**ALGORITHM AND DATA STRUCTURE PRACTICUM**

**MODULE 7**

**REGULAR EXPRESSIONS**



**CREATED BY :**

**KURNIAWAN BAGASKARA**

**L200214253**

**INFORMATICS STUDY PROGRAM**

**FACULTY OF COMMUNICATION AND INFORMATION SCIENCE**

**MUHAMMADIYAH SURAKARTA UNIVERSITY**

Experimen 8.3

class Stack:

    def \_\_init\_\_(*self*):

*self*.items = []

    def isEmpty(*self*):

        return len(*self*) == 0

    def \_\_len\_\_(*self*):

        return len(*self*.items)

    def peek(*self*):

        assert not *self*.isEmpty()

        return *self*.items[-1]

    def pop(*self*):

        assert not *self*.isEmpty()

        return *self*.items.pop()

    def push(*self*, *data*):

*self*.items.append(*data*)

class StackLL:

    def \_\_init\_\_(*self*):

*self*.top = None

*self*.size = 0

    def isEmpty(*self*):

        return *self*.top is None

    def \_\_len\_\_(*self*):

        return *self*.size

    def peek(*self*):

        assert not *self*.isEmpty()

        return *self*.top.item

    def pop(*self*):

        assert not *self*.isEmpty()

        node = *self*.top

*self*.top = *self*.top.next

*self*.size -= 1

        return node.item

    def push(*self*):

*self*.top = \_StackNode(data, *self*.top)

*self*.size += 1

class \_StackNode:

    def \_\_init\_\_(*self*, *data*, *link*):

*self*.item = *data*

*self*.next = *link*

##PROMPT = "Masukkan bilangan positif (<0 untuk mengakhiri) : "

##myStack = Stack()

##value = int(input(PROMPT))

##while value >= 0:

##    myStack.push(value)

##    value = int(input(PROMPT))

##while not myStack.isEmpty():

##    value = myStack.pop()

##    print(value)

9

Experimen 8.4

from latihan83 import Stack

def cetakBiner(*d*):

    f = Stack()

    if *d*==0: f.push(0)*;*

    while *d* !=0:

        sisa = *d*%2

*d* = *d*//2

        f.push(sisa)

    st = ""

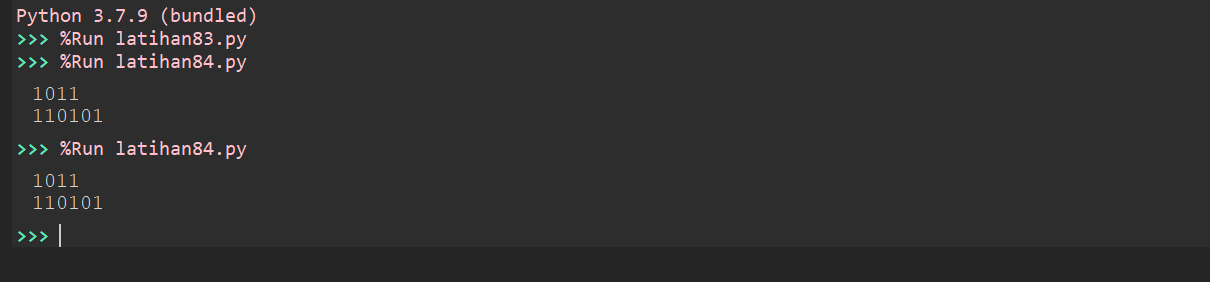
    for i in range(len(f)):

        st = st + str(f.pop())

    return st

print(cetakBiner(11))

print(cetakBiner(53))



Experimen 8.6

class Queue(object):

    def \_\_init\_\_(*self*):

*self*.qlist = []

    def isEmpty(*self*):

        return len(*self*) == 0

    def \_\_len\_\_(*self*):

        return len(*self*.qlist)

    def enqueue(*self*, *data*):

*self*.qlist.append(*data*)

    def dequeue(*self*):

        assert not *self*.isEmpty(), "Antrian sedang kosong"

        return *self*.qlist.pop(0)

Q = Queue()

Q.enqueue(28)

Q.enqueue(19)

Q.enqueue(45)

Q.enqueue(13)

Q.enqueue(7)

print(Q.qlist)

Q.dequeue()

Q.dequeue()

Q.dequeue()

Q.dequeue()

Q.dequeue()

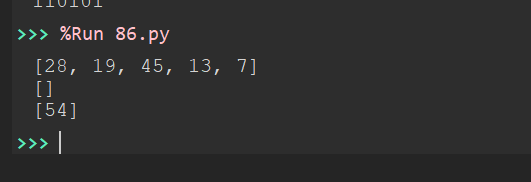
print(Q.qlist)

Q.enqueue(98)

Q.enqueue(54)

Q.dequeue()

print(Q.qlist)



Experimen 8.7

class PriorityQueue(object):

    def \_\_init\_\_(*self*):

*self*.qlist = []

    def \_\_len\_\_(*self*):

        return len(*self*.qlist)

    def isEmpty(*self*):

        return len(*self*) == 0

    def enqueue(*self*, *data*, *priority*):

        entry = \_PriorityQEntry(*data*, *priority*)

*self*.qlist.append(entry)

    def dequeue(*self*):

        pass

class \_PriorityQEntry(object):

    def \_\_init\_\_(*self*, *data*, *priority*):

*self*.item = *data*

*self*.priority = *priority*

    def \_\_str\_\_(*self*):

        return 'Item: {}\nPriority: {}'.format(*self*.item, *self*.priority)

S = PriorityQueue()

S.enqueue('Jeruk', 4)

S.enqueue('Tomat', 2)

S.enqueue('Mangga', 0)

S.enqueue('Duku', 5)

S.enqueue('Papaya', 2)

for i in S.qlist:

    print(i)

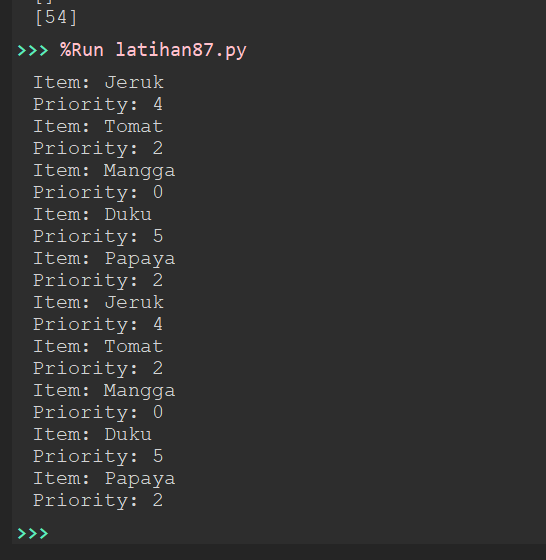
S.dequeue()

S.dequeue()

S.dequeue()

for i in S.qlist:

    print(i)



**TASK 1**

class Stack(object):

def \_\_init\_\_(self):

self.items = []

def isEmpty(self):

return len(self) == 0

def \_\_len\_\_(self):

return len(self.items)

def peek(self):

assert not self.isEmpty(), "Tidak bisa diintip. Stack kosong"

return self.items[-1]

def pop(self):

assert not self.isEmpty(), "Tidka bisa dipop dari Stack kosong"

return self.items.pop()

def push(self, data):

self.items.append(data)

def cetakHexa(d):

f = Stack()

if d == 0: f.push(0);

while d != 0:

sisa = d%16

d = d//16

if sisa == 10:

sisa = "A"

elif sisa == 11:

sisa = "B"

elif sisa == 12:

sisa = "C"

elif sisa == 13:

sisa = "D"

elif sisa == 14:

sisa = "E"

elif sisa == 15:

sisa = "F"

f.push(sisa)

st = ""

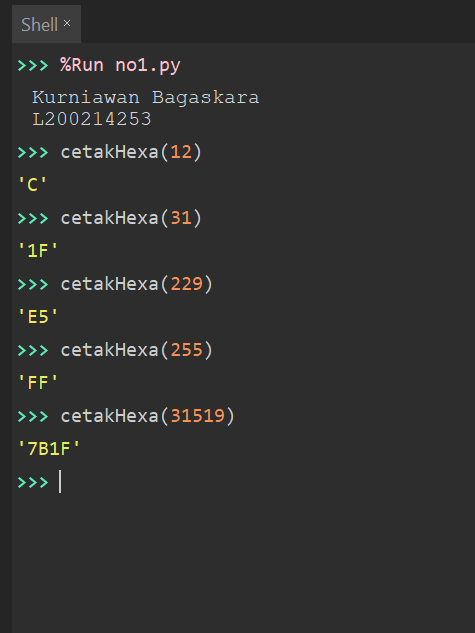
for i in range (len(f)):

st = st + str(f.pop())

return st

print ("Kurniawan Bagaskara")

print ("L200214253")



**TASK 2**

class Stack(object):

def \_\_init\_\_(self):

self.items = []

def isEmpty(self):

return len(self) == 0

def \_\_len\_\_(self):

return len(self.items)

def peek(self):

assert not self.isEmpty(), "Tidak bisa diintip. Stack kosong"

return self.items[-1]

def pop(self):

assert not self.isEmpty(), "Tidak bisa dipop dari Stack kosong"

return self.items.pop()

def push(self, data):

self.items.append(data)

nilai = Stack()

for i in range(16):

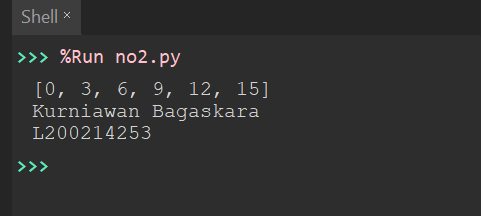
if i%3 == 0:

nilai.push(i)

print(nilai.items)

print ("Kurniawan Bagaskara")

print ("L200214253")



**TASK 3**

class Stack(object):

def \_\_init\_\_(self):

self.items = []

def isEmpty(self):

return len(self) == 0

def \_\_len\_\_(self):

return len(self.items)

def peek(self):

assert not self.isEmpty(), "Tidka bisa diintip. Stack kosong"

return self.items[-1]

def pop(self):

assert not self.isEmpty(), "Tidka bisa dipop dari Stack kosong"

return self.items.pop()

def push(self, data):

self.items.append(data)

nilai = Stack()

for i in range (16):

if i%3 == 0:

nilai.push(i)

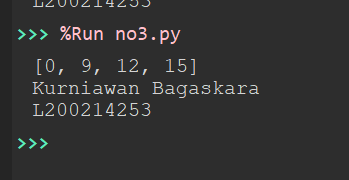
elif i%4 == 0:

nilai.pop()

print(nilai.items)

print ("Kurniawan Bagaskara")

print ("L200214253")



**TASK 4**

class Queue(object):

def \_\_init\_\_(self):

self.qlist = []

def isEmpty(self):

return len(self) == 0

def \_\_len\_\_(self):

return len(self.qlist)

def enqueue(self, data):

self.qlist.append(data)

def dequeue(self):

assert not self.isEmpty(), "Antrian sedang kosong"

return self.qlist.pop(0)

def getFrontMost(self):

return self.qlist[0]

def getRearMost(self):

return self.qlist[-1]

class PriorityQueue(object):

def \_\_init\_\_(self):

self.qlist = []

def isEmpty(self):

return len(self) == 0

def \_\_len\_\_(self):

return len(self.qlist)

def enqueue(self, data, priority):

entry = \_PriorityQEntry(data, priority)

self.qlist.append(entry)

def getFrontMost(self):

x = 0

while self.qlist[x].priority != 0:

x+=1

return self.qlist[x].item

def getRearMost(self):

a = []

for i in self.qlist:

a.append(i.priority)

print (self.qlist[a.index(max(a))].item)

class \_PriorityQEntry(object):

def \_\_init\_\_(self, data, priority):

self.item = data

self.priority = priority

A = Queue()

A.enqueue(28)

A.enqueue(19)

A.enqueue(45)

A.enqueue(13)

A.enqueue(7)

S = PriorityQueue()

S.enqueue("Jeruk", 4)

S.enqueue("Tomat", 2)

S.enqueue("Mangga", 0)

S.enqueue("Duku", 5)

S.enqueue("Pepaya", 2)

print ("Kurniawan Bagaskara")

print ("L200214253")

**TASK 5**

class PriorityQueue(object):

def \_\_init\_\_(self):

self.qlist = []

def isEmpty(self):

return len(self) == 0

def \_\_len\_\_(self):

return len(self.qlist)

def enqueue(self, data, priority):

entry = \_PriorityQEntry(data, priority)

self.qlist.append(entry)

def dequeue(self):

assert not self.isEmpty(), "Antrian sedang kosong"

a = []

for i in self.qlist:

a.append(i.priority)

print (self.qlist.pop(a.index(min(a))).item)

class \_PriorityQEntry(object):

def \_\_init\_\_(self, data, priority):

self.item = data

self.priority = priority

S = PriorityQueue()

S.enqueue("Jeruk", 4)

S.enqueue("Tomat", 2)

S.enqueue("Mangga", 0)

S.enqueue("Duku", 4)

S.enqueue("Pepaya", 2)

print ("Kurniawan Bagaskara")

print ("L200214253")