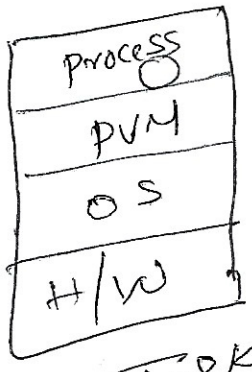
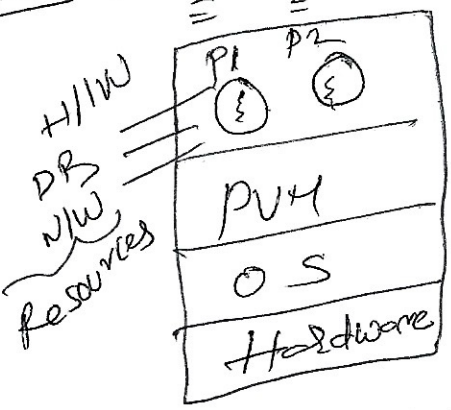
Threads

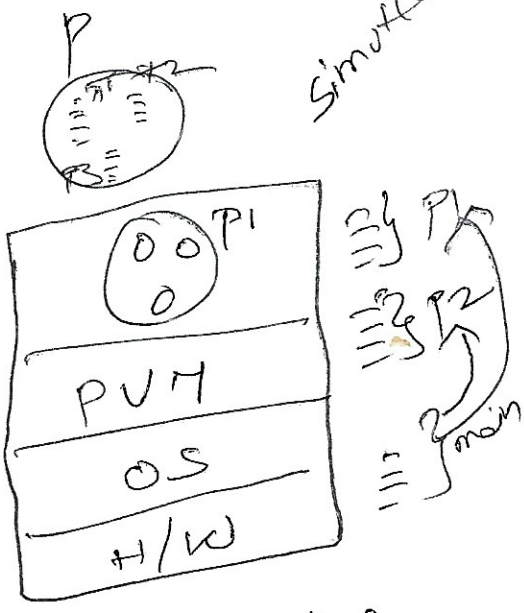
Multithreading

1. What is Multithreading?
2. Ways to create a thread
3. Mutex
4. Semaphores
5. Inter process Communication
6. IPC using Condition
7. IPC using Queue

What is Multithreading







Unit Tasking

Multitasking

Multithreading

→ Running Problem
→ PVM - Python virtual machine is called process

Resources such H/W, DB, NW used by process

Multithreading :- Animation, gaming, server, client

Ways to create Thread

- Using Function / Method
- Using a class

A 65
B 66
C 67
:
Z 90

```
for i in range(65,97):
    pt(i)
```

```
from threading import *
from time import *
def display():
    for i in range(65,91):
        pt(chr(i))
t = Thread(target=display, name='Alphabets')
t.start()
for i in range(65,91):
    pt(i)
t.join()
```

```
from threading import *
from time import *
def display():
    for i in range(65,91):
        print(chr(i))
        sleep(1)
t = Thread(target=display, name='Alphabets')
t.start()
for i in range(65,91):
    print(i)
    sleep(1)
t.join()
```

A 65
H 72
J 75
77 M

Class

```
from threading import *
from time import *
class Alphabets(Thread):
    def run(self):
        for i in range(65,91):
            print(chr(i))
            sleep(1)
t = Alphabets()
t.start()
for i in range(65,91):
    print(i)
    sleep(1)
```

A 65
H 72
J 75
77 M

Mutex :-

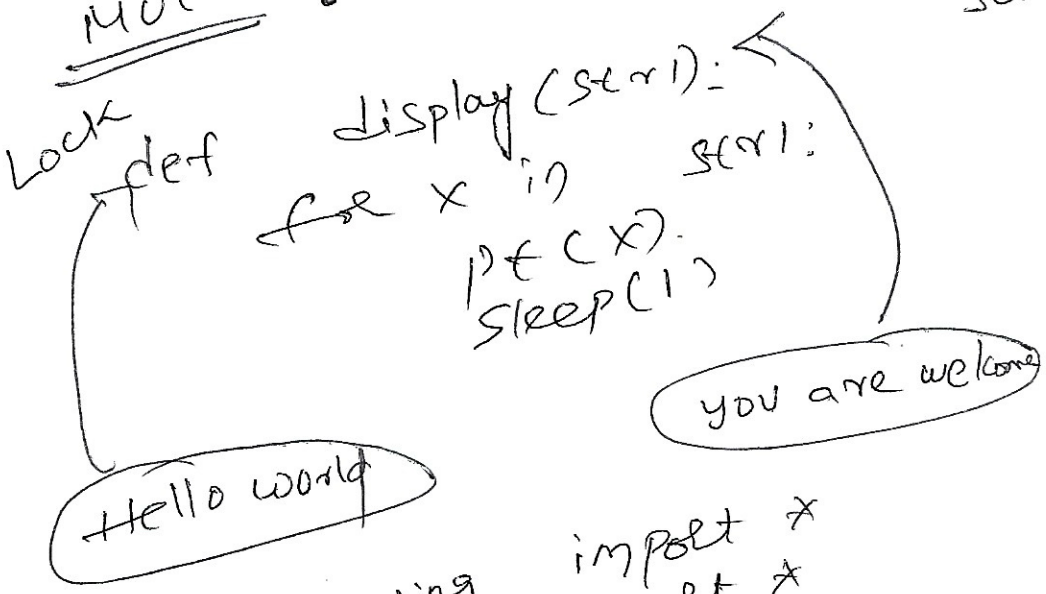
↳ Thread x Synchronization

Race condition x

Mutex
Semaphore

Mutex :-

Lock




```
from threading import *
from time import *

def display(msg):
    for i in msg:
        print(i, end='')
        sleep(1)

t1 = Thread(target=display, args=('HELLO WORLD',))
t2 = Thread(target=display, args=('you are welcome',))
```

↳ Mutex is a lock

↳ from threading import *
from time import *

```
def display(msg):          → 1. acquire()
    for i in msg:
        pt(i)
        sleep(1)           → 1. release()
```

```
l = lock()
t1 = Thread(target = display, args = ('HELLLO WORLD',))
t2 = Thread(target = display, args = ('you are welcome',))

t1.start()
t2.start()

t1.join()
t2.join()
```

sem [t1 | t2 | t3]
sem [2]

Semaphore
↳ Token. It is a Variable. sem [1] 0 1
0 - Block
1 → Allow

```
l = Semaphore(1)
t3 = Thread(
t3.start()
t3.join()
```

1 → 0 0

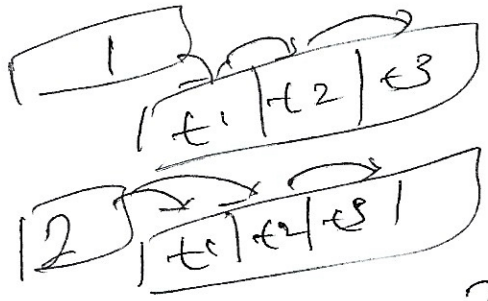
```
l = Semaphore(2)
```

t1
t2
t3
race condition

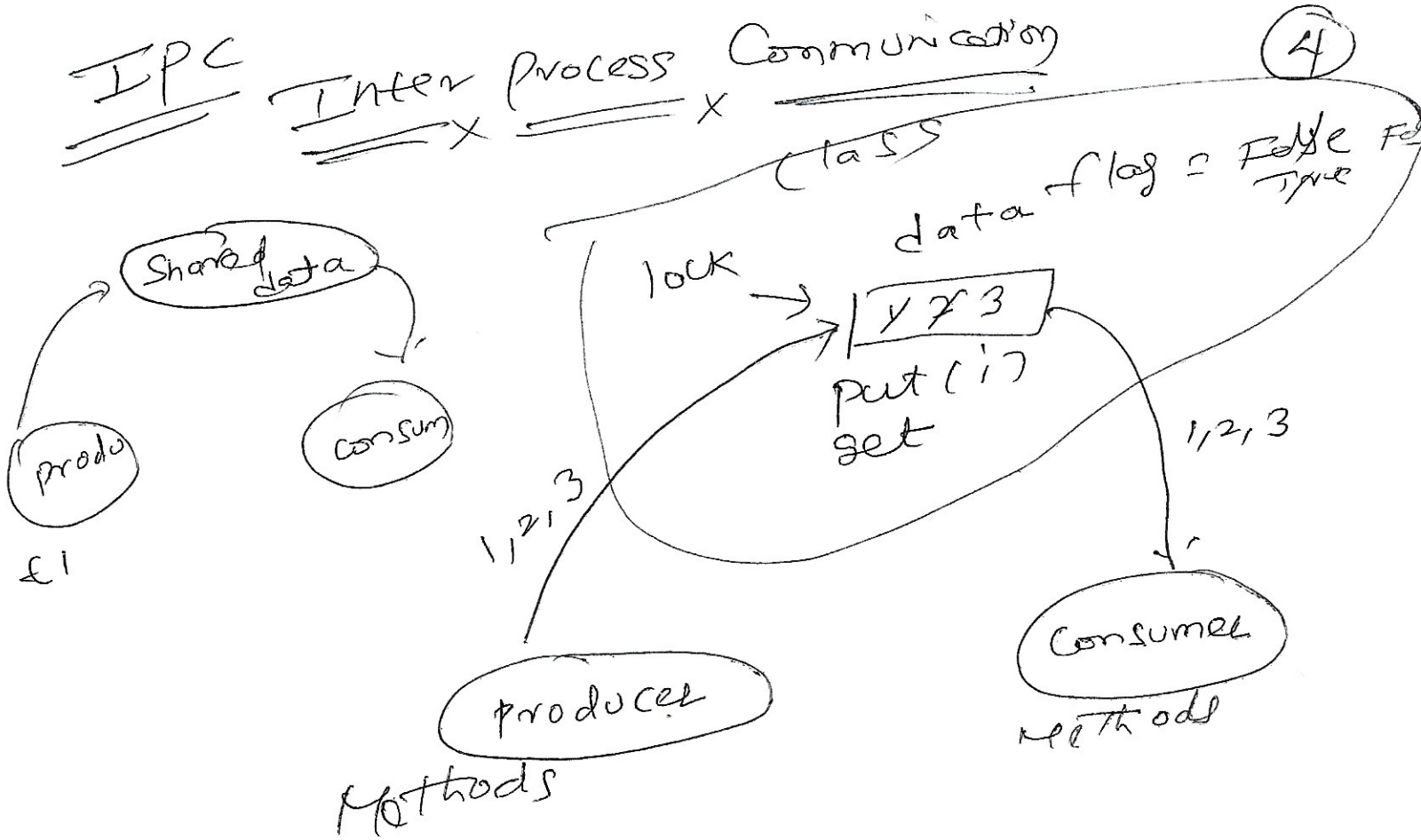
IPC

Semaphores ✓

```
from threading import *
from time import *
def display(str):
    l.acquire()
    for i in str:
        print(i)
        sleep(1)
    l.release()

l = Semaphore(1)  || l = semapore(2)   race condition
t1=Thread(target=display, args=('HELLO WORLD',))
t2=Thread(target=display, args=('you are welcome',))
t3=Thread(target=display, args=('0123456',))

t1.start()
t2.start()
t3.start()

t1.join()
t2.join()
t2.join
```



IPC — Inter Process Communication




```
Class MyData:
    def __init__(self):
        self.data=0
        self.flag=False
        self.lock=LOCK

    def put(self,d):
        while(self.flag!=False):
            pass
        self.lock=acquire()
        self.data=d
        self.data=True
        self.lock.release()
```

```
def get(self):
    while(self.flag!=True):
        pass
    self.lock.acquire()
    x=self.data
    self.flag=True
    self.lock.release()
```

```
from threading import *
from time import *

class MyData():
    def __init__(self):
        self.data=0
        self.flag=False
        self.lock=Lock()

    def put(self,d):
        while(self.flag!=False):
            pass
        self.lock.acquire()
        self.data=d
        self.flag=True
        self.lock.release()
        sleep()
```

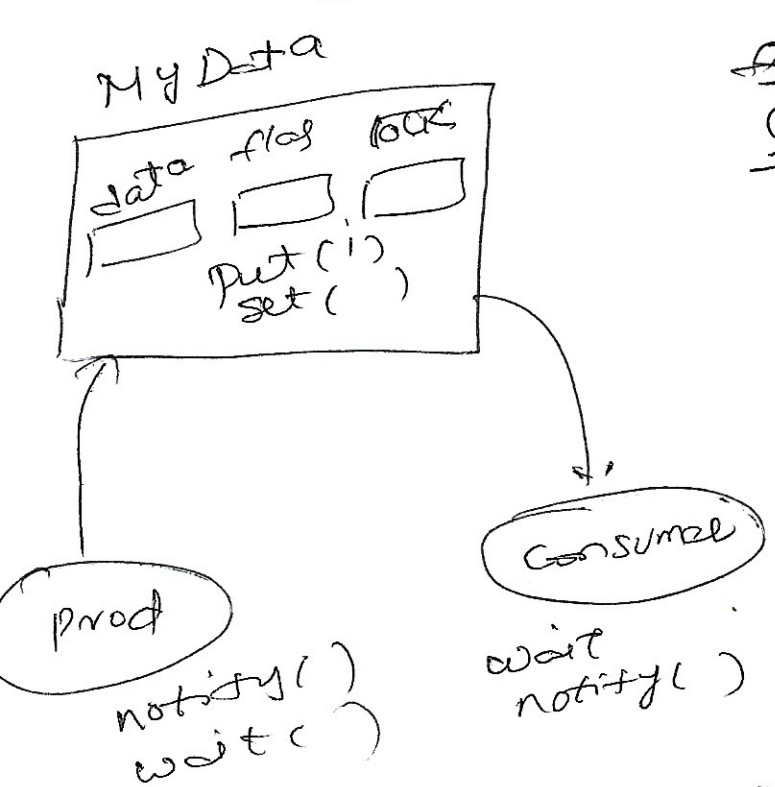
```
def get(self):
    while (self.flag != True):
        pass
    self.lock.acquire()
    x = self.data
    self.flag = False
    self.lock.release()
    sleep(1)
    return x

def producer(data):
    i = 1
    while(true):
        data.put(i)
        pt ('producer:',i)
        i += 1

def consumer(dat
    while(true):
        x = data.get(
        pt('consumer',

data = MyData()
t1 = Thread(target = lambda : producer(data))
t2 = Thread(target = lambda : consumer(data))
t1.start()
t2.start()
t1.join()
t2.join()
```

```
Producer: 1
Consumer: 1
          2
          2
          3 3
          4 4
          5
          5 6
          6
```

IPC using Condition

flag / lock

CV




```
from threading import *
from time import *

class MyData():
    def __init__(self):
        self.data=0
        self.cv = Condition()

    def put(self,d):
        self.cv.acquire()
        self.cv.wait(timeout=0)
        self.data=d
        self.cv.notify()
        self.cv.release()
        sleep(1)
```

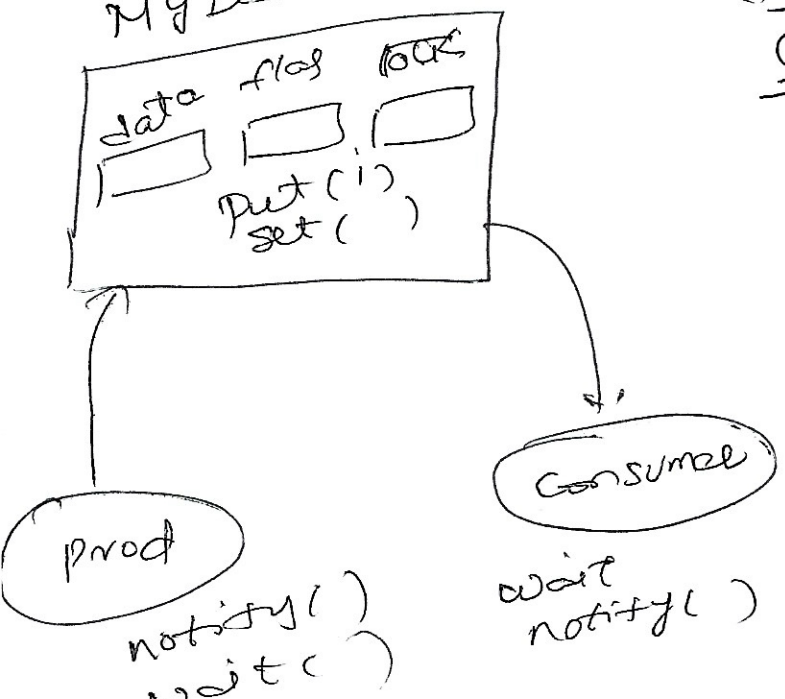
```
def get(self):
    while (self.flag != True):
        pass
    self.lock.acquire()
    x = self.data
    self.flag = False
    self.lock.release()
    sleep(1)
    return x

def producer(data):
    i = 1
    while(True):
        data.put(i)
        pt('producer:',i)
        i += 1

def consumer(data):
    while(True):
        x = data.get()
        pt('consumer',x)

data = MyData()
t1 = Thread(target = lambda : producer(data))
t2 = Thread(target = lambda : consumer(data))
t1.start()
t2.start()
t1.join()
t2.join()
```

```
producer: 1
consumer: 1
          2
          2
          3 3
          4 4
          5
          5 6
          6
```

IPC using Condition :

flag / lock

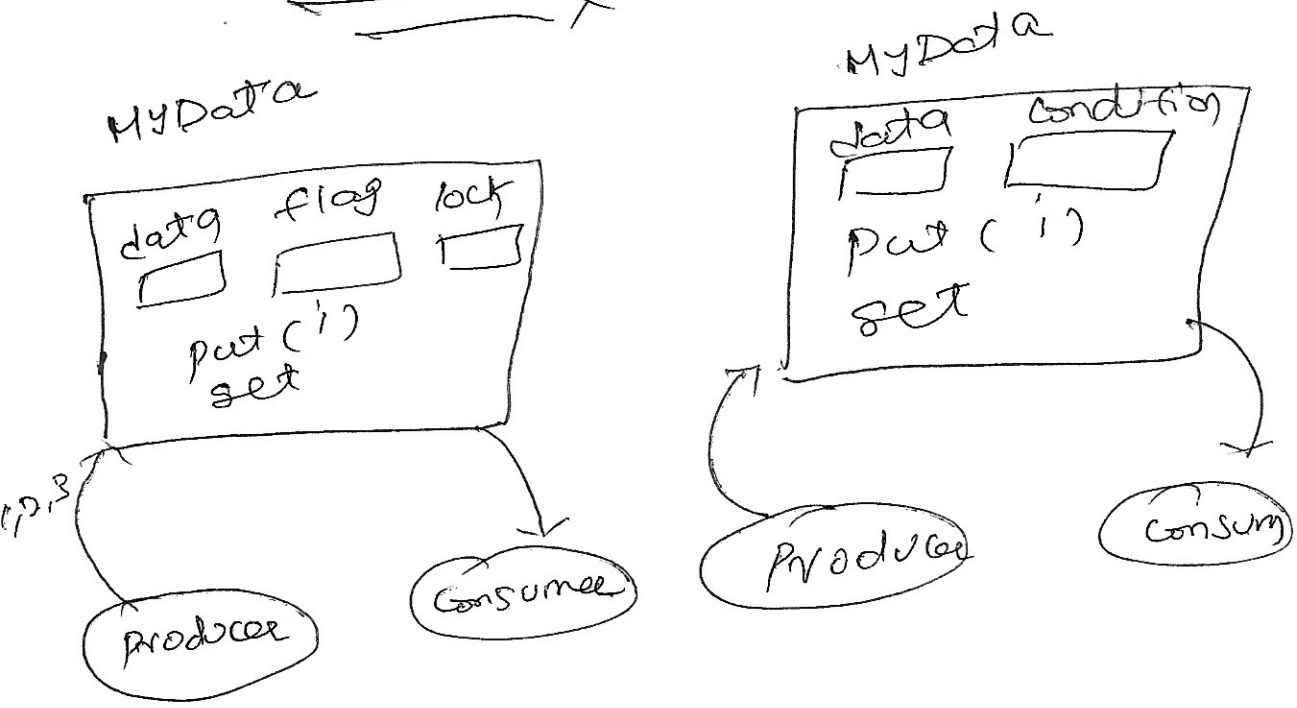
CV

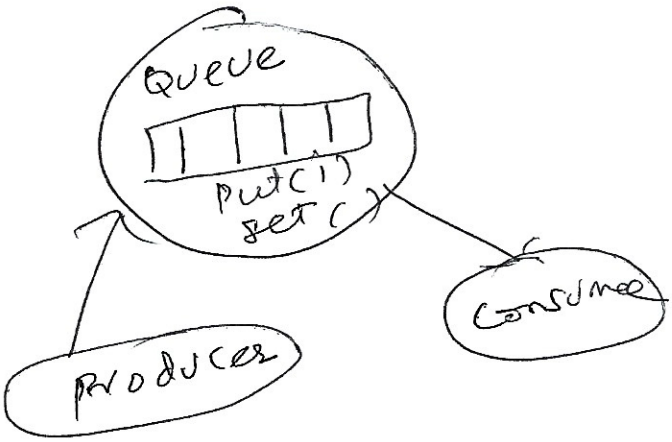



```
from threading import *
from time import *

class MyData():
    def __init__(self):
        self.data=0
        self.cv = Condition()

    def put(self,d):
        self.cv.acquire()
        self.cv.wait(timeout=0)
        self.data=d
        self.cv.notify()
        self.cv.release()
        sleep(1)
```

```
def get(self):
    self.cv.acquire()
    self.cv.wait(timeout=0)
    x = self.data
    self.cv.notify()
    self.cv.release()
    sleep(1)
    return x

def producer(data):
    i=1
    while(True):
        data.put(i)
        pt('Produced:', i)
        i+=1

def Consumer(data):
    while True:
        x = data.get()
        pt('Consumed:', x)

data = MYData()
t1 = Thread(target=lamda: producer(data))
t2 = Thread(target= lamda: Consumer(data))
t1.start()
t2.start()
t1.join()
t2.join()
```

IPC using Queue





```
from threading import *
from time import *
from queue import *

Q = Queue()

def Producee(que):
    i = 1
    while True:
        que.put(i)
        pt('Producee', i)
        i += 1
        sleep(1)

def Consumee(que):
    while(True):
        x = que.get()
        pt('Consumee', x)
        sleep(1)

t1 = Thread(target = lambda q: producee(q))
t2 = Thread(target= lambda: Consumee(q))
t1.start()
t2.start()
t1.join()
t2.join()
```