Lab 1: Python: List, String, Tuple, Set, Dictionary dan Class

1.0 Obyektif

- Mereview ide-ide informatika, pemrograman dan penyelesaian masalah
- Memahami abstraksi dan peranannya dalam proses penyelesaian masalah
- Memahami dan mengimplementasikan *notion* dari tipe data abstrak
- Mereview bahasa pemrograman python.

1.1 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 obyek data Python. List ditulis sebagai nilai-nilai terpisahkan koma di dalam kurung siku. List kosong diwakili oleh [ ]. List bersifat heterogen, artinya obyek-obyek data di dalamnya tidak harus berasal dari kelas yang sama dan koleksi tersebut dapat berikan ke suatu variabel.

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

```
>>>[1,3,True,6.5]
>>>myList = [1,3,True,6.5]
>>>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

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:

```
>>>myList = [1024, 3, True, 6.5]
>>>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(6.5))
>>>print(myList.index(4.5))
>>>myList.remove(6.5)
>>>print(myList)
>>>del myList[0]
>>>print(myList)
```

1.2 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).
Cobalah 5 baris di bawah ini dan perhatikan hasilnya:

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

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. Kita dapat mengubah item dalam list menggunakan indexing dan assignment, tetapi hal tersebut tidak berlakukan untuk string.

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

1.3 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.

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

1.4 Set

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

```
>>>{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 semua 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?

```
>>>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

1.5 Dictionary

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

```
>>>capitals={'Iowa':'DesMoines','Wisconsin':'Madison'}
>>>capitals
```

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

```
capitals =
{'Iowa':'DesMoines','Wisconsin':'Madison'}
print(capitals['Iowa'])

capitals['Utah']='SaltLakeCi
ty' print(capitals)

capitals['California']='Sacramen
to' print(len(capitals))

for k in capitals:
  print(capitals[k]," is the capital of ", 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 obyek dict_keys
values	adict.values()	Mengembalikan values dari dictionary dalam obyek dict_values
items	adict.items()	Mengembalikan pasangan key-value pairs dalam obyek 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

```
>>>phoneext={'saalimah':1410, 'hidayah':1137}
>>>phoneext
>>>phoneext.keys()
>>>list(phoneext.keys())

>>>phoneext.values()
```

```
>>>list(phoneext.values())
>>>phoneext.items()
```

```
>>>list(phoneext.items())

>>>phoneext.get("kent")
>>>phoneext.get("kent","NO ENTRY")
```

Listing 1.2 Fungsi Menghitung Akar Kuadrat. Tambahkan exception handling agar root atau n dapat menerima nilai nol.

```
def squareroot(n):
        root = n / 2            #tebakan awal akan berupa 1/2 of n
        for k in range(20):
                root = (1 / 2) * (root + (n / root))
        return root

>>>squareroot(9)
>>>squareroot(4563)
```

1.6 Struktur Data Kelas: Mengenal Pendekatan Berorientasi Obyek

Listing 1.3 Class Fraction. Metode-metode aritmatika dan relasional ditinggalkan sebagai latihan

```
def gcd(m,n):
  while m%n != 0:
    oldm =
    m oldn =
    n

    m = oldn
    n =
  oldm%oldn
  return n

class Fraction:
   def_init_(self,top,bottom):
     self.num = top
     self.den =
     bottom

   def_str_(self):
     return str(self.num)+"/"+str(self.den)

   def show(self):
     print(self.num,"/",self.de
     n)

   def_add_(self,otherfraction):
     newnum = self.num*otherfraction.den
             + \ self.den*otherfraction.num
     newden = self.den *
     otherfraction.den common =
     gcd(newnum,newden)
     return
   Fraction(newnum//common,newden//common) def
```

eq_(self, other):

```
        firstnum = self.num * other.den
        secondnum = other.num *
        self.den

        return firstnum ==

secondnum x = Fraction(1,2)
y = Fraction(2,3)
print(x+y)
print(x ==
y)
```

1.8 Rangkuman

- Informatika adalah kajian mengenai penyelesaian masalah berbasis komputer.
- Informatika menggunakan abstraksi sebagai perangkat untuk merepresentasikan proses dan data.
- Tipe data abstrak memungkinkan pemrogram mengelola kompleksitas dari suatu domain masalah dengan menyembunyikan detail dari datanya.
- Python adalah bahasa yang powerful, mudah digunakan dan object-oriented.
- Lists, tuples dan strings adalah koleksi sequential Python yang sudah built-in.
- Dictionaries dan sets adalah koleksi data non-sequential.
- Classes memungkinkan programmer mengimplementasikan tipe data abstrak.
- Programmer dapat meng-override metode-metode standard selain dapat membuat metode baru sendiri.
- Classes dapat diorganisasikan dalam bentuk hirarki.
- Konstruktor dari class akan selalu mengeksekusi konstruktor dari induknya sebelum menjalankan proses di dalam dirinya sendiri.

1.9 Tugas Pendahuluan

1. Implementasikan metode sederhana getNum dan getDen yang akan mengembalikan pembilang (numerator) dan penyebut (denominator) dari suatu pecahan (fraction).

1.10 Tugas Lanjutan

2. Implementasikan operator aritmatika sederhana (__sub__, __mul__, dan __truediv__).
3. Implementasikan operator relasional yang diperlukan: (__gt__ge__, __lt__, __le__, , dan ne)
4. Modifikasi konstruktor untuk kelas fraction sehingga mampu memeriksa untuk memastikan bahwa pembilang dan penyebut keduanya bertipe integers. Jika salah satunya bukan integer, maka konstruktor akan memunculkan suatu exception.

1.11 Latihan Mandiri

5. Dalam definisi fraksi, kita menganggap bahwa pecahan negatif mempunyai pembilang negatif dan penyebut positif. Menggunakan suatu penyebut negatif dapat mengakibatkan beberapa operator relasi memberikan hasil tidak benar.

Secara

umum, ini merupakan konstrain yang tidak penting. Lakukan perubahan terhadap konstruktor agar memungkinkan pengguna melewatkan suatu penyebut negatif sehingga semua operator tetap berlanjut bekerja dengan benar.

6. Telitilah metode _radd_ . Bagaimana perbedaannya dengan _add_ ? Kapan digunakan? Implementasikan metode radd .
7. Ulangi pertanyaan terakhir tetapi kali ini untuk metode iadd .
8. Telitilah metode __repr__. Apa bedanya dengan __str_? Kapan digunakan? Implementasikan _repr_.
9. Rancanglah suatu kelas untuk merepresentasikan suatu permainan kartu. Rancanglah kelas unutk mereprentasikan kumpulan kartu. Menggunakan dua kelas ini, implementasikan suatu game kartu favorit anda.
10. Temukan permainan sudoku di majalah atau situs web. Tulislah program untuk menyelesaikan masalah puzzle tersebut.