Dasar-Dasar Pemrograman 1

Lab 04

Formatted strings, Text I/O, Exceptions

Revisi 1: Akses untuk test case sudah dibuka untuk email selain UI
Revisi 2: Merevisi batasan ke-1 tentang format nama input file

---

Dek Depe *Back to Campus*

Hari yang dinantikan Dek Depe sudah tiba. Setelah pandemi mulai membaik, Dek Depe sangat bahagia karena akhirnya bisa belajar secara langsung di kampus tercinta, Universitas Impian. Namun, Dek Depe lupa bahwa ia belum mempersiapkan apapun untuk mengikuti perkuliahan luring. Sebagai teman yang baik, kamu akhirnya memutuskan untuk menolong Dek Depe mendaftarkan barang-barang yang perlu ia beli di salah satu toko ternama, Pacil Mart.

Ketentuan Program

Kamu diminta untuk membuat sebuah program yang dapat menerima input dari sebuah file `.txt` yang setiap barisnya berisi **nama barang**, **uang** yang dialokasikan untuk barang tersebut, dan **harga barang** tersebut. Output program adalah *formatted string* di **terminal** yang memberikan keterangan mengenai berapa kuantitas maksimal setiap barang yang bisa dibeli dengan uang yang dialokasikan.

- Program akan meminta input berupa **nama file .txt**. File tersebut berisi barang yang ingin dibeli, uang yang dibayarkan untuk membeli barang tersebut, dan harga satuan dari barang tersebut.
- Jika file tidak ada, terminal akan mencetak **File tidak tersedia**. Jika file ada tapi isinya kosong, cetak **File input ada tapi kosong**. Pada kedua kasus ini, daftar belanjaan tidak perlu dicetak.
- Setiap baris file input akan berisi **[Nama barang] [Uang yang dialokasikan] [Harga satuan barang].** Program akan menghitung berapa maksimal barang yang bisa dibeli dengan uang yang dialokasikan serta kembaliannya.

💡 **Hint :** Setiap nama barang **dijamin** hanya terdiri dari **satu kata**.
Anda dapat memanfaatkan method split() untuk membantu program Anda.

- Program akan mencetak daftar belanjaan berupa tabel yang memiliki kolom **nama** barang, **jumlah** barang yang dibeli dan **kembalian** dari masing-masing pembelian.
- Output dari program harus memenuhi **ketentuan** berikut:
  1. **Nama barang** harus dibuat **rata kiri** dengan **space** sebanyak **12**. Jika panjang karakter nama barang **kurang dari 12**, sisa space yang kosong harus diisi dengan **spasi**.
  2. **Jumlah barang** yang dibeli harus dibuat **rata kanan** dengan space sebanyak **8**. Jika **banyak digit** pada jumlah barang **kurang dari 8**, sisa space yang kosong harus diisi dengan **spasi**.
  3. **Kembalian** harus dibuat dengan **rata kanan** dengan space sebanyak **10**. Jika **banyak digit** pada kembalian **kurang dari 10**, sisa space yang kosong harus diisi dengan **spasi.**
  4. Setiap data pada output **harus dipisahkan** dengan **tanda pemisah "|"**

  Ketiga data pada poin (1), (2), dan (3) **dijamin** tidak akan melebihi space yang tersedia sehingga output terjamin rapi. Silakan lihat contoh output untuk lebih memahami ketentuan di atas.

💡 **Hint :** Gunakan *string formatting* untuk mengimplementasikan output program

Berikut contoh penggunaan method **`split()`** pada string:

```
x = "satu, dua, tiga"
y = x.split()
z = x.split(",")
print(y)
print(z)
print(y[0])
print(z[1])
print(y[2])
print(z[2])
```

Output:

```
['satu,', 'dua,', 'tiga']
['satu', ' dua', ' tiga']
satu,
 dua
tiga
 tiga
```

Penjelasan:

- **`x.split()`**
  akan membuat list dari kata-kata penyusun string yang dipisahkan oleh `" "`(spasi), `"satu, dua, tiga" -> ['satu,', 'dua,', 'tiga'].`
- **`x.split(",")`**
  akan membuat list dari kata-kata penyusun string yang dipisahkan oleh `","` (koma), `"satu, dua, tiga" -> ['satu', ' dua', ' tiga'].`

Untuk mengakses hasil splitnya, kalian dapat menggunakan indexing karena hasil method split() akan berbentuk list **(Ingat juga bahwa indexing dimulai dari 0)**.

Sebagai contoh dari `y = x.split() = ['satu,', 'dua,', 'tiga']`:

`y[0] => 'satu,'`

`y[1] => 'dua,'`

`y[2] => 'tiga'`

Batasan

1. Format nama input file **dijamin valid**.
1. Isi file .txt dijamin **sesuai format input atau kosong**.
2. Harga satuan barang dijamin **> 0**
3. Uang yang dialokasikan dijamin **> 0**

Test Case

File input pada test case dapat diunduh melalui tautan ini (sudah bisa dibuka menggunakan email selain email UI). Ingat bahwa test case yang diuji tidak terbatas pada test case yang diberikan.

> 💡 Teks berwarna `merah` adalah masukan dari pengguna.

Input 1:

```
Selamat datang di Pacil Mart!

Masukkan nama file input: text1.txt
```

Isi text1.txt:

```
Pensil 5000 2000
Pulpen 7000 5000
Tipe-X 20000 10000
Buku 70000 15000
```

Output 1:

```
Berikut adalah daftar belanjaanmu:

Nama Barang |  Jumlah| Kembalian
—------------------------------
Pensil      |       2|      1000
Pulpen      |       1|      2000
Tipe-X      |       2|         0
Buku        |       4|     10000

Terima kasih sudah belanja di Pacil Mart!
```

Penjelasan Output 1:

- Pensil
  Jumlah = Uang yang dialokasikan // Harga satuan barang
  = 5000 // 2000
  = 2
  Kembalian = Uang yang dialokasikan - (Harga satuan barang * Jumlah)
  = 5000 - (2000 * 2)
  = 1000
- Pulpen
  Jumlah = Uang yang dialokasikan // Harga satuan barang
  = 7000 // 5000
  = 1
  Kembalian = Uang yang dialokasikan - (Harga satuan barang * Jumlah)
  = 7000 - (5000 * 1)
  = 2000
- Tipe-X
  Jumlah = Uang yang dialokasikan // Harga satuan barang
  = 20000 // 10000
  = 2
- Kembalian = Uang yang dialokasikan - (Harga satuan barang * Jumlah)
  = 20000 - (10000 * 2)
  = 0
- Buku
  Jumlah = Uang yang dialokasikan // Harga satuan barang
  = 70000 // 15000
  = 4
  Kembalian = Uang yang dialokasikan - (Harga satuan barang* Jumlah)
  = 70000 - (15000 * 4)
  = 10000

Input 2:

```
Selamat datang di Pacil Mart!

Masukkan nama file input: text2.txt
```

Isi text2.txt (kosong):

```

```

Output 2:

```
File input ada tapi kosong
```

Input 3:

```
Selamat datang di Pacil Mart!

Masukkan nama file input: file3.txt
```

Output 3 (file3.txt tidak ada):

```
File tidak tersedia
```

Komponen Penilaian

- **50%** Kebenaran fungsionalitas program
- **25%** Menggunakan **exception**, **formatted string**, dan **text I/O**
- **5%** Dokumentasi Kode
- **10%** Memenuhi kriteria standar penulisan kode Python*
- **10%** Mengumpulkan dengan format dan penamaan file yang benar

*Standar penulisan kode yang harus dipenuhi yaitu:

1. Indentasi yang konsisten
2. Aturan penamaan variabel mengikuti Python Naming Convention (sumber)
3. Penamaan Module, Class, Method, dan Variabel yang tidak ambigu

Deliverables

Kumpulkan berkas `[Kelas]_[KodeAsdos]_[NPM]_[NamaLengkap]_Lab04.py` yang telah di-**zip** dengan format penamaan seperti berikut.
`[Kelas]_[KodeAsdos]_[NPM]_[NamaLengkap]_Lab04.zip`
Contoh: `A_ABC_2206123456_ThamiEndamora_Lab04.zip`