Nama : Windiapriani Ginayawati

NIM : L200170157

Kelas : D

Modul : IV

---

1. Membuat sebuah fungsi pencarian pada list.
   Berikut ini adalah class *Mahasiswa* dengan list *Daftar*

```
class Mahasiswa(object):
    """Class Manusia yang dibangun dari class manusia"""
    def __init__(self,nama,NIM,kota,us):
        """Metode inisiasi ini menutupi metode inisiasi di class Manusia"""
        self.nama = nama
        self.NIM = NIM
        self.kotaTinggal = kota
        self.uangSaku = us
c0 = Mahasiswa ('Ika',10,'Sukoharjo', 240000)
c1 = Mahasiswa ('Budi',51,'Sragen', 230000)
c2 = Mahasiswa ('Ahmad',2,'Surakarta', 250000)
c3 = Mahasiswa ('Chandra',18,'Surakarta', 230000)
c4 = Mahasiswa ('Eka',4,'Boyolali', 240000)
c5 = Mahasiswa ('Fandi',31,'Salatiga', 250000)
c6 = Mahasiswa ('Deni',13,'Klaten', 245000)
c7 = Mahasiswa ('Galuh',5,'Wonogiri', 245000)
c8 = Mahasiswa ('Janto',23,'Klaten', 245000)
c9 = Mahasiswa ('Hasan',64,'Karanganyar', 270000)
c10 = Mahasiswa ('Khalid',29,'Purwodadi', 265000)

Daftar = [c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10]
```

   Code untuk no. 1

```
#1
print ("No. 1")
def cari(data,target):
    a = []
    b = 0
    for i in data:
        if i.kotaTinggal == target:
            a.append(b)
            b += 1
        else :
            b += 1
    return a
print(cari(Daftar, "Klaten"))
```

2. Dari list Daftar tadi, buat sebuah fungsi untuk menemukan uang saku yang terkecil.
   Code untuk no. 2

```
#2
print ("\nNo. 2")
def cariTerkecil (self):
    terkecil = self[0].uangSaku
    for i in self:
        if i.uangSaku < terkecil :
            terkecil = i.uangSaku
    return terkecil
print("Uang saku terkecil di dalam daftar: ",cariTerkecil(Daftar))
```

3. Mengubah program sebelumnya agar mengembalikan objek mahasiswa yang mempunyai uang saku terkecil. Jika ada lebih dari satu mahasiswa yang uang sakunya terkecil, semua objek mahasiswa itu dikembalikan.
   Code untuk no. 3

```
#3
print ("\nNo. 3")
def cariTerkecil (self):
    terkecil = self[0].uangSaku
    c = []
    for i in self:
        if i.uangSaku < terkecil :
            c.append ((i.nama, i.NIM, i.kotaTinggal, i.uangSaku))
    return c
print(cariTerkecil(Daftar))
```

4. Membuat fungsi untuk mengembalikkan semua objek mahasiswa yang uang sakunya kurang dari 250000
   Code untuk no. 4

```
#4
print ("\nNo. 4")
def terkecil (self):
    terkecil = 250000
    d = []
    for i in self:
        if i.uangSaku < 250000 :
            d.append((i.nama, i.NIM, i.kotaTinggal, i.uangSaku))
    for i in d :
        print (i)
print(terkecil(Daftar))
```

5. Buat suatu program untuk mencari suatu item di sebuah linked list
   Code untuk no. 5

```
#5
print ("\nNo. 5")
class node (object):
    def __init__ (self, data, next = None):
        self.data = data
        self.next = next
    def cari (self, cari):
        curNode = self
        while curNode is not None :
            if curNode.next != None :
                if curNode.data != cari :
                    curNode = curNode.next
                else :
                    print ("Data", cari, "ada dalam Linked List")
                    break
            elif curNode.next == None :
                print ("Data", cari, "tidak ada linked list")
                break
a = node (12)
menu = a
a.next = node (34)
a = a.next
a.next = node (16)
a = a.next
a.next = node (45)
menu.cari(16)
menu.cari(110)
```

6. Membuat fungsi Binary search agar mengembalikan index lokasi elemen yang ditemukan.
   Code untuk no. 6

```
#6
print ("\nNo. 6")
def binSe(kumpulan, target):
    #Mulai dari seluruh runtutan elemen
    low = 0
    high = len(kumpulan) -1
    data = []

    #Secara berulang belah runtutan itu menjadi separuhnya
    #sampai targetnya ditemukan
    while low <= high:
        #Temukan pertengahan runtut itu
        mid = (high + low) //2
        #Apakah pertengahannya memuat target?
        if kumpulan[mid] == target:
            data.append(kumpulan.index(target))
            return True
        #ataukah targetnya di sebelah kirinya?
        elif target < kumpulan[mid]:
            high = mid -1
        #ataukah targetnya di sebelah kanannya?
        else :
            low = mid +1
        #Jika runtutnya tidak bisa dibelah lagi, berarti targetnya tidak ada
    return False

list = [35, 67, 89, 57, 689]
target1 = 345
target2 = 67
print ("nilai target :", target1)
print (binSe(list, target1))
print ("\n nilai target :", target2)
print (binSe(list, target2))
```

7. Mengubah fungsi sebelumnya agar mengembalikan semua index lokasi elemen yang ditemukan.
   Code untuk no. 7

```
#7
print ("\nNo. 7")
def binSearch(kumpulan, target):
    #Mulai dari seluruh runtutan elemen
    low = 0
    high = len(kumpulan) -1
    data = []

    #Secara berulang belah runtutan itu menjadi separuhnya
    #sampai targetnya ditemukan
    while low != high:
        #Temukan pertengahan runtut itu
        mid = (high + low) //2
        #Apakah pertengahannya memuat target?
        if kumpulan[mid] == target:
            break
        #ataukah targetnya di sebelah kirinya?
        elif target < kumpulan[mid]:
            high = mid -1
        #ataukah targetnya di sebelah kanannya?
        else :
            low = mid +1
    for i in range (low, high):
        if target == kumpulan[i]:
            data.append(i)
    return data
lis = [2, 3, 4, 5, 6, 7, 8, 9, 9, 12]
cari = 7
print ("posisi data ", cari, "pada list ", lis, "adalah ")
print (binSearch(lis, cari))
```

8. Code untuk no. 8

```
#8
print ("""\nNo. 8
Ada 2 kemungkinan pola yang bisa digunakan.
Misalkan, angka yang akan ditebak adalah 70.

Pola pertama :
    a = nilai tebakan pertama // 2
    tebakan selanjutnya = nilai tebakan "lebih dari" + a

    "jika hasil tebakab selanjutnya "kurang dari", maka nilai yang dipakai tetap
    nilai lebih dari sebelumnya"

    a = a // 2

Simulasi
    tebakan 1 : 50 (mengambil nilai tengah) jawaban "lebih dari itu"
    tebakan 2 : 75 (lebih dari 50) jawaban "kurang dari itu"
    tebakan 3 : 62 (kurang dari 75) jawaban "lebih dari itu"
    tebakan 4 : 68 (lebih dari 62) jawaban "lebih dari itu"
    tebakan 5 : 71 (lebih dari 68) jawaban "kurang dari itu"
    tebakan 6 : 69 (kurang dari 71) jawaban "lebih dari itu"
    tebakan 7 : antara 71 dan 69, jadi jawabannya 70

Pola kedua :
    menggunakan barisan geometri Sn = 2^n

    Barisan yang terjadi 2, 4, 8, 16, 32, 64
    Misal angka yang akan ditebak adalah 68
    tebakan 1 : 64 jawaban "lebih dari itu"
    tebakan 2 : 96 (64 + 32) jawaban "kurang dari itu"
    tebakan 3 : 80 (64 + 16) jawaban "kurang dari itu"
    tebakan 4 : 72 (64 + 8) jawaban "kurang dari itu"
    tebakan 5 : 68 (64 + 4) jawaban "lebih dari itu"
    tebakan 6 : 70 (64 + 2) jawaban "Pas"
""")
```

Hasil Run program dari no 1-6

```
No. 1
[6, 8]

No. 2
Uang saku terkecil di dalam daftar:  230000

No. 3
[('Budi', 51, 'Sragen', 230000), ('Chandra', 18, 'Surakarta', 230000)]

No. 4
('Ika', 10, 'Sukoharjo', 240000)
('Budi', 51, 'Sragen', 230000)
('Chandra', 18, 'Surakarta', 230000)
('Eka', 4, 'Boyolali', 240000)
('Deni', 13, 'Klaten', 245000)
('Galuh', 5, 'Wonogiri', 245000)
('Janto', 23, 'Klaten', 245000)
None

No. 5
Data 16 ada dalam Linked List
Data 110 tidak ada linked list

No. 6
nilai target : 345
False

 nilai target : 67
True
```

Hasil Run program dari no 7-8

```
No. 7
posisi data  7 pada list  [2, 3, 4, 5, 6, 7, 8, 9, 9, 12] adalah
[5]

No. 8
Ada 2 kemungkinan pola yang bisa digunakan.
Misalkan, angka yang akan ditebak adalah 70.

Pola pertama :
    a = nilai tebakan pertama // 2
    tebakan selanjutnya = nilai tebakan "lebih dari" + a

    "jika hasil tebakab selanjutnya "kurang dari", maka nilai yang dipakai tetap
    nilai lebih dari sebelumnya"

    a = a // 2

Simulasi
    tebakan 1 : 50 (mengambil nilai tengah) jawaban "lebih dari itu"
    tebakan 2 : 75 (lebih dari 50) jawaban "kurang dari itu"
    tebakan 3 : 62 (kurang dari 75) jawaban "lebih dari itu"
    tebakan 4 : 68 (lebih dari 62) jawaban "lebih dari itu"
    tebakan 5 : 71 (lebih dari 68) jawaban "kurang dari itu"
    tebakan 6 : 69 (kurang dari 71) jawaban "lebih dari itu"
    tebakan 7 : antara 71 dan 69, jadi jawabannya 70

Pola kedua :
    menggunakan barisan geometri Sn = 2^n

    Barisan yang terjadi 2, 4, 8, 16, 32, 64
    Misal angka yang akan ditebak adalah 68
    tebakan 1 : 64 jawaban "lebih dari itu"
    tebakan 2 : 96 (64 + 32) jawaban "kurang dari itu"
    tebakan 3 : 80 (64 + 16) jawaban "kurang dari itu"
    tebakan 4 : 72 (64 + 8) jawaban "kurang dari itu"
    tebakan 5 : 68 (64 + 4) jawaban "lebih dari itu"
    tebakan 6 : 70 (64 + 2) jawaban "Pas"
```