Del 1

Du får inte ändra given kod om inte annant anges.

Uppgift 1 – Använd behållare

I dent givna filen **uppgift1.py** finns en lista av namn. Skriv ett program som skapar en associativ behållare (dictionary) med dessa namn som nycklar. Varje nyckel ska associeras med en tom lista som värde. Fyll sedan varje lista i behållaren med 5 slumptal i intervallet 0 – 9 och skriv ut behållaren enligt formatet nedan. Inga ytterligare behållare får skapas.

Funktionella krav (användarinmatning i fetstil):

```
Matthew: 0 5 6 7 2
Sarah: 8 1 4 9 3
...
```

Uppgift 2 – Implementera algoritm

Du ska här implementera en enkel (men inte särkilt effektiv) sorteringsalgoritm för att sortera en lista av tal i stigande ordning, utgå från den givna filen.

uppgift2.py.

- Fram tills dess att listan är i rätt ordning:
  - Gå igenom listan av tal parvis:
    - Om ordningen av talen inte stämmer, byt plats på dem (se figur till höger)
- Skriv ut listan

Funktionella krav (användarinmatning i fetstil):

```
[1, 2, 3, 4, 5, 6 ,7 ,8, 9]
```

Uppgift 3 – Abstraktion

Utgå från den givna filen **uppgift3.py**, du ska här implementera fyra funktionerna som anropas i det givna huvuprogrammet enligt följande beskrivning. Funktionen **create_deck** ska skapa och returnera en tom kortlek, en kortlek representeras med en lista. Funktionen **create_card** ska skapa och returnera ett kort, ett kort representeras av en dictionary med tre nyckel-värde-par där värdet är av typen sträng. Funktionen **add_card** lägger till ett kort i en kortlek. Funktionen **display_deck** skriver ut en kortlek enligt körexemplet.

Undersök det givna programmet för att ta reda på vilken typ och antal data som skickas till varje funktion. Se också funktionella krav nedan.

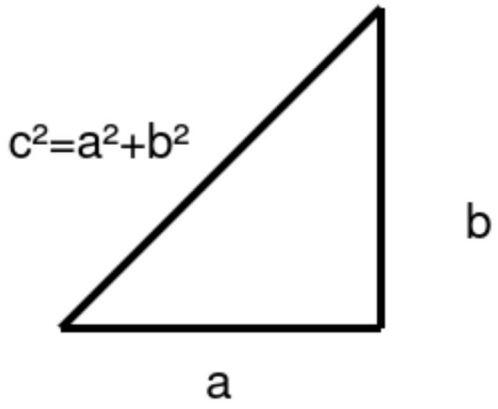
Funktionella krav (användarinmatning i fetstil):

```
Fireball   RX   Description of fireball...
Wrath of God   WW2   Description of wrath of god...
Savannah Lions   W   Description of savannah...
```

Del 2

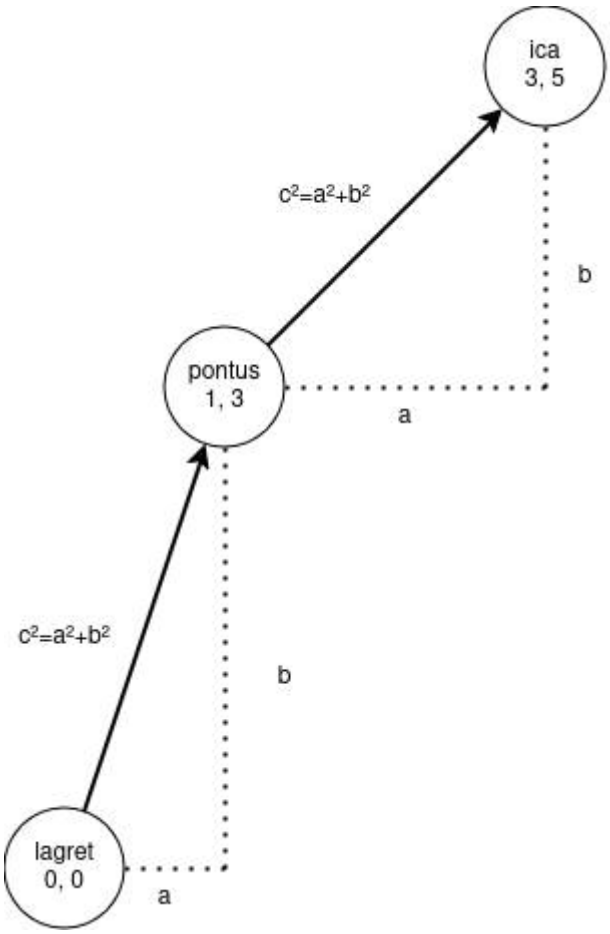
Uppgift 4 -- Skapa algoritm

I den givna filen **locations.txt** finns en uppsättning av platser som det ska levereras paket till. Varje rad i filen består av tre data som seppareras med mellanslag: platsens namn, var platsen befinner sig i x-led och var platsen befinner sig i y-led. Din uppgift är att läsa in filen och sedan beräkna kortaste sträckan för att besöka varje plats. Paketen levereras med en drönare och kan därför alltid ta fågelvägen mellan punkterna. Vi antar här att vi får ut kortaste sträckan genom att flyga från nuvarande punkt till den punkten som är närmast och som ännu inte har besökts. Ordningen som platserna besöks ska skrivas ut tillsammans med avståndet och det totala avståndet i slutet av programmet. Drönaren börjar i lagret på plats 0, 0 i x/y-led.



Mindre avrundningsfel spelar ingen roll i lösningen.

Fågelvägen mellan två punkter beräknas enkelt med pythagoras sats. Hypotenusan representerar då avståndet mellan två platser. Hypotenusan kan man få fram genom att ta summan av kvadraten av båda sidorna i triangeln och sedan ta roten ur den summan. ”Roten ur” finns i modulen **math** och är en funktion som heter **sqrt** (exempelvis: math.sqrt(9) för kvadratroten av 9).



Krav: Minst två funktioner med tydligt avgränsade syften som följer principerna för procedurell abstraktion skall användas för att lösa problemet

Krav: När platserna lästs in från fil så får behållaren som innehåller platserna inte modifieras, men du får skapa ytterligare behållare för att hålla koll på vilka platser som besökts

Funktionella krav:

```
pontus_hem 3.1622776601683795
ica 2.8284271247461903
stationen 10.198039027185569
Total distance traveled: 16.18874381210014
```

Uppgift 5 – Referenssemantik

I den givna filen **uppgift5.py** finns ett givet huvudprogram med en given datastruktur och funktionsarnrop. Datastrukturen representerar ett antal studenter som har klarat en delmängd av 4 uppgifter som finns i en kurs. Siffran 1 indikerar att uppgiften är avklarad och siffran noll indikerar att uppgiften inte är avklarad. Studenten ”ponha45” har alltså inte klarat uppgift 1 eller 4, men har klarat uppgift 2 och 3. Ditt jobb är att implementera funktionerna så att programmet uppfyller de funktionella kraven. Följande ickefunktionella krav skall också uppfyllas.

Funktionen **group_students** ska knyta ihop två studenters resultat så att de delar på en och samma lista av resultat. Båda studenterna vars namn skickas med till funktionen skall alltså ha en varsin referens till samma lista. Om studenternas resultat skiljer sig åt vid denna gruppering så ska deras resultat för en uppgift räknas som godkänt om en eller flera av studenterna har godkänt på den uppgiften.

Funktionen **split_group** ska bryta isär två studenters resultat så att de har varsin referens till två olika listor. Innehållet i vardera lista ska vara likadant som den gemensamma listan av resultat var.

Funktionen **set_grade** ska uppdatera ett betyg för en student. Observera i sista delen av exemplet att betyget på uppgift 4 (index 3) uppdateras för både ”ponha45” och ”hilra94” eftersom de har referenser till samma resultatslista.

Du alltid anta att anrop till **group_students** kommer inkludera namn på 2 olika studenter som inte är knutna till andra studenter. Du kan alltid anta att **split_group** kommer inkludera namn på 2 olika studenter som är knutna till varandra.

Funktionella krav:

```
Before grouping any students:
ponha45: [0, 1, 1, 0]
aroni08: [0, 1, 0, 1]
lovar08: [1, 1, 0, 0]
hilra94: [1, 0, 1, 0]
aleno42: [1, 0, 0, 1]

After grouping hilra and ponha:
ponha45: [1, 1, 1, 0]
aroni08: [0, 1, 0, 1]
lovar08: [1, 1, 0, 0]
hilra94: [1, 1, 1, 0]
aleno42: [1, 0, 0, 1]

After setting grade for ponha45:
ponha45: [1, 1, 1, 1]
aroni08: [0, 1, 0, 1]
lovar08: [1, 1, 0, 0]
hilra94: [1, 1, 1, 1]
aleno42: [1, 0, 0, 1]
```

Uppgift 6 – Inkapslning av data

I filen **uppgift6.py** hittar du pseudokod för ett huvudprogram. Ditt jobb är att designa och implementera modulerna (separata filer) **coord** och **canvas,** samt implementera huvudprogrammet så att delarna uppfyller de funktionella och ickefunktionella kraven.

Modulen **coord** ska representera en x/y-koordinat med en lämplig behållare och tillhörande publika funktioner (och eventuella privata funktioner). Det ska (minst) gå att skapa en x/y-koordinat, hämta x-koordinaten och hämta y-koordinaten.

Modulen **canvas** ska representera ett papper med en associativ (dict) underliggande behållare och tillhörande publika funktioner (och eventuella privata funktioner). Det ska (minst) gå att skapa ett canvas med en given bredd och höjd, alla koordinater har tecknet ” ” (mellanslag) lagrat när ett canvas skapas. Hämta bredd för canvas. Hämta höjd för canvas. Hämta vilken symbol som finns vid en viss koordinat. Sätta in en symbol vid en koordinat. Radera en symbol vid en viss koordinat (ersätt symbolen med ” ” (mellanslag)). Visa canvas för användaren (skriv ut till terminalen, se funktionella krav för utseende).

Ytterligare ickefunktionella krav:

- **coord** och **canvas** är olika abstrakta datatyper och ska därmed inte känna till varandras interna representation.
- Funktionen för att visa canvas för användaren ska använda sig av andra lämpliga funktioner i modulen för att undvika koduppрepning.

Funktionella krav:

```
#######
#     #
#     #
# *   #
#     #
#######
```