

Nama : Rohmad Khoirudin
NIM : L200180101
Kelas : D

Prak-ASD

Modul 8

1). Buatlah Program untuk mengubah representasi suatu bilangan dari basis sepuluh ke basis dua.

```
Python 3.7.6 Shell
File Edit Shell Debug Options Window Help
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_8\1.py =====>>>
>>> cetakHex(12)
'C'
>>> cetakHex(31)
'1F'
>>> cetakHex(229)
'E5'
>>> cetakHex(225)
'E1'
>>> cetakHex(255)
'FF'
>>> cetakHex(31519)
'7B1F'
>>>

1.py - D:\smt 4\prak algo\modul_8\1.py (3.7.6)
File Edit Format Run Options Window Help
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)

def cetakHex(d):
    f = Stack()
    if d == 0: f.push(0);
    while d != 0:
        if d%16 == 10:
            sisa = "A"
        elif d%16 == 11:
            sisa = "B"
        elif d%16 == 12:
            sisa = "C"
        elif d%16 == 13:
            sisa = "D"
        elif d%16 == 14:
            sisa = "E"
        elif d%16 == 15:
            sisa = "F"
        else:
            sisa = d%16
        d=d//16
        f.push(sisa)
    st = ""
    for i in range(len(f)):
        st = st + str(f.pop())
    return st

Ln: 17 Col: 4
Ln: 25 Col: 24
```

2). Eksekusi program berikut. Dan tunjukan isi stacknya

```
Python 3.7.6 Shell
File Edit Shell Debug Options Window Help
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_8\2.py =====>>>
>>> for i in range(16):
>>>     if i%3==0:
>>>         nilai.push(i)
>>>
>>> nilai.pop()
15
>>> nilai.pop()
12
>>> nilai.pop()
9
>>> nilai.pop()
6
>>> nilai.pop()
3
>>> nilai.pop()
0
>>>

2.py - D:\smt 4\prak algo\modul_8\2.py (3.7.6)
File Edit Format Run Options Window Help
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)

nilai = Stack()
for i in range(16):
    if i%3==0:
        nilai.push(i)

Ln: 21 Col: 4
Ln: 22 Col: 0
```

3). Eksekusi program berikut. Dan tunjukan isi stacknya

The image shows two side-by-side Python 3.7.6 Shell windows. The left window displays the execution of a script named 'modul_8\3.py'. The script defines a 'Stack' class with methods: __init__, __len__, __isEmpty__, __peek__, __pop__, and __push__. It then creates a 'nilai' instance of the Stack class and tests its methods. The right window shows the source code of the '3.py' file, which is identical to the code being executed in the left window.

```
Python 3.7.6 Shell
File Edit Shell Debug Options Window Help
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_8\3.py =====
>>> for i in range(16):
    if i % 3 == 0:
        nilai.push(i)
    elif i%4==0:
        nilai.pop()

3
6
>>> nilai.pop()
15
>>> nilai.pop()
12
>>> nilai.pop()
9
>>> nilai.pop()
0
>>>
```

```
3.py - D:\smt 4\prak algo\modul_8\3.py (3.7.6)
File Edit Format Run Options Window Help

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)

nilai = Stack()
for i in range(16):
    if i % 3 == 0:
        nilai.push(i)
    elif i%4==0:
        nilai.pop()
```

4). Tulis dua metode berikut ke kelas Queue dan class PriorityQueue

- Metode untuk mengetahui item yang paling depan dan belakang tanpa menghapusnya pada Queue

The image shows two side-by-side terminal windows. The left window is a Python 3.7.6 Shell with the following content:

```
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_8\4.py =====
>>> q = Queue()
>>> q.enqueue(7)
>>> q.enqueue(11)
>>> q.enqueue(12)
>>> q.enqueue(18)
>>> q.getFrontMost()
7
>>> q.getRearMost()
18
>>>
```

The right window is a Python 3.7.6 Shell with the following content:

```
class Queue():
    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[len(self.qlist)-1]
```

- Metode untuk mengetahui item yang paling depan dan belakang tanpa menghapusnya pada PriorityQueue

```

Python 3.7.6 Shell
File Edit Shell Debug Options Window Help
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_8\4.py =====
>>> q = PriorityQueue()
>>> q.enqueue("rohmad", 5)
>>> q.enqueue("khoir", 2)
>>> q.enqueue("udin", 11)
>>> q.enqueue("khoirudin", 6)
>>> q.getFrontMost()
(11, 'udin')
>>> q.getRearMost()
(2, 'khoir')
>>>

4.py - D:\smt 4\prak algo\modul_8\4.py (3.7.6)
File Edit Format Run Options Window Help
import heapq
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):
        heapq.heappush(self.qlist, (priority, data))
        self.qlist.sort()
    def dequeue(self):
        return self.qlist.pop(-1)
    def getFrontMost(self):
        return self.qlist[-1]
    def getRearMost(self):
        return self.qlist[0]

q = PriorityQueue()
q.enqueue("rohmad", 5)
q.enqueue("khoir", 2)
q.enqueue("udin", 11)
q.enqueue("khoirudin", 6)
  
```

5). Pada Class PriorityQueue pada latihan, metode dequeue() belum diimplementasikan. Tulisakan metode dequeuennya

```

Python 3.7.6 Shell
File Edit Shell Debug Options Window Help
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_8\5.py =====
>>> q.dequeue()
khoir
>>> q.dequeue()
rohmad
>>> q.dequeue()
khoirudin
>>> q.dequeue()
udin
>>>

5.py - D:\smt 4\prak algo\modul_8\5.py (3.7.6)
File Edit Format Run Options Window Help
class PriorityQueue():
    def __init__(self):
        self.qlist=[]
    def isEmpty(self):
        return len(self)==0
    def __len__(self):
        return len(self.qlist)
    def enqueue(self, item, priority):
        entry = PriorityQue(item, priority)
        self.qlist.append(entry)
    def dequeue(self):
        n = []
        for i in self.qlist:
            n.append(i.priority)
        print (self.qlist.pop(n.index(min(n))).item)
    def getFrontMost(self):
        return self.qlist[0]
    def getRearMost(self):
        return self.qlist[len(self.qlist)-1]

class _PriorityQue():
    def __init__(self, data, priority):
        self.item = data
        self.priority= priority

q = PriorityQueue()
q.enqueue("rohmad", 5)
q.enqueue("khoir", 2)
q.enqueue("udin", 11)
q.enqueue("khoirudin", 6)
  
```

Atau bisa juga dengan

Python 3.7.6 Shell

File Edit Shell Debug Options Window Help

Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\smt 4\prak algo\modul_8\5.py =====>>>
(2, 'khoir')
(11, 'udin')
[(2, 'khoir'), (5, 'rohmad'), (6, 'khoirudin'), (11, 'udin')]
>>> q.dequeue()
(2, 'khoir')
>>> q.dequeue()
(5, 'rohmad')
>>> q.dequeue()
(6, 'khoirudin')
>>> q.dequeue()
(11, 'udin')
>>>

Ln: 14 Col: 4

*5.py - D:\smt 4\prak algo\modul_8\5.py (3.7.6)

File Edit Format Run Options Window Help

import heapq
class PriorityQueue(object):
 ##### cara pertama
 def __init__(self):
 self.qlist = []
 def __len__(self):
 return len(self.qlist)
 def isEmpty(self):
 return len(self) == 0
 def enqueue(self, data, priority):
 heapq.heappush(self.qlist, (priority, data))
 self.qlist.sort()
 def dequeue(self):
 return self.qlist.pop(0)
 def getFrontMost(self):
 return self.qlist[0]
 def getRearMost(self):
 return self.qlist[-1]

q = PriorityQueue()
q.enqueue("rohmad", 5)
q.enqueue("khoir", 2)
q.enqueue("udin", 11)
q.enqueue("khoirudin", 6)

print(q.getFrontMost())
print(q.getRearMost())
print(q.qlist)

Ln: 19 Col: 4