Graus en Enginyeria

Projecte de curs de l'assignatura
INFORMÀTICA 2025/26
Quadrimestre de tardor

Starmap

un sistema simple de gestió de mapes estel · lars.

Departament d'Enginyeria Minera,
Industrial i TIC (EMIT)

12 de setembre de 2025

Índex

1 Objectiu

L'objectiu d'aquest exercici consisteix a dissenyar i implementar un programa que permeti gestionar un mapa estel · lar de constel · lacions.

A més de la gestió bàsica d'estrelles i adjacències, el projecte incorpora:

- Assignació de **coordenades (x,y)** a cada estrella per poder representar gràficament les constel · lacions.
- **Dibuix** de constel · lacions amb la llibreria estàndard `turtle` de Python.
- **Persistència** en `JSON`: guardar i carregar constel · lacions.
- Arquitectura modular: `functions.py` (API de domini) i `starmap.py` (programa principal).

1.1 Coneixements previs

Per tal de poder dissenyar un programa que permeti simular un gestor de cartografia estel · lar, cal conèixer una sèrie de conceptes que la nostra aplicació haurà de complir:

- Una constel · lació s'identifica per un **nom**.
- Una constel · lació està formada per **estrelles**.
- Una estrella s'identifica per un **nom únic dins la constel · lació**.
- Cada estrella disposa de **coordenades cartesianes** (x, y) en un rang raonable (p. ex. $[-300, 300]$).
- Una estrella es pot trobar adjacent cap a una o més estrelles.
- Les **adjacències són bidireccionals** i no s'admeten auto-adjacències ni duplicats.
- No hi ha límit al nombre d'estrelles que poden estar adjacents a una estrella.

Així doncs, podríem representar la constel · lació de l'Osa Major coneixent les següents dades:

- **Nom**: Great Bear
- **Estrelles de la constel · lació**:
  - Alkaid
  - Mizar and Alcor
  - Alioth
  - Megrez
  - Dubhe
  - Merak
  - Phad
- **Adjacències**:
  - Estrelles adjacents a Alkaid: Mizar and Alcor

- Estrelles adjacents a Mizar and Alcor: Alkaid, Alioth
- Estrelles adjacents a Alioth: Mizar and Alcor, Megrez
- Estrelles adjacents a Megrez: Alioth, Dubhe, Phad
- Estrelles adjacents a Dubhe: Megrez, Merak
- Estrelles adjacents a Merak: Dubhe, Phad
- Estrelles adjacents a Phad: Megrez, Merak

Amb la informació que s'ha proporcionat, podríeu donar nom a les estrelles de la següent imatge de l'Osa Major? Quina estructura de dades faríeu servir per a representar el mapa estel · lar de constel · lacions?

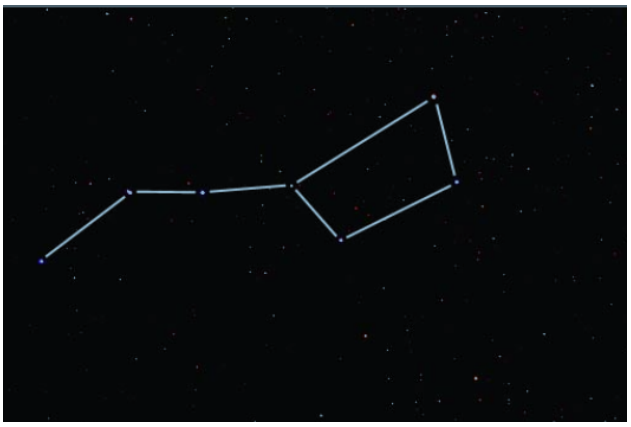

Figura 1: L'Osa Major.

2 Descripció del sistema

Estructura de dades

Una possible estructura per emmagatzemar el mapa en el nostre programa Python és la següent:

```
starmap = {
    "constellationName": {
        "stars": {
            "Alkaid": [x1, y1],
            "Mizar and Alcor": [x2, y2],
            # ...
        },
        "adjacencies": {
            "Alkaid": ["Mizar and Alcor", "Alioth"],
            "Mizar and Alcor": ["Alkaid"],
            # ...
        }
    },
    # Other constellations...
}
```

Important

- Si B és adjacent d'A, aleshores A és adjacent de B.
- No s'admeten auto-adjacències ($A \not\sim A$) ni duplicats.
- Els noms d'estrella són únics dins d'una constel · lació.

  Nota: en els fitxers JSON, les adjacències s'emmagatzemen com a `llistes`, ja que el format JSON no admet conjunts. Tot i que això pot provocar alguns comportaments no desitjats, mantindrem aquest mateix format al nostre codi Python.

Exemple complet de constel · lació en format JSON

El format de fitxer `JSON` per a una constel · lació d'exemple és el següent:

```
{
  "Great Bear": {
    "stars": {
      "Alkaid": [200, -20],
      "Mizar and Alcor": [150, 10],
      "Alioth": [100, 40],
      "Megrez": [40, 60],
      "Dubhe": [-80, 120],
      "Merak": [-100, 40],
      "Phad": [-20, 20]
    },
    "adjacencies": {
      "Alkaid": ["Mizar and Alcor"],
      "Mizar and Alcor": ["Alkaid", "Alioth"],
      "Alioth": ["Mizar and Alcor", "Megrez"],
      "Megrez": ["Alioth", "Dubhe", "Phad"],
      "Dubhe": ["Megrez", "Merak"],
      "Merak": ["Dubhe", "Phad"],
      "Phad": ["Megrez", "Merak"]
    }
  }
}
```

2.1 Funcionament del programa

El programa mostrarà un menú d'opcions:

1. Afegir constel · lació
2. Afegir estrelles a constel · lació
3. Afegir adjacència entre estrelles de constel · lació
4. Eliminar adjacència entre estrelles de constel · lació
5. Eliminar estrella de constel · lació
6. Llistar constel · lació

7. Llistar constel · lacions
8. Dibuixar constel · lació (Turtle)
9. Guardar constel · lació en JSON
10. Carregar constel · lació des de JSON
11. Modificar posició d'una estrella
12. Eliminar constel · lació
13. Sortir de l'aplicació

La descripció de cada opció s'amplia amb:

- **Dibuixar constel · lació (Turtle)**: obre una finestra de `turtle` i representa les estrelles als seus (x, y), amb el nom etiquetat, i dibuixa segments per a cada adjacència.
- **Guardar en JSON**: demana un camí de fitxer i desa la constel · lació seleccionada en el format indicat.
- **Carregar des de JSON**: llegeix una constel · lació d'un fitxer `.json` i l'afegeix al mapa.
- **Modificar posició d'una estrella**: permet canviar les coordenades (x, y) d'una estrella existent.
- **Eliminar constel · lació**: suprimeix la constel · lació (estrelles i adjacències).

2.2 Exemple d'execució

Per tal que comprengueu més bé el que s'està demanant que implementeu en aquest exercici, s'inclou un vídeo amb un exemple complet d'execució.

https://youtu.be/CWFEeo7eJHI

3 Eines necessàries

Es fan servir les eines habituals del curs: editor de text, l'intèrpret de Python i les seves llibreries, i l'eina per passar els tests.

Per a la visualització, s'utilitza la llibreria estàndard `turtle` de Python.

3.1 Instal · lació i ús de `turtle`

La llibreria `turtle` forma part de la distribució estàndard de Python (CPython). Per executar els exemples:

1. Assegura't de tenir Python 3 instal · lat.
2. Executa el programa principal amb:

   ```
   $ python starmap.py
   ```

3. Quan escullis l'opció de dibuixar, s'obrirà una finestra gràfica. Per tancar-la, prem la "X" de la finestra o acaba el programa.

En entorns sense servidor gràfic (p. ex. màquines remotes sense GUI), la finestra `turtle` no es podrà mostrar.

3.1.1 Si `turtle` no està disponible en la nostra instal · lació

`turtle` forma part de la biblioteca estàndard de Python, però depèn de `tkinter` (la interfície de Tk). En algunes instal · lacions de Linux, `tkinter` no ve preinstal · lat.

Per comprovar si realment està disponible al nostre equip, podem executar la següent comanda al terminal:

```
$ python3 -c "import turtle; print('OK')"
```

Si dóna error, cal instal · lar el paquet de `tkinter` segons el vostre sistema:

- **Ubuntu/Debian i derivats**:
  `sudo apt-get install python3-tk`
- **Windows (instal · lador oficial de python.org)**: normalment ja està inclòs; no cal res.
- **macOS**:
  - Amb el Python oficial de `python.org`, `tkinter` sol venir activat.
  - Si feu servir `Homebrew` o `pyenv` i us falta Tk, instal · leu `tcl-tk` i assegureu-vos que el vostre Python té suport per Tk: `brew install tcl-tk`
    Després comproveu amb `python3 -m tkinter` que s'obre una finestra de prova.

3.1.2 Exemple: punt etiquetat amb `turtle`

El codi següent dibuixa un punt a les coordenades (x, y) i n'escriu l'etiqueta al costat:

```
import turtle

def labeledPoint(x, y, label, size=8):
    t = turtle.Turtle()
    t.hideturtle()
    t.speed(0)
    t.penup()
    t.goto(x, y)
    t.dot(size)             # filled point
    t.goto(x + 8, y + 8)    # slight offset for readability
    t.write(label, font=("Arial", 10, "normal"))

# Minimal usage example:
if __name__ == "__main__":
    screen = turtle.Screen()
    screen.setup(width=800, height=600)
    labeledPoint(-120, 180, "Dubhe")
    labeledPoint(-60, 100, "Merak")
    turtle.done()
```

3.2 Introducció a JSON

JSON (JavaScript Object Notation) és un format lleuger d'intercanvi de dades, fàcil de llegir i escriure per humans, i fàcil de generar i analitzar per màquines. Tot i que prové del llenguatge JavaScript, és un format independent del llenguatge i àmpliament utilitzat en el desenvolupament de programari per emmagatzemar informació estructurada.

En aquest projecte, fem servir JSON per guardar i carregar la informació d'un mapa estel · lar (les constel · lacions i les seves estrelles) de manera persistent, és a dir, per tal que es pugui reprendre el programa en sessions posteriors.

Estructura bàsica de JSON

El format JSON es basa en dues estructures:

- **Objectes**: col · leccions de parelles clau-valor, escrites entre claus { }. Són similars als diccionaris de Python.
- **Llistes (arrays)**: col · leccions ordenades de valors, escrites entre claudàtors [ ].

Manipulació de `JSON`

Per guardar i carregar constel · lacions en fitxers `.json` es fa servir la llibreria estàndard `json` de Python. Els mètodes principals són:

- `json.dump(obj, fitxer, ...)` → escriu un objecte Python en un fitxer en format JSON.
- `json.load(fitxer)` → llegeix el contingut d'un fitxer JSON i retorna un objecte Python.

- `json.dumps(obj, ...)` → retorna una cadena de text amb l'objecte codificat en JSON.
- `json.loads(cadena)` → transforma una cadena JSON en un objecte Python.

Aquests mètodes permeten convertir entre les estructures de dades de Python (diccionaris, llistes) i el format `JSON`.

4 Aspectes organitzatius

El projecte està dimensionat i dissenyat per a ésser treballat en equip. Cal que seguiu el següent esquema al treballar-hi:

1. Llegiu amb atenció individualment tot l'enunciat de principi a fi. Estudieu els coneixements previs.
2. Reuniu-vos i poseu en comú el que heu estudiat. Escriviu conjuntament els docstrings i doctests (en aquelles funcions en què es pugui) per tal de verificar que tots els membres del grup coincidiu en què ha de realitzar cadascuna d'elles.
3. Repartiu-vos la tasca de desenvolupament del projecte.
4. Repartiu-vos la tasca de documentar el projecte.

   Recordeu que tota funció que pot ser documentada ha de tenir els seus respectius doctests i documentació.

4.1 Lliurables

Un dia abans de la presentació, caldrà fer entrega a través d'Atenea del codi del vostre projecte, així com del document de la memòria del projecte (en format *.pdf*) de la qual es detallen els apartats més avall. El vostre projecte ha d'incloure, com a mínim, la funcionalitat bàsica. **Tingueu en compte que la part bàsica funcionant us atorga, com a molt, una nota de 7 al projecte**.

Les tasques opcionals milloren la nota del projecte i es recomana que, com a mínim, n'implementeu una. **Cada tasca opcional que dugueu a terme correctament i funcioni augmentarà la nota total del projecte en un punt**.

La realització i entrega d'una memòria correcta, així com l'ús correcte de la documentació, poden augmentar també la nota del projecte, a criteri del corrector.

El dia de la presentació també caldrà que porteu imprès la memòria del projecte en què han de constar els següents apartats:

1. Manual d'usuari. (un full) Un petit document en què s'explica a un usuari final per què serveix i com cal usar el programa que heu implementat.
2. Memòria tècnica: codi font del projecte documentat convenientment. (Tot el vostre codi ha d'estar incorporat a la memòria)

3. Memòria d'execució: Cal que entregueu una taula que reflecteixi l'execució del projecte. Aquesta taula ha de tenir 4 columnes: una per la data i una per a cada persona que forma l'equip. En aquesta taula hi consignareu una casella per cada dia que treballeu en el projecte. En aquesta casella indicareu molt resumidament què heu fet i quant temps hi heu dedicat. Finalment sumareu tots els temps dedicats. La taula ha de semblar aquesta:

Data	Pep	Maria	Aina
2/12/2025		Llegir enunciat (2h)	Llegir enunciat (2h) Pensar exemple (1h)
3/12/2025	Reunió treball (1h)	Reunió treball (1h)	Reunió treball (1h)
	1h feina	3h feina	4h feina

Addicionalment, us recomanem afegir un apartat de conclusions en què cada membre resumeixi les dificultats i aprenentatges extrets de la realització del projecte.

5 Tasques del projecte

El projecte es divideix en dos mòduls:

- ***functions.py***: conté totes les funcions de gestió del domini (constel · lacions, estrelles, adjacències, coordenades, persistència i dibuix).
- ***starmap.py***: importa `functions.py` i implementa el menú i la lògica d'interacció amb l'usuari.

A continuació es detallen les tasques que permetran implementar les funcions necessàries per donar solució al projecte. Totes les funcions han d'estar **documentades** i, quan sigui possible, incloure **doctests**. Per comprovar la seva correcció podeu executar:

```
$ python -m doctest -v functions.py
```

5.1 Fitxer functions.py

Tasca 1
Dissenyeu una funció de nom ***addConstellation(starmap, constellation)*** *que, donat un mapa estel · lar i un nom de constel · lació, afegeixi una nova constel · lació buida al mapa.*

```
>>> starmap = {}
>>> addConstellation(starmap, "Great Bear")
>>> starmap
{'Great Bear': {'stars': {}, 'adjacencies': {}}}
```

Tasca 2
Dissenyeu una funció de nom ***addStars(starmap, constellation, starNames)*** *que, donat un mapa estel · lar, una constel · lació i una llista d'estrelles, permeti afegir les estrelles donades a una constel · lació existent. Cada estrella ha de començar sense adjacències i amb coordenades buides.*

```
>>> starmap = {'Great Bear': {'stars': {}, 'adjacencies': {}}}
>>> addStars(starmap, "Great Bear", ["Alkaid", "Dubhe"])
>>> starmap
{'Great Bear': {'stars': {'Alkaid': [], 'Dubhe': []},
... 'adjacencies': {'Alkaid': [], 'Dubhe': []}}}
```

Tasca 3
Dissenyeu una funció de nom ***addAdjacencies(starmap, constellation, baseStar, neighbors)*** *que, donat un mapa estel · lar, una constel · lació, una estrella base i una llista d'estrelles adjacents, estableixi les connexions bidireccionals. No s'han de permetre duplicats ni auto-adjacències. No es crearà l'adjacència si alguna de les dues estrelles no existeix encara a la constel · lació.*

```
>>> starmap = {'Great Bear': {'stars': {'Alkaid': [], 'Dubhe': []},
... 'adjacencies': {'Alkaid': [], 'Dubhe': []}}}
>>> addAdjacencies(starmap, "Great Bear", "Alkaid", ["Dubhe"])
>>> starmap["Great Bear"]["adjacencies"]["Alkaid"]
['Dubhe']
```

Tasca 4

Dissenyeu una funció de nom ***deleteAdjacency(starmap, constellation, star1, star2)****, que donat un mapa estel · lar, un nom de constel · lació, i dos noms d'estrella, elimini la connexió entre les dues estrelles donades.*

```
>>> starmap = {'Great Bear': {'stars': {'Alkaid': [], 'Dubhe': []},
... 'adjacencies': {'Alkaid': ['Dubhe'], 'Dubhe': ['Alkaid']}}}
>>> deleteAdjacency(starmap, "Great Bear", "Alkaid", "Dubhe")
>>> starmap["Great Bear"]["adjacencies"]["Alkaid"]
[]
```

Tasca 5

Dissenyeu una funció de nom ***deleteStar(starmap, constellation, star)*** *que, donat un mapa estel · lar, una constel · lació i un nom d'estrella, elimini l'estrella de la constel · lació (així com totes les seves adjacències).*

```
>>> starmap = {'Great Bear': {'stars': {'Alkaid': [], 'Dubhe': []},
... 'adjacencies': {'Alkaid': ['Dubhe'], 'Dubhe': ['Alkaid']}}}
>>> deleteStar(starmap, "Great Bear", "Dubhe")
>>> "Dubhe" in starmap["Great Bear"]["stars"]
False
>>> starmap
{'Great Bear': {'stars': {'Alkaid': []}, 'adjacencies': {'Alkaid': []}}}
```

Tasca 6

Dissenyeu una funció de nom ***listAllStars(starmap, constellation)****, que donat un mapa estel · lar i una constel · lació, mostri per pantalla totes les estrelles de la constel · lació i les seves adjacències.*

```
>>> starmap = {'Great Bear': {'stars': {'Alkaid': [], 'Dubhe': []},
... 'adjacencies': {'Alkaid': ['Dubhe'], 'Dubhe': ['Alkaid']}}}
>>> listAllStars(starmap, "Great Bear")
Great Bear:
  Alkaid adjacent a: Dubhe
  Dubhe adjacent a: Alkaid
```

Tasca 7

Dissenyeu una funció de nom ***listAllConstellations(starmap)*** *que, donat un mapa estel·lar, mostri per pantalla el nom de totes les constel·lacions guardades al mapa.*

```
>>> starmap = {'Great Bear': {'stars': {'Alkaid': [], 'Dubhe': []},
... 'adjacencies': {'Alkaid': ['Dubhe'], 'Dubhe': ['Alkaid']}}}
>>> listAllConstellations(starmap)
Great Bear
>>> starmap = {}
>>> listAllConstellations(starmap)
No hi ha constel·lacions
```

Tasca 8

Dissenyeu una funció de nom ***assignCoordinates(starmap, constellation, star, x, y)*** *que, donat un mapa estel·lar, una constel·lació, una estrella i les coordenades x i y, assigni les coordenades (x, y) a l'estrella de la constel·lació donada.*

```
>>> starmap = {'Great Bear': {'stars': {'Alkaid': [], 'Dubhe': []},
... 'adjacencies': {'Alkaid': ['Dubhe'], 'Dubhe': ['Alkaid']}}}
>>> assignCoordinates(starmap, "Great Bear", "Alkaid", 10, 20)
>>> starmap["Great Bear"]["stars"]["Alkaid"]
[10, 20]
>>> starmap
{'Great Bear': {'stars': {'Alkaid': [10, 20], 'Dubhe': []},
'adjacencies': {'Alkaid': ['Dubhe'], 'Dubhe': ['Alkaid']}}}
```

Tasca 9

Dissenyeu una funció de nom ***getCoordinates(starmap, constellation, star)*** *que, donat un mapa estel·lar, una constel·lació i una estrella, retorni les coordenades de l'estrella donada.*

```
>>> starmap = {'Great Bear': {'stars': {'Alkaid': [10, 20], 'Dubhe': []},
... 'adjacencies': {'Alkaid': ['Dubhe'], 'Dubhe': ['Alkaid']}}}
>>> getCoordinates(starmap, "Great Bear", "Alkaid")
[10, 20]
```

Tasca 10

Dissenyeu una funció de nom ***isValidCoord(coord)*** *que, donada una coordenada, retorni* `True` *si la coordenada donada és una llista de longitud 2 (p. ex.* `[x, y]`*), i* `False` *altrament.*

```
>>> isValidCoord([10, 20])
True
>>> isValidCoord((0, 0))
True
```

```
>>> isValidCoord([1])
False
>>> isValidCoord("10,20")
False
```

Tasca 11
Dissenyeu una funció de nom ***shouldDrawOnce(a, b)*** *que, donats dos textos corresponents a noms d'estrelles, retorni* `True` *només si el nom* `a` *va abans que* `b` *(ordre alfabètic).*

NOTA: Aquesta funció s'utilitza per evitar dibuixar dues vegades el mateix segment (bidireccional).

```
>>> shouldDrawOnce("Alkaid", "Dubhe")
True
>>> shouldDrawOnce("Dubhe", "Alkaid")
False
>>> shouldDrawOnce("A", "A")
False
```

Tasca 12
Dissenyeu una funció de nom ***setupWindow(constellation, width=800, height=600)*** *que creï la finestra de* `turtle`*, en fixi el títol i la mida, i retorni l'objecte* `Screen`*.*

Nota*: aquestes funcions visuals no tenen doctest.*

Tasca 13
Dissenyeu una funció de nom ***makePen()*** *que creï i retorni una tortuga a punt per dibuixar. La configuració de la tortuga ha de ser: amagada, màxima velocitat, llapis aixecat i gruix de línia 2.*

Tasca 14
Dissenyeu una funció de nom ***drawStar(pen, name, coord, pointSize=8, offset=8)*** *que, donada una tortuga, un nom d'estrella, unes coordenades, la mida de punt i un desplaçament, dibuixi un punt a* `coord` *i escrigui* `name` *desplaçat lleugerament per no tapar el punt.*

Tasca 15
Dissenyeu una funció de nom ***drawAllStars(pen, stars)*** *que, donada una tortuga i un diccionari d'estrelles (amb coordenades), dibuixi totes les estrelles amb coordenades vàlides* `[x, y]`*.*

Tasca 16
Dissenyeu una funció de nom ***drawSegment(pen, aXY, bXY)*** *que, donada una tortuga i dues coordenades, dibuixi una línia des de* `aXY=(aX, aY)` *fins a* `bXY=(bX, bY)`*.*

Tasca 17
Dissenyeu una funció de nom ***drawAllSegments(pen, stars, adjacencies)*** *que, donada una tortuga, un diccionari d'estrelles, i un diccionari d'adjacències, recorri totes les adjacències i dibuixi segments només quan* `should_draw_once(a,b)` *sigui* `True`*. Ignoreu noms sense coordenades vàlides.*

Tasca 18
Dissenyeu una funció de nom ***drawConstellation(starmap, constellation*** *que, fent ús de les funcions definides anteriorment:*

1. *Comprova que la constel·lació existeix.*
2. *Obre finestra i crea llapis.*
3. *Dibuixa totes les estrelles i segments.*
4. *Manté la finestra oberta amb* `turtle.done()`*.*

En aquest cas us proporcionem el codi d'aquesta funció que podeu fer servir a la vostra solució i que, si totes les funcions anteriors han estat correctament implementades, hauria de funcionar.

```
def drawConstellation(starmap, constellation):
    """
    Dibuixa la constel·lació indicant estrelles (punt+nom) i segments (adjacències).
    Obre una finestra Turtle i es queda oberta fins que la tanques.
    """
    if constellation not in starmap:
        raise KeyError("Constel·lació inexistent")

    data  = starmap[constellation]
    stars = data.get('stars', {})
    adjs  = data.get('adjacencies', {})

    try:
        turtle.TurtleScreen._RUNNING = True
    except Exception:
        pass

    try:
        turtle.Screen().clearscreen()
    except turtle.Terminator:
        try:
            turtle.TurtleScreen._RUNNING = True
        except Exception:
            pass
```

```
setupWindow(constellation)
pen = makePen()

drawAllStars(pen, stars)
drawAllSegments(pen, stars, adjs)

turtle.done()
```

5.2 Fitxer starmap.py

Aquest fitxer ha de contenir el programa principal amb el menú d'opcions. Es recomana implementar-lo de manera incremental: primer el bucle principal i, després, una a una les opcions.

És molt important que us fixeu en l'exemple d'execució, doncs la vostra solució ha de replicar de la manera més precisa possible el seu funcionament.

6 Tasques Opcionals

A continuació, es presenten algunes funcionalitats extres que podeu implementar per ampliar el projecte:

Tasca 19
Càlculs geomètrics — Distància entre dues estrelles

Implementeu una funció `distanceBetween(starmap, constellation, starA, starB)` *que retorni la distància euclidiana entre dues estrelles amb coordenades definides* (x, y).

```
>>> starmap = {'Great Bear': {'stars': {'A':[0,0], 'B':[3,4]},
'adjacencies':{}}}
>>> round(distanceBetween(starmap, "Great Bear", "A", "B"), 2)
5.0
```

A tenir en compte*: Valideu que ambdues estrelles tinguin coordenades. Gestioneu els errors amb missatges clars.*

Tasca 20
Càlculs geomètrics — Estrella més propera

Implementeu `nearestStar(starmap, constellation, star)` *que retorni una llista amb el nom de les estrelles diferents de* `star` *amb distància mínima.*

```
>>> starmap = {'Great Bear': {'stars': {'A':[0,0],'B':[3,4],'C':[10,0]},
'adjacencies':{}}}
>>> nearestStar(starmap, "Great Bear", "C")
['B']
```

Tasca 21
Turtle — Canviar estil segons propietats

Afegiu el que creieu convenient per establir estils d'estrelles (color, mida del punt, tipus de lletra) segons propietats opcionals (p. ex. magnitud).

Requisits*:*

- *Nova funció* `setStarStyle(starmap, constellation, star, color=None, size=None)`.
- `drawConstellation(...)` *ha de respectar aquests estils si existeixen.*

Tasca 22
Persistència avançada — Guardar* tot *el mapa

Implementeu `saveStarmapJSON(starmap, filepath)` *i* `loadStarmapJSON(filepath)` *per desar i carregar el* mapa sencer *(totes les constel · lacions).*

```
>>> saveStarmapJSON(starmap, "full_map.json")
>>> newmap = loadStarmapJSON("full_map.json")
>>> isinstance(newmap, dict) and "Great Bear" in newmap
True
```

Tasca 23
Persistència avançada — Còpia de seguretat

Abans d'una operació destructiva (`deleteStar`*,* `deleteAdjacency`*,* `deleteConstellation`*), creeu automàticament un fitxer* `.bak.json` *amb l'estat actual de la constel · lació (o del mapa sencer si ho preferiu).*

Tasca 24
Consultes — Comptar estrelles

Implementeu `countStars(starmap, constellation)` *que retorni el nombre total d'estrelles.*

```
>>> starmap = {'Great Bear': {'stars': {'A':[], 'B':[], 'C':[]}, 'adjacencies':{}}}
>>> countStars(starmap, "Great Bear")
3
```

Tasca 25
Consultes — Estrelles aïllades

Implementeu `listIsolatedStars(starmap, constellation)` *que retorni una llista d'estrelles sense adjacències.*

```
>>> starmap = {'Great Bear': {'stars': {'A':[], 'B':[], 'C':[]},
...                           'adjacencies': {'A':['B'], 'B':['A']}}}
>>> listIsolatedStars(starmap, "Great Bear")
['C']
```