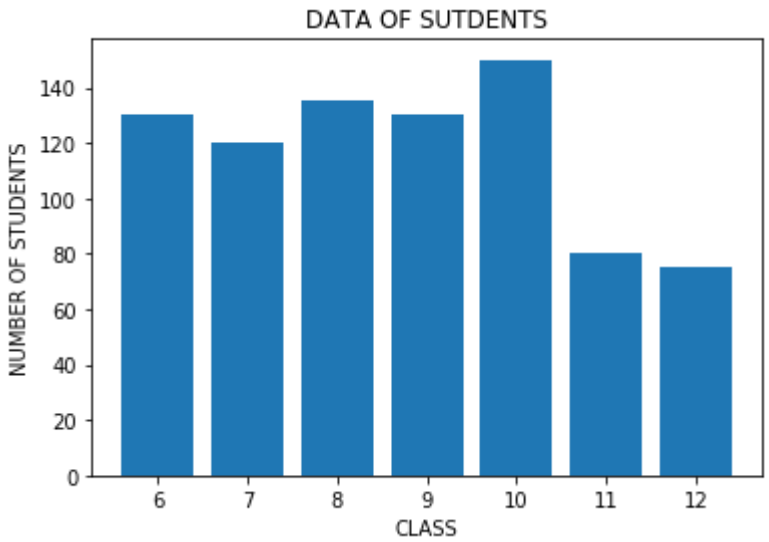
```
#Bar graph representing bed-sheets manufactured by a factory
import matplotlib.pyplot as plt
import numpy as np
x=[1,2,3,4,5]
y=[600,850,700,300,900]
plt.bar(x,y)
plt.xlabel("WEEK")
plt.ylabel("Usage")
plt.title("Bedsheets")
plt.show()
```



In [14]:

```
#Program for Graph of the give data
import matplotlib.pyplot as plt
import numpy as np

a=[6,7,8,9,10,11,12]
b=[130,120,135,130,150,80,75]
plt.bar(a,b)
plt.xlabel("CLASS")
plt.ylabel("NUMBER OF STUDENTS")
plt.title("DATA OF SUTDENTS")
plt.show()
```

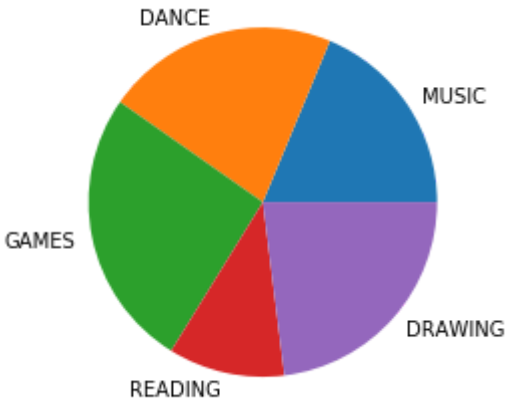


In [16]:

```
#Program to pie graph of the given data
import matplotlib.pyplot as plt
labels = ['MUSIC','DANCE', 'GAMES', 'READING','DRAWING']
number_of_students = [130,150,180,75,160]
plt.pie(number_of_students,labels=labels)
```

Out[16]:

```
([<matplotlib.patches.Wedge at 0x873dc08>,
  <matplotlib.patches.Wedge at 0x8746388>,
  <matplotlib.patches.Wedge at 0x8746348>,
  <matplotlib.patches.Wedge at 0x874b708>,
  <matplotlib.patches.Wedge at 0x8752588>],
 [Text(0.9154789352338673, 0.6098346654160165, 'MUSIC'),
  Text(-0.30665142637607196, 1.0563923999639153, 'DANCE'),
  Text(-1.0773212340458904, -0.22221376796643377, 'GAMES'),
  Text(-0.23437408375699953, -1.0747412660092042, 'READING'),
  Text(0.824628636069477, -0.7280024811593669, 'DRAWING')])
```



In [17]:

```
#Program to make a graph of the given data
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
N = 500
x = [2002,2005,2010,2015]
y = [50,40,30,60]
colors = (0,0,0)
plt.title('PRODUCTION')
plt.xlabel('YEAR')
plt.ylabel('PRODUCTION IN TONS')
plt.scatter(x, y)
plt.show()
```