PROGRAMMING II PYTHON

STACKS

MORGAN BERGEN

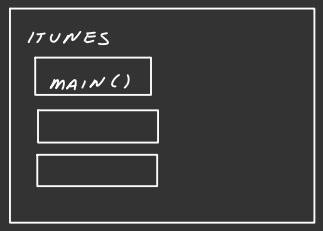
FEB 24 2022

NODE-BASED IMPLEMENTATION OF A STACK.

```
# NODE.PY
CLASS NODE:

    DEF __INIT__(SELF, DATA):
        SELF.DATA = DATA
        SELF.NEXT = NONE

    DEF SETDATA(SELF, DATA):
        SELF.DATA = DATA

    DEF SETNEXT(SELF, NEXT):
        SELF.NEXT = NEXT

    DEF GETDATA(SELF):
        RETURN(SELF.DATA)

    DEF GETNEXT(SELF):
        RETURN(SELF.NEXT)
```

```
# MAIN.PY

DEF MAIN():
    FIRST = NODE('    ')
    SECOND = NODE('    ')

IF __NAME__ == "__MAIN__":
    MAIN()
```

START <PROCESS NAME> START ITUNES ITUNES ADDED TO QUEUE

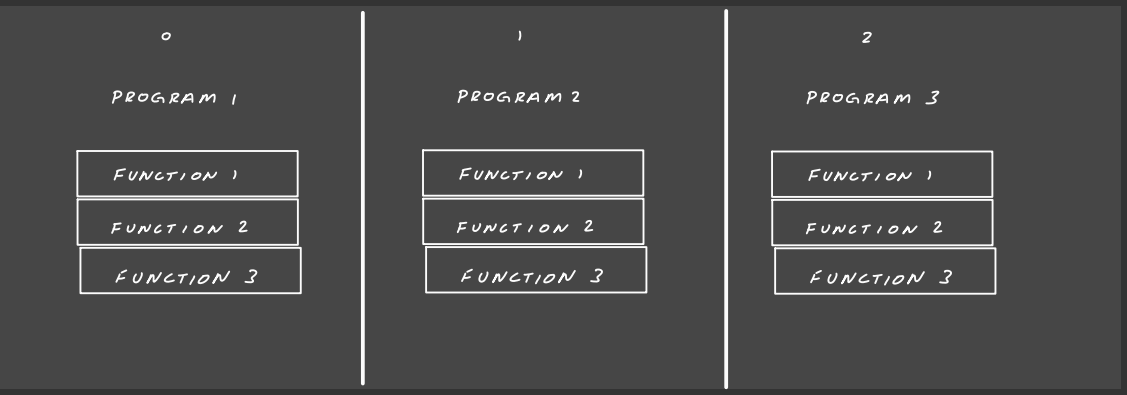


PROCESS STACK

CALL <FUNCTION NAME> ORGANIZED BY STACKS
START <PROCESS NAME> ORGANIZED BY QUEUES

PROCESS AT THE FRONT OF THE QUEUE
HAS A FUNCTION AT THE TOP OF ITS CALL STACK

PROGRAMS WILL GET IN A QUEUE & WAIT TO HAVE THEIR TIME IN THE PROCESSOR
WHILE A PROGRAM IS USING THE CPU IT CALLS A FUNCTION AND ADD THAT FUNCTION TO IT'S CALL STACK.
QUEUES CONTAIN STACKS, STACKS CONTAIN NODES



1 QUEUE CONTAIN STACKS
EACH STACK HAS A NAME (i.e. iTUNES, FIREFOX)

ONE QUEUE ORGANIZES 3 STACKS
STACKS ARE PROGRAMS/PROCESS NAMES
STACK 1 IS NAMED iTUNES
STACK 2 IS NAMED FIREFOX
STACK 3 IS NAMED PUTTY
EACH STACK CONTAINS NODES
NODES CONTAIN <FUNCTION NAMES>

STACK 1 ITUNES CONTAINS NODES WITH FUNCTION NAMES
STACK 1 ITUNES CONTAINS MAIN() (WITH ONLY ONE NODE)
WHEN CALL COMMAND IS USED THE PROGRAM AT THE FRONT OF THE QUEUE ADDS A FUNCTION TO THE TOP OF THE STACK & MOVES THAT PROGRAM TO THE BACK OF THE QUEUE

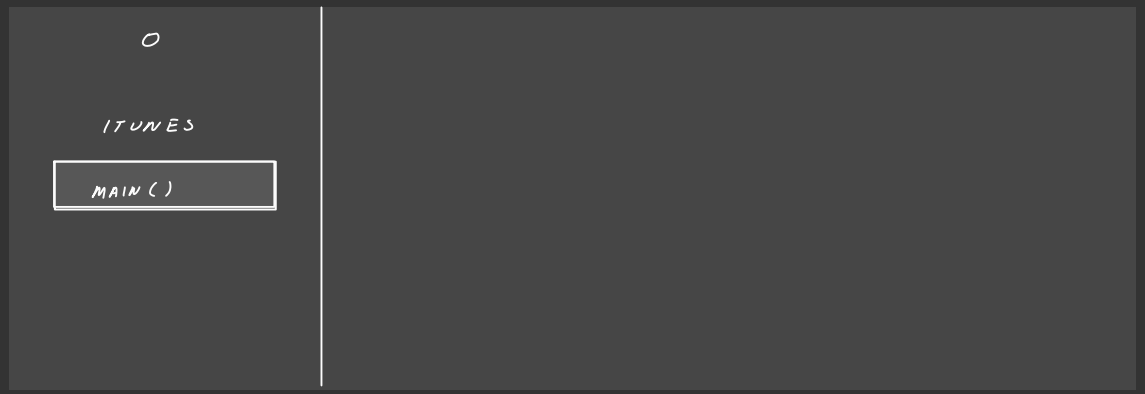
<CLASS CPUSCHEDULER>

<CLASS QUEUE>

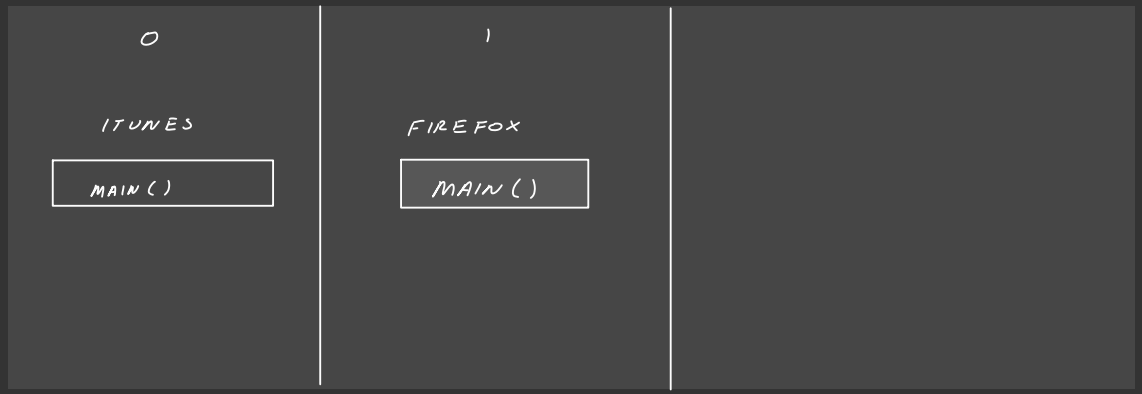
<CLASS PROCESS>

<CLASS STACK>

<CLASS NODE>

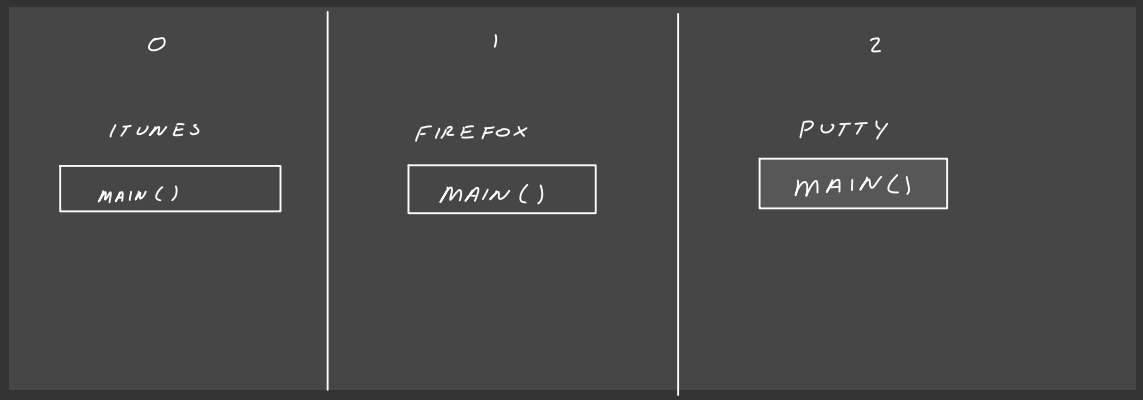


START ITUNES ⟶ ITUNES ADDED TO QUEUE



START ITUNES ⟶ ITUNES ADDED TO QUEUE

START FIREFOX → FIREFOX ADDED TO QUEUE



START ITUNES ⟶ ITUNES ADDED TO QUEUE

START FIREFOX → FIREFOX ADDED TO QUEUE

START PUTTY → PUTTY ADDED TO QUEUE

< CLASS CPUSCHEDULER > ORGANIZES QUEUES

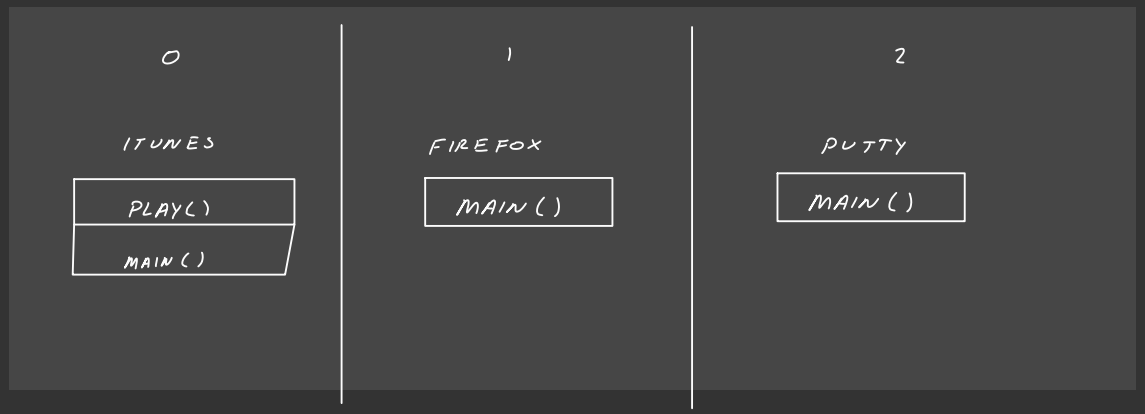
< CLASS QUEUE >

< CLASS PROCESS >

< CLASS STACK >

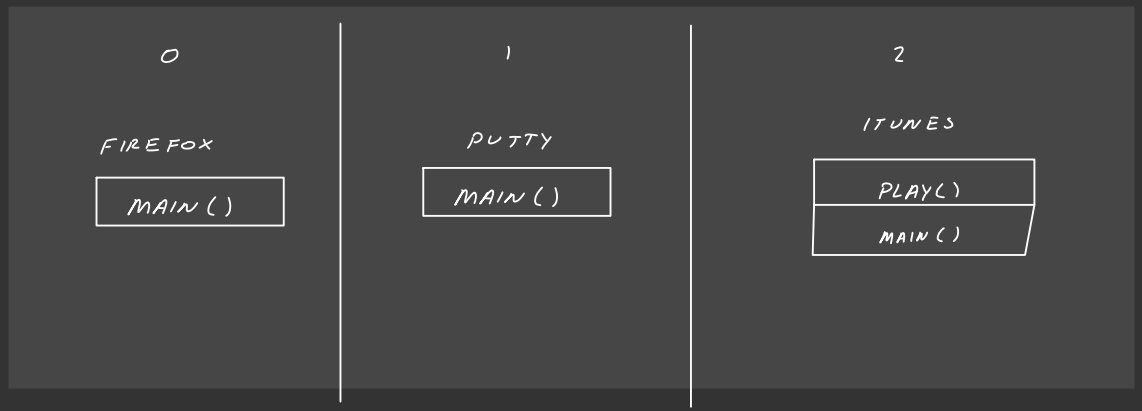
< CLASS NODE >

FUNCTION NAMES ARE NODE ELEMENTS



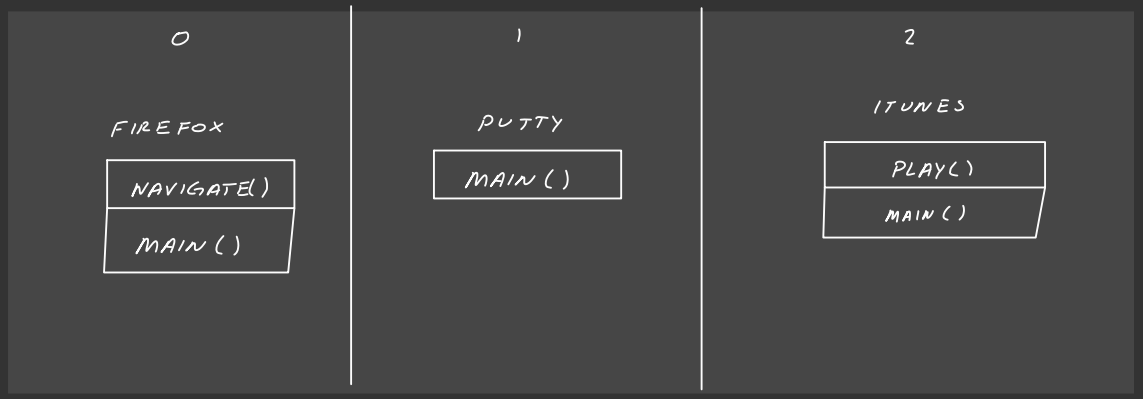
START ITUNES → "ITUNES ADDED TO QUEUE"
START FIREFOX → "FIREFOX ADDED TO QUEUE"
START PUTTY → "PUTTY ADDED TO QUEUE"
CALL PLAY → "ITUNES CALLS PLAY"

1/2 (# PUSH("PLAY()"))



START ITUNES → "ITUNES ADDED TO QUEUE"
START FIREFOX → "FIREFOX ADDED TO QUEUE"
START PUTTY → "PUTTY ADDED TO QUEUE"
CALL PLAY → "ITUNES CALLS PLAY"

2/2



START ITUNES → "ITUNES ADDED TO QUEUE"
START FIREFOX → "FIREFOX ADDED TO QUEUE"
START PUTTY → "PUTTY ADDED TO QUEUE"
CALL PLAY → "ITUNES CALLS PLAY"
CALL NAVIGATE → "FIREFOX CALLS NAVIGATE"

1/2 # PUSH("NAVIGATE()")

<CLASS CPUSCHEDULER> ORGANIZES QUEUES

<CLASS QUEUE>

<CLASS PROCESS>

<CLASS STACK>

<CLASS NODE>

#FUNCTION NAMES ARE NODE ELEMENTS

#PROCESS NAMES ARE STACKS

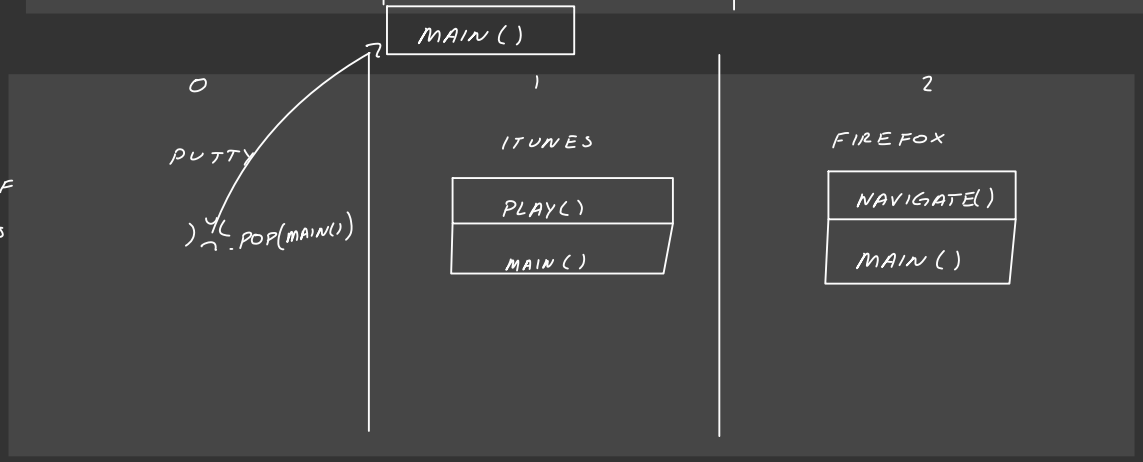
CPU_SCHEDULER IS IN CHARGE OF ADDING A PROCESS TO A QUEUE

IF COMMAND == "START ITUNES":

ENQUEUE (PROGRAM) <STACK>

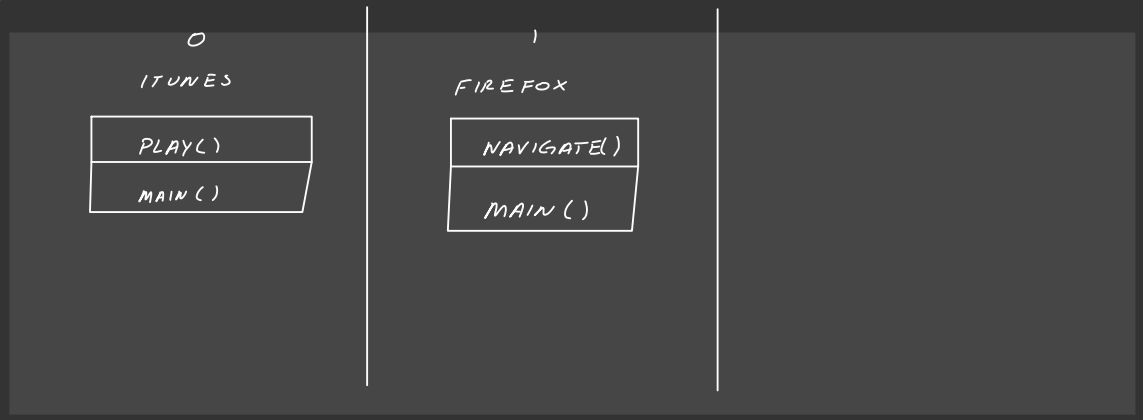
START ITUNES → "ITUNES ADDED TO QUEUE"
START FIREFOX → "FIREFOX ADDED TO QUEUE"
START PUTTY → "PUTTY ADDED TO QUEUE"
CALL PLAY → "ITUNES CALLS PLAY"
CALL NAVIGATE → "FIREFOX CALLS NAVIGATE"

2/2



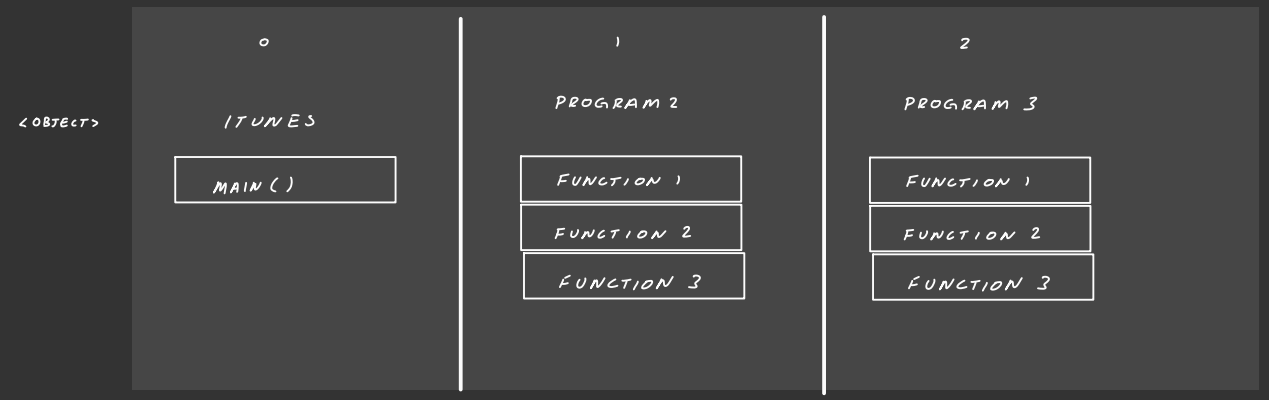
START ITUNES → "ITUNES ADDED TO QUEUE"
START FIREFOX → "FIREFOX ADDED TO QUEUE"
START PUTTY → "PUTTY ADDED TO QUEUE"
CALL PLAY → "ITUNES CALLS PLAY"
CALL NAVIGATE → "FIREFOX CALLS NAVIGATE"
RETURN → "PUTTY RETURNS FROM MAIN"
"PUTTY PROCESS HAS ENDED"

1/2



START ITUNES → "ITUNES ADDED TO QUEUE"
START FIREFOX → "FIREFOX ADDED TO QUEUE"
START PUTTY → "PUTTY ADDED TO QUEUE"
CALL PLAY → "ITUNES CALLS PLAY"
CALL NAVIGATE → "FIREFOX CALLS NAVIGATE"
RETURN → "PUTTY RETURNS FROM MAIN"
"PUTTY PROCESS HAS ENDED"

1/2


<OBJECT>
0
ITUNES
MAIN()
1
PROGRAM 2
FUNCTION 1
FUNCTION 2
FUNCTION 3
2
PROGRAM 3
FUNCTION 1
FUNCTION 2
FUNCTION 3