Uppdelning av kod och arbetsprocesser

bottom text

Exempel på ett problem:

Från övningsprov i Prog1

Summera talen mellan två positioner, `start` och `end` i listan `numbers` nedan,
$0 \leq$ `start` $\leq$ `end` $<$ `len(numbers)`.

```
numbers = [2, -6, 9, -1, -4, 2, 8, 5, 3, -7, 8, -7, -9, -10, 0, -3, 8]
```

Skriv ett program som hittar `start` och `end` som ger den största summan.

Exempel: Mellan index 1 och 3 är summan (-6) + 9 + (-1) = 2

Vad behövs?

Diskutera kort i par, dela sen era idéer för klassen!

Våra tankar:

- Funktion för att summera tal mellan två index
- Funktion för att iterera över olika intervall
- Variabel som håller reda på största summa hittills
- Main-funktion som kör allt

- Bonus: funktion som genererar listor med tal
  - Kan användas för att testa ens program mer ordentligt

Discordbot - API

Vi vill ha info om avgångar från SL. Vi behöver:

- Hitta relevanta API-er (i detta fall två)
- Ta fram siteID för stationen med ‘SL Platsuppslag’ (funktionen sl_siteID)
- Ta fram avgångar med ‘SL Realtidsinformation 4’ (funktionen sl_departure)
- Returnera en rimlig sträng från sl_departure

```
def sl_siteID(message):
   stationSearch = message.content.split(' ')
   if len(stationSearch) < 2:
      return 'need station!!!'

   else:
      s = ''
      for i in range (1,len(stationSearch)-1):
         s = s + stationSearch[i] + '+'
      s = s + stationSearch[-1]

      stationUrl = 'https://api.sl.se/api2/typeahead.json?key=4c123e93d7f147039fac4c15e0356257&searchstring=' + s
      stationRaw = urllib.request.urlopen(stationUrl)
      stationJson = json.loads(stationRaw.read())
      stationID = stationJson['ResponseData'][0]['SiteId']

      return stationID

def sl_departure(stationID):
      departureUrl =
'https://api.sl.se/api2/realtimedeparturesV4.json?key=10a7d32e89ec413994ee70661fa72a91&siteid=' + stationID +
'&timewindow=60'
      departureRaw = urllib.request.urlopen(departureUrl)
      departureJson = json.loads(departureRaw.read())

      departureDestination = departureJson['ResponseData']['Metros'][0]['Destination']
      departureTime = departureJson['ResponseData']['Metros'][0]['ExpectedDateTime'][11:]
      departureName = departureJson['ResponseData']['Metros'][0]['StopAreaName']

      responseString = 'next metro from ' + departureName + ' departs at ' + departureTime + ' towards ' +
departureDestination
      return responseString
```

Discordbot - Botgrejer

Vi vill ha en bot som:

- Reagerar på ett specifikt kommando (::SL)
- Läser en del av kommandot (stationsnamn)
- Skickar stationsnamnet till första API-funktionen (sl_siteID)
- Tar strängen från andra API-funktionen (sl_departure)
- Svarar användaren

```python
@client.event
async def on_message(message):
    if message.author == client.user:
        return

    if message.content.startswith('::'):
        await message.channel.send('*goblin noises* \n')
        if 'help' in message.content:
            await message.channel.send('::help displays this message \n'
                                       '::goblin tells goblin lore \n'
                                       '::todo <number> shows dumb API test shit \n'
                                       '::SL <station> helps your journey \n'
                                       '::SMHI displays stockholm weather \n')
        if message.content == '::goblin':
            await message.channel.send('i am goblin')

        if message.content.startswith('::todo'):
            response = todo(message)
            await message.channel.send(response)


        if message.content.startswith('::SL '):
            siteID = sl_siteID(message)
            response = sl_departure(siteID)
            await message.channel.send(response)

        if message.content.startswith('::SMHI'):
            response = smhi()
            await message.channel.send(response)

helpFile = open("readme.txt", "r")
token = helpFile.readline()
client.run(token)
```

PROJEKTSPEC

Namn på gruppmedlemmar Skola
Kurs Datum

Specifikation till uppgift XX

Skriv här en kort beskrivning av vad programmet ska göra.

Funktioner

Under den här rubriken skriver ni ner vilka funktioner som programmet kommer behöva. Om det är möjligt är det bra att redan här specificera varje funktions parametrar och vad de ska returnera.

Exempel:

```
# den här funktionen summerar alla tal i listan mellan index startIndex och slutIndex
def interval_sum(lista med int:s, startIndex, slutIndex):
   ...
```

```
# den här funktionen konkatenerar alla strängar i listan
def concat(lista med str:s):
   ...
```

```
# den här funktionen använder SL:s API för att returnera det site ID som bäst motsvarar söksträngen station
def getSiteID(station):
   ...
```

```
# main-funktionen kör programmet
def main():
   ...
```

Datastrukturer

Under den här rubriken skriver ni vilka datastrukturer ni tänker att programmet kan behöva använda sig av. Exempel på datastrukturer är listor, dictionaries, tupplar och tekniskt sett även strängar. Relevant här är både datastrukturer som ni själva introducerar och sådana som "tvingas på er" av API:s eller liknande. Använd gärna bilder för att visualisera hur datastrukturerna ser ut, eftersom det då blir tydligare hur de ska användas.

Exempel:

- "För att använda SL:s API behöver vi hantera JSON-filer"
- "JSON-filer tolkas i Python som antingen listor eller dictionaries"
- "När vi söker efter ett siteID behöver vi en söksträng"
- "Vi kan lagra alla svar vi får från API:t i en lista"

Algoritmer

Här skriver ni ner algoritmer som kommer behövas i programmet. Det kan till exempel vara en algoritm för att plocka ur en relevant del ur en JSON-fil, en algoritm för att välja ett specifikt element ur en lista eller något annat.

Namn på gruppmedlemmar Skola
Kurs Datum

Exempel:

Algoritm för att hitta den sekvens vars summa är störst, från en lista av integers:

1. Initialisera variablerna *start* och *end* till 0 och *max* till värdet av listans första element.
2. Gå igenom varje möjligt startindex *i*, som går från 0 till1 mindre än längden av listan.
3. För varje möjligt startindex *i*, gå igenom varje möjligt slutindex *j*. *j* kan alltså gå från *i* till 1 mindre än längden av listan.
4. Summera alla tal i intervallet. Om summan är större än *max*, sätt *max* till det nya värdet, *start* till *i* och *end* till *j - 1*.
5. Resultatet finns i variablerna *start* och *end*.

Tidsplan

Tanken med att göra en tidsplan är att fundera över i vilken ordning saker ska göras och att uppskatta hur lång tid varje steg tar. Det är en bra övning att göra för att kunna bedöma om er projektidé är genomförbar eller inte. Ni kan dela upp tidsplanen i veckor, dagar, eller något annat som ni tycker är relevant.

Exempel:

Vecka 7 (3 h):
Under den här veckan ska vi studera våra API:s och fundera över vilka datastrukturer vi kan behöva använda för att plocka ut relevant data. Vi ska även få GitHub att fungera.

Vecka 8 (3 h):
Den här veckan ska vi skriva funktionerna.....