UPPSALA UNIVERSITET

Avrundning OU3, testning, OU5

Carl Nettelblad

2020-02-24

Info

- Tack för mittkursvärderingen!

- Konkreta åtgärder
  - Flera korrekturfixar på sidorna
    - Framför allt fixar i minilektionerna
  - Påverkade planering föreläsningarna
    - Ska försöka ta det lugnare för nytt material

Några tankar

- Invändningar mot repetition på föreläsning
  - Om det känns lätt andra gången beror det på att det har satt sig
  - De saker jag valt att repetera har varit kärnpunkter jag ser att många fortfarande missar konkreta i uppgifterna

UPPSALA UNIVERSITET

Om uppgifterna

- Kommentar i utvärderingen ungefär ”är målet att lösa kluriga problem eller använda Python”
  - Det *är båda*
    - Ett av kursmålen ”analysera och lösa problem med hjälp av programmeringskonstruktioner”
  - Även om ni tittar på svaren på övningarna tidigare i lektionen, försök tänka tillbaka hur ni hade kunnat komma på det själva

Tid

- Om man går på alla labbpass, inte bara vid deadline, finns gott om tid för hjälp
  - Jag har själv varit i någon sal på en majoritet av alla pass
  - Större delen av tiden är listan för att få hjälp eller att redovisa tom
  - Har du kört fast med OU4, kör du fast med OU5, kom och be om hjälp
    - Eller maila, i andra hand

Tid

- Om man är en van programmerare går varje lektion fort
- Våra tidsuppskattningar har verkat stämma hyfsat *i medeltal*
- Men avsikten är inte att labbpassen ska räcka för att göra alla lektionerna
  - En termin på 30 hp är 20 veckor heltid
  - Ni har ca 60 h schemalagt i labb, 14 h föreläsningar
  - 800 h
  - Uppgifterna är 3 hp, 80 h
    - Men av tentans 2 hp, 50 h till, borde mycket tid ägnas åt lektionerna

UPPSALA UNIVERSITET

OU3

- Varför läsa från fil?
  - Du har använt massvis av program
  - Det normala är att man inte behöver ändra i koden för ett program för att köra det på nya data
- Stäng dina filer
  - Gärna med `with`
  - Om du skriver inte säkert att innehållet finns där
  - Operativsystem sätter gränser för antal öppna filer
    - Några hundra till hundratusen
    - Men tänk om ditt program ska läsa igenom alla filer på hårddisken (tiotals miljoner)

En gång

- Läsa filer är krävande
- Gör det inte i onödan

- Kod blir också ofta tydligare om man kan följa flödet

- Vad vill vi göra?
  - Läs varje rad
  - Skriv ut raden med radnummer
  - Ta bort kommentar
  - Håll reda på vilka ord som förekommer, utan stoppord
  - Vi ser att resultatet ser ut som ord som nycklar och listor med radnummer som värden, så vi skapar ett sådant lexikon

UPPSALA UNIVERSITET

Kompakt

```
import keyword
import re
ref = {}
stoppord = set(keyword.kwlist)
…
for i, line in enumerate(fil, start=1):
    print('{i:2d}{line}', end=’’)
    line = re.sub(r'#.*$', '', line)
    for w in re.finditer('[a-zA-ZåäöÅÄÖ]+', line):
        if w in stoppord:
            continue
        ref.setdefault(w, []).append(i)
```

UPPSALA UNIVERSITET

Förklaring

- **`set`** är som ett lexikon med bara nycklar
  - `in`-operatorn blir mer effektiv än för bara en lista
- **`enumerate`** ger oss tupler med radnummer och rader från fil som bara ger rader, packas upp till `i` och `line`
- **`finditer`** räknar upp träffarna utan att först lägga dem i en lista, perfekt för `for`
- **`continue`** används för att gå vidare till nästa steg i loopen
  - Vanligt att man kan ha flera olika villkor i början, kan vara tydligare än `not in`
- **`setdefault`** sätter ett värde om nyckeln inte redan finns, returnerar värdet efteråt (gamla eller nya)
  - Lägg in en tom lista och gör sedan `append` på den lista som finns

Vad blir fördelen?

- Om vi vill se vad som händer med varje rad räcker det med att läsa den loopen
- Jämför med:

```
linelist = []
for line in fil:
        # Bygg upp linelist med varje rad

# Filtrera bort kommentarer på alla rader till linelist2

# Dela upp linelist2 till ord i wordlist

# Gå igenom wordlist och skapa wordlist2 utan stoppord

# Stoppa in radnummer från wordlist2 i ref
```

UPPSALA UNIVERSITET

Vad vill man *inte* göra?

- Först identifiera alla unika ord
- Sedan gå igenom alla rader och se vilka av de orden som finns på varje rad

```
for w in wordlist:
        linenums = []
        for i, line in enumerate(lines, start = 1):
                if w in line:
                        linenums.append(i)
        print(f'{w:16}{linenums}')
```

UPPSALA UNIVERSITET

Problem

- Vi har inte splittat raden i ord, så ordet ”ko” anses finnas på en rad som innehåller ”skola”
- Lätt fixat!

```
for w in wordlist:
    linenums = []
    for i, line in enumerate(lines, start = 1):
        wordlist = re.findall('[a-zA-ZåäöÅÄÖ]+', line)
        if w in wordlist:
            linenums.append(i)
    print(f'{w:16}{linenums}')
```

UPPSALA UNIVERSITET

Större problem

- Tänk om vi har 10 000 unika ord på 100 000 rader
  - Hur många gånger gör vi `findall` totalt?
    - En miljard!!!
    - Om varje tar en mikrosekund får vi ändå vänta i mer än en kvart (1000 sekunder = 16 2/3 minut)
  - Ändå inte enormt lång
  - ”Rätta” lösningen gör 100 000 sökningar
    - 0,1 sekund om varje tar 1 µs

UPPSALA UNIVERSITET

`read` eller `readlines` eller `for line in fil`

- Samma princip gäller att läsa in filen
- Python kan få problem med riktigt långa strängar eller listor
  - De tar plats i minnet
- Om vi inte har många olika unika ord skulle vårt program lätt kunna läsa in en fil på 20 GB
  - Ungefär storleken på hela engelska Wikipedia
- `for line in fil:` läser en rad i taget
  - Gott om plats
- `readlines` skapar en lista med alla rader, read en sträng med hela filen
  - Hela den listan/strängen måste få plats samtidigt
  - Har din dator mindre än 20 GB RAM kommer det garanterat att gå väldigt långsamt, om det går alls

UPPSALA UNIVERSITET

Men…

- Våra exempelfiler var inte så långa
- Man kan alltid bygga program för att klara av extrema fall
  - Tar ofta onödigt lång tid, onödigt komplicerat
- I det här fallet är den effektiva, ”skalbara” lösningen också enklare och tydligare
  - Undvik dåliga vanor!
  - Du blir inte sjuk direkt om du låter bli att tvätta händerna
  - Egentligen behöver du bara tvätta händerna om du faktiskt är smutsig
  - Vad är enklast? Att hålla reda på om du faktiskt blivit smutsig med något som ger risk för kontaktsmitta, eller att tvätta dig?
  - Att ha onödigt många loopar i varandra, eller att läsa in en godtyckligt stor fil på en gång, är *nästan aldrig* bra

UPPSALA UNIVERSITET

part2

```
def part2(e):
    return e[1]
```

- Vad gör `part2`?
- Vad returnerar:

```
part2([3,6,9])
part2('Hejsan')
part2(('Hej', 5))
part2('Hej',5)
part2((('Hej', 5)))
```

- Vad `part2` gör beror på typen och innehållet i `e`
- `part2` sorterar *inte*

UPPSALA UNIVERSITET

sorted

- Vad gör `sorted(lex.items(), key=part2)`?
- `key` är en namngiven parameter, ur dokumentation:

Key Functions ¶

Both `list.sort()` and `sorted()` have a *key* parameter to specify a function to be called on each list element prior to making comparisons.

For example, here's a case-insensitive string comparison:

```
>>> sorted("This is a test string from Andrew".split(), key=str.lower)
['a', 'Andrew', 'from', 'is', 'string', 'test', 'This']
```

The value of the *key* parameter should be a function that takes a single argument and returns a key to use for sorting purposes. This technique is fast because the key function is called exactly once for each input record.

A common pattern is to sort complex objects using some of the object's indices as keys. For example:

```
>>> student_tuples = [
...     ('john', 'A', 15),
...     ('jane', 'B', 12),
...     ('dave', 'B', 10),
... ]
>>> sorted(student_tuples, key=lambda student: student[2])   # sort by age
[('dave', 'B', 10), ('jane', 'B', 12), ('john', 'A', 15)]
```

UPPSALA UNIVERSITET

Vad gör `part2`?

- `part2` returnerar det andra elementet (element nummer 1) för sin parameter `e`
- När den används som sorteringsnyckel med tupler från ett lexikon innebär det att sortera på värdet
  - `sorted` anropar `part2` i varje steg när två element ska jämföras och använder returvärdet i stället för elementen själva för att avgöra ordningen

UPPSALA UNIVERSITET

Problemlösnings-strategier

- Be om hjälp om du kör fast
  - Funkar på kurs, funkar ofta i verkligheten också!
  - Funkar inte på tentan…
- När något går fel:
  - Printa det du kan, eller kör debugger
    - Eller ”kör koden i huvudet”
  - Kontrollera värden
    - ”Min loop är konstig” – vilka värden får variablerna i varje iteration av loopen
  - Skriv ut *typer* för värden
  - Läs felmeddelandet
    - Finns det något i det som står som du förstår? Vad får du om du googlar det? Vilka rader nämns?
    - Vad står på raden ovanför?

UPPSALA UNIVERSITET

Problemlösnings-strategier

- När man kör fast beror det oftast på att det man tror händer, det man tror att man förstår, inte stämmer.
- Att skriva ut värden och typer hjälper till att kontrollera vad som stämmer.
- Att själva försöka ”köra koden i huvudet” gör samma sak (skriv ned det du inte håller i huvudet). Kan tvinga fram en förståelse som missas av att bara följa med i datorns körning.
  - Ditt huvud finns med på tentan…
  - Eller om man diskuterar möjliga idéer innan man kodar dem.

Problemlösnings-strategier

- Att förklara för en assistent vad som går fel hjälper ibland. När man sätter ord på vad som händer ser man misstaget.
- Ibland fungerar det att förklara för en kompis.
- Ibland fungerar det att förklara för sin katt, eller en badanka. Allt som tvingar dig att inte titta på exakt vad som står och tänka igenom det.
- Begrepp ”rubber ducking”, även för erfarna programmerare.

När man inte ens har något som gör fel…

- ”Hur ska man göra det här”
- Försök dela upp i steg
  - ”För att göra det här kommer jag att behöva det här”
  - Prova att sitta i ett anteckningsdokument eller på papper
  - Skriv upp allt du behöver göra för att få fram resultatet

Den tomma filens tyranni

- Skriv kod som gör *något*
  - Läser in filen, hittar ord, filtrerar bort stoppord ur en lista
  - Lättare att se vilka bitar som fastnar i ett pussel när man börjat lägga det
  - Men man måste bara beredd att flytta eller ta bort delar som inte alls stämmer
- Få ihop kod som gör *något*, skriv sedan om den för att göra *rätt* och göra det på ett snyggt och begriplig sätt

UPPSALA UNIVERSITET

`assert`

- I OU2 kanske vi ville kolla att `n` till `smooth_a` var icke-negativt
- Kan förstås göra med `if`
  - Vad gör vi om det är så?
    - `print`?
    - Rapportera fel
      - Vilket fel?

UPPSALA UNIVERSITET

assert

- Tänk om vi bara kunde säga saker vi räknar med ska stämma
  - Som vilken sorts värden vi får in
  - Rapportera på ett enhetligt sätt om något sådant villkor inte uppfylls

```
def smooth_a(lista, n):
    assert n >= 0
```

UPPSALA UNIVERSITET

Fördelar

- Rapporteras alltid som ett `AssertionError`
- Den som *läser* koden förstår att det är något som ska gälla
- En del verktyg kan markera om man anropar funktionen på ett sätt som uppenbart bryter mot det

UPPSALA UNIVERSITET

Regressionstest

- Jämför två versioner av ”samma” kod
  - Vi använde för att hitta fel i `isprime_fast` för några föreläsningar sedan

```
def test_primefast():
      for i in range(100):
            assert isprime(i) == isprimefast(i)
```

UPPSALA UNIVERSITET

pytest

- pytest är ett elegant sätt att köra igenom testfall
- I enklaste läget, leta upp alla kodfiler som innehåller funktioner med vissa namn

```
python3 -m pip install pytest
python3 -m pytest -q
```

- Installera modulen `pytest`
- Kör modulen `pytest` i sökläge

UPPSALA UNIVERSITET

Köra pytest

```
(base) cnettel@IT-WL-CARNE864:/mnt/c/Users/Carl Nettelblad/Box Sync/prog1$ python -
m pytest -q
F                                                                    [100%]
================================== FAILURES ===================================
_____________________________ test_primefast ______________________________

    def test_primefast():
        for i in range(100):
>               assert isprime(i) == isprimefast(i)
E     assert False == True
E      +  where False = isprime(4)
E      +  and   True = isprimefast(4)

test_prime.py:23: AssertionError
1 failed in 1.10s
(base) cnettel@IT-WL-CARNE864:/mnt/c/Users/Carl Nettelblad/Box Sync/prog1$ python -
m pytest -q
.                                                                    [100%]
1 passed in 1.35s
(base) cnettel@IT-WL-CARNE864:/mnt/c/Users/Carl Nettelblad/Box Sync/prog1$
```

UPPSALA UNIVERSITET

Kommentar

- Vår kod testar bara olika värden och säger med `assert` att resultatet ska vara lika
- `pytest` plockar fram det fall där vår assertion misslyckades och berättar i vilken funktion det var, vilka värden som testades och vad resultaten blev
- Lätt att skriva en hel *testsvit*
  - Många olika tester
  - Skriv `assert` för allt du vet ska hända
  - Fixar du en bugg? Skriv nya test med `assert` som testar att rätt saker händer i den fixade versionen

UPPSALA UNIVERSITET

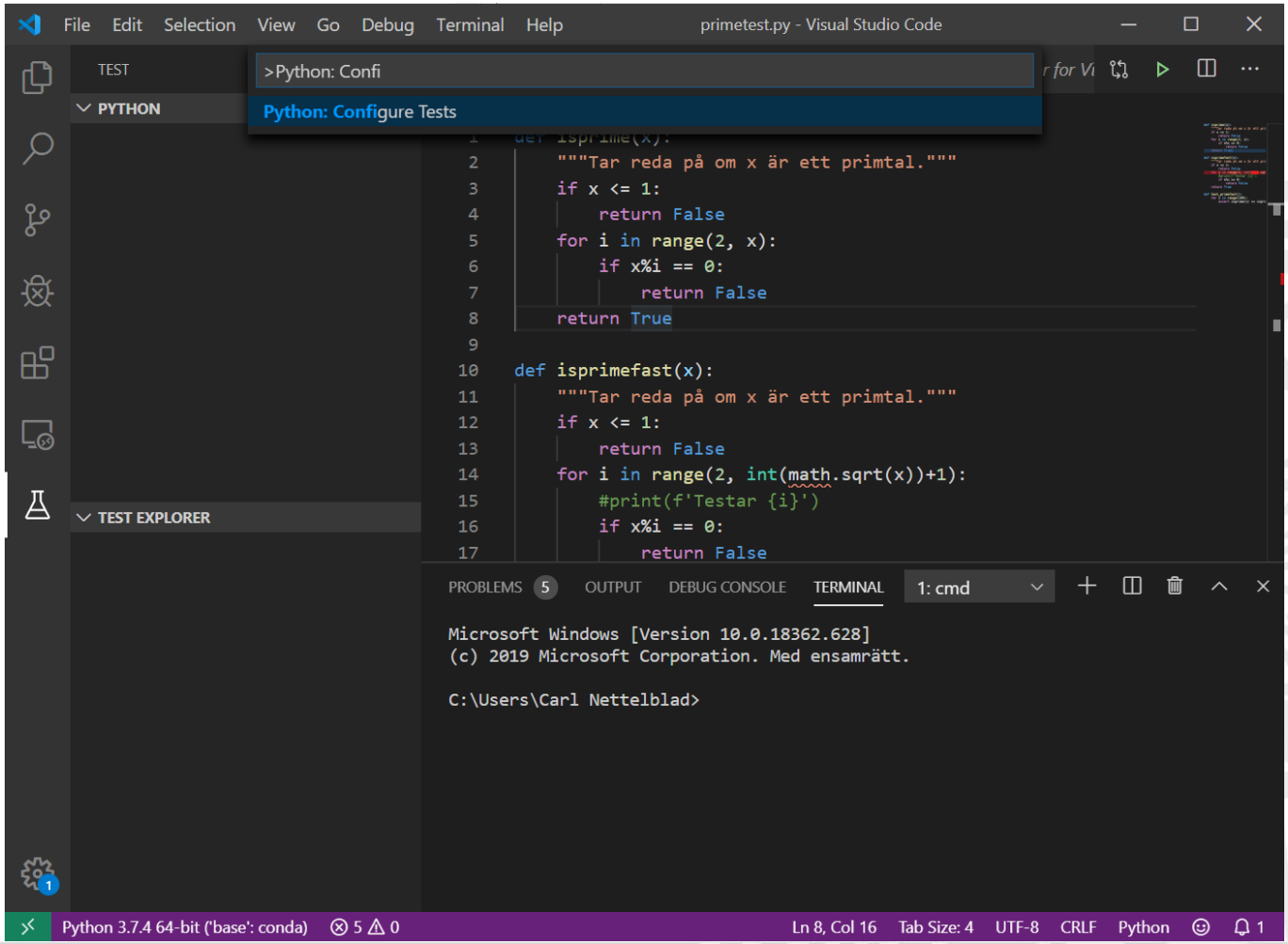
Testdriven utveckling

- Skriv testen först!
- ”Jag tänker mig att jag har en funktion som gör det här och ger de här resultaten i de här fallen”
  - Skriv kod som faktiskt anropar den där tänkta funktionen i form av tester
  - Skriv funktionen steg för steg
  - Klar när alla testerna lyckas!

UPPSALA UNIVERSITET

I Visual Studio Code

- Ctrl+Skift+P
- Python: Configure Tests

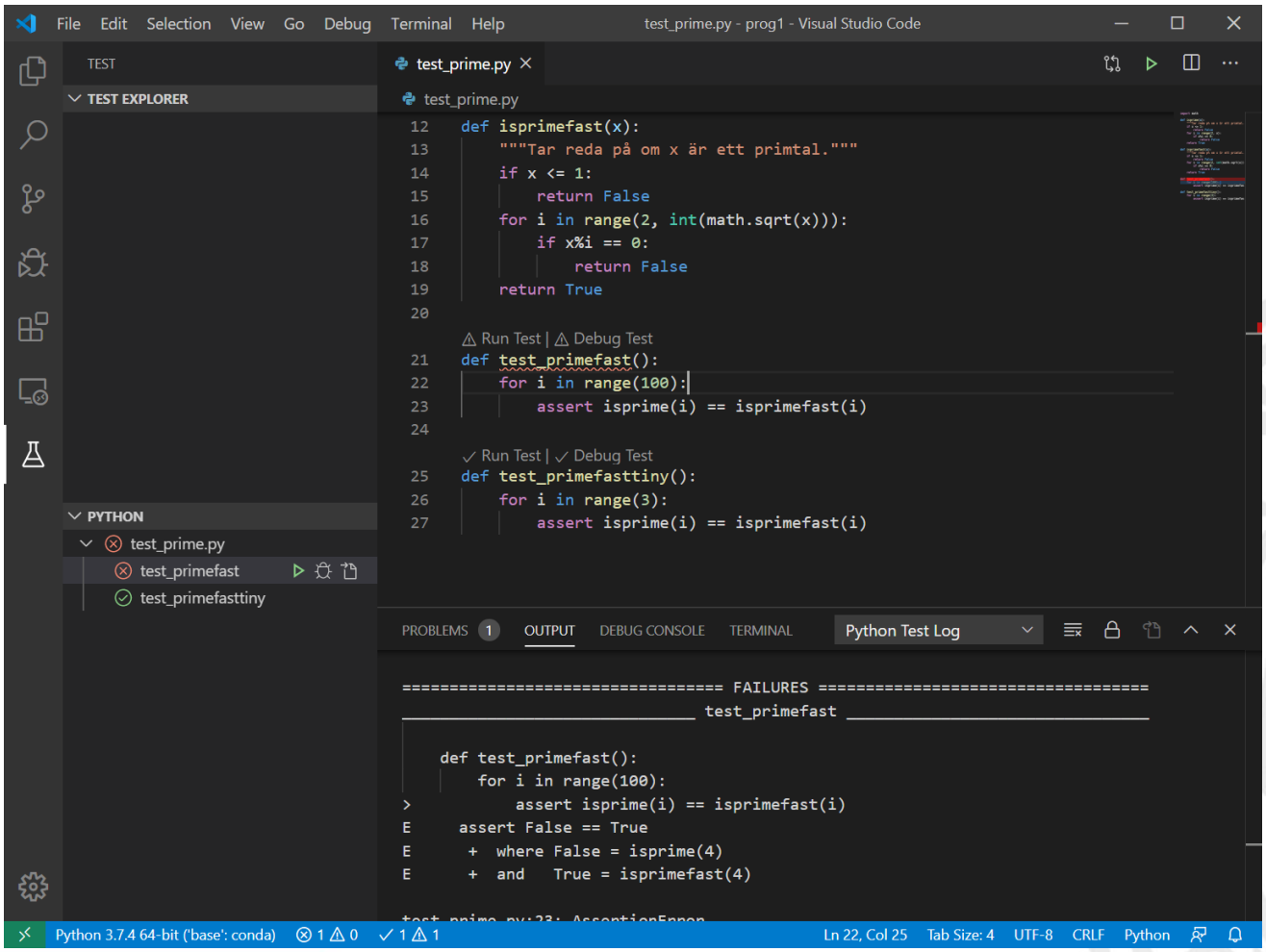


Testning i Visual Studio Code

- Om Configure Tests inte fungerar kan man under File, Preferences, Settings söka på `python.testing.pytestEnabled` och kryssa för
- Öppna en hel mapp (File -> Open Folder, inte bara en fil)
  - Tester i filer med namn som börjar på test dyker automatiskt upp i testvyn

UPPSALA UNIVERSITET

Testvyn



UPPSALA UNIVERSITET

matplotlib

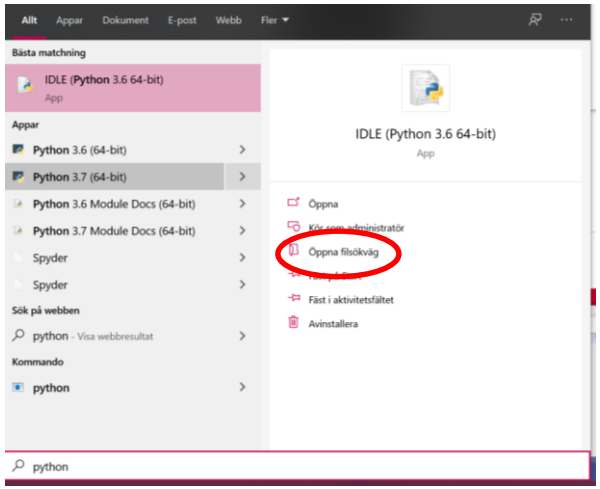
- Mycket populärt paket för att generera figurer i Python
- Ingår *inte* i standard-Python
  - Finns med i Anaconda
  - Finns i datorsalarna
  - Kan ofta installeras med pip
- Möjligt exempel för att installera:

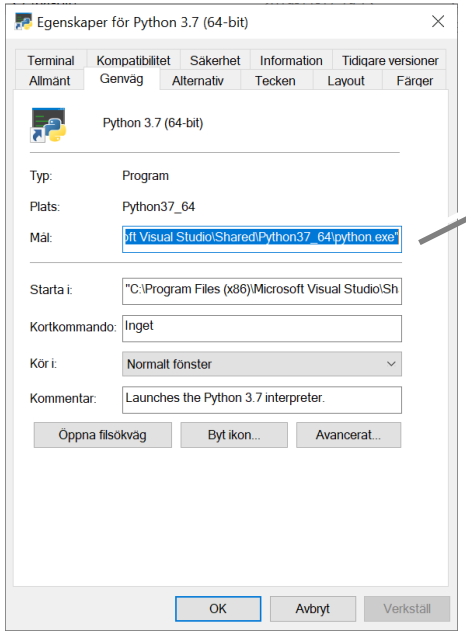
  ```
  python3 -m pip install matplotlib
  ```

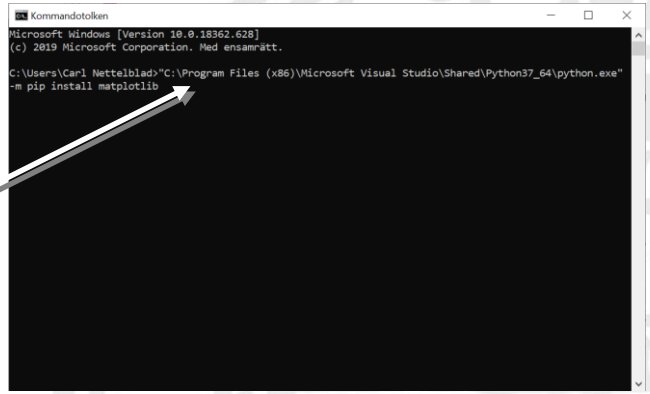
  från terminalen, inte inne i Python

`matplotlib` i Windows

- I Anaconda finns `matplotlib` färdigt, om det är korrekt installerat
- Har du Python separat? Fungerar det inte att skriva Python i terminalen?







UPPSALA UNIVERSITET

CSV i svenska program

- CSV använder komma
- Flera program för att läsa textfiler använder ditt användarkontos språkinställningar
  - Med amerikanskt språk används . för decimaler och , som avskiljare
  - Med svenskt språk används , för decimaler och ; som avskiljare
  - Excel tolkar en CSV-fil som en semikolonseparerad fil om du bara öppnar den
- Får du problem att tolka filen i Python
  - Kolla att du använder filen direkt från lektionen, inte sparad genom något annat program

UPPSALA UNIVERSITET

Skippa rubrikraden

```
import csv

with open('people.csv', 'r') as csvFile:
    reader = csv.reader(csvFile)
    for _, _ in zip(range(1), reader):
        pass
    for row in reader:
        print(row)
```

- `zip` fortsätter så länge *båda* källorna har nya element.
  - Läs en rad och gå vidare.
  - Att ange variabelnamnet _ är ett vanligt sätt att säga ”det här är skräp som jag inte använder”

UPPSALA UNIVERSITET

Större utvecklingsprojekt

- Programmering komplex på flera nivåer
- ”Hur uppnår jag det här, givet det jag har?”
  - Implementera specifik funktion, korta program
- ”Vad skulle jag vilja ha för verktyg för att kunna uppnå det jag vill?”
  - Definiera en ny klass eller hjälpfunktion utifrån visst behov
    - Se att vi kan skapa funktion `rectangle` eller klassen `Rectangle` som hjälp när vi vill rita flaggor

UPPSALA UNIVERSITET

Större utvecklingsprojekt

- ”Vad kommer jag vilja att min kod kan göra i framtiden?”
  - Definiera system med klasser där delarna kan återanvändas
  - Se till att man kan ange koordinaterna till `Rectangle`
  - Se till att `Rectangle` kan hantera olika färger
  - Skapa `filledFigure` som kan användas av både
- *Ibland* kan man svara på frågorna direkt, ibland får man arbeta iterativt (börja i någon del för att se vad som behövs)

UPPSALA UNIVERSITET

Designa klasser

- Klasser kan motsvara mängder av verkliga objekt (eller objekt på skärmen)
  - Verkligt som i ”finns utanför programmet”, som en fil
  - Ursprunglig motivering för objektorientering
- Användbart på fler sätt
  - Du kan inte gå ut i verkligheten och hitta en `csv.reader`
  - Dela upp det vi gör i separata enheter med separata uppgifter blir det viktiga
    - ”Det här är en egen sak som tar strängar från till exempel en fil och delar upp dem i kolumner”
- Bygg vår egen verklighet i programmet och gör den begriplig!

UPPSALA UNIVERSITET

Hur designa?

- Top down
  - Utgå från krav/behov
  - Omsätt till mindre och mindre delar
    - ”För att göra W behöver vi delar som gör X, Y, Z”
  - ”Waterfallmodell”, vattnet rinner uppifrån och ned och när man beskrivit allt som text kommer man till botten
    - ”Bara” att skriva koden som gör det
    - *Stor* risk för ”tänkte inte på det”
    - Att skriva och köra kod ger nya insikter i sig

UPPSALA UNIVERSITET

Hur designa?

- Bottom up
  - Börja i någon tydlig ände, få den att fungera
  - Testa ofta
  - Utgå från det du redan gjort när du gör nästa del
  - Ingen fullständig design innan man kodar
  - Mindre ”elegant” arkitektur i slutänden

- Modernt i dag
  - Arbeta ”agilt”
  - Små team som levererar i en form av bottom-up
  - ”*Ingen* vet vad som behövs innan det är byggt”
  - Helst: Var beredd att slänga bort och börja om när något inte blev bra…

UPPSALA UNIVERSITET

Simuleringar

- Bland de allra första syftena med datorer var vapenberäkningar
  - Sikta artilleri givet position för vapen och mål samt väderförhållanden
  - Förutse effekterna av kärnvapensprängningar
  - Enkla ekvationer, komplexa beteenden
- Även ett av de första syftena med objektorientering
  - Beskriv en ”värld” med objekt
  - Se och styr vad som händer i världen
  - Inte enkla differentialekvationer längre, komplexa och flexibla samband

Generella frågor för simuleringar

- Hur representerar vi ”tid” i vår simulering?
- I vilken ordning ska saker ske?
- Hur hittar olika objekt varandra?

- Vad är syftet med vår simulering?
  - Vilka beteenden vill vi studera?
  - Vilka delar av verkligheten?
  - Vad vill vi kunna mäta/visualisera? Ska det gå att spåra ett enskilt objekt genom simuleringen (en atom, en person, ett fordon, en mutation)

UPPSALA UNIVERSITET

Typiska lösningar

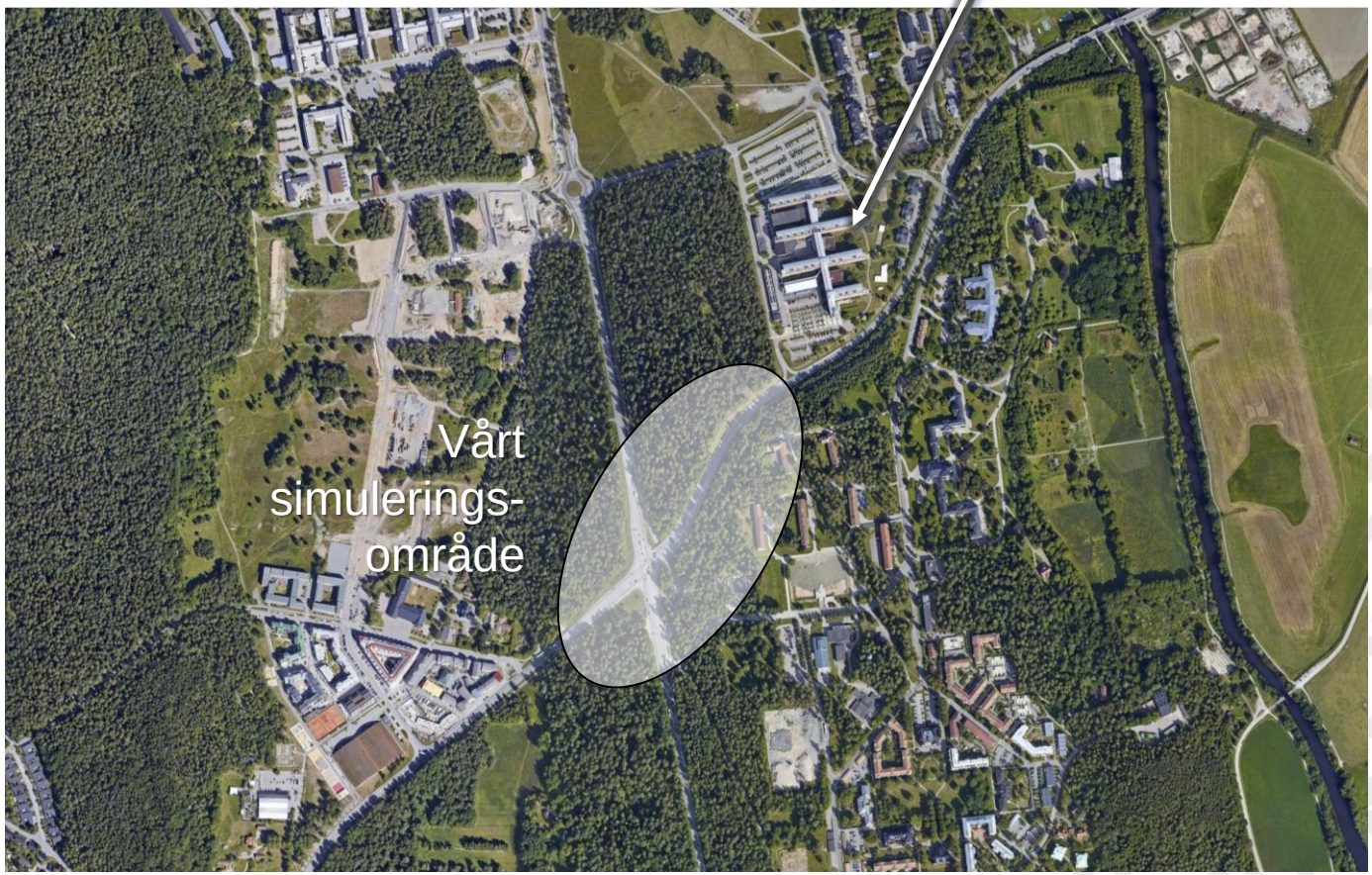
- Ha en ”motor”, en klass som representerar ”hela världen”
- Världen har referenser till alla objekt
  - Som variabler, lexikon, listor, …
- Världen har en ”klocka”
  - Varje objekt kan ha en metod `step` som anropas för varje ”klocktick”
  - Fungerar bra så länge alla saker händer på samma tidsskalor
    - Svårt att simulera trafikkorsningar (sekundskala) och kemiska reaktioner (nanosekundskala) i samma simulering

UPPSALA UNIVERSITET

Simulera världen

- Du kan inte beskriva hela världen
- I vårt fall:
  - Vi vill förstå effekterna av olika sätt att reglera trafikljus och lägga filer i en korsning
    - Hur länge får man vänta?
    - Hur långa köer blir det?
- Vad avgör det?
  - Vilka fordon som kommer in i korsningen i vilken ordning
  - Vilka filer finns det
  - Trafikljus för varje fil

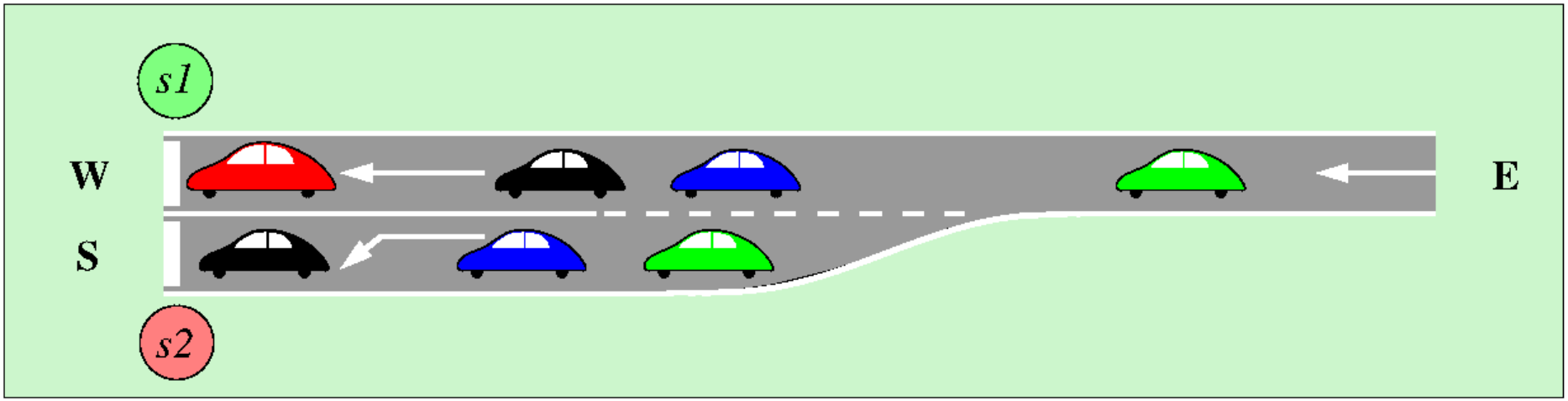
Verkligheten



Själva korsningen

Vårt problem

- Förenklad bild



- Hur påverkas trafikflödet av hur lång filerna är, inställningar på trafiksignalerna?
- Anta att alla fordon håller konstant fart med konstanta avstånd
  - Kan bromsa accelerera på så kort tid att den är irrelevant på vår tidsskala

UPPSALA UNIVERSITET

Beskriva världen

- En fil/körfält (`Lane`)
  - Har en längd i termer av antal bilar som får plats
  - Alla bilar håller farten 1 plats/tidssteg (om tomt framför)
- Ett fordon (`Vehicle`)
  - Ett fordon ”vet” vart det vill. Systemet ser till att det styrs till rätt fil. Varje fil håller reda på vilka fordon den innehåller, i vilken ordning.
- En fil är enkelriktad och når en punkt med en/flera ny filer, eventuellt med trafiksignaler
  - Filen tar hand om att flytta sina fordon framåt

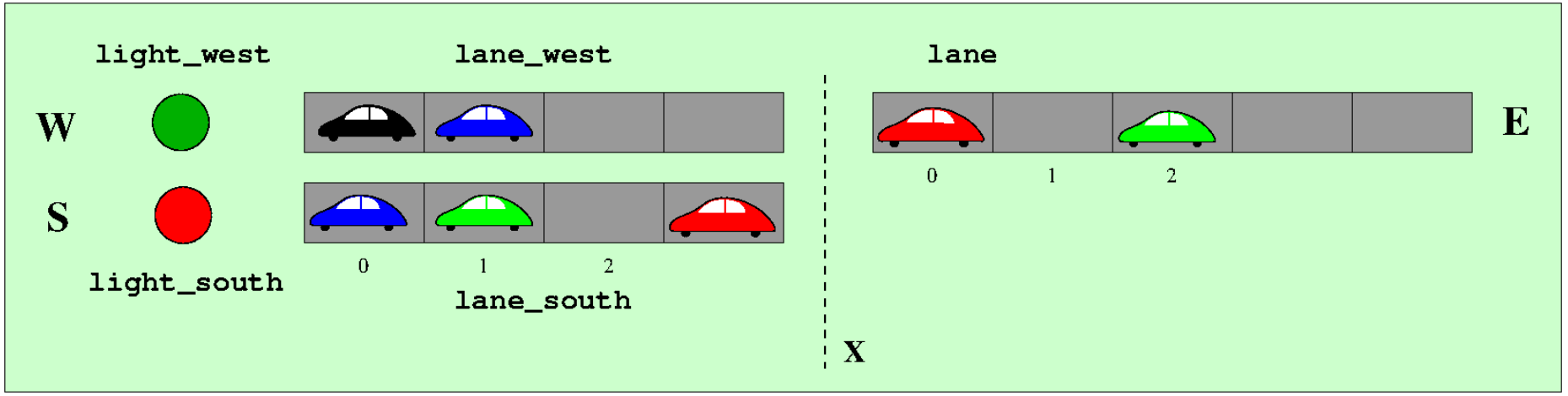
Trafiksignaler

- ”Dumma” trafikljus (`Light`)
  - Ingen dynamisk anpassning efter klockslag ellet trafikflöde
- Växlar regelbundet rött/grönt (inget gult)
- Total *period*
  - Antal tidssteg grönt + rött
- *Grönperiod*
  - Antal tidssteg i början av perioden då trafikljuset lyser grönt

UPPSALA UNIVERSITET

Exaktare modell

- Klassen `TrafficSystem` ska knyta ihop olika filer och ljus



UPPSALA UNIVERSITET

Koden

- Skal till klasserna `Lane`, `Vehicle`, `Light` i filen `trafficComponents.py`
  - Finns separata specifikationer för varje klass
  - Kort demokod som testar implementation
- Klassen `Destinations` (som genererar mål för nya fordon) *färdig*
- Skal till klassen `TrafficSystem`
  - I en egen fil, ska importera `trafficComponents` och `destinations`

UPPSALA UNIVERSITET

Skriva egna moduler

- När vi importerar en modul
  - Med `import x`
  - Eller `import x as y`
  - Eller `from x import klassnamn`
  - Eller `from x import *`
- …kommer Python att först leta efter en `.py`-fil med samma namn i den katalog där programmet finns
  - Läsa in och köra den filen för att få alla definitioner

UPPSALA UNIVERSITET

All kod körs

- All kod som inte finns i en funktion körs när modulen importeras
- Det vill vi oftast inte!
- Vi vill att *olika* saker ska hända om vi kör ett program eller importerar det som en modul

- Lösning:
  - Kolla om du körs som huvudprogram

```
if __name__ == "__main__":
```

- Demokoden i `trafficComponents` finns i ett sådant `if`-block

UPPSALA UNIVERSITET

Egna klasser

- I kodskelettet ser vi de saker vi gått igenom om egna klasser
- `class KlassNamn:`
- Magiska metoder `__init__`, `__str__`, `__repr__` för att initiera och representationen med `str` respektive `print`
- Egna metoder, som `enter` i `Lane` som lägger in en ny bil längst bak

```
class Lane:
    …
    def enter(self, v):
        """…
```

UPPSALA UNIVERSITET

Tidssteg

- `TrafficSystem` håller ihop världen, anropar step för alla delar
- Logiken som ”kopplar ihop” delar i `TrafficSystem`
- Börja ”framifrån”
  - Signaler skiftar färg
  - Fordon passerar (gröna) signaler
  - Fordon rör sig framåt i sin fil
  - Ett fordon längst fram i en fil som övergår i flera filer flyttas till rätt ny fil
  - Fordon tillkommer från `Destinations` och läggs i kön
  - Om det finns plats i filen flyttas eventuella fordon från kön

UPPSALA UNIVERSITET

Statistik

- Vi vill veta många saker om trafiksystemet:
  - Genomsnitt, median och maximal tid från att fordon skapas tills de passerar (rätt) trafikljus
  - Antal fordon som skapats total, är i systemet just nu, kört ut genom respektive fil
  - Andel av tidsstegen då ett fordon inte kunnat köra in i rätt trafikljusfil för att den varit full, andel tidssteg då kön med nya fordon från `Destinations` inte varit tom
  - Strukturera det här snyggt som information som lagras i `TrafficSystem`
    - Vilken information behövs? Vad kan beräknas?
- `printStatistics` ska redovisa nuvarande statistik
- `snapshot` skapar en textvisualisering av systemet
  - Färg på trafikljus
  - Är systemet blockerat?
  - Mål och placering för fordon i filerna

UPPSALA UNIVERSITET

Angreppssätt

- Designen är färdig i stora drag
  - Bottom-up-utveckling utifrån färdig specifikation
  - Utveckla klasser var för sig
    - Testa var för sig
    - Med egna och färdiga tester
    - Skriv ned om något verkar oklart så ni kan gå tillbaka när ”allt sitter ihop” och se om det stämmer

      `# TODO: Check this!`
    - Fast förklara gärna vad det är som konstigt…
- Bygg upp första trafiksystemet när alla mindre klasser är färdiga
- När det är gjort, bygg upp det stora systemet

Två trafiksystem, samma klasser

- Först det enklare `trafficSystem0`
  - En enda fil och en signal
- Andra systemet det vi visat här
  - `step`-metoden kan lätt få upprepningar här
  - Kan vi representera en signal, en fil och dess statistik på ett snyggt sätt? Tupel, lista, lexikon? Metod som gör en del av arbetet åt `step` och kanske kan användas flera gånger?
- BÅDA systemen ska använda exakt samma `Vehicle`, `Lane`, `Light` i separat fil `trafficComponents.py`
- `Destinations` ska inte ändras alls

UPPSALA UNIVERSITET

Ytterligare kommentarer

- Snyggast om trafiksystemet kan ta in nya inställningar (längd på filer, perioder)
  - Initierarmetoden
  - Standardvärden för parametrar från uppgiften
- Ännu snyggare (valfritt!)
  - Tänk om samma konfigurationsfil kan användas för att beskriva båda trafiksystemen
  - Listor med filer, trafikljus och deras mål?
  - Använda `pickle` för att spara trafiksystem
- Kön med genererade fordon ska oftast vara tom
  - Om den blir längre och längre är något fel

UPPSALA UNIVERSITET

Valfri overkill

- Animera systemet med `Turtle`...
  - Ska varje klass ha en egen `paint`-metod? Vad behöver den veta?

UPPSALA UNIVERSITET

Nästa gång

- Repetition av centrala koncept
- Genomgång av övningstenta från i höstas
- Finns snart totalt tre tentor i Python upplagda
  - Samma huvudsakliga struktur, men gamla Java-tentor kan också ge exempel på variation