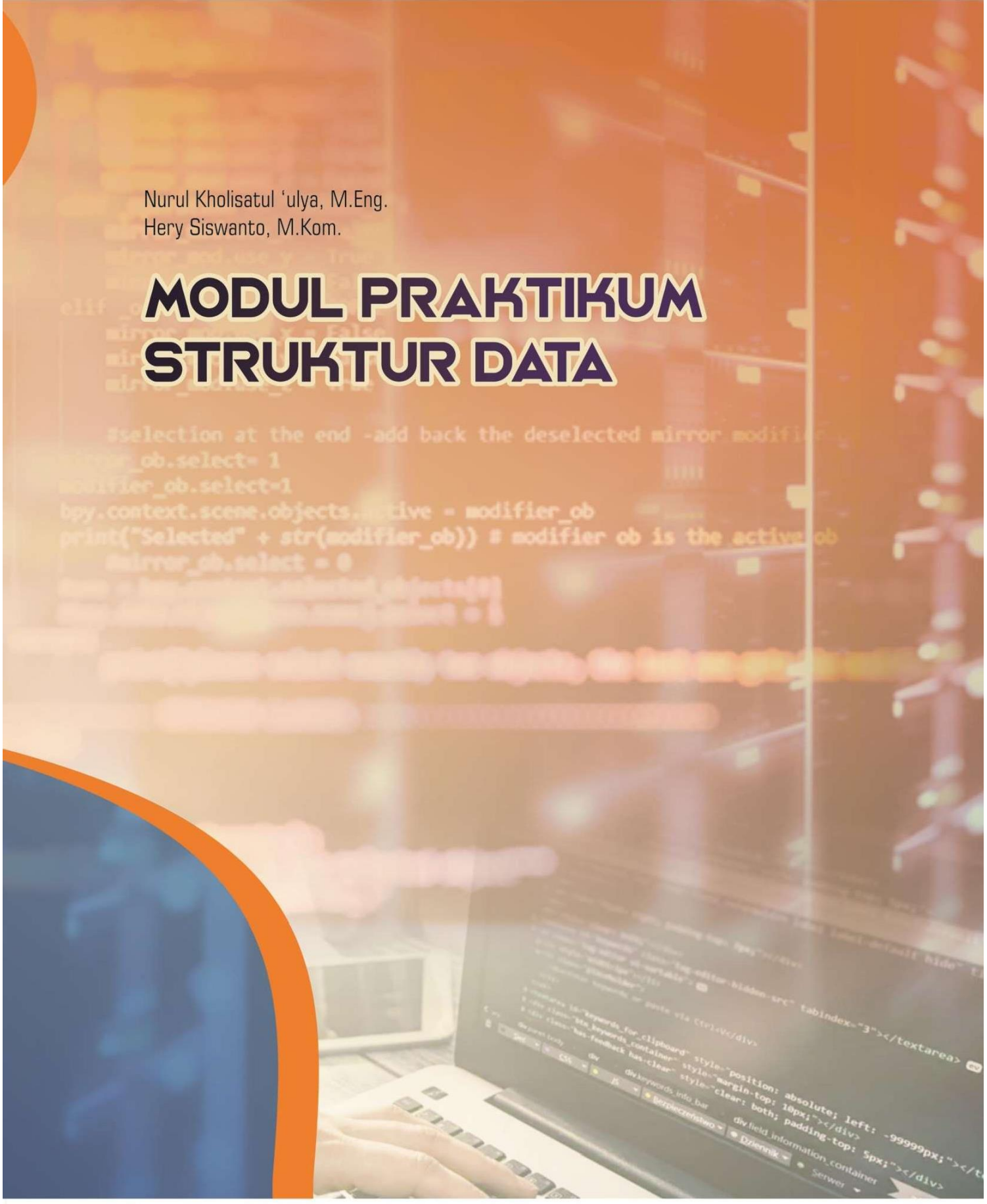
Nurul Kholisatul 'ulya, M.Eng.
Hery Siswanto, M.Kom.

MODUL PRAKTIKUM STRUKTUR DATA



PROGRAM STUDI S1 INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
ITS PKU MUHAMMADIYAH SURAKARTA
TAHUN AKADEMIK 2021/2022

MODUL PRAKTIKUM
STRUKTUR DATA

TIM PENYUSUN:
Nurul Kholisatul ,,ulya, M.Eng.
Hery Siswanto, M.Kom

PROGRAM STUDI INFORMATIKA PROGRAM SARJANA
FAKULTAS SAINS DAN TEKNOLOGI
ITS PKU MUHAMMADIYAH SURAKARTA
TAHUN AKADEMIK 2021/2022

MODUL PRAKTIKUM
STRUKTUR DATA

ISBN: 978-623-5516-55-4

Penyusun

Nurul Kholisatul „ulya, M.Eng.
Hery Siswanto, M.Kom.

Penyunting:

Septi Aprilia, M.Pd.

Penerbit

CV. AE Media Grafika
Website: www.aemediagrafika.com
Email: aemediagrafika@gmail.com

Bekerjasama dengan

Prodi Informatika Program Sarjana
Fakultas Sains dan Teknologi
ITS PKU Muhammadiyah Surakarta
Jl. TulangBawang Selatan No. 26 Tegalsari Rt. 01/32 Kadipiro Surakarta
Telp.(0271) 734955, Faks. (0271) 73495557136,
Websit1e:www.itspku.ac.id, E-mail : info@itspku.ac.id

VISI DAN MISI

PRODI INFORMATIKA PROGRAM SARJANA

FAKULTAS SAINS DAN TEKNOLOGI

ITS PKU MUHAMMADIYAH SURAKARTA

VISI

Menjadi Program Studi Informatika Program Sarjana yang unggul di bidang *software engineering* dan *technopreneur*, Berkarakter Islam Berkemajuan pada tahun 2031

MISI

1. Menyelenggarakan dan mengembangkan pendidikan informatika dengan unggulan di bidang *software engineering* dan *technopreneur* yang profesional untuk menghasilkan lulusan yang unggul dan memiliki nilai-nilai Islami.
2. Menyelenggarakan dan mengembangkan penelitian informatika dengan bidang unggulan di bidang *software engineering* dan *technopreneur* dalam rangka mendukung pembangunan nasional.
3. Menyelenggarakan dan mengembangkan pelayanan pengabdian kepada masyarakat berbasis pada hasil penelitian informatika di bidang *software engineering* dan *technopreneur* untuk menyelesaikan masalah.
4. Menyelenggarakan pembinaan dan penguatan nilai-nilai Al Islam dan Kemuhammadiyahan pada semua civitas akademika

KATA PENGANTAR

Alhamdulillah, puji dan syukur atas kehadirat Allah yang telah memberikan kemampuan serta kesempatan kepada penulis untuk menyusun buku “Praktikum Struktur Data” ini.

Bab-bab dalam buku ini berisi tentang lanjutan praktikum Struktur Data yang diterapkan menggunakan bahasa pemrograman python. Bahasa pemrograman python merupakan bahasa pemrograman yang cukup mudah bagi pemula, selain itukarena kehandalannya dalam pembuatan berbagai aplikasi maka bahasa ini tepat untuk dipelajari.

Penulis berharap buku ini dapat digunakan secara maksismal dalam memahamkan mahasiswa tentang penerapan struktur data . Tak lupa penulis mengucapkan terima kasih kepada berbagai pihak yang membantu dalam penyelesaian buku ini. Kritik dan saran sangat penulis harapkan demi penyempurnaan buku ini.

Semoga buku ini bermanfaat

Surakarta, September 2021

Penulis

TATA TERTIB DAN TATA CARA
PRAKTIKUM STRUKTUR DATA

Demi menjaga kelancaran jalannya praktikum Struktur Data, praktikan diwajibkan memenuhi tata tertib dan tata cara seperti yang tertera di bawah ini:

TATA TERTIB

1. Praktikan dapat mengikuti praktikum bila memenuhi syarat-syarat sebagai berikut:
   a. Terdaftar pada KRS.
   b. Membawa Kartu Laporan Hasil Praktikum
   c. Berpakaian sopan. Pria menggunakan kemeja bahan berkerah dan celana bahan. Wanita menggunakan rok dan kemeja atau gamis serta wajib menggunakan jilbab. Tidakdiperkenankan menggunakan yang berbahan kaos atau jeans ketat.
   d. Menggunakan sepatu tertutup.
2. Praktikan harus hadir sesuai jadwal praktikum (jam dan hari). Toleransi keterlambatan diberikan 10 menit setelah praktikum dimulai.
3. Ketika memasuki Laboratorium:
   a. Harus tenang, tertib, dan sopan.
   b. Dilarang membawa makanan, minuman, rokok dan barang-barang yang tidak diperlukan pada saat praktikum.
4. Selama praktikum berlangsung, praktikan :
   a. Dilarang meninggalkan Laboratorium tanpa seijin penanggung jawab praktikum pada hari tersebut.
   b. Harus dapat menjaga keselamatan diri, alat-alat dan kebersihan Laboratorium.
5. Praktikan harus mengganti alat-alat yang rusak/hilang selama praktikum berlangsung dengan alat yang sama.

6. Setelah praktikum selesai dan disetujui, praktikan:
   a. Melaporkan kelengkapan alat-alat yang digunakan pada yang bersangkutan.
   b. Menuliskan laporan hasil praktikum pada tiap pertemuan.
   c. Harus meminta paraf/tanda tangan pada penanggung jawab praktikum untuk mengesahkan hasil praktikum.
7. Bagi praktikan yang berhalangan hadir karena sakit dapat menunjukkan surat keterangan dokter paling lambat pada saat praktikum berlangsung (diwakilkan). Melampaui waktu tersebut praktikan dinyatakan **GAGAL 1 KALI**. Batas maksimum untuk tidak hadir praktikum (absen) adalah sebanyak tiga kali dan wajib mengulang di praktikum semester berikutnya.
8. Tata tertib ini untuk dilaksanakan dengan penuh kesadaran.

TATA CARA

1. Pada setiap pertemuan praktikum terdapat tugas yang wajib dikerjakan oleh praktikan.
2. Kriteria penilaian praktikum meliputi kelengkapan dan kerapihan tugas. Apabila tugas tersebut tidak memenuhi syarat, maka akan ada pemberitahuan untuk diperbaiki dan dikumpulkan kembali.
3. Tugas-tugas atau Laporan Praktikum tidak boleh sama antara sesama Praktikan, apabila ditemukan ada tugas dan laporan yang sama, maka yang bersangkutan akan diberlakukan pengurangan nilai.
4. Jika praktikan tidak mengumpulkan atau tidak mengerjakan salah satu dari tugas-tugas dan laporan yang diberikan atau tidak mengumpulkan tugas dan laporan pada hari yang telah ditentukan, maka diberlakukan pengurangan nilai dan tidak ada proses perbaikan.
5. Segala peraturan yang dirasa perlu, dapat ditambah sewaktu-waktu.
6. Praktikan dapat diberikan peringatan, dikeluarkan ataupun digagalkan jika melanggar tata tertib dan tata cara praktikum ini.
7. **SELESAI**

DAFTAR ISI

BAB 1

OVERVIEW PYTHON: TIPE DATA PADA STRUKTUR DATA PYTHON

A. TUJUAN

1. Mahasiswa mampu mereview bahasa pemrograman python
2. Mahasiswa mampu mereview tipe data pada struktur data python

B. DASAR TEORI

1. List

Selain kelas numerik dan boolean, Python mempunyai sejumlah kelas koleksi built-in yang sangat powerful. Lists, strings dan tuples adalah koleksi terurut (*ordered*) yang sangat mirip secara struktur tetapi mempunyai perbedaan spesifik yang harus dipahami agar dapat digunakan dengan tepat. Sets dan dictionaries adalah koleksi tak-terurut (*unordered*).

List (daftar) adalah suatu koleksi berurut beranggotakan nol atau lebih rujukan ke objek data Python. List ditulis sebagai nilai-nilai terpisahkan koma di dalam kurung siku. List kosong diwakili oleh [ ]. List bersifat heterogen, artinya objek-objek data di dalamnya tidak harus berasal dari kelas yang sama dan koleksi tersebut dapat berikan ke suatu variabel.

Kegiatan 1

Coba tuliskan 3 baris instruksi ini di dalam modus interaktif (menggunakan software IDLE) dan perhatikan hasilnya:

```
>>>[2,5 ,4.7, False]
>>>myList = [2,5 ,4.7, False]
>>>myList
```

Tabel 1.1 Operasi terhadap Sequence di dalam Python

Nama Operasi	Operator	Penjelasan
Indexing	[]	Mengakses suatu elemen sequence
Concatenation	+	Menggabungkan sequence berama-sama
Repetition	*	Menggabungkan sejumlah berulang kali
Membership	In	Apakah suatu item adalah di dalam sequence
Length	Len	Jumlah item di dalam sequence
Slicing	[:]	Mengekstrak bagian dari suatu sequence

Kegiatan 2

Coba jalankan instruksi-instruksi berikut pada modus Interaktif (IDLE):

```
>>> myList = [1,2,3,4]
>>>A = [myList]*3
>>>print(A)
>>>myList[2]=45
>>>print(A)
```

Tabel 1.2 Metode-metode yang disediakan oleh List di dalam Python

Nama Metode	Cara Penggunaan	Penjelasan
Append	alist.append(item)	Menambahkan suatu item baru ke akhir list
Insert	alist.insert(i,item)	Menyisipkan suatu item ke dalam list pada posisi ke-i
Pop	alist.pop()	Menghapus & mengembalikan item terakhir dari dalam list
Pop	alist.pop(i)	Menghapus dan mengembalikan item ke-i dari dalam list
Sort	alist.sort()	Mengubah suatu list agar terurut
Reverse	alist.reverse()	Mengubah suatu list dalam urutan terbalik
Del	del alist[i]	Menghapus item pada posisi ke-i
Index	alist.index(item)	Mengembalikan index dari kemunculan pertama dari item
Count	alist.count(item)	Mengembalikan jumlah kehadiran dari item
Remove	alist.remove(item)	Menghapus kemunculan pertama dari item

Berbagai operasi di atas dapat dilihat percobaannya di bawah ini:

Kegiatan 3

```
>>>myList = [1400, 5, True, 7.3]
>>>myList.append(False)
>>>print(myList)
>>>myList.insert(2,4.5)
>>>print(myList)
>>>print(myList.pop())
>>>print(myList)
>>>print(myList.pop(1))
>>>print(myList)
```

```
>>>myList.pop(2)
>>>print(myList)
>>>myList.sort()
>>>print(myList)
>>>myList.reverse()
>>>print(myList)
>>>print(myList.count(7.3))
>>>print(myList.index(4.5))
>>>myList.remove(6.5)
>>>print(myList)
>>>del myList[0]
>>>print(myList)
```

2. String

String adalah koleksi sequential dari nol atau lebih huruf, bilangan dan simbol-simbol lain. Huruf- huruf, bilangan dan simbol-simbol lain tersebut dinamakan sebagai karakter. Nilai string literal dibedakan dari pengenal (identifiers) dengan menggunakan tanda quotation (single atau double).

Kegiatan 4

Cobalah 5 baris di bawah ini dan perhatikan hasilnya:

```
>>> "Kholisatul"
>>>myName="Kholisatul"
>>>myName[3]
>>>myName*2
>>>len(myName)
```

Kegiatan 5

Berikut ini adalah contoh pemanfaatan metode yang tersedia dalam kelas Strings:

```
>>>myName
>>>myName.upper()
>>>myName.center(10)
>>>myName.find('v')
>>>myName.split('v')
```

Tabel 1.3 Metode bagi Strings di dalam Python

Nama Metode	Cara Penggunaan	Penjelasan
Center	astring.center(w)	Mengembalikan string di tengah dalam field berukuran w
Count	astring.count(item)	Mengembalikan jumlah kehadiran item

		dalam string
Ljust	astring.ljust(w)	Mengembalikan string left-justified dalam field berukuran w
Lower	astring.lower()	Mengembalikan string dalam huruf kecil
Rjust	astring.rjust(w)	Mengembalikan string right-justified dalam field ukuran w
Find	astring.find(item)	Mengembalikan index dari kemuculan pertama dari item
Split	astring.split(schar)	Memecah string menjadi substring-substring pada schar

Perbedaan besar antara lists dan strings adalah bahwa lists dapat dimodifikasi sedangkan string tidak. Ini dikenal sebagai **mutability**. Lists bersifat mutable; strings dikatakan immutable. dapat mengubah item dalam list menggunakan indexing dan assignment, tetapi hal tersebut tidak berlakukan untuk string.

Kegiatan 6

```
>>>myList
>>>myList[0]=2**10
>>>myList
>>>myName
>>>myName[0]='X'
```

3. Tuple

Tuples sangat mirip dengan list dalam hal keberagaman rangkaian data. Perbedaannya adalah tuple bersifat *immutable*, seperti string. Tuples ditulis sebagai nilai-nilai dipisahan koma di dalam kurung. Karena berbentuk sequences maka operasi-operasi di atas dapat diterapkan terhadap tuples.

Kegiatan 7

```
>>>myTuple=(2,True,4.96)
>>>myTuple
>>>len(myTuple)
>>>myTuple[0]
>>>myTuple*3
>>>myTuple[0:2]
```

4. Set

Suatu **set** adalah koleksi tak berurut dari nol atau lebih objek data Python yang *immutable*. Sets tidak membolehkan duplikasi dan nilai-nilai dipisahkankoma di dalam kurung kurawal. Himpunan kosong diwakili oleh set(). Set bersifat heterogen, dan koleksi ini dapat diberikan ke suatu variabel.

Kegiatan 8

```
>>>{3,6,"cat",4.5,False}
>>>mySet={3,6,"cat",4.5,False}
>>>mySet
```

Set tidak bersifat sequential, tetapi masih dapat menggunanakan operasi-operasi di atas. Tabel 4 merangkum operasi-operasi tersebut.

Tabel 1.4 Operasi terhadap Set di dalam Python

Nama Operasi	Operator	Penjelasan
Membership	In	Keanggotaan Himpunan (set)
Length	Len	Mengembalikan kardinalitas dari himpunan
|	aset | otherset	Mengembalikan himpunan baru dengan elemen dari kedua set
&	aset & otherset	Mengembalikan set baru dengan hanya elemen-elemen yang hadir di kedua set
-	aset – otherset	Mengembalikan himpunan baru dengan semua item dari set pertama yang tidak ada di set kedua
<=	aset <= otherset	Menanyakan apakah semua elemen dari set pertama ada dalam set kedua?

Kegiatan 9 (melanjutkan Latihan 8)

```
>>> mySet
>>>len(mySet)
>>>False in mySet
>>>"dog" in mySet
>>>mySet
>>>yourSet={99,3,100}
>>>mySet.union(yourSet)
>>>mySet|yourSet
>>>mySet.intersection(yourSet)
>>>mySet&yourSet
>>>mySet.difference(yourSet)
>>>mySet-yourSet
>>>{3,100}.issubset(yourSet)
>>>{3,100}<=yourSet
>>>mySet.add("house")
>>>mySet
>>>mySet.remove(4.5)
>>>mySet
>>>mySet.pop()
>>>mySet
>>>mySet.clear()
>>>mySet
```

Tabel 1.5 Metode yang disediakan oleh Set di dalam Python

Nama Metode	Cara Penggunaan	Penjelasan
Union	aset.union(otherset)	Mengembalikan himpunan baru dengan semua elemen dari kedua set
intersection	aset.intersection(otherset)	Mengembalikan himpunan baru dengan hanya elemen yang hadir di kedua set
difference	aset.difference(otherset)	Mengembalikan himpunan baru dengan semua item yang hadir dalam set pertama tetapi tidak hadir di dalam set kedua
Issubset	aset.issubset(otherset)	Menanyakan apakah semua elemen dari set tertentu ada dalam set yang lain
Add	aset.add(item)	Menambahkan item ke suatu set
Remove	aset.remove(item)	Menghapus item dari suatu set

Pop	aset.pop()	Menghapus elemen tertentu dari suatu set
Clear	aset.clear()	Menghapus semua elemen dari suatu set

5. Dictionary

Koleksi Python terakhir dari struktur tak-berurut adalah **dictionary**. Dictionaries adalah koleksi pasangan item-item berasosiasi dimana setiappasangan terdiri dari suatu key dan value. Pasangan key-value ini ditulis seabagai key:value. Dictionaries ditulis dipisahkan koma dalam kurung kurawal.

```
>>>kota={'Denpasar':'Bandung','Magelang':'Solo'}
>>>kota
```

Manipulasi terhadap dictionary dilakukan dengan mengakses nilai melalui key-nya atau dengan menambahkan pasangan key-value berikutnya. Mirip dengan mengaksesan sequence, tetapi tidak menggunakan inex, melainkan harus menggunakan key.

Listing 1.1 Program Python yang memanfaatkan dictionary

```
kota={'Bali':'Denpasar','Jateng':'Solo'}
print(kota['Bali'])
kota['Jabar']='Bandung'
print(kota)
kota['Jatim']='Kediri'
print(len(kota))
for k in kota:
    print(kota[k]," berada di ", k)
```

Tabel 1.6 Operator yang disediakan oleh Dictionaries di dalam Python

Operator	Penggunaan	Penjelasan
[]	mydict[k]	Mengembalikan nilai berasosiasi dengan k, jika tidak maka muncul error
in	key in adict	Mengembalikan True jika key ada dalam Dictionary, jika tidak False
del	del adict[key]	Menghapus entry dari Dictionary

Tabel 1.7 Metode yang disediakan Dictionaries di dalam Python

Nama Metode	Penggunaan	Penjelasan
Keys	adict.keys()	Mengembalikan keys dari dictionary dalam objek dict_keys
values	adict.values()	Mengembalikan values dari dictionary dalam objek dict_values
items	adict.items()	Mengembalikan pasangan key-value pairs dalam objek dict_items
Get	adict.get(k)	Mengembalikan value yang berasosiasi dengan k, jika tidak None
Get	adict.get(k, alt)	Mengembalikan nilai yang berasosiasi dengan k, jika tidak alt

Kegiatan 10

```
>>> phoneext={'nurul':1490, 'lisa':1157}
>>>phoneext
>>>phoneext.keys()
>>>list(phoneext.keys())
>>>phoneext.values()
>>>list(phoneext.values())
>>>phoneext.items()
>>>list(phoneext.items())
>>>phoneext.get("ulya")
>>>phoneext.get("ulya","NO ENTRY")
```

BAB 2
CLASS PYTHON: PENDEKATAN BERORIENTASI OBJEK

A. TUJUAN

1. Mahasiswa mampu memahani cara membuat dan menerapkan *class* pada python
2. Mahasiswa mampu menerapkan *class python* dalam menyelesaikan masalah

B. DASAR TEORI

1. Pemrograman Berorientasi Objek (PBO)

Pada program yang selama ini di buat, saat mendesain program biasanya berdasarkan fungsi (blokstatemen yang memanipulasi data). Hal ini disebutpemrograman *procedure-oriented*. Ada cara lain untukmengorganisasi program dengan menggabungkan data dan operasi yang dibungkus dalam suatu objek yaitu paradigm pemrograman berorientasi objek. Semua unit dalam program bisa dianggap sebagai objek. Objek besar dibangun dari objek – objek yang lebih kecil. Objek yang satu berinteraksi dengan objek yang lain, sehingga semua menjadi sebuah kesatuan yang utuh.

Jika dimisalkan sebuah mobil. Mobil memiliki cirri punya ban, stang,kursi, pedal gas, rem, dan lain sebagainya. Ada juga cirri warna, atau tahunkeluaran berapa. Selain punya ciri, mobil juga punya aksi atau sesuatu yang bisa dilakukan olehnya. Misalnya, ketika pedal diinjakapa yang terjadi. Ketika di rem apa yang terjadi, dan lain sebagainya

Objek memiliki field berupa variabel objek dan method berupa fungsi objek. Keduanya disebut atribut objek. Class juga dapat memiliki field class (variabel class) dan method class. Class didefinisikandengan keyword class.

2. Istilah pada PBO

Sebelum mempelajari lebih jauh tentang PBO, ada baiknya mengetahui istilah – istilah dalam PBO, yaitu sebagai berikut:

- **Kelas** – Kelas adalah cetak biru atau prototype dar iobjek dimana mendefinisikan atribut dari suatu objek. Atribut ini terdiri dari data member (variabel) dan fungsi (metode).

- **Variabel Kelas** – Variabel kelas adalah variabel yang di*share* atau dibagi oleh semua instance (turunan) dari kelas. Variabel kelas didefinisikan di dalam kelas, tapi di luar metode-metode yang ada dalam kelas tersebut.
- **Data member** – Data member adalah variabel yang menyimpan data yang berhubungan dengan kelas dan objeknya
- **Overloading Fungsi** – Overloading fungsi adalah fungsi yang memiliki nama yang sama di dalam kelas, tapi dengan jumlah dan tipe argumen yang berbeda sehingga dapat melakukan beberapa hal yang berbeda.
- **Overloading operator** – Overloading operator adalah pembuatan beberapa fungsi atau kegunaan untuk suatu operator. Misalnya operator + dibuat tidak hanya untuk penjumlahan, tapi juga untuk fungsi lain.
- **Variabel instansiasi** – Variabel instansiasi adalah variabel yang didefinisikan di dalam suatu metode dan hanya menjadi milik dari instance kelas.
- **Pewarisan/Inheritansi** – Inheritansi adalah pewarisan karakteristik sebuah kelas kekelas lain yang menjadi turunannya.
- ***Instance*** – *Instance* adalah istilah lain dari objek suatu kelas. Sebuah objek yang dibuat dari prototype kelas Lingkaran misalnya disebut sebagai instance dari kelas tersebut.
- **Instansiasi** – Instansiasi adalah pembuatan instance/objek dari suatu kelas
- **Metode** – Metode adalah fungsi yang didefinisikan di dalam suatu kelas
- **Objek** – Objek adalah instansiasi atau perwujudan dari sebuah kelas. Bila kelas adalah prototipenya, dan objek adalah barang jadinya.

3. Membuat Class

Mendefinisikan sebuah kelas dengan menggunakan kata kunci class diikuti oleh nama kelas tersebut. Berikut adalah sintaks pembuatan kelas di Python.

```
class ClassName:
    '''class docstring'''
class_body
```

Kelas memiliki doc string atau string dokumentasi yang bersifat opsiona lartinya bisa ada atau tidak. Doc string bisa diakses menggunakanformat ClassName.__doc__. class_body terdiri dari semua pernyataan berupa attribut, fungsi, dan data dari kelas

4. Instansiasi Objek

Untuk membuat objek dari sebuah kelas, bisa dilakukan dengan memanggil nama kelas dengan argument sesuai dengan fungsi__init__() pada saat mendefinisikannya.

```
mobil1 = Mobil()
mobil2 = Mobil()
```

5. Mengakses Atribut Objek

Agar bisa mengakses atribut objek dengan menggunakan operator titik. Variabel kelas bisa diakses dengan menggunakan nama kelasnya.

```
mobil.warna
mobil1.maju()
mobil2.mundur()
```

C. LANGKAH PRAKTIKUM

Kegiatan 1: Membuat Class dan Objek

```
class Mobil:
    def __init__(self):  # Perbaikan nama metode dari _init ke __init__
        self.warna = 'Hitam'  # Penulisan atribut warna
        self.tipe = 'Sedan'  # Penulisan atribut tipe (tambahkan tanda sama dengan)
        print('Objek telah dibuat')

    def maju(self):  # Fungsi maju
        print('Mobil maju')

    def mundur(self):  # Fungsi mundur
        print('Mobil mundur')

    def melaju(self, speed):  # Fungsi melaju dengan parameter speed
        print(f'Melaju dengan kecepatan {speed} km/jam')  # Menggunakan f-string untuk format string

# Membuat objek pertama
mobil1 = Mobil()  # Menggunakan tanda sama dengan untuk assignment
print(mobil1.warna)
mobil1.maju()  # Tidak perlu menggunakan print() saat memanggil fungsi yang tidak mengembalikan nilai
mobil1.melaju(100)

# Membuat objek kedua
mobil2 = Mobil()
print(mobil2.tipe)
mobil2.mundur()
mobil2.melaju(150)
```

LATIHAN 1

1. Modifikasi `class Mobil` diatas dan buat 3 objek baru didalamnya!
2. Buatlah suatu class yang dapat merepresentasikan sifat-sifat dari objek Kucing Objek ini memiliki field/variable/properties berupa umur, warna bulu dan method berupa `meong()` dan `umur()`

Kegiatan 2: Method Objek dan Init

```
class Orang:
    # Method
    def __init__(self, nama):
        self.nama = nama

    # Deklarasi method
    def katakanSalam(self):
        print("Assalamu'alaikum,", self.nama)

# Membuat objek dengan nama
org = Orang("Budi")
org.katakanSalam()
```

Pada python terdapat nama-nama method special. `_init_` adalah salah satunya. Method ini akan dijalankan ketika objek dibuat. Method ini berguna untuk melakukan inisialisasi.

LATIHAN 2

1. Buatlah `class PersegiPanjang`. Lalu tambahkan 2 method yakni `luas()` dan `keliling()`

Kegiatan 3: Variabel class dan objek (instance)

Variabel class yaitu variabel yang dimiliki oleh class, sedangkan variabel objek adalah variabel yang dimiliki oleh tiap-tiap objek instance dari class

```
class Orang:
# variabel class, untuk menghitung jumlah orang
    total=0
    def __init__(self,nama):
# inisiasi data, data yang dibuat pada self merupakan variabel obyek
        self.nama = nama

# ketika ada orang yang dibuat, tambahkan total orang
        Orang.total += 1

    def __del__(self):
# kurangi total orang jika objek dihapus
        Orang.total -=1

    def katakanHalo(self):
        print ("Halo, nama saya",self.nama, ", apa kabar?")
```

```
    def total_populasi(cls):
        print ('Total Orang adalah ', cls.total)

# method class
    total_populasi= classmethod(total_populasi)

org=Orang('budi')
org.katakanHalo()
Orang.total_populasi()

org2=Orang('andi')
org2.katakanHalo()
Orang.total_populasi()

print ('objek dihapus')
del org
del org2

Orang.total_populasi()
```

D. TUGAS

1. Buatlah class Hewan dan juga ciptakan object dari class tersebut sehingga bisa membuat berbagai macam object hewan dengan karakternya masing-masing!
   Kode program akan menghasilkan output seperti berikut.

```
Nama Hewan : Harimau
Jumlah Kaki : 4
Makanan : Daging
Type Hewan : Karnivora

Nama Hewan : Kerbau
Jumlah Kaki : 4
Makanan : Rumput
Type Hewan : Karnivora
```

2. Buatlah class berdasarkan class diagram berikut

BAB 3
SEARCHING (TEKNIK PENCARIAN)

A. TUJUAN

1. Mahasiswa mampu memahami dan mengimplementasikan teknik pencarian pada python
2. Mahasiswa mampu membedakan berbagai teknik pencarian

B. DASAR TEORI

1. Linier Search

Linear Search adalah salah satu algoritma untuk mencari data yang berada dalam sebuah data yang tertata. Proses dari *linear search* yaitu mencari data dari salah satu ujungkiri/kanan lalu mencari tiap data secara runtut sampai ketemu data yang dinginkan.

2. Binary Search

Binary Search adalah algoritma pencarian data dengan mengurutkan data terlebih dahulu secara *ascending/descending.* Setelah data diurutkan akan mencari data tengah terlebih dahulu. Apabila data yang dicari kurang dari data tengah, maka pencarian data dimulai dari index ke 0 sampai index sebelum data tengah. Sedangkan apabila data yang dicari lebih dari data tengah,maka pencarian data dimulai dari index setelah data tengah sampai index terakhir. Begitu seterusnya sampai data yang dicari tersebut ditemukan.

C. LANGKAH PRAKTIKUM

Kegiatan 1: Linier Search

```
def  l_search(a,x,l,n):
    if l<n:
        if a[l]==x:
            print("Data di temukan pada posisi",l+1)
        else:l_search(a,x,l+1,n)
    else:
        print("Data tidak ditemukan")
print("Masukkan list data:")
a=[int(b) for b in input().split()]
```

```
x=int(input("Masukkan data yang dicari:"))
n=len(a)
l_search(a,x,0,n)
```

ANALISA PROGRAM

1. Analisa sintaks program diatas dengan menjelaskan maksud isi sintaks perbaris
2. Tampilkan hasil dari program tersebut

Kegiatan 2: Binary Search

```
def b_search(a,x,l,n):
    if l<=n:
        mid=(l+n)//2
        if a[mid]==x:
            print("Data di temukan pada posisi",mid+1)
        else:
            if a[mid]>x:
                b_search(a,x,l,mid-1)
            else:
                b_search(a,x,mid+1,n)
    else:
        print("Data tidak ditemukan")
print("Masukkan list data:")

a=[int(b) for b in input().split()]
list.sort(a)
print("Urutan data adalah",a)
x=int(input("Masukkan data yang dicari"))
n=len(a)
b_search(a,x,0,n)
```

ANALISA PROGRAM

1. Analisa sintaks program diatas dengan menjelaskan maksud isi sintaks perbaris
2. Tampilkan hasil dari program tersebut

D. TUGAS

1. Buatlah program untuk menampilkan index data dari sejumlah list data menggunakan (boleh modifikasi dari program di atas)
   a. linier search
   b. binary search (pengurutan dari yang terbesar)
2. Seseorang mendaftarkan diri sebagai pemilih. Setelah mendaftar dia mendapatkan nomor id. Orang tersebut perlu memeriksa apakah nomor itu ada di data atau tidak. Gunakan teknik pencarian linier search dan binary search untuk mengetahuinya

BAB 4
SORTING (1):
BUBLE SORT, INSERTION SORT, SELECTION SORT

A. TUJUAN

1. Mahasiswa mampu menerapkan algoritma sorting pada python
2. Mahasiswa mampu membandingkan buble sort, insertion sort dan selection sort

B. DASAR TEORI

Sering di temui dalam belajar sebuah algoritma adalah bagaimanamengurutkan sebuah data yang acak, atau sering dikenal dengan istilah sorting. Sorting adalah suatu proses untuk menyusun kembali himpunan obyek menggunakan aturan tertentu. Sorting disebut juga sebagai suatu algoritma untuk meletakkan kumpulan elemen data ke dalam urutan tertentu berdasarkan satu atau beberapa kunci dalam tiap-tiap elemen. Ada dua macam urutan yang biasa digunakan dalam suatu proses sorting yaitu:

a. urut naik (ascending)

Mengurutkan dari data yang mempunyai nilai paling kecil sampai paling besar

b. urut turun (descending)

Mengurutkan dari data yang mempunyai nilai paling besar sampai paling kecil.

Data yang terurut mudah untuk dicari, mudah untuk diperiksa, dan mudah untuk dibetulkan jika terdapat kesalahan. Data yang terurut dengan baik juga mudah untuk dihapus jika sewaktu-waktu data tersebut tidak diperlukan lagi. Selain itu, dengan mengurutkan data maka semakin mudah untuk menyisipkan data atapunmelakukan penggabungan data.

MACAM TEKNIK SORTING:

a. Buble Sort

Merupakan algoritma pengurutan paling tua dengan metode pengurutan paling sederhana. Pengurutan yang dilakukan dengan membandingkan masing- masing *item* dalam suatu list secara berpasangan, menukar *item* jika diperlukan, dan mengulaginya sampai akhir list secara berurutan, sehingga tidak ada lagi *item* yang dapat ditukar.

b. Selection Sort

Ide utama dari algoritma *selection sort* adalah memilih elemen dengan nilai paling rendah dan menukar elemen yang terpilih dengan elemen ke-*i*. Nilai dari *i* dimulai dari 1 ke *n*, dimana *n* adalah jumlah total elemen dikurangi 1.

c. Insertion Sort

Algoritma *insertion sort* pada dasarnya memilah data yang akan diurutkan menjadi dua bagian, yang belum diurutkan dan yang sudah diurutkan. Elemen pertama diambil dari bagian array yang belum diurutkan dan kemudian diletakkansesuai posisinya pada bagian lain dari array yang telah diurutkan. Langkah ini dilakukan secara berulang hingga tidak ada lagi elemen yang tersisa pada bagian array yang belum diurutkan.

C. LANGKAH PRAKTIKUM

Kegiatan 1: BUBBLE SORT

```
def tukar(lists,i,j):
    lists[i],lists[j]=lists[j], lists[i]

def bubble(listku):
    perubahan = True
    #banyaknya sesi yang digunakan untuk mengecek data dari awal
    sesi = len(listku)
    while sesi > 1 and perubahan:
        perubahan = False
        j = 1
        while j < sesi:
            if listku[j]<listku[j-1]:
                perubahan = True
                tukar(listku,j,j-1)
            j+=1
        print(listku)
        #Jika penanda 'perubahan' tidak terjadi maka perulangan berhenti
        #artinya data sudah terurut tanpa melakukan perulangan lagi.
        if not perubahan:
            print("hasil akhir = %s" %str(listku))
        sesi-=1
print("====================================================================")
print("Sebelum Bubble Sort")
mylist=[55,21,11,90,13,76,49,30]
print(mylist)
print("Setelah Bubble Sort")
bubble(mylist)
```

Kegiatan 2: SELECTION SORT

```
def selectionSort(alist):
     for slot in range(len(alist)-1):
       position=slot
       for location in range(len(alist)-1,slot,-1):
           if alist[location]<alist[position]:
               position = location

       temp = alist[slot]
       alist[slot] = alist[position]
       alist[position] = temp
alist = [54,26,93,17,77,31,44,55,20]
selectionSort(alist)
print(alist)
```

Kegiatan 3: INSERTION SORT

```
def insertionSort(myList):
    for index in range(1, len(myList)):
        Kanan = myList[index]
        Kiri = index - 1
        while Kiri >=0 and myList[Kiri] > Kanan:
            myList[Kiri + 1] = myList[Kiri]
            Kiri -= 1
        myList[Kiri + 1] = Kanan

index = 0
panjangList = int(input("Panjang data yang diinginkan = "))
myList=[]
for i in range(1,panjangList+1):
    angka = int(input("Masukan data yang ke %i untuk List = " %i))
    myList.append(angka)

print ("Sebelum di insertion sort =" )
print (myList)
insertionSort(myList)
print ("Setelah di insertion sort =" )
print (myList)
```

LATIHAN

1. Modifikasi sintaks program **bubble sort** agar dapat user dapat melakukan input data secara mandiri
2. Buatlah tampilan pada program **selection sort** agar hasil program menampilkan sebelum dan sesudahnya
3. Modifikasi sintaks program pada **Insertion Sort** agar menampilkan data secara *descending*

D. TUGAS

1. Buatlah sebuah program untuk menampilkan data karyawan diurutkan sesuai gaji yang diterima (gunakan **insertion** dan **selection sort**)
2. Sebuah kelas terdiri dari 20 mahasiswa yang mengikuti 3 mata kuliah yaitu Basis Data, Sistem Operasi dan Struktur Data. Tampilkan 5 mahasiswa terbaik di setiap mata kuliah diurutkan dari nilai menggunakan algoritma **bubble sort**

BAB 5
SORTING (2): QUICK SORT DAN MERGE SORT

A. TUJUAN

1. Mahasiswa mampu menerapkan algoritma sorting pada python
2. Mahasiswa mampu membandingkan quick sort dan merge sort

B. DASAR TEORI

Sebelumnya telah dipelajari teknik *simple* sorting yang terdiri dari bubble sort, selection sort dan insertion sort. Pada modul ini akan dibahas teknik sorting *advance* yaitu quick sort dan merge short.

1. Quick Sort

Quick Sort adalah salah satu algoritma pengurutan data yang paling cepat, yaitu dengan membagi list menggunakan sebuah pivot atau menggunakan teknik pemecahan data menjadi partisi-partisi, sehingga metode ini disebut juga dengan nama *partition exchange sort*. Quick Sort juga menggunakan rekursif dalam algoritmanya.

Dalam algoritma ini dilakukan partisi data yakni membagi data tersebut menjadi dua bagian berdasarkan nilai batas tertentu. Item yang memiliki nilai lebih kecil dari nilai batas menjadi satu kelompok sedangkan item yang memiliki nilai lebih besar dari nilai batas menjadi satu kelompok yang lain. Batas yang menjadi penentu kelompok data tersebut dikenal sebagai ***pivot value***.

Untuk memulai iterasi pengurutan, pertama-tama sebuah elemen dipilih dari data, kemudian elemen-elemen data akan diurutkan diatur sedemikian rupa. Data yang kurang dari pivot sudah ditentukan akan diletakkan disebelah kirinya pivot sedangkan data yang lebih besar dari pivot maka diletakkan disebelah kananpivot.

2. Merge Short

Merge sort merupakan algoritma pengurutan yang dirancang untuk memenuhi kebutuhan pengurutan atas suatu rangkaian data yang tidak memungkinkan untuk ditampung dalam memori komputer karena jumlahnya yang terlalu besar. Algoritma ini ditemukan oleh **John von Neumann** pada tahun 1945. Algoritma dirumuskan dalam 3 langkah berpola **divide-and-conquer**. Berikut langkah kerjanya:

a. **Divide**,

   Memilah elemen – elemen dari rangkaian data menjadi dua bagian.

b. **Conquer**,

   Conquer setiap bagian dengan memanggil prosedur merge sort secara rekursi

c. **Kombinasi,**

   Mengkombinasikan dua bagian tersebut secara rekursif untuk mendapat kan rangkaian data berurutan

Proses rekursi berhenti jika mencapai elemen dasar. Hal ini terjadi bilamana bagian yang akan diurutkan menyisakan tepat satu elemen. Sisa pengurutan satu elemen tersebut menandakan bahwa bagian tersebut telah terurut sesuai rangkaian.

C. LANGKAH PRAKTIKUM

Kegiatan 1: QUICK SORT

```
def q_sort(a, low, high):
    if low < high:
        pivotpos = partition(a, low, high)
        q_sort(a, low, pivotpos - 1)
        q_sort(a, pivotpos + 1, high)

def partition(a, low, high):
    pivotvalue = a[low]
    up = low + 1
    down = high
    done = False
    while not done:
        while up <= down and a[up] <= pivotvalue:
            up += 1
        while a[down] >= pivotvalue and down >= up:
            down -= 1
        if down < up:
            done = True
        else:
            temp = a[up]
            a[up] = a[down]
            a[down] = temp

    temp = a[low]
    a[low] = a[down]
    a[down] = temp
    return down

print("Masukkan data:")
a = [int(x) for x in input().split()]
high = len(a)
q_sort(a, 0, high - 1)
print("Data yang telah diurutkan:", a)
```

Kegiatan 2: MERGE SORT

```
def merge_sort(a):
    if len(a) > 1:
        mid = len(a) // 2
        l_half = a[:mid]
        r_half = a[mid:]

        merge_sort(l_half)
        merge_sort(r_half)

        i = j = k = 0

        while i < len(l_half) and j < len(r_half):
            if l_half[i] < r_half[j]:
                a[k] = l_half[i]
                i += 1
            else:
                a[k] = r_half[j]
                j += 1
            k += 1

        while i < len(l_half):
            a[k] = l_half[i]
            i += 1
            k += 1

        while j < len(r_half):
            a[k] = r_half[j]
            j += 1
            k += 1

print("Masukkan data: ")
a = [int(x) for x in input().split()]
merge_sort(a)
print("Data yang telah diurutkan:", a)
```

LATIHAN

1. Analisa sintaks program diatas dengan menjelaskan maksud isi sintaks perbaris
2. Tampilkan hasil dari program **quick sort** dan **merge sort** tersebut secara descending

D. TUGAS

1. Buatlah program array (cari referensi di internet) berisi 10 data mahasiswa terdiri dari nama, nim, asal.
   - **a.** Lakukan pengurutan berdasarkan NIM menggunakan algoritma **quick sort**
   - **b.** Lakukan pengurutan berdasarkan "asal" menggunakan algoritma **merge sort**

BAB 6
REKURSI

A. TUJUAN

1. Mahasiswa mampu menjelaskan dan menerapkan rekursi pada python
2. Mahasiswa mampu mempelajari formulasi rekursif dalam penyelesaian masalah

B. DASAR TEORI

Rekursi adalah teknik dalam pemrograman dimana suatu method/fungsi memanggil dirinya sendiri. Fungsi ini akan terus berjalan sampai kondisi berhenti terpenuhi, oleh karena itu dalam sebuah fungsi rekursi perlu terdapat 2 blok penting, yaitu blok yang menjadi titik berhenti dari sebuah proses rekursi dan blok yang memanggil dirinya sendiri. Konsep ini dapat digunakan untuk merumuskan solusi sederhana dalam sebuah permasalahan yang sulit untuk diselesaikan secara **iteratif** dengan menggunakan **loop for, while, do-while**. Pada saat tertentu konsep ini dapat digunakan untuk mendefinisikan permasalahan dengan konsisten dan sederhana.

Penggunaan paling umum dari rekursi yaitu dalam ilmu komputer, yang mengacu kepada suatu metode mendefinisikan fungsi dimana fungsi tersebutmemanggil definisinya sendiri. Bilangan Fibonacci adalah contoh klasik dari rekursi:

- Fib(0) adalah 0 [kasus dasar]
- Fib(1) adalah 1 [kasus dasar]

C. LANGKAH PRAKTIKUM

Kegiatan 1: Penjumlahan bentuk iterasi dan rekursi

1. Iterasi

```
def listsum(numList):
    theSum = 0
    for i in numList:
        theSum = theSum + i
    return theSum

print("Masukkan list data:")
numList=[int(x)for x in input().split()]
print("Hasil Penjumlahan:",listsum(numList))
```

2. Rekursi

```
def listsum(numList):
    if len(numList) == 1:
        return numList[0]
    else:
        return numList[0] + listsum(numList[1:])
print("Masukkan list data:")
numList=[int(x)for x in input().split()]
print("Hasil Penjumlahan:",listsum(numList))
```

Kegiatan 3: konversi integer ke string

```
def toStr(n,base):
    convertString = "0123456789ABCDEF"
    if n < base:
        return convertString[n]
    else:
        return toStr(n//base,base) + convertString[n%base]
n = int(input("Masukkan bilangan integer:"))
base = int(input("Masukkan basis bilangan (2 atau 16):"))
print("Hasil konversi bilangan ",n," dengan basis",base,
      "adalah:",toStr(n,base))
```

Kegiatan 4: Pencarian dengan Fibonacci

```
def f_search(a,x,n):
    f0=0
    f1=1
    f2=f0+f1
    while f2<n:
        f0=f1
        f1=f2
        f2=f0+f1
        offset=-1
    while f2>1:
        i = min(offset+f2, n-1)
        if (a[i]<x):
            f2=f1
            f1=f0
            f0=f2-f1
            offset = i
        elif (a[i]>x):
            f2=f0
            f1=f1-f2
            f0=f2-f1
        else :
            return i
    if(f1 and a[offset+1]==x):
        return offset+1
    return -1
```

```
print("Masukkan list:")
a=[int(b) for b in input().split()]
list.sort(a)
print("Data yang telah diurutkan",a)
x=eval(input("Masukkan data yang dicari:"))
n=len(a)
pos = f_search (a,x,n)
if pos >=0:
    print("Data berada pada posisi",pos+1)
else:
    print("Data tidak ditemukan")
```

LATIHAN

1. Pahami dan bandingkan kasus penjumlahan dengan teknik iterasi dan teknik rekursi. Jelaskan langkah perhitungannya!
2. Analisa sintaks program pada **kegiatan 3- 4** diatas dengan menjelaskan maksud isi sintaks perbaris

D. TUGAS

1. Buatlah suatu fungsi rekursif untuk menyelesaikan faktorial dari suatu bilangan.
2. Buatlah suatu fungsi rekursif untuk menyelesaikan deret Fibonacci.

BAB 7
STACK DAN QUEUE

A. TUJUAN

1. Mahasiswa mampu menerapkan operasi dasar stack dan queue
2. Mahasiswa mampu menerapkan struktur data stack dan queue pada program

B. DASAR TEORI

1. STACK (TUMPUKAN)

S**tack/tumpukan** (disebut pula "push-down stack") adalah koleksi berurut dari item-item dimana penambahan item baru dan penghapusan item yang telah ada selalu terjadi di ujung yang sama. Ujung ini dinamakan sebagai "top." Ujung berlawanan dari top dikenal sebagai "base." Stack diurutkan mengikuti konsep LIFO (*last in first out*). Berikut ini adalah beberapa operasi terhadap stack:

- **stack()** membuat suatu stack baru yang kosong. Tidak memerlukan parameter dan mengembalikan suatu stack kosong.
- **push(item)** menambahkan suatu item baru ke atas (top) dari stack. Perlu item dan tidak mengembalikan apapun.
- **pop()** menghapus item teratas dari stack. Tidak perlu parameter dan mengembalikan item. Stack berubah.
- **peek()** mengembalikan top item dari stack tetapi tidak menghapusnya. Tidak memerlukan parameter dan stack tidak berubah.
- **isEmpty()** memeriksa apakah stack dalam keadaan kosong. Tidak memerlukan parameter dan mengembalikan nilai boolean.
- **size()** mengembalikan jumlah item di dalam stack. Tidak memerlukan parameter dan mengembalikan suatu integer.

2. QUEUE (ANTRIAN)

Queue adalah kumpulan data yang berurut dimana penambahan data baru berada di satu ujung bernama ekor atau *rear*. Sedangkan penghapusan data berada di ujung kepala atau *front*. Queue menggunakan metode pengurutan FIFO (First In, First Out) yaitu data yang masuk pertama maka data tersebut juga keluar pertama kali. Berikut ini adalah beberapa operasi Queue:

- **queue()** membuat suatu antrian baru yang kosong. Tidak memerlukan

parameter dan mengembalikan suatu antrian kosong.

- **enqueue(item)** menambahkan suatu item baru ke ujung saru antrian. Perlu item dan tidak mengembalikan sesuatu.
- **dequeue()** menghapus item depan dari antrian. Tidak memerlukan parameter dan mengembalikan itemnya. Antrian termodifikasi.
- **isEmpty()** menguji untuk melihat apakah antrian dalam keadaan kosong. Tidak memerlukan parameter dan mengembalian nilai boolean.
- **size()** mengembalikan jumlah item yang ada di dalam antrian. Tidak memerlukan parameter dan mengembalikan suatu integer.

C. LANGKAH PRAKTIKUM

Kegiatan 1: Operasi Stack

```
top=0
mymax=eval(input("Masukkan jumlah maksimum stack:"))
def createStack():
    stack=[]
    return stack
def isEmpty(stack):
    return len(stack)==0
def Push(stack,item):
    stack.append(item)
    print("--Memasukkan",item,"ke stack--")
def Pop(stack):
    if isEmpty(stack):
        return "**Maaf,stack sudah kosong**"
    return stack.pop()
stack=createStack()
while True:
    print("1.Push")
    print("2.Pop")
    print("3.Tampilkan")
    print("4.Keluar")
    ch=int(input("Masukkan pilihan:"))
    if ch==1:
        if top<mymax:
            item=input("Masukkan stack sembarang:")
            Push(stack,item)
            top+=1
        else:
            print("**Stack sudah Penuh**")
    elif ch==2:
        print(Pop(stack))
    elif ch==3:
        print(stack)
    else:
        print("Keluar..Sampai jumpa lagi...")
        break
```

Kegiatan 2: Operasi Queue

```
front=0
rear=0
mymax=eval(input("Masukkan jumlah maksimum antrian:"))
def createQueue():
    queue=[]
    return queue
def isEmpty(queue):
    return len(queue)==0
def enqueue(queue,item):
    queue.append(item)
    print("--Memasukkan",item,"ke antrian--")
```

```
def dequeue(queue):
    if isEmpty(queue):
        return "**Maaf, Antrian sudah habis**"
    item=queue[0]
    del queue[0]
    return item
queue=createQueue()
while True:
    print("1.Enqueue")
    print("2.Dequeue")
    print("3.Display")
    print("4.Quit")
    ch=int(input("Masukkan pilihan:"))
    if ch==1:
        if rear<mymax:
            item=input("Masukkan antrian sembarang:")
            enqueue(queue,item)
            rear+=1
        else:
            print("**Antrian sudah penuh**")
    elif ch==2:
        print(dequeue(queue))
    elif ch==3:
        print(queue)
    else:
        print("Keluar... Sampai jumpa lagi..")
        break
```

LATIHAN

1. Pahami dan bandingkan operasi pada program stack dan queue diatas. Jelaskan langkah dengan mengisi tabel berikut (tambahkan jumlah tabel jika diperlukan)!

    a. STACK

Operasi Stack	**Isi Stack (list)**	**Nilai Kembalian (return)**

b. QUEUE

Operasi Antrian	**Isi Antrian (list)**	**Nilai Kembalian (return)**

D. TUGAS

1. Jelaskan perbedaan stack dan queue!
2. Buatlah program stack dan queue dengan pendekataan class python (program sebelumnya bisa menjadi referensi anda)
3. Buatlah program antrian untuk permasalahan berikut
   a. Posisi antrian mobil pada suatu tempat pencucian mobil (*car wash*)
   b. Antrian pelanggan pada kasir supermarket (*grocery store*)

BAB 8
SINGLE LINKED LIST

A. TUJUAN

1. Mahasiswa mampu menerapkan operasi dasar singly linked list
2. Mahasiswa mampu menerapkan struktur data linked list

B. DASAR TEORI

Linked List adalah salah satu bentuk struktur data, berisi kumpulan data (node) yang tersusun secara sekuensial, saling sambung-menyambung, dinamis dan tidak terbatas. Pada Linked List, setiap item berada di dalam sebuah node atau objectlink. Setiap objek Link memuat sebuah referensi (biasa disebut next) menuju link setelahnya pada list.

Dalam pengaplikasiannya, secara struktur link, linked list dibagi menjadi 2 macam, yaitu:

1. Singly-Linked List, merupakan list dengan penghubung 1 referensi antar data yang menunjuk data selanjutnya (next).
2. Doubly-Linked List, merupakan list dengan penghubung 2 referensi antar data yang menunjuk data sebelum (prev) dan setelahnya (next).

Dan dari 2 jenis di atas, secara bentuk list yang terbentuk, masing-masing dapat diaplikasikan dengan 2 bentuk, yaitu:

1. Linear: List yang berbentuk linear dengan head/first sebagai penunjuk awal list dan akan diteruskan dengan obyek-obyek yang saling berhubungan, diakhiri dengan tail/ last sebagai penunjuk akhir list, yang mana data terakhir ini akan memiliki referensi null sebagai indikator akhir list.
2. Circular: List yang berbentuk circular dengan head/first sebagai penunjuk awal list dan akan diteruskan dengan obyek-obyek yang saling berhubungan, diakhiri dengan tail/ last sebagai penunjuk akhir list. Bedanya akhir list akan memiliki referensi yang akan menunjuk kembali ke awal list (head/first) dan atau head/first pun akan memiliki referensi yang menunjuk bahwa data sebelumnya adalah tail/lastlist.

C. LANGKAH PRAKTIKUM

Kegiatan 1: Membuat kelas node dan singly linked list

```
# node structure
class Node:
  #constructor to create a new node
  def __init__(self, data):
    self.data = data
    self.next = None

#class Linked List
class LinkedList:
  #constructor to create an empty LinkedList
  def __init__(self):
    self.head = None

  #display the content of the list
  def PrintList(self):
    temp = self.head
    if(temp is None):
      print("Empty List!!")
    while temp:
        print(temp.data, end=" ")
        temp = temp.next
        if temp:
          print ("->", end=" ")
    print()
```

Buatlah **object** untuk dapat menampilkan hasil dari kegiatan 1

Kegiatan 2: Operasi insert

a. Insert data di awal dan akhir

```
def insert_at_start(self, data):
  temp = Node(data)
  temp.next = self.head
  self.head= temp
def insert_at_end(self, data):
  temp = Node(data)
  if self.head is None:
      self.head = temp
      return
  n = self.head
  while n.next is not None:
      n= n.next
  n.next = temp;
def insert_after_data(self, x, data):
  n = self.head
  while n is not None:
      if n.data == x:
          break
      n = n.next
  if n is None:
      print("item not in the list")
  else:
      temp = Node(data)
      temp.next = n.next
      n.next = temp
```

b. Insert data sebelum data tertentu

```
def insert_before_data(self, x, data):
  if self.head is None:
      print("List has no element")
      return
  if x == self.head.data:
      temp = Node(data)
      temp.next = self.head
      self.head = temp
      return
  n = self.head
  while n.next is not None:
      if n.next.data == x:
          break
      n = n.next
  if n.next is None:
      print("item not in the list")
  else:
      temp = Node(data)
      temp.next = n.next
      n.next = temp
```

Kegiatan 3: Operasi delete

a. Delete data di awal

```
def del_front(self):
  if(self.head != None):
    temp = self.head
    self.head = self.head.next
    temp = None
```

b. Delete data di akhir

```
def del_back(self):
  if(self.head != None):
    if(self.head.next == None):
      self.head = None
    else:
      temp = self.head
      while(temp.next.next != None):
        temp = temp.next
      lastNode = temp.next
      temp.next = None
      lastNode = None
```

D. TUGAS

Buatlah program untuk:

1. Menghitung jumlah node pada linked list
2. Mencari data tertentu pada linked list

   Misal hasil yang ditampilkan: **data ditemukan pada index ke 1**
3. Menghapus semua node

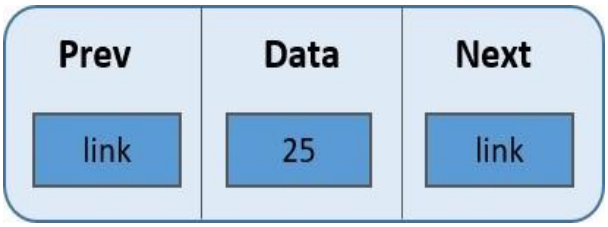
BAB 9
DOUBLE LINKED LIST

A. TUJUAN

1. Mahasiswa mampu menerapkan operasi double linked list
2. Mahasiswa mampu menerapkan doubly linked list pada python

B. DASAR TEORI

Doubly Linked List adalah struktur data linier. Dimana elemen disimpan dalam bentuk node. Setiap node berisi tiga elemen. Bagian data yang menyimpan nilaielemen, bagian sebelumnya yang menyimpan tautan ke node sebelumnya. Dan bagian selanjutnya yang menyimpan tautan ke node berikutnya seperti yang ditunjukkan pada gambar berikut:



Gambar. Doubly linked list

Doubly linked list dapat diaplikasikan dengan 2 bentuk:

1. Linear: List yang berbentuk linear dengan head/first sebagai penunjuk awal list dan akan diteruskan dengan obyek-obyek yang saling berhubungan, diakhiri dengan tail/ last sebagai penunjuk akhir list, yang mana data terakhir ini akan memiliki referensi null sebagai indikator akhir list.
2. Circular: List yang berbentuk circular dengan head/first sebagai penunjuk awal list dan akan diteruskan dengan obyek-obyek yang saling berhubungan, diakhiri dengan tail/ last sebagai penunjuk akhir list. Bedanya akhir list akan memiliki referensi yang akan menunjuk kembali ke awal list (head/first) dan atau head/first pun akan memiliki referensi yang menunjuk bahwa data sebelumnya adalah tail/lastlist.

C. LANGKAH PRAKTIKUM

Kegiatan 1: Membuat kelas node dan singly linked list

```
class Node:
  def __init__(self, data):
    self.data = data
    self.next = None
    self.prev = None

class LinkedList:
  def __init__(self):
    self.head = None

  def PrintList(self):
    temp = self.head
    if(temp is None):
      print("List Kosong!!")
    while temp:
        print(temp.data, end=" ")
        temp = temp.next
        if temp:
          print ("<->", end=" ")
    print()
```

Buatlah **object** untuk dapat menampilkan hasil dari kegiatan 1

Kegiatan 2: Operasi Insert

a. Insert data di awal

```
def insert_first(self, data):
  new = Node(data)
  if(self.head == None):
      self.head = new
      return
  else:
      self.head.prev = new
      new.next = self.head
      self.head = new
```

b. Insert data di awal

```
def insert_end(self, data):
  new = Node(data)
  if self.head is None:
      self.head = new
      return
  else:
    n = self.head
    while n.next is not None:
      n= n.next
    n.next = new
    new.prev = n
```

Kegiatan 3: Operasi Delete

a. Delete data di awal

```
def del_first(self):
  if(self.head != None):
    temp = self.head
    self.head = self.head.next
    temp = None
    if(self.head != None):
      self.head.prev = None
```

b. Delete data di akhir

Gunakan sintax pada operasi delete di akhir pada singly linked list

```
------------Doubly Linked List-------------
1.Insert first
2.Del first
3.PrintList
4.Exit
Enter choice:
```

D. TUGAS

1. Modifikasi program menggunakan percabangan agar menampilkan program seperti berikut

```
------------Doubly Linked List-------------
1.Insert first
2.Del first
3.PrintList
4.Exit
Enter choice:
```

2. Buatlah program untuk memanajemen playlist pada aplikasi media player seperti: (Bisa gunakan referensi program pada modul ini)
   a. Memasukkan list pada playlist
   b. Menghapus List
   c. Daftar list yang di mainkan

BAB 10
BINARY TREE

A. TUJUAN

1. Mahasiswa mampu menerapkan tree
2. Mahasiswa mampu menerapkan tree pada python

B. DASAR TEORI

Tree

Tree adalah struktur data non-linier yang memiliki hubungan hierarkis antara elemen-elemennya. Ini terdiri dari node, dihubungkan oleh tepi, yang menyimpan item data. Susunan elemen sedemikian rupa sehingga pembentukan siklus dihindari.

Kombinasi node memberi pohon struktur data hierarkisnya. Setiap node dalam pohon memiliki dua komponen: data dan link ke node yang berdekatan. Sebuah node dapat memiliki parent node, child node, atau keduanya.

Binary tree

Binary Trees adalah struktur data hierarkis di mana setiap node memiliki paling banyak dua anak, sering disebut sebagai anak kiri dan kanan.

Ada 2 teknik yang dikenal luas untuk melintasi pohon yaitu

1. Depth First Traversal

   Depth First search atau traversal adalah teknik di mana masuk sedalam mungkin ke dalam tree dan menjelajahi node, yaitu memindahkan tree dari satu node ke node lain. Begitu mencapai node yang tidak bisa turun lebih jauh, maka selanjutnya mundur dan pindah ke node satu sebelumnya dan seterusnya.

   Ini memiliki 3 teknik:

   - **Inorder (Left, Root, Right)**

     Traversal berurutan berarti mengunjungi cabang kiri, lalu simpul saat ini, dan akhirnya, cabang kanan.

   - **Preorder (Root, Left, Right)**

     Dalam traversal pre-order, root selalu menjadi node pertama yang dikunjungi.

- **Postorder (Left, Right, Root)**

  Traversal post-order berarti mengunjungi cabang kiri, lalu simpul saat ini, dan akhirnya, cabang kanan. Dalam traversal post-order, root selalu merupakan node terakhir yang dikunjungi.

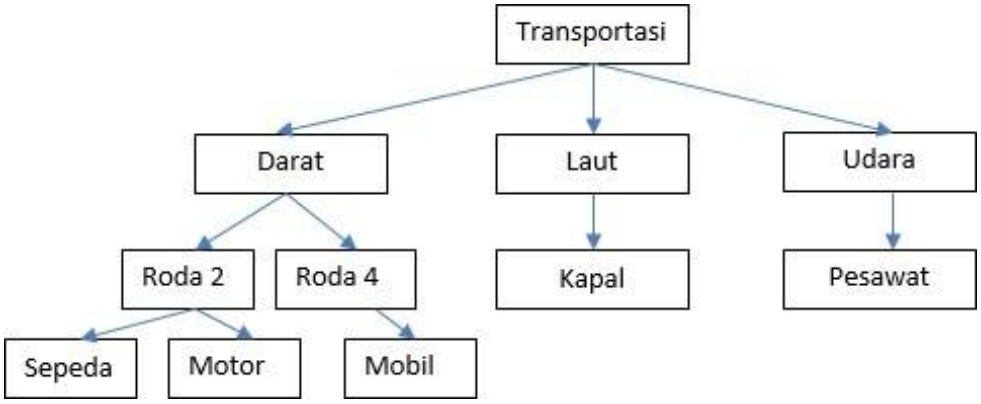
2. Breadth-First Traversal

   algoritma traversal untuk mencetak semua node dalam sebuah graf. Dalam algoritma ini, dimulai dengan sebuah node dan mencetak nilainya. Kemudian mencetak semua neighbor dari node saat ini. Setelah itu, memilih setiap neighbor dari node saat ini dan mencetak semua neighbornya. Proses ini berlanjut sampai semua node pada tree tercetak.tekniknya disebut Level Order Traversal

C. LANGKAH PRAKTIKUM

Kegiatan 1: Membuat Tree

Perhatikan Skema Tree berikut:



```
class TreeNode:
    def __init__(self, data, children=[]):
        self.data = data
        self.children = children

    def __str__(self, level=0):
        ret = "  " * level + str(self.data) + "\n"
        for child in self.children:
            ret += child.__str__(level + 1)
        return ret

    def addChild(self, TreeNode):
        self.children.append(TreeNode)

tree = TreeNode('Transportasi', [])
darat = TreeNode('Darat', [])
laut = TreeNode('Laut', [])
udara = TreeNode('Udara',[])
tree.addChild(darat)
tree.addChild(laut)
tree.addChild(udara)
dua = TreeNode('Roda 2', [])
empat = TreeNode('Roda 4', [])
darat.addChild(dua)
darat.addChild(empat)
print(tree)
```

Lengkapi agar hasil sesuai dengan skema di atas

Kegiatan 2: Depth First Traversal

a. Binary Tree

```
class TreeNode:
    def __init__(self, data):
        self.data = data
        self.leftChild = None
        self.rightChild = None
```

b. Preorder, Inorder, Postorder

```
def preOrderTraversal(root_node):
    if not root_node:
        return
    print(root_node.data)
    preOrderTraversal(root_node.leftChild)
    preOrderTraversal(root_node.rightChild)
```

```
def inOrderTraversal(root_node):
    if not root_node:
        return
    inOrderTraversal(root_node.leftChild)
    print(root_node.data)
    inOrderTraversal(root_node.rightChild)

def postOrderTraversal(root_node):
    if not root_node:
        return
    postOrderTraversal(root_node.leftChild)
    postOrderTraversal(root_node.rightChild)
    print(root_node.data)
```

Masukkan data pada skema tree, kemudian panggil fungsi preorder, inorder dan postorder. Berikutnya tampilkan hasilnya!

D. TUGAS

1. Buatlah program untuk memasukkan data berikut pada binary tree: {8,13,12,4,3,28,40}
2. Berikutnya tampilkan hasil pada program no 1 menggunakan teknik preorder, inorder dan postorder (ref: buka web tutorialspoint.com)

BAB 11
GRAPH

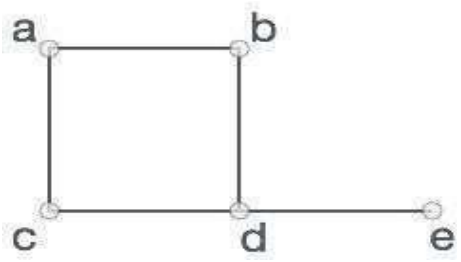
A. TUJUAN

1. Mahasiswa mampu menerapkan graph
2. Mahasiswa mampu menerapkan graph pada python

B. DASAR TEORI

Graf adalah representasi bergambar dari sekumpulan objek di mana beberapa pasang objek dihubungkan oleh tautan. Objek yang saling berhubungan diwakili oleh titik yang disebut sebagai simpul(node), dan tautan yang menghubungkan simpuldisebut tepi(vertex). Graf dapat dengan mudah disajikan menggunakan tipe data dictionary python. simpul sebagai kunci dictionary dan hubungan antara simpul juga disebut tepi sebagai nilai dalam dictionary. Selain itu dapat juga di sajikan dalam bentuk list.

Perhatikan grafik berikut



Pada graf di atas,

V = {a, b, c, d, e}

E = {ab, ac, bd, cd, de}

Sehingga bisa dituliskan dalam syntax seperti berikut:

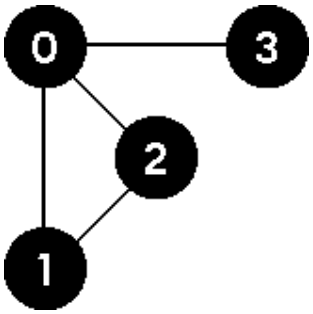
```
# Create the dictionary with graph elements
graph = {

   "a" : ["b","c"],

   "b" : ["a", "d"],

   "c" : ["a", "d"],

   "d" : ["e"],
```

Output:

{'c': ['a', 'd'], 'a': ['b', 'c'], 'e': ['d'], 'd': ['e'],'b': ['a', 'd']}

C. LANGKAH PRAKTIKUM

Perhatikan graf berikut:



Berapa jumlah vertex (tepi)? Berapa jumlah edge (simpul)?

```
vertices = {"0", "1", "2", "3"}
edges = {("0", "1"), ("0", "2"), ("0", "3"), ("2", "1")}
graph = dict()
for vertex in vertices:
    graph[vertex] = []
for edge in edges:
    v1 = edge[0]
    v2 = edge[1]
    graph[v1].append(v2)
    graph[v2].append(v1)
print("Himpunan Vertex:", vertices)
print("Himpunan Node:", edges)
print("Representasi graph:")
print(graph)
```

Kegiatan 1: Graph Sederhana tanpa class

```
class Graph:
    def __init__(self, num):
        self.V = num
        self.graph = [None] * self.V

    # Add edges
    def add_edge(self, s, d):
        node = AdjNode(d)
        node.next = self.graph[s]
        self.graph[s] = node

        node = AdjNode(s)
        node.next = self.graph[d]
        self.graph[d] = node

    # Print the graph
    def print_agraph(self):
        for i in range(self.V):
            print("Vertex " + str(i) + ":", end="")
            temp = self.graph[i]
            while temp:
                print(" -> {}".format(temp.vertex), end="")
                temp = temp.next
            print(" \n")
```

Kegiatan 2: Representasi graph dengan list

Implementasikan graph pada kegiatan 1, lalu representasikan graph dalam bentuk list.

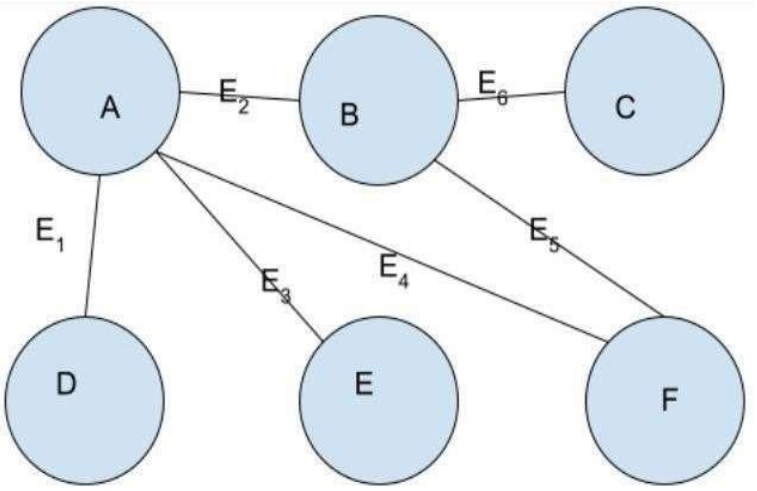
Kegiatan 3: Representasi graph dengan matriks

```
class Graph(object):

    # Initialize the matrix
    def __init__(self, size):
        self.adjMatrix = []
        for i in range(size):
            self.adjMatrix.append([0 for i in range(size)])
        self.size = size

    # Add edges
    def add_edge(self, v1, v2):
        if v1 == v2:
            print("Same vertex %d and %d" % (v1, v2))
        self.adjMatrix[v1][v2] = 1
        self.adjMatrix[v2][v1] = 1

    # Remove edges
    def remove_edge(self, v1, v2):
        if self.adjMatrix[v1][v2] == 0:
            print("No edge between %d and %d" % (v1, v2))
            return
        self.adjMatrix[v1][v2] = 0
        self.adjMatrix[v2][v1] = 0

    def __len__(self):
        return self.size

    # Print the matrix
    def print_matrix(self):
            print('\n'.join([str(lst) for lst in self.adjMatrix]))
```

Implementasikan graph pada kegiatan 1, lalu representasikan graph dalam bentuk matriks.

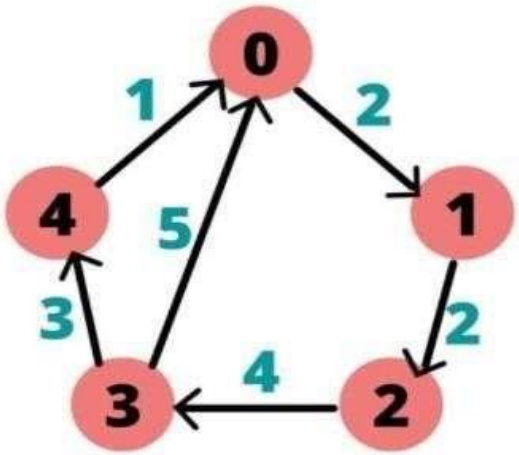
D. TUGAS

1. Representasikan graf berikut dalam bentuk list dan matriks



2. Modifikasi sintax berikut untuk menerapkan weighted graph berikut, agar menampilkan

```
0  --->  [[1, 2]]
1  --->  [[2, 2]]
2  --->  [[3, 4]]
3  --->  [[0, 5], [4, 3]]
4  --->  [[0, 1]]
```



```
adj_list = {}
mylist = []
def add_node(node):
  if node not in mylist:
    mylist.append(node)
  else:
    print("Node ",node," already exists!")

def add_edge(node1, node2):
  temp = []
  if node1 in mylist and node2 in mylist:
    if node1 not in adj_list:
      temp.append(node2)
      adj_list[node1] = temp

    elif node1 in adj_list:
      temp.extend(adj_list[node1])
      temp.append(node2)
      adj_list[node1] = temp

  else:
    print("Nodes don't exist!")

def graph():
  for node in adj_list:
    print(node, " ---> ", [i for i in adj_list[node]])
```

Kemudian terapkan pada graf berikut dan tampilkan hasilnya!

BAB 12 HASHING

A. TUJUAN

1. Mahasiswa mampu menerapkan hashing sederhana
2. Mahasiswa mampu menerapkan hashing pada python

B. DASAR TEORI Hashing

Hashing merupakan teknik yang digunakan untuk menyusun dan mengakses elemen data dalam List dengan waktu yang relatif konstan melalui manipulasi key untuk mengidentifikasilokasi dalam List.

Hash Table

Hash table adalah sebuah struktur data yang terdiri atas sebuah tabel dan fungsi yang bertujuan untuk memetakan nilai kunci yang unik untuk setiap record menjadi angka (hash) lokasi record tersebut dalam sebuah tabel.

Hash table menggunakan memori penyimpanan utama berbentuk array dengan tambahan algoritma untuk mempercepat pemrosesan data. Pada intinya hash table merupakanpenyimpanan data menggunakan key value yang didapat dari nilai data itu sendiri. Dengan key value tersebut didapat hash value. Jadi hash function merupakan suatu fungsi sederhana untuk mendapatkan hash value dari key value suatu data. Yang perlu diperhatikan untuk membuat hash function adalah:

- ukuran array/table size(m),
- key value/nilai yang didapat dari data(k),
- hash value/hash index/indeks yang dituju(h).

Berikut contoh penggunaan hash table dengan hash function sederhana yaitu memodulus keyvalue dengan ukuran array : h = k (mod m)

Misal kita memiliki array dengan ukuran 13, maka hash function : h = **k (mod 13)**

Tabel Contoh Hash

k	H
7	7
13	0
25	12
27	1
39	0

Range dari h untuk sembarang nilai k.Maka data 7 akan disimpan pada index 7, data 13 akan disimpan pada index 0, dst..Untuk mencari kembali suatu data, maka kita hanya perlu menggunakan hash function yang sama sehingga mendapatkan hash indexyang sama pula.Misal : mencari data 25 → h = 25 (mod 13) = 12

Fungsi Hash

Hash function merupakan fungsi yang digunakan untuk memanipulasi key dari elemen data dalam List untuk mengidentifikasi lokasi aslinya di list. Fungsi ini akan memetakan list data yang ukurannya berubah-ubah ke ukuran tetap. Nilai kembalian dari fungsi hash disebut dengan Hash Values.

Ini berkaitan dengan menghasilkan slot atau indeks ke nilai "kunci" apa pun. Fungsi hashing sempurna atau hash sempurna adalah fungsi yang memberikan slot unik untuk setiap nilai kunci. Terkadang, ada kasus di mana fungsi hash menghasilkan indeks yang sama untuk beberapa nilai kunci. Ukuran tabel hash dapat ditingkatkan untuk meningkatkan kesempurnaan fungsi hash.

Collision/Tabrakan

Tabrakan terjadi ketika dua item/nilai mendapatkan slot/indeks yang sama, yaitu fungsi hashing menghasilkan nomor slot yang sama untuk beberapa item. Jika langkah-langkah resolusi tabrakan yang tepat tidak diambil maka item sebelumnya di slot akan diganti dengan item baru setiap kali tabrakan terjadi.

Chaining

Salah satu cara untuk mengatasi tabrakan adalah Chaining. Ini memungkinkan beberapa item ada di slot/indeks yang sama. Ini dapat membuat rantai/kumpulan item

dalam satu slot. Saat tabrakan terjadi, item disimpan di slot yang sama menggunakan mekanisme chaining.

C. LANGKAH PRAKTIKUM

Kegiatan 1: Review tipe data dictionary

Buatlah data kota yang ada dalam karesidenan surakarta menggunakan dictionary sehingga tampil data seperti berikut

```
10 -> Solo
15 -> Sukoharjo
20 -> Wonogiri
25 -> Sragen
30 -> Karanganyar
35 -> Boyolali
```

Kegiatan 2: Review tipe data list

Modifikasi data pada kegiatan 1 dengan menyimpannya dalam bentuk list. Kemudian tambahkan syntax berikut:

```
def insert(item_list, key, value):
        item_list.append((key, value))

def search(item_list, key):
        for item in item_list:
                if item[0] == key:
                        return item[1]
```

Lakukan kedua fungsi operasi tersebut, kemudian tampilkan hasilnya

Kegiatan 3: Tabel hash, Fungsi Hash

Tuliskan sintax berikut!

```
hash_table = ["Hello"]*10
print (hash_table)

def hash_func(key):
    return key % len(hash_table)
```

a. Berapakah jumlah tabel yang dibuat?

b. Panggil fungsi hash_func, isi dengan angka 10, 20 dan 25. Apa hasilnya?Jelaskan!

Kegiatan 4: Operasi insert pada tabel hash

```
def insert(hash_table, key, value):
        hash_key = hash_func(key)
        hash_table[hash_key] = value
```

Lanjutkan kegiatan 3 dengan menambahkan syntax berikut:

a. Lakukan insert data pada tabel (sesuai parameter pada fungsi) yaitu Solo, Wonogiri, Sragen

b. Tampilkan hasil pada tabel hash yang telah diisi

c. Apakah data pada tabel terisi sesuai inputan?

Kegiatan 5: Penanganan Collision dengan chaining (rantai)

```
hash_table = [[] for _ in range(10)]
print (hash_table)

def hashing_func(key):
        return key % len(hash_table)

print (hashing_func(10))
print (hashing_func(20))
print (hashing_func(25))
```

```
def insert(hash_table, key, value):
        hash_key = hashing_func(key)
        hash_table[hash_key].append(value)

insert(hash_table, 10, 'Solo')
insert(hash_table, 25, 'Sragen')
insert(hash_table, 20, 'Wonogiri')
print (hash_table)
```

Kegiatan 6: Tabel hash nested list (list bersarang)

a. Membuat Tabel hash

```
hash_table = [[] for _ in range(10)]
print (hash_table)
```

b. Operasi insert

```
def insert(hash_table, key, value):
        hash_key = hash(key) % len(hash_table)
        key_exists = False
        bucket = hash_table[hash_key]
        for i, kv in enumerate(bucket):
                k, v = kv
                if key == k:
                        key_exists = True
                        break
        if key_exists:
                bucket[i] = ((key, value))
        else:
                bucket.append((key, value))
```

c. Operasi search

```
def search(hash_table, key):
        hash_key = hash(key) % len(hash_table)
        bucket = hash_table[hash_key]
        for i, kv in enumerate(bucket):
                k, v = kv
                if key == k:
                        return v
```

```
def delete(hash_table, key):
        hash_key = hash(key) % len(hash_table)
        key_exists = False
        bucket = hash_table[hash_key]
        for i, kv in enumerate(bucket):
                k, v = kv
                if key == k:
                        key_exists = True
                        break
        if key_exists:
                del bucket[i]
                print ('Key {} deleted'.format(key))
        else:
                print ('Key {} not found'.format(key))
```

d. Operasi delete

Lakukan ketiga operasi tersebut dan tampilkan hasilnya! Dimana saja letak posisi data setelah dilakukan operasi insert?

D. TUGAS

1. Buatlah program yang berisi data berikut menggunakan tabel hash
2. Dalam sebuah perpustakaan memiliki 200 buku dan rak buku berjumlah 5. Carilah nomor rak dari nomor buku pada tabel daftar buku berikut: (Gunakan teknik nesteddictionary pada penyimpanan data)

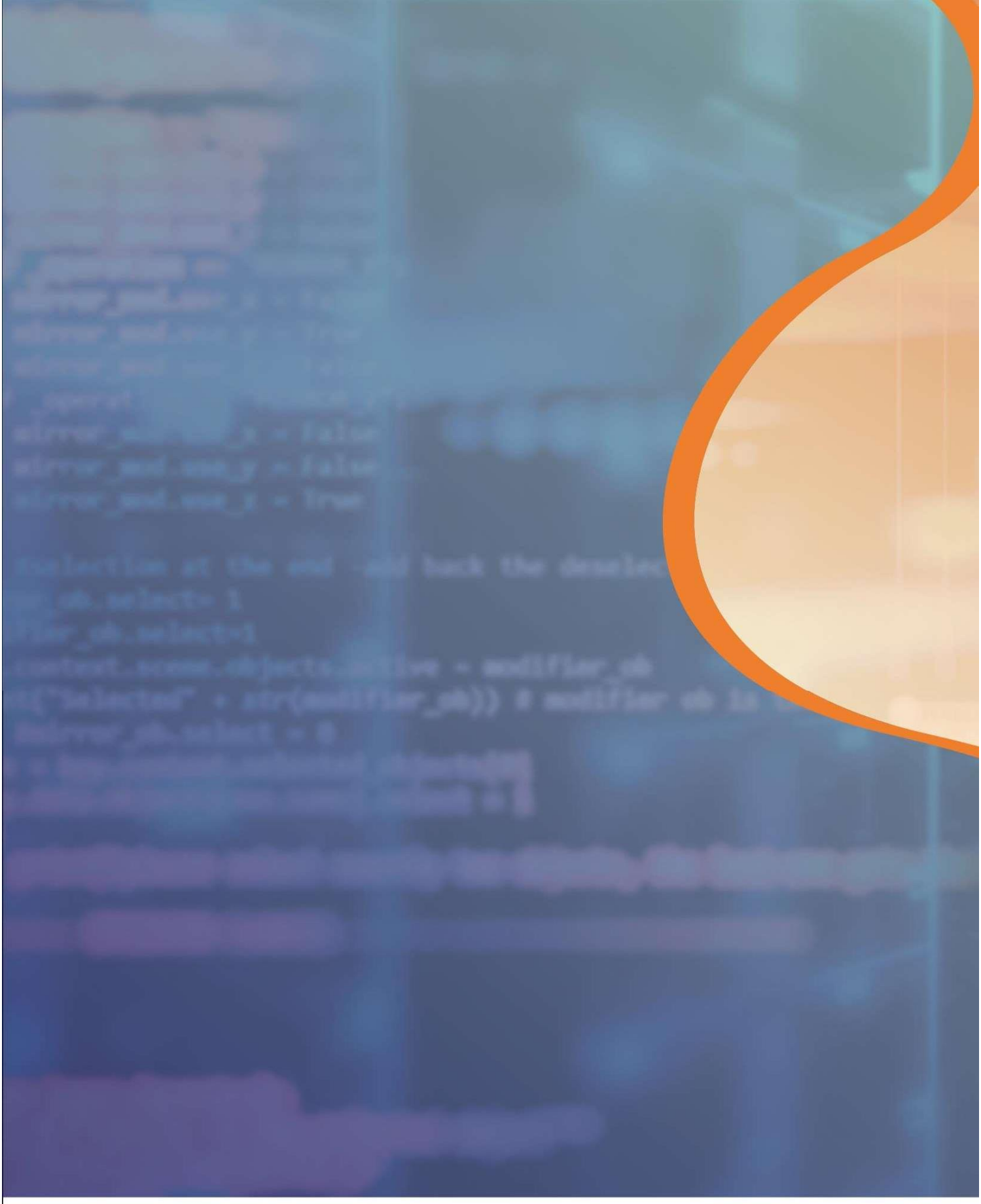
Kode	Judul	Pengarang	Penerbit	Nomor Buku
A	Pemrograman Dasar	Abdul Kadir	Andi	35
B	Belajar Python	Budi Raharjo	Graha Ilmu	20
C	Struktur Data	Chomaria	Informatika	64
D	Internet	Deni Santoso	Biobses	131
E	Database	Eko Cahyanto	Cemerlang	78

DAFTAR PUSTAKA

D. Malhotra and N. Malhotra (2021) Data Structures and Program Design using Python. David Pallai.

Dr. Basant Agarwal, Benjamin Baka (2018). Hands on Data Structures and algorithms with python. Packt.

Lambert, Kenneth (2019). Fundamentals of Python: Data Structures. Cengage Learning.



AE MEDIA GRAFIKA
aemediagrafika@gmail.com
www.aemediagrafika.com

itspku
Institut Teknologi Sains dan Kesehatan
PKU Muhammadiyah Surakarta

ISBN: 978-623-5516-55-4
9 786235 516554