729G46. Dugga. DUG1, 1hp.

2025-01-08 kl. 14.00-16.00

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_next_power_of_two(val)` vars argument är ett heltal större än 0. Funktionen ska returnera närmsta tvåpotens (2^x) som är *större* än `val`.

Uppgiften skall lösas med en lämplig loop. Dvs, du får t.ex. inte använda logaritmer eller metoden `bit_length()` för att räkna ut lösningen.

Exempel

```
>>> get_next_power_of_two(70)
128  # 2^7
>>> get_next_power_of_two(8)
16  # 2^4
```

Uppgift 1.2 - lös med lämplig loop

Skriv funktionen `count_swedish_vowels(word)` vars argument `word` är en sträng. Antag att alla bokstäver i `word` är gemener. Funktionen ska returnera hur många av bokstäverna i `word` som är svenska vokaler. Följande bokstäver är vokaler: aouåeiyäö.

Uppgiften skall lösas med en lämplig loop.

Tips: Operatorn `in` kan användas på strängar; `'a' in 'abc'` → `True`.

Exempel

```
>>> count_swedish_vowels('bokomslag')
3
>>> count_swedish_vowels('pythonprogram')
4
>>> count_swedish_vowels('hmm')
0
```

Uppgift 2 - Rekursion

- Spara lösningen till Uppgift 2 i en fil med namnet `uppg2.py`.
- Filen skickas in via Studentklienten som '*Assignment #2*'.
- *Uppgiften ska lösas med rekursion*

Skriv funktionen `count_occurrences_rec(values, target)` som får in en lista av strängar och tal. Funktionen ska returnera antalet element i `values` som har värdet `target`. Svaret som returneras ska vara ett heltal.

Notera att `target` endast ska matcha hela element i `values`. Se t.ex. första exemplet nedan där inga fall av strängen `'e'` hittas eftersom `'e'` endast förekommer som en delsträng i `'hej'`.

Uppgiften skall lösas med rekursion och det är inte tillåtet att använda t.ex. list-metoden `count`.

Exempel

```
>>> count_occurrences_rec([1, 2, 3, 3, 'hej', 2, 'h', 0.4, 2, '2'], 'e')
0
>>> count_occurrences_rec([1, 2, 3, 3, 'hej', 2, 'h', 0.4, 2, '2'], '2')
1
>>> count_occurrences_rec([1, 2, 3, 3, 'hej', 2, 'h', 0.4, 2, '2'], 2)
3
```

Uppgift 3 - Dictionaries och nästling

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

Ett dictionary används för att lagra information om en läsgrupps lästa böcker. Detta dictionary innehåller boktitlar som nycklar vilka har varsitt inre dictionary som värde. Dessa inre dictionaries innehåller antalet sidor boken har under nyckeln `'pages'` samt vilka personer som läst boken under nyckeln `'read_by'`. Exempelvis:

```
cs_reading_group = {
    'The Wizard Book': {
        'pages': 657, 'read_by': ['Alan', 'Grace']
    },
    'The Dragon Book': {
        'pages': 1040, 'read_by': ['Margaret', 'Grace']
    },
    'The Dinosaur Book': {
        'pages': 896, 'read_by': ['John', 'Alan', 'Margaret']
    },
    'The Cinderella Book': {
        'pages': 500, 'read_by': ['Alan', 'Grace', 'John']
    }
}
```

(Om du av något skäl inte kan kopiera från denna pdf så finns samma dictionary i den givna filen `cs_reading_group.py`.)

Skriv funktionen `total_pages_read(reading_log, person)` som givet ett dictionary enligt ovan returnerar det totala antalet sidor som en person har läst. Om personen inte läst några av böckerna returneras 0 (se sista exemplet nedan).

Exempel Givet `'cs_reading_group'` enligt ovan:

```
>>> total_pages_read(cs_reading_group, 'Alan')
2053
>>> total_pages_read(cs_reading_group, 'Margaret')
1936
>>> total_pages_read(cs_reading_group, 'Daniel')
0
```