1. class node:

```
    def __init__(self,data):
        self.val=data
        self.next=None
        self.prev=None
class dll:
    def __init__(self):
        self.head=None
        self.tail=None

    def insertatbeg(self,data):
        if self.head==None:
            self.head=node(data)
            self.tail=self.head
        else:
            new=node(data)
            new.next=self.head
            self.head.prev=new
            self.head=new

    def insertatend(self,data):
        if self.head==None:
            self.head=node(data)
            self.tail=self.head
        else:
            new=node(data)
            self.tail.next=new
            new.prev=self.tail
            self.tail=new

    def maximum(self):
        max=0
        curr=self.head
        while curr:
            if curr.val>max:
                max=curr.val
            curr=curr.next
        print(max)

    def insertatpos(self,data,pos):
        new=node(data)
        curr=self.head
        for i in range(pos-2):
            curr=curr.next
        new.next=curr.next
        curr.next.prev=new
        curr.next=new
        new.prev=curr
```

```
    def delatbeg(self):
        self.head=self.head.next
        self.head.prev=None

    def delatend(self):
        self.tail=self.tail.prev
        self.tail.next=None

    def delofval(self,val):
        curr=self.head
        while curr:
            if curr.next.val==val:
                break
            curr=curr.next
        curr.next=curr.next.next
        curr.next.prev=curr

    def delnlast(self,n):
        curr=self.tail
        for i in range(n):
            curr=curr.prev
        curr.next=curr.next.next
        curr.next.prev=curr

    def printing(self):
        curr=self.head
        while curr:
            print(curr.val,end="->")
            curr=curr.next

    def reverse(self):
        curr=self.head
        while curr:
            curr.next,curr.prev=curr.prev,curr.next
            curr=curr.prev
        self.head,self.tail=self.tail,self.head

o=dll()
for i in range(6):
    o.insertatend(i)
o.printing()
print()
o.delnlast(3)
o.printing()
```

2. given array of integers and integer k, find out subarray with k elements which will give maximum sum

brute force-

```
l=list(map(int,input().split()))
k=int(input())
m=0
for i in range(len(l)-k+1):
    s=sum(l[i:i+k])
    m=max(s,m)
print(m)
```

sliding window-

```
l=list(map(int,input().split()))
k=int(input())
s=sum(l[:k])
m=s
for i in range(1,len(l)-k+1):
    s=s-l[i-1]+l[i+k-1]
    m=max(s,m)
print(m)
```

3. https://leetcode.com/problems/maximum-average-subarray-i/?envType=problem-list-v2&envId=sliding-window

```
class Solution:
    def findMaxAverage(self, nums: List[int], k: int) ->
float:
        s=sum(nums[:k])
        m=s/k
        for i in range(1,len(nums)-k+1):
            s=s-nums[i-1]+nums[i+k-1]
            m=max(s/k,m)
        return m
```

4. https://leetcode.com/problems/substrings-of-size-three-with-distinct-characters/description/?envType=problem-list-v2&envId=sliding-window

```
class Solution:
    def countGoodSubstrings(self, s: str) -> int:
        count=0
        for i in range(len(s)-2):
            if s[i]!=s[i+1] and s[i+1]!=s[i+2] and s[i]!
=s[i+2]:
                count+=1
        return count
```

5. https://leetcode.com/problems/maximum-number-of-vowels-in-a-substring-of-given-length/description/?envType=problem-list-v2&envId=sliding-window

```
class Solution:
    def maxVowels(self, s: str, k: int) -> int:
        c=0
        v="aeiou"
        for i in range(k):
            if s[i] in v:
                c+=1
        m=c
        for i in range(1,len(s)-k+1):
            if s[i-1] in v:
                c-=1
            if s[i+k-1] in v:
                c+=1
            m=max(m,c)
        return m
```

6. https://leetcode.com/problems/maximum-points-you-can-obtain-from-cards/?envType=problem-list-v2&envId=sliding-window

```
class Solution:
    def maxScore(self, l: List[int], k: int) -> int:
        s=sum(l[:k])
        m=s
        n=len(l)
        for i in range(k):
            s=s+l[n-i-1]-l[k-i-1]
            m=max(m,s)
        return m
```

7. https://leetcode.com/problems/number-of-sub-arrays-of-size-k-and-average-greater-than-or-equal-to-threshold/description/?envType=problem-list-v2&envId=sliding-window

```
class Solution:
    def numOfSubarrays(self, arr: List[int], k: int, threshold: int) -> int:
        s=sum(arr[:k])
        count=0
        if(s/k>=threshold):
                count+=1
        for i in range(1,len(arr)-k+1):
```

```
            s=s+arr[i+k-1]-arr[i-1]
            if(s/k>=threshold):
                count+=1
        return count
```

8. mini project- digital clock

```
import turtle
import datetime
import time
screen=turtle.Screen()
screen.tracer(0)
screen.bgcolor("black")

s=turtle.Turtle()
t=turtle.Turtle()

s.speed(0)
t.speed(0)
s.color("white")
s.hideturtle()
s.pensize(2)
s.penup()
s.goto(-200,-50)
s.pendown()
s.forward(400)
s.left(90)
s.forward(100)
s.left(90)
s.forward(400)
s.left(90)
s.forward(100)

t.color("white")
t.hideturtle()
t.penup()
t.goto(-115,-35)
t.pendown()

s=datetime.datetime.now().second
m=datetime.datetime.now().minute
hr=datetime.datetime.now().hour
while(True):

t.write(str(hr).zfill(2)+":"+str(m).zfill(2)+":"+str(s).zfill
(2),font=(("Arial", 60, "normal")))
    s+=1
    if s==60:
```

```
        s=0
        m+=1
    if m==60:
        m=0
        hr+=1
    if hr==13:
        hr=1
    time.sleep(1)
    screen.update()
    t.undo()
```