Rekursive

Tutorial 8 [G,F] - Dasar-Dasar Pemrograman 1 Gasal 2019/2020

Selamat datang di Tutorial 8 DDP1! Kita akan mengerjakan latihan pemrograman singkat pada sesi tutorial kali ini.

Judul	File Submission	Bobot Nilai
Latihan 19 : Undian Berhadiah	latihan19.py	80
Latihan 20 : Terbesar	latihan20.py	20
Challenge 6 : The Holy Necronomicon	challenge6.py	0

Mohon kumpulkan semua file jawaban Anda dalam bentuk zip dengan format Lab8_Nama_NPM_KodeAsdos.zip. Contoh: Lab8_WindiChandra_1606862721_YE.zip

Latihan #19: Undian Berhadiah (latihan19.py, skor: +80)

Konsep penting: rekursi, fungsi

Alternate Ilustrasi Kak Cephas Knight UwU Kewl Charming But Need Money made with Picrew

Setelah menjalani kepanitiaan PacilFest, Kak Cephas kehabisan uang karena selama kepanitiaannya, ia selalu memesan makanan lewat G*-F**d. Kak Cephas sangat butuh uang dengan cepat karena ia ingin membeli figurin edisi terbatas dari N*k*pa*a di internet.

Kak Cephas pun mencoba peruntungannya dengan mengikuti undian tebak angka yang diadakan oleh sebuah perusahaan cat bernama Lol1Cat. Karena Kak Cephas hanya memiliki satu kesempatan dalam undian ini, Kak Cephas ingin memilih angka yang tepat sehingga ia menang undian tersebut.

Untungnya, Kak Cephas memiliki pengalaman cukup banyak dari pelajaran Matematika Diskrit dan S3 di bidang Cocoklogi. Kak Cephas mulai menganalisa angka pemenang sebelumnya berdasarkan tanggal undian tersebut diadakan:

Tanggal Undian	Angka pemenang
1 Januari **2001**	**01** 1 **20**
2 Februari **2001**	**01** 2 **20**
1 Januari **2002**	**02** 1 **20**
2 Februari **2002**	**02** 2 **20**
1 Januari **2003**	**03** 2 **20**
2 Februari **2003**	**03** 4 **20**

1 Januari **2004**	**04** 3 **20**
2 Februari **2004**	**04** 6 **20**
1 Januari **2005**	**05** 5 **20**
2 Februari **2005**	**05** 10 **20**

Ternyata, angka pertama dan ketiga dari angka pemenang bergantung terhadap tahun undian tersebut diadakan! **Angka pertama adalah dua digit terakhir dari tahun undian, dan angka ketiga adalah dua digit pertama dari tahun undian.** Bagaimana dengan angka kedua?

Setelah dilakukan analisis cocoklogi™ tingkat dewa oleh Kak Cephas, akhirnya ia menemukan bagaimana cara mendapatkan angka kedua tersebut.

Kak Cephas yakin bahwa angka kedua tersebut didapatkan dari deret Cibonacci (*customized Fibonacci*). Deret tersebut mirip dengan deret Fibonacci. Apabila deret Fibonacci dimulai dari angka 1 dan 1 (1, 1, 2, 3, 5, dst), maka deret Cibonacci dimulai dari angka X dan Y.

Hint: Deret Fibonacci merupakan deret yang dimulai dari 2 angka yaitu 1 dan 1, dan angka berikutnya adalah jumlah dari dua angka sebelumnya, contoh: 1, 1, 2, 3, 5, 8, 13, dan seterusnya. Secara matematis, bilangan ke-N dari deret Fibonacci dapat didefinisikan sebagai $a_N = a_{N-1} + a_{N-2}$, *dengan* $a_1 = a_2 = 1$.

Ternyata, angka kedua dari undian bisa didapat dari bilangan Cibonacci ke-N yang dimulai dari angka X dan Y, dengan:

- N = dua digit terakhir pada tahun, dan
- X = Tanggal berapa undian diadakan, dan
- Y = Bulan berapa undian diadakan.

Perlu diketahui bilangan Cibonacci ke-1 dan 2 masing-masing adalah X dan Y. Untuk bilangan Cibonacci ke-0, disepakati bahwa nilainya adalah 0 (nol).

Sebagai contoh, kita ambil kasus pada tanggal 2 Februari 2005. Berdasarkan analisa Kak Cephas, angka pertama dan ketiga masing-masing adalah 05 dan 20 (dari tahun 2005). Angka kedua didapat dari bilangan Cibonacci ke-5 yang dimulai dari angka 2 dan 2.

Berikut adalah deret bilangan Cibonacci yang dimulai dari angka 2 dan 2:

2, 2, 4, 6, **10**, 16, 26, …

… sehingga terlihat bahwa bilangan Cibonacci ke-5 adalah 10.

Maka, angka pemenangnya adalah 05 10 20, sesuai dengan data pemenang di atas.

Sekarang, bantulah Kak Cephas dengan membuat program yang menerima sebuah tanggal undian tersebut diadakan, kemudian menampilkan tiga angka yang akan memenangkan undian tersebut! Anda dibebaskan dalam mengatur format masukan maupun keluaran.

Contoh interaksi program:

```
Masukkan tanggal undian: 2,2,2004
Angka pemenang: 04 6 20
```

Contoh interaksi lain:

```
Masukkan tanggal undian: 2,2,2005
Angka pemenang: 05 10 20
```

Contoh interaksi lain:

```
Masukkan tanggal undian: 1,7,2000
Angka pemenang: 00 0 20
```

Contoh interaksi lain:

```
Masukkan tanggal undian: 14,11,2019
Angka pemenang: 19 50782 20
```

Perhatikan bahwa tulisan tebal berwarna biru adalah masukan dari pengguna.
Format masukan yang digunakan pada contoh adalah:

```
<tanggal undian>,<bulan undian>,<tahun undian>
```

Catatan: Untuk mempermudah, Anda boleh mengasumsikan bahwa tahun pasti terdiri dari 4 digit angka. Anda diwajibkan untuk menggunakan rekursi (pada perhitungan Cibonacci ke-N). **Apabila tidak menggunakan rekursi, maka program Anda hanya akan mendapatkan 20 poin.**

Latihan #20: Terbesar (latihan20.py, skor: +20)

Konsep penting: rekursi, fungsi

Diberikan sebuah tuple, tuple tersebut mengandung satu atau lebih elemen. Setiap elemen dalam tuple tersebut bisa merupakan sebuah bilangan, atau merupakan tuple yang mempunyai sifat yang sama (*nested tuple*, bisa mengandung bilangan atau tuple lagi). Buatlah sebuah fungsi yang menerima *nested tuple* tersebut, dan menghasilkan bilangan terbesar yang ada di *nested tuple* tersebut!

Apabila *nested tuple* tidak mengandung bilangan apa-apa, fungsi Anda harus menghasilkan **None**.

Dalam pengerjaan soal ini, tidak diperlukan proses input/output. Disediakan *template* untuk mengerjakan soal latihan ini:

```
def max_nested_tuple(nested_tuple):
    # Tambahkan kode Anda disini


if __name__ == '__main__':
    tup1 = (1,2,3,4)
    print(tup1, '->', max_nested_tuple(tup1))
    tup2 = (1,(2,5),(1,(((9)))))
    print(tup2, '->', max_nested_tuple(tup2))
    tup3 = ((), ((((((()))))), ((), ((()), ())))
    print(tup3, '->', max_nested_tuple(tup3))
    tup4 = ()
    print(tup4, '->', max_nested_tuple(tup4))
```

Anda diperbolehkan untuk menambahkan kode dimanapun, asalkan fungsi `max_nested_tuple(nested_tuple)` tetap menerima sebuah *nested tuple* dan menghasilkan bilangan terbesar dalam *nested tuple* tersebut.

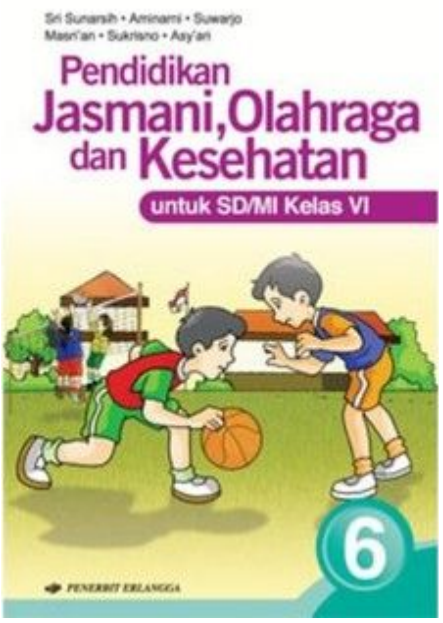
Berikut adalah keluaran yang diharapkan untuk program di atas:

```
(1, 2, 3, 4) -> 4
(1, (2, 5), (1, 9)) -> 9
((), (), ((), (), ())) -> None
() -> None
```

Anda diperbolehkan untuk menambahkan keluaran (untuk contoh kasus) untuk memastikan bahwa fungsi Anda benar.

Catatan: Anda diwajibkan untuk menggunakan rekursi (pada fungsi `max_nested_tuple(nested_tuple)`). **Apabila tidak menggunakan rekursi, maka program Anda hanya akan mendapatkan 10 poin.**

Challenge #6: The Holy Necronomicon (challenge6.py, skor: +0)



Ilustrasi *The Holy Necronomicon*

Setelah Sang *Dragon Slayer* **Cephas Knight** mengalahkan The Grandmaster dalam pertempuran galaktik nan sengit, tiba saatnya bagi Cephas Knight untuk menghadapi musuh terbesarnya, **Yog-Sohoth, The Opener of Gates, Son of the Nameless Mist**. Tetapi untuk menghadapi musuh ini, Cephas Knight harus menerima *enchanted item* dari buku suci *The Holy Necronomicon*, yaitu **The Tears of Shub-Niggurath.**

Buku *Holy Necronomicon* adalah sebuah buku yang dapat membedakan kata *holy* dan kata *corrupted*. Bila seseorang membacakannya beberapa kata *holy* dan kemudian membacakan kalimat, maka *The Holy Necronomicon* akan memberitahu ada berapa kata *corrupted* yang berada dalam kalimat tersebut.

Cephas Knight membutuhkan jumlah kata *corrupted* dalam beberapa kalimat untuk mendapatkan *enchanted item* yang akan membantunya mengalahkan musuhnya .

Suatu kata dikatakan *corrupted* apabila kata tersebut merupakan kata *holy* yang ditambahkan satu atau lebih karakter pada sebelum atau sesudah kata tersebut.

Perhatikan bahwa kata holy dalam kata corrupted **tidak case sensitive,** jadi bila kata holy adalah **Bat** maka kata corrupted nya adalah **Bat**ter**,** lam**bat,** sam**bat,** s**bat**ter**,** s**Bat**ter**, etc**

Contoh:
Kata *holy*: **Bat**
Kata *corrupted*:

- **Sbat (“S” + “bat”)**
- **Rambatter (“Ram” + “bat” + “ter”)**
- **Wombats (“Wom” + “bat” + “s”)**
- **Batter (“Bat” + “ter”)**

Contoh Lain:
Kata *holy*: **Bit**
Kata *corrupted*:

- **Gigabit (“Giga” + “bit”)**
- **bitter (“bit” + “ter”)**
- **Summerbitz (“Summer” + “bit” + “z”)**
- **Bitjayangkamueaaa (“Bit”+”tjayangkamueaaa”)**

Buatlah sebuah program yang dapat menghitung banyaknya kata *corrupted* pada satu atau lebih kalimat yang diberikan oleh pengguna untuk setiap kata *holy* yang diberikan!

Anda dibebaskan untuk menentukan format masukan atau keluaran, asalkan program Anda menerima daftar kata *holy*, dan satu atau lebih kalimat yang dipisahkan oleh *newline*. Program juga harus berhenti menerima masukan apabila diberikan string kosong untuk kalimat tersebut. Untuk mempermudah, Anda dapat mengasumsikan setiap kalimat tidak akan mengandung tanda baca apapun.

Contoh interaksi program:

```
Masukkan kata holy: root,sit,lad,bat
Masukkan kalimat: The bat sits on the ladder
Masukkan kalimat: The root of all evil is the sister batter
Masukkan kalimat:
----------------------------
THE HOLY NECRONOMICON SPEAKS
----------------------------
root: 1
sit: 1
lad: 1
bat: 2
```

Perhatikan bahwa tulisan tebal berwarna biru adalah masukan dari pengguna. Apabila terdapat satu kata *corrupted* yang bisa dibentuk oleh dua atau lebih kata *holy* yang berbeda, maka masing-masing dihitung 1 untuk setiap kata *holy* yang bisa membentuk kata *corrupted* tersebut.