Lab 5
Keranjang Boba

Materi List

List adalah sebuah struktur data *built-in* di bahasa pemrograman Python. List direpresentasikan sebagai sebuah barisan terurut yang mengandung elemen-elemen di dalamnya. Ibaratkan list adalah sebuah rentetan loker yang di setiap lokernya terdapat nomor-nomor, dan masing-masing loker dapat menyimpan sesuatu barang.

List juga merupakan sebuah objek yang mutable dan iterable. List berupa objek mutable artinya isi / kandungannya dapat berubah. List berupa objek iterable artinya kita bisa melakukan iterasi / looping pada setiap elemen-elemen di dalam list.

Deklarasi List

Kita bisa membuat sebuah list dengan mendeklarasikan langsung seperti di bawah ini:

```
lst1 = []
lst2 = [1, 2, 3]
lst3 = [1, "Hello", [1, 2, 3]]

print(lst1) #output: []
print(lst2) #output: [1, 2, 3]
print(lst3) #output: [1, "Hello", [1, 2, 3]]
```

Atau, kita bisa membuat list dengan menggunakan *constructor* dan memasukkan sebuah object iterable ke dalam parameternya

```
lst1 = list()
lst2 = list("abcde")
lst3 = list((1, 2, 3))

print(lst1) #output: []
print(lst2) #output: ["a", "b", "c", "d", "e"]
print(lst3) #output: [1, 2, 3]
```

Perhatikan bahwa ternyata sebuah list bisa mengandung objek-objek dengan tipe data yang berbeda, bahkan sebuah list juga bisa mengandung list lain di dalamnya! :O

Selain itu, kita bisa memecah sebuah string menjadi sebuah list dengan method .split()

```
my_str = "Selamat pagi semua"
my_lst = my_str.split()

print(my_lst)
#Output: ["Selamat", "pagi", "semua"]
```

Method tersebut akan memecah string dengan delimiter whitespace (`" "`).

Mengakses Data Pada List

Data pada list dapat kita akses melalui indeksnya. Perhatikan bahwa indeks dari list dimulai dari 0.

```
my_lst = [1, 2, 3, 4, 5]
print(my_lst[0]) # Output: 1
print(my_lst[3]) # Output: 4
print(my_lst[-1]) # Output: 5
print(my_lst[len(my_lst)]) #IndexError: list index out of range
```

Perhatikan apabila kita mencoba mengakses list melalui indeksnya, namun indeks yang kita masukkan berada di luar batas list yang bersangkutan, maka python akan melempar IndexError

Mengubah Data Pada List

Kita bisa melakukan pengubahan data pada list melalui indeksnya seperti di bawah ini

```
my_lst = [1, 2, [1, 2, 3]]

my_lst[0] = 100
print(my_lst) #Output: [100, 2, [1, 2, 3]]

my_lst[2][1] = 0
print(my_lst) #Output: [100, 2, [1, 0, 3]]
```

Memasukkan Data Pada List

Memasukkan data pada list dapat dilakukan dengan method `.append()`. Dengan method tersebut, item yang dimasukkan ke parameter akan ditambahkan ke belakang list.

```
my_lst = []

my_lst.append(1)
print(my_lst) # Output: [1]

my_lst.append(2)
print(my_lst) # Output: [1, 2]

my_lst.append("Hello")
print(my_lst) # Output: [1, 2, "Hello"]
```

Selain `.append()`, kita juga bisa menggunakan method `.insert()`. Method `.insert()` mengambil 2 parameter (i, elem) sedemikian sehingga elemen `elem` akan dimasukkan ke dalam list di index ke `i.`

```
my_lst = ["a", "b", "c", "d", "e"]

my_lst.insert(1, "o")
print(my_lst) # Output: ["a", "o", "b", "c", "d", "e"]

my_lst.insert(5, "p")
print(my_lst) # Output: ["a", "o", "b", "c", "d", "p", "e"]
```

Menghapus Data Pada List

Untuk menghapus elemen di dalam list, method `.pop()` dapat kita gunakan. Method bisa `.pop()` dipanggil dengan 1 parameter ataupun tanpa parameter. Jika kita tidak memasukkan parameter, maka elemen yang dihapus adalah elemen paling belakang dari list. Jika kita memberikan parameter berupa sebuah bilangan bulat, maka yang dihapus adalah elemen yang sesuai dengan indeks yang dimasukkan ke parameter.

```
my_lst = [1, 2, 3, 4, 5]
```

```
my_lst.pop()
print(my_lst) # Output: [1, 2, 3, 4]

my_lst.pop(2)
print(my_lst) # Output: [1, 2, 4]
```

Selain `.pop()`, kita juga bisa menggunakan method `.remove()` untuk menghapus elemen di dalam list. Method `.remove()` membutuhkan sebuah parameter, kemudian menghapus elemen yang sesuai dengan parameter yang diberikan. Jika ada 2 atau lebih elemen yang sama, maka yang dihapus adalah elemen dengan indeks yang lebih kecil.

```
my_lst = ["a", "b", "c", "b", "z"]

my_lst.remove("a")
print(my_lst) # Output: ["b", "c", "b", "z"]

my_lst.remove("b")
print(my_lst) # Output: ["c", "b", "z"]
```

Materi Function

Function dalam bahasa Python dapat diartikan menjadi sebuah subprogram, yang di dalamnya terdapat beberapa baris program. Misalkan kita mempunyai beberapa *statement* program, lalu kita beri nama kumpulan *statement* tersebut. Nantinya kita bisa menjalankan program yang sudah kita beri nama kapanpun dan dimanapun, dengan cara memanggil namanya tersebut.

Bagian dari program yang membuat sebuah *function* disebut ***function definition*.** Function yang kita panggil bisa disebut ***called*** atau ***invoked***. Setiap function bisa menerima ***parameter(s)*** sebagai variabel yang dibutuhkan. Namun perlu diperhatikan bahwa variabel-variabel yang ada di dalam function hanya bisa dipanggil **di dalam** function tersebut.

Deklarasi Function

Struktur function secara umum adalah seperti ini:

```
def<function-name>(<parameters>):
    <body>
```

- Function-name adalah nama dari fungsi tersebut.
- Parameters adalah variabel-variabel yang akan dipakai di dalam function (opsional)
- Body adalah statements yang akan dijalankan ketika function tersebut dipanggil

Penggunaan Function

Dengan memanfaatkan function, kita bisa mendapatkan beberapa keuntungan yaitu:

- Program yang dijalankan berkali-kali bisa kita tulis dalam satu kali penulisan
- Merapikan struktur program secara umum
- Lebih mudah untuk dibaca dan dilacak jika terjadi sebuah error

Pemanggilan Function

```
def selamat():
    print("Selamat, kamu telah berhasil!")

selamat()
selamat()
```

Outputnya akan menjadi:

```
Selamat, kamu telah berhasil!
Selamat, kamu telah berhasil!
```

Pada program diatas terdapat sebuah function selamat() dan dipanggil sebanyak dua kali di bawahnya. Sehingga setiap pemanggilan function selamat(), statement dalam function tersebut akan dijalankan, yaitu mencetak string “Selamat, kamu telah berhasil!”. Karena function dipanggil sebanyak dua kali, maka program akan mencetak string sebanyak dua kali.

Function dengan Parameter

Bayangkan kita ingin melakukan beberapa statement secara berulang namun kita juga membutuhkan nilai dari variabel untuk menjalankan statement tersebut. Oleh karena itu kita bisa membuat function dengan parameter untuk menjalankan program tersebut.

```
def selamat(nama):
    print("Selamat ulang tahun", nama, "!")
    print("Semoga panjang umur!")

selamat("James")
selamat("Lucy")
```

Outputnya akan menjadi:

```
Selamat ulang tahun James !
Semoga panjang umur!
Selamat ulang tahun Lucy !
Semoga panjang umur!
```

Bisa dilihat bahwa dengan menggunakan fungsi, kita bisa melakukan statement yang sama dengan ringkas. Bayangkan jika ternyata statement yang diulang sangat banyak, kita akan menghabiskan banyak waktu dan program yang dibuat pun akan sulit untuk dibaca!

Function dengan Return

Ketika kita ingin menjalankan beberapa *statements* yang menghasilkan sebuah nilai, kita bisa membuat function yang akan me-*return* atau mengembalikan sebuah nilai dan nilainya bisa kita simpan ke dalam sebuah variabel lain. Misalkan kita ingin membuat function tentang perkalian 2 buah integer.

```
def kali(bil_1, bil_2):
    return bil_1 * bil_2

hasil_1 = (2, 3)
hasil_2 = (0, 10)
print("Hasil perkalian 2 x 3 adalah " + hasil_1)
print("Hasil perkalian 0 x 10 adalah " + hasil_2)
```

Outputnya akan menjadi:

```
Hasil perkalian 2 x 3 adalah 6
Hasil perkalian 0 x 10 adalah 0
```

Kita juga dapat langsung mencetak atau melakukan print nilai hasil dari return suatu function tanpa perlu menyimpan hasil return tersebut ke dalam suatu variabel.

```
def kali(bil_1, bil_2):
    return bil_1 * bil_2

print("Hasil perkalian 3 x 5 adalah " + str(kali(3, 5)))
```

Outputnya akan menjadi:

```
Hasil perkalian 3 x 5 adalah 15
```

Apabila kita mencoba mencetak / mengambil nilai dari sebuah function tanpa return value, maka hasilnya adalah None

```
def selamat_pagi(nama):
    print("Ohayou " + nama)

print(selamat_pagi("Dek Depe"))
# Output:
# Ohayou Dek Depe
# None

x = selamat_pagi("Dek Depe") #Muncul output: Ohayou Dek Depe
print(x) #Muncul output: None
```

Keranjang Boba

Ilustrasi pohon boba
https://s2.pastiseru.com/idimgs/201811/15/4/1542271712983.jpg

Deskripsi Soal:

Keluarga **Dek Depe** memiliki banyak ladang dan berpikir untuk membuat perkebunan. Suatu hari, dia menemukan **pohon** spesial yang dapat membuahkan **boba**. Karena bisnis boba sedang hits, Dek Depe akhirnya menanami ladangnya dengan pohon-pohon boba. Dek Depe membutuhkan beberapa **keranjang** untuk memanen ladang-ladang boba tersebut. Tiap keranjang direpresentasikan sebagai **list** dengan **panjang 2** seperti berikut: **[S, C]**. S merupakan **nama** dari keranjang dan C adalah **kapasitas** biji boba yang bisa ditampung. Dijamin nama setiap keranjang **uni**k dan panjangnya tidak lebih dari **20 karakter** (berbeda satu sama lain).

Tak butuh waktu lama untuk perkebunan keluarga Dek Depe menjadi sebuah perkebunan yang **besar**! Oleh karena itu, Dek Depe memerlukan suatu **sistem manajemen** untuk mengatur keranjang-keranjangnya. Ada beberapa operasi yang bisa dilakukan Dek Depe terhadap keranjang-keranjang yang ia miliki:

a. **BELI**
   Apabila operasi yang dilakukan adalah **BELI**, maka akan ditambahkan keranjang baru ke daftar keranjang yang dimiliki Dek Depe. Dipastikan bahwa nama keranjang baru yang akan ditambahkan belum ada di daftar keranjang sebelumnya.
b. **JUAL**
   Apabila operasi yang dilakukan adalah **JUAL**, maka keranjang yang bersangkutan akan dihapus dari daftar keranjang. Dipastikan bahwa keranjang yang akan dibuang sudah ada di daftar keranjang sebelumnya.
c. **UBAH_KAPASITAS**
   Apabila operasi yang dilakukan adalah **UBAH_KAPASITAS**, maka keranjang yang bersangkutan akan diubah kapasitasnya. Dipastikan bahwa keranjang yang ingin diubah kapasitasnya sudah ada di daftar keranjang.

d. **UBAH_NAMA**
   Apabila operasi yang dilakukan adalah **UBAH_NAMA,** maka keranjang yang bersangkutan akan diubah namanya menjadi nama yang baru. Dipastikan bahwa nama keranjang yang akan diubah namanya sudah ada di daftar keranjang.
e. **LIHAT**
   Apabila operasi yang dilakukan adalah **LIHAT**, maka akan dicetak informasi mengenai keranjang yang bersangkutan. Dipastikan bahwa nama keranjang yang akan dilihat sudah ada di daftar keranjang.
f. **LIHAT_SEMUA**
   Apabila operasi yang dilakukan adalah **LIHAT_SEMUA**, makan akan dicetak informasi mengenai seluruh keranjang dalam bentuk table. (*Hint: gunakan formatting*)
g. **TOTAL_KAPASITAS**
   Apabila operasi yang dilakukan adalah **TOTAL_KAPASITAS**, maka program harus mencetak total kapasitas dari semua keranjang yang dimiliki Dek Depe.

Panduan Pengerjaan:

1. Pada awalnya, daftar keranjang berupa sebuah list kosong. Lalu akan dijalankan N buah operasi.
2. Setiap operasi dipastikan berupa salah satu dari **BELI**, **JUAL**, **UBAH_KAPASITAS**, **UBAH_NAMA**, **LIHAT**, **LIHAT_SEMUA** atau **TOTAL_KAPASITAS**.
3. Setiap operasi berupa sebuah string yang berisi informasi mengenai operasi yang ingin dilakukan
4. Jika operasi yang dijalankan adalah **BELI**, maka operasi tersebut akan diikuti sebuah string dan sebuah bilangan bulat yang merepresentasikan nama keranjang dan kapasitas keranjang yang ingin ditambahkan
5. Jika operasi yang dijalankan adalah **JUAL**, maka operasi tersebut akan diikuti sebuah bilangan bulat yang merepresentasikan indeks keranjang yang ingin dijual. Dijamin indeks keranjang yang ingin dijual ada dalam daftar_keranjang
6. Jika operasi yang dijalankan adalah **UBAH_KAPASITAS**, maka operasi tersebut akan diikuti oleh 2 buah bilangan bulat yang masing-masing menyatakan indeks keranjang yang ingin diubah dan kapasitas barunya. Dijamin indeks keranjang yang ingin diubah ada dalam daftar_keranjang
7. Jika operasi yang dijalankan adalah **UBAH_NAMA**, maka operasi tersebut akan diikuti oleh sebuah bilangan bulat dan sebuah string yang masing-masing menyatakan indeks keranjang yang ingin diubah dan nama barunya. Dijamin indeks keranjang yang ingin diubah ada dalam daftar_keranjang.
8. Jika operasi yang dijalankan adalah **LIHAT**, maka operasi tersebut akan diikuti oleh sebuah bilangan bulat yang menyatakan indeks keranjang yang ingin dilihat. Dijamin indeks keranjang yang ingin dilihat ada dalam daftar_keranjang.
9. Jika operasi yang dijalankan adalah **LIHAT_SEMUA**, maka akan dicetak semua keranjang dalam format tabel. Sediakan minimal 20 karakter untuk nama keranjang dan 10 karakter untuk kapasitas keranjang.

10. Jika operasi yang dijalankan adalah **TOTAL_KAPASITAS**, maka program mencetak total kapasitas dari semua keranjang yang dimiliki Dek Depe.
11. Diberikan template untuk soal ini. Anda hanya perlu mengimplementasikan function-function yang sudah dibuat. Template dapat diakses di sini.
12. Berikan dokumentasi yang baik untuk program anda. Jika anda tidak memberikan dokumentasi, anda akan terkena penalti berupa pengurangan nilai.

Contoh Interaksi 1:

```
Masukkan banyak operasi: 7

Operasi 1: BELI KRJ1 5
Berhasil menambahkan KRJ1 dengan kapasitas 5

Operasi 2: BELI KRJ2 10
Berhasil menambahkan KRJ2 dengan kapasitas 10

Operasi 3: BELI Kantong_Dora*mon 9999
Berhasil menambahkan Kantong_Dora*mon dengan kapasitas 9999

Operasi 4: TOTAL_KAPASITAS
Total kapasitas keranjang Dek Depe adalah 10014

Operasi 5: LIHAT 1
Keranjang KRJ2 memiliki kapasitas sebesar 10

Operasi 6: JUAL 1
Berhasil menjual KRJ2 yang memiliki kapasitas sebesar 10

Operasi 7: LIHAT_SEMUA
Keranjang Dek Depe
--------------------------
KRJ1                | 5
Kantong_Dorae*mon   | 9999
```

Penjelasan:
Setelah operasi 1, 2, dan 3 dijalankan, list `daftar_keranjang` berisi `[["KRJ1", 5], ["KRJ2", 10], ["Kantong Dora*mon", 9999]]`

Ketika operasi 4 dijalankan, maka program akan menghitung total kapasitas dari setiap keranjang yaitu = 5 + 10 + 9999 = 10014

Ketika operasi 5 dijalankan, maka program akan mencetak informasi berupa nama dan kapasitas keranjang yang ingin dilihat oleh Dek Depe

Ketika operasi 6 dijalankan, maka program akan menghapus `["KRJ2", 10]` dari list `daftar_keranjang`

Ketika operasi 7 dijalankan, maka program akan mencetak semua keranjang yang dimiliki Dek Depe dalam bentuk tabel.

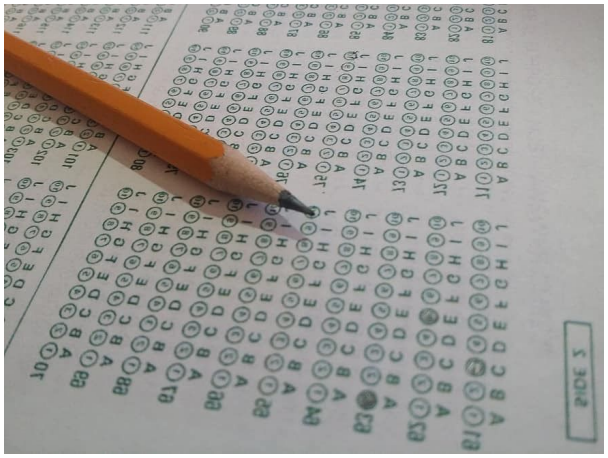
Contoh Interaksi 2:

```
Masukkan banyak operasi: 6

Operasi 1: BELI KRJ4 10
Berhasil menambahkan KRJ4 dengan kapasitas 10

Operasi 2: TOTAL_KAPASITAS
Total kapasitas keranjang Dek Depe adalah 10

Operasi 3: UBAH_KAPASITAS 0 100
Berhasil mengubah kapasitas KRJ4 menjadi 100

Operasi 4: TOTAL_KAPASITAS
Total kapasitas keranjang Dek Depe adalah 100

Operasi 5: UBAH_NAMA 0 KERJ4
Berhasil mengubah nama KRJ4 menjadi KERJ4

Operasi 6: LIHAT 0
Keranjang KERJ4 memiliki kapasitas sebesar 100
```

Contoh Interaksi 3:

```
Masukkan banyak operasi: 4

Operasi 1: BELI KRJ1 5
Berhasil menambahkan KRJ1 dengan kapasitas 5

Operasi 2: LIHAT 0
Keranjang KRJ1 memiliki kapasitas sebesar 5

Operasi 3: UBAH_NAMA 0 KRJ0
Berhasil mengubah nama KRJ1 menjadi KRJ0

Operasi 4: LIHAT 0
Keranjang KRJ0 memiliki kapasitas sebesar 5
```

Bonus Task: Cari Soal



Ilustrasi Ujian Online

Deskripsi Soal:

Pada suatu hari Dek Depe sedang mengerjakan **ujian online** di Fakultas Ternak Lele tercinta. Saat sedang asik - asiknya mengerjakan ujian dengan sangat serius, tiba - tiba Dek Depe diberikan informasi dari Fakultas, bahwa **jawaban** dari soal ujian tersebut **bocor** dan sudah tersebar ke seluruh mahasiswa dikarenakan **kecerobohan** seorang **asisten** pengajar. Hal ini membuat para tenaga pengajar harus melakukan **pergantian** soal ujian agar para mahasiswa tetap menjaga integritas dan kejujuran **#NoPlagiarisme**

Ekspresi asisten yang tidak sengaja membocorkan jawaban

Untung saja, tenaga pengajar sudah membuat beberapa versi soal ujian dengan berbagai macam tingkat kesulitan yang direpresentasikan dengan sebuah **barisan L**. Nah, soal ujian yang mengalami kebocoran adalah soal ujian yang merupakan **elemen pertama** dari **barisan L** tersebut.

Karena soal ujian bocor, maka tim pengajar harus melakukan pencarian soal ujian lain yang memiliki tingkat kesulitan **paling mirip** dengan soal ujian yang bocor tersebut. Soal **paling**

mirip adalah soal yang **selisih** tingkat kesulitannya dengan soal yang bocor **paling sedikit** dibanding soal-soal lainnya.

Tim pengajar harus cepat cepat dalam mencari soal pengganti! Oleh karena itu, mereka meminta kamu sebagai **programmer andal** di dunia persilatan ini untuk membantu tenaga pengajar dalam menjunjung integritas dan kejujuran.

Panduan Pengerjaan:

1. Input berubah sebuah barisan bilangan bulat yang menyatakan tingkat kesulitan tiap soal yang dipisahkan oleh spasi.
2. Output nantinya adalah **nomor** soal yang merupakan soal yang tingkat kesulitannya **paling mirip**.
3. Jika ada **dua atau lebih** soal yang **paling mirip** maka nomor soal yang diambil adalah soal dengan nomor soal **terkecil yang ditemukan**.
4. Tugas anda adalah mengimplementasikan sebuah function yang menerima parameter berupa sebuah list yang berisi bilangan bulat. Function tersebut me-return sebuah bilangan yang berupa nomor soal paling mirip.

Simulasi Program

Contoh masukkan 1:

Masukkan tingkat kesulitan soal: **1 5 4 9 7**

Contoh keluaran 1:

Versi soal paling mirip adalah soal ke: **3**

Penjelasan:
Soal-soal yang dimiliki oleh tim pengajar memiliki tingkat kesulitan 1, 5, 4, 9, dan 7. Lalu, soal yang bocor adalah soal pertama (dengan tingkat kesulitan 1).
Selisih soal 2 dengan soal 1 = 5 - 1 = 4
Selisih soal 3 dengan soal 1 = 4 - 1 = 3
Selisih soal 4 dengan soal 1 = 9 - 1 = 7
Selisih soal 5 dengan soal 1 = 7 - 1 = 6
Soal yang dipilih adalah soal 3 dengan selisih 3

Contoh masukkan 2:

Masukkan tingkat kesulitan soal: **7 5 4 6 6 6 1 1 2 10**

Contoh keluaran 2:

Versi soal paling mirip adalah soal ke: **4**

Contoh masukkan 3:

Masukkan tingkat kesulitan soal: **5 5 5 5**

Contoh keluaran 3:

Versi soal paling mirip adalah soal ke: **2**

Contoh masukkan 4:

Masukkan banyak versi soal: **20 5 2 7 8 2 18**

Contoh keluaran 4:

Versi soal paling mirip adalah soal ke: **7**

Deliverable

Buatlah file zip dengan format nama **[KodeAsdos]_[Nama]_[NPM]_[Kelas]_lab5.zip** yang berisi file `lab05.py`, dan `lab05_bonus.py`

Contoh: **`DNS_LouisAkbar_1234567890_lab3.zip`**