**\*\*FCFS\*\***

class process:

def \_\_init\_\_(self, pid, arrival\_time, burst\_time):

self.pid=pid

self.arrival\_time=arrival\_time

self.burst\_time=burst\_time

self.complition\_time=0

self.turnaround\_time=0

self.wait\_time=0

def get\_arrival\_time(process):

return process.arrival\_time

def fcfs(pro):

pro.sort(key=get\_arrival\_time)

current\_time=0

for process in pro:

if current\_time < process.arrival\_time:

current\_time = process.arrival\_time

process.complition\_time = current\_time + process.burst\_time

process.turnaround\_time = process.complition\_time - process.arrival\_time

process.wait\_time = process.turnaround\_time - process.burst\_time

current\_time=process.complition\_time

def display(pro):

print("PID \t AT \t BT \t CT \t TAT \t WT")

for p in pro:

print(f"{p.pid} \t {p.arrival\_time} \t {p.burst\_time} \t {p.complition\_time} \t {p.turnaround\_time} \t {p.wait\_time}")

# main funtion

if \_\_name\_\_ == "\_\_main\_\_":

process\_list = [

process("P1", 0, 5),

process("P2", 1, 3),

process("P3", 2, 8),

process("P4", 3, 6)

]

fcfs(process\_list)

display(process\_list)

**\*\*np-sjf\*\***

class process:

def \_\_init\_\_(self, pid, arrival\_time, burst\_time):

self.pid=pid

self.arrival\_time=arrival\_time

self.burst\_time=burst\_time

self.complition\_time=0

self.turnaround\_time=0

self.wait\_time=0

def get\_arrival\_time(process):

return process.arrival\_time

def fcfs(pro):

pro.sort(key=get\_arrival\_time)

current\_time=0

for process in pro:

if current\_time < process.arrival\_time:

current\_time = process.arrival\_time

process.complition\_time = current\_time + process.burst\_time

process.turnaround\_time = process.complition\_time - process.arrival\_time

process.wait\_time = process.turnaround\_time - process.burst\_time

current\_time=process.complition\_time

def display(pro):

print("PID \t AT \t BT \t CT \t TAT \t WT")

for p in pro:

print(f"{p.pid} \t {p.arrival\_time} \t {p.burst\_time} \t {p.complition\_time} \t {p.turnaround\_time} \t {p.wait\_time}")

# main funtion

if \_\_name\_\_ == "\_\_main\_\_":

process\_list = [

process("P1", 0, 5),

process("P2", 1, 3),

process("P3", 2, 8),

process("P4", 3, 6)

]

fcfs(process\_list)

display(process\_list)

**\*\*p-sjf\*\***

class process:

def \_\_init\_\_ (self, pid, at, bt):

self.pid = pid

self.at= at

self.bt=bt

self.rt=bt

self.ct=0

self.tat=0

self.wt=0

self.start\_time=-1

def sjfp (process):

n=len(process)

current\_time=0

complete =0

shortest = None

while complete < n:

ready\_queue= [p for p in process if p.at<= current\_time and p.rt>0]

if ready\_queue:

shortest\_job = min(ready\_queue, key=lambda x: x.rt)

if shortest\_job.start\_time == -1: #if first job enters

shortest\_job.start\_time = current\_time

shortest\_job.rt -= 1 #run for 1 unit time

current\_time += 1

if shortest\_job.rt == 0:

shortest\_job.ct = current\_time

shortest\_job.tat = shortest\_job.ct - shortest\_job.at

shortest\_job.wt = shortest\_job.tat - shortest\_job.bt

complete += 1

else :

current\_time += 1

def display(process):

print("pid \t at \t bt \t ct \t tat \t wt")

for p in process:

print(f"{p.pid} \t {p.at} \t {p.bt} \t {p.ct} \t {p.tat} \t {p.wt}" )

if \_\_name\_\_ == "\_\_main\_\_":

process\_list = [

process("P1", 0, 8),

process("P2", 1, 4),

process("P3", 2, 9),

process("P4", 3, 5)

]

sjfp(process\_list)

display(process\_list)

**\*\*ps\*\***

class process:

def \_\_init\_\_(self, pid, at, bt, priority):

self.pid = pid

self.at= at

self.bt=bt

self.priority=priority

self.ct=0

self.tat=0

self.wt=0

self.is\_complete= False

def priority\_shedule(process):

n = len(process)

current\_time = 0

complete = 0

while complete < n:

ready\_queue = [p for p in process if p.at<=current\_time and not p.is\_complete]

if ready\_queue:

high\_priority = min(ready\_queue, key=lambda x: x.priority)

current\_time += high\_priority.bt

high\_priority.ct = current\_time

high\_priority.tat = high\_priority.ct - high\_priority.at

high\_priority.wt = high\_priority.tat - high\_priority.bt

complete += 1

high\_priority.is\_complete = True

else:

current\_time += 1

def display(process):

print("pid \t at \t bt \t ct \t tat \t wt \t priority")

for p in process:

print(f"{p.pid} \t {p.at} \t {p.bt} \t {p.ct} \t {p.tat} \t {p.wt} \t {p.priority}" )

if \_\_name\_\_ == "\_\_main\_\_":

process\_list = [

process("P1", 0, 6, 2),

process("P2", 1, 8, 1),

process("P3", 2, 7, 3),

process("P4", 3, 3, 2)

]

priority\_shedule(process\_list)

display(process\_list)

**\*\*rr\*\***

class process:

def \_\_init\_\_ (self, pid, at, bt):

self.pid = pid

self.at= at

self.bt=bt

self.rt=bt

self.ct=0

self.tat=0

self.wt=0

def rr(process, time\_quantum):

n = len(process)

queue=[]

time = 0

complete=0

process.sort(key=lambda x: x.at)

visited =[False] \* n

while complete < n:

for i in range(n):

if process[i].at<time and not visited[i]:

queue.append(process[i])

if not queue:

time+=1

continue

current\_process = queue.pop(0)

run\_time = min(time\_quantum, current\_process.rt)

time += run\_time

current\_process.rt -= run\_time

for i in range(n):

if (process[i].at>time-run\_time and process[i].at<time and not visited[i]):

queue.append(process[i])

visited[i] = True

if current\_process.rt ==0:

current\_process.ct=time

current\_process.tat = time - current\_process.at

current\_process.wt = current\_process.tat -current\_process.bt

complete += 1

else:

queue.append(current\_process)

def display(process):

print("pid \t at \t bt \t ct \t tat \t wt")

for p in process:

print(f"{p.pid} \t {p.at} \t {p.bt} \t {p.ct} \t {p.tat} \t {p.wt}" )

if \_\_name\_\_ == "\_\_main\_\_":

process\_list = [

process("P1", 0, 5),

process("P2", 1, 4),

process("P3", 2, 2),

process("P4", 3, 1)

]

time\_quantum = 2

rr(process\_list, time\_quantum)

display(process\_list)