729G46. Dugga. DUG1, 1hp.

2024-13-37 kl. 24.00-26.00

Se separat papper för information om inloggning, utloggning, inlämning av uppgifter och frågor under duggan.

Tillåtna hjälpmedel penna, papper, linjal, suddgummi, godkänd(a) bok/böcker (lärare kontrollerar böcker innan duggan), en enkelsidig A4 med handskrivna anteckningar

Filer och kataloger

Istället för din vanliga hemkatalog får du en temporär hemkatalog under duggan. Du kommer fortfarande åt den med `~/`. Lägg kod du skriver direkt i denna katalog eller i en underkatalog du själv skapar. **Du kan inte spara din fil i `~/Desktop/given_files`**.

Skriva kod

Spara koden för alla deluppgifter i varje uppgift i en separat fil:

- Spara svaren till Uppgift 1 i filen `uppg1.py`.
- Spara svaren till Uppgift 2 i filen `uppg2.py`.
- Spara svaren till Uppgift 3 i filen `uppg3.py`.

Inga doc-strängar behöver skrivas. Inga avdrag kommer ges om koden bryter mot `PEP8` eller `PEP257`.

Begränsningar av hur en uppgift får lösas

I de fall som en uppgift ska lösas på ett speciellt sätt står detta angivet i uppgiften; t.ex. "*Lös uppgiften rekursivt.*" eller "*Lös uppgiften med hjälp av en `for`-loop*".

Om specifika funktioner inte får användas i lösningen står detta också angivet i uppgiften, t.ex. "*Funktionen `max()` får inte användas.*"

Tillåtna moduler att använda på duggan: Följande Python-moduler *tillåtna* att användas.

- `copy`
- `math`
- `random`
- `sys`

Godkänt och rättning

Duggan består av tre uppgifter. För att få godkänt på duggan måste **alla uppgifter** vara godkända.

- För att få **godkänt** på en uppgift med deluppgifter (t.ex. uppgift 1) så måste samtliga deluppgifter vara godkända om inte annat anges.
- Ej godkända uppgifter kan kompletteras i mån av tid.
- *Upprepade inlämningar där du uppenbart gissat dig fram kan komma att underkännas och kan då inte längre kompletteras.* Om du inte förstår feedback du fått så be om ett förtydligande istället för att chansa.

Notation i uppgifter

När strängar visas i uppgifterna kommer de alltid att vara omgivna av citattecken. Om det står att strängen `"hej"` ska skrivas ut, ska utskriften motsvara `print("hej")`.

- termen *heltal* syftar alltid på datatypen `int` om inget annat anges
- termen *sträng* syftar alltid på datatypen `str` om inget annat anges

Exempel i uppgifter

- Tänk på att eventuella exempel till uppgifterna *inte täcker alla möjliga fall*.
- Ibland kan t.ex. divisioner resultera i svar med precisionsfel. Om du t.ex. får `1.99999999` som svar vid divisionen `4/2.0` behöver du inte oroa dig.

Kom ihåg att testköra din kod

Inkompletta och delvis felaktiga lösningar *kan* ge delpoäng men kom ihåg att testköra din kod innan du skickar in! *Filer som kraschar vid körning ger normalt sett 0 poäng på hela uppgiften.* Alltså, om du vill skicka in inkompletta lösningar som kraschar när du testkör så kommentera ut eventuella anrop till den kraschande koden innan du skickar in.

Starta Thonny

I tentasystemet startar du Thonny genom att öppna ett terminalfönster och skriva:

```
$ 729g46
```

Uppgift 1 - Loopar

- För att få godkänt på denna uppgift skall **båda** deluppgifterna lösas.
- En av uppgifterna skall lösas med en `while`-loop och en skall lösas med `for`-loop.
- Comprehensions och generatorer får *inte* användas för att lösa uppgifterna.
- Spara lösningarna till båda deluppgifterna i Uppgift 1 i en fil med namnet `uppg1.py`.
- Filen skickas in via Studentklienten som '*Assignment #1*'.

Uppgift 1.1 - lös med lämplig loop

Skriv funktionen `get_unique_random_integers(start, stop, number_of_ints)` vars argument är alla heltal större än 0 och `start` $<$ `stop`. Funktionen ska returnera en lista med som innehåller slumpmässiga heltal mellan `start` och `stop` (inklusive `start` och `stop`). Inget av heltalen får förekomma mer än en gång. Antag också att `number_of_ints-1` $\leqslant$ `stop-start`.

Använd funktionen `randint(a, b)` från modulen `random` som returnerar ett slumpmässigt heltal `c`, där `a` $\leqslant$ `c` $\leqslant$ `b`.

Exempel

```
>>> get_unique_random_integers(1, 100, 10)
[93, 23, 6, 26, 90, 29, 59, 12, 15, 86]  # tio slumpmässigt valda heltal utan
                                         # dubletter
```

Uppgift 1.2 - lös med lämplig loop

Skriv funktionen `remove_all_strings(values)` som tar in en lista med noll eller fler element och som kan innehålla strängar, heltal och flyttal. Funktionen ska returnera en lista som innehåller alla värden i `values` som inte är strängar.

Exempel

```
>>> remove_all_strings([])
[]
>>> remove_all_strings(['hejsan'])
[]
>>> remove_all_strings(['hejsan', 1, 0.5, 'hopp'])
[1, 0.5]
```

Uppgift 2 - Dictionaries och nästling

- För att få godkänt på denna uppgift skall **båda** deluppgifterna lösas.
- Spara lösningarna till båda deluppgifterna i Uppgift 2 i en fil med namnet `uppg2.py`.
- Filen skickas in via Studentklienten som '*Assignment #2*'.

Uppgift 2.1 - Dictionary

Skriv funktionen `get_popular_words(word_freqs)` som tar in ett dictionary vars nycklar är ord (datatyp `str`) och vars värden är antalet förekomster av det ordet i en text (datatyp `int`). Funktionen ska returnera en lista med alla ord som förekommer fler än 50 gånger.

Exempel

```
>>> wf = { 'hej': 1, 'och': 4, 'förlåt': 204, 'havregröt': 50, 'träd': 102 }
>>> get_popular_words(wf)
['förlåt', 'träd' ]
```

Uppgift 2.2 - Nästlad data

Skriv funktionen `count_pets(people)` som ska returnera det totala antalet djur som personerna i listan `people` har tillsammans. Varje element i listan `people` är ett dictionary med nycklarna `'name'`, `'age'` och `'number of pets'`. Värdet tillhörande nyckeln `'name'` är personens namn. Värdet tillhörande nyckeln `'age'` är personens ålder, värdet tillhörande nyckeln `'number of pets'` är antalet husdjur som personen äger. Exemplet nedan är en lista som innehåller tre personer:

```
my_friends = [ { 'name': 'Ada', 'age': 5, 'number of pets': 2 },
               { 'name': 'Bertil', 'age': 8, 'number of pets': 0 },
               { 'name': 'Celeste', 'age': 32, 'number of pets': 4 } ]
```

Exempel

```
# my_friends definierat enligt ovan
>>> count_pets(my_friends)
6
```

Uppgift 3 - Rekursion

- Spara lösningen till Uppgift 3 i en fil med namnet `uppg3.py`.
- Filen skickas in via Studentklienten som '*Assignment #3*'.
- *Uppgiften ska lösas med rekursion*

Skriv funktionen `count_strings(value_list)` som ska returnera antalet element av datatypen `str` som finns i listan `value_list`. Det returnerade värdet ska vara ett heltal.

Exempel

```
>>> count_strings(['30', True, 'True', 3, 3.0, 6, 'a', 100, 2, 30])
3
>>> count_strings([])
0
```