Problemes LP – Curs 2018-19
Tema 7: Python

IMPORTANT: En tots els exercicis intenteu aprofitar al màxim les funcionalitats de Python que us hem explicat per accedir i recórrer llistes, diccionaris i arrays, utilitzant tots els mètodes i funcions predefinides que pugueu i minimitzant l'ús d'estructures `for` i `while`.

EXERCICIS DE CLASSES

Exercici 1 (nivell bàsic)

Implementeu una calculadora de nombres complexos seguint els passos següents:

1. Definiu una classe `Complex` que permeti guardar un nombre complex i tingui mètodes que permetin:
   - Mostrar el nombre complex per pantalla en format a + bi
   - Llegir per teclat els valors d'un nombre complex
   - Obtenir el conjugat d'un nombre complex
   - Sumar dos nombres complexos
   - Restar dos nombres complexos
   - Multiplicar dos nombres complexos

   Les operacions de suma, resta i multiplicació s'han d'implementar sobrecarregant els operadors +, - i *, respectivament
2. Implementeu el programa principal. Ha d'anar demanant operacions (suma, resta, multiplicació o obtenir el conjugat) per fer fins que es seleccioni l'opció de sortir. En cada operació es demanaran per teclat els nombres complexos d'entrada i es mostrarà el resultat final per pantalla.

Exercici 2 (nivell bàsic)

Crear una classe `Vector` que permeti representar un vector en un espai de N dimensions, $v = (x_1.x_2, \dots, x_N)$. El número de dimensions ha de poder ser diferent per cada vector.

1. El constructor ha de rebre com a paràmetre una llista numèrica amb les coordenades del vector.
2. Redefiniu el mètode __str__ perquè es mostri per pantalla el seu contingut en format $[x_1, x_2, \dots, x_N]$.
3. Sobrecarregueu l'operador de suma perquè sumi dos vectors component a component, si tenen la mateixa dimensió. Si no, s'ha de mostrar un missatge d'error.

4. Sobrecarregueu l'operador de resta perquè calculi la distància euclidiana entre els dos vectors: $d(u,v) = \sqrt{\sum_{i=1}^{N}(u_i - v_i)^2}$
5. Sobrecarregueu l'operador de multiplicació perquè multipliqui totes les components del vector per un número enter que es passa com a paràmetre.

Exercici 3 (nivell mig)

Implementeu un programa de gestió d'una assignatura i els seus alumnes seguint els passos següents:

1. Definiu una classe `Alumne` que permeti guardar el nom d'un alumne i una llista amb les notes de totes les proves de l'assignatura. Definiu tots els atributs i mètodes que considereu adients per la classe, segons el que es demana al programa principal.
2. Definiu una classe `Assignatura` que permeti guardar el nom de l'assignatura i una llista amb tots els alumnes de l'assignatura. Definiu tots els atributs i mètodes que considereu adients per la classe, segons el que es demana al programa principal.
3. Implementeu un programa principal que primer de tot llegeixi el nom de l'assignatura i després mostri repetidament un menú que permeti escollir entre les opcions següents (fins que es seleccioni sortir):
    - Afegir un alumne: demanarà el nom d'un alumne i l'afegirà a la llista d'alumnes de l'assignatura
    - Afegir una nota a un alumne: demanarà el nom de l'alumne i la nota i afegirà la nota a la llista de notes de l'alumne. Si l'alumne no existeix es mostrarà un missatge d'error.
    - Consultar la informació d'un alumne: demanarà el nom de l'alumne i mostrarà per pantalla el nom, la llista de notes i la nota mitjana de l'alumne. Si l'alumne no existeix es mostrarà un missatge d'error.
    - Llistar tots els alumnes de l'assignatura: Mostrarà per pantalla el nom i la nota mitjana de tots els alumnes de l'assignatura. També mostrarà la nota mitjana global de l'assignatura, tenint en compte les notes mitjanes de tots els alumnes.
    - Sortir

EXERCICIS DE FITXERS

Exercici 4 (nivell bàsic)

Implementeu una funció que rebi com a paràmetre el nom d'un fitxer de text i llegeixi el fitxer i mostri per pantalla el número de línies del fitxer, el número de paraules i el número de caràcters totals del fitxer (sense comptar els espais en blanc).

Exercici 5 (nivell bàsic)

Implementeu una funció que rebi com a paràmetre el nom d'un fitxer de text que conté números enters i retorni una llista que contingui quants números hi ha a cada línia del fitxer i la suma dels números de cada línia.

EXERCICIS DE LLISTES

Exercici 6 (nivell mig)

Donada una llista de nombres enters, que es passa com a paràmetre, escriure funcions per:

a) Retornar la suma i el valor mig de tots els valors de la llista.
b) Retornar una llista amb la suma acumulada de la llista original a cada element, és a dir, una llista on el primer element sigui el mateix, el segon sigui la suma del primer i el segon, el tercer la suma dels tres primers elements i així successivament. Per exemple si la llista d'entrada és [1,2,3,4] la llista de sortida hauria de ser [1,3,6,10]
c) Retornar una nova llista amb el factorial de cadascun dels nombres de la llista original.
d) Retornar una nova llista amb tots els nombres primers de la llista original.
e) Retornar si hi ha algun element duplicat dins de la llista.
f) Retornar una nova llista sense elements duplicats.
g) Retornar tres llistes, una amb els valors més petits que un número *n* que es passa com a paràmetre, una altra amb els valors més grans i una tercera amb els valors iguals.
h) Retornar una llista amb els elements que són divisibles per un número *n* que es passa com a paràmetre.
i) Suposant que la llista original només conté 0's i 1's representant un número binari (amb el dígit més significatiu a la primera posició de la llista), retornar el valor decimal corresponent.

Exercici 7 (nivell mig)

Donades dues llistes de nombres enters, que es passen com a paràmetre, escriure funcions per:

a) Retornar una nova llista amb la intersecció de les dues llistes, és a dir, que contingui els elements de la primera llista que també estan a la segona.
b) Retornar una nova llista en què cada element sigui la multiplicació dels dos elements corresponents de les dues llistes. Si les dues llistes tenen mida diferent, la mida de la llista resultat ha de ser la mida de la més gran (omplint amb zeros tots els elements pels que no es pot calcular la multiplicació)
c) Retornar una nova llista que contingui la multiplicació de cadascun dels elements de la primera llista per tots els elements de la segona llista. Els elements de la nova llista seran llistes i cada llista contindrà totes les multiplicacions d'un element de la primera llista amb tots els elements de la segona.

Exercici 8 (nivell mig)

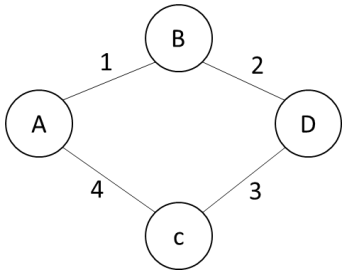
Escriure un programa que permeti jugar a una versió simplificada del joc del Master Mind. El joc consistirà en endevinar una seqüència de números. Al principi, el programa ha de demanar la longitud de la seqüència i generar una llista aleatòria de números entre l'1 i el 4 de la longitud indicada. Després, el programa ha d'anar demanant seqüències de números i anar informant de quants números de la seqüència s'han endevinat (valor i posició) fins que s'encerta la seqüència aleatòria o s'arriba a un número màxim d'intents igual al doble de la longitud de la seqüència.

Per exemple, si la seqüència generada de forma aleatòria és [2, 4, 3, 3], si s'introdueix la seqüència [1, 2, 3, 4] el programa ha d'informar que hi ha un número correcte, amb la seqüència [4, 2, 3, 3] ha d'informar que hi ha dos números correctes i amb la seqüència [3, 3, 4, 2] que no hi ha cap número correcte.

Exercici 9 (nivell mig) – EXERCICI AVALUABLE

Com ja sabem un graf ponderat es pot representar amb llistes d'adjacència on per cada node guardem una llista amb totes les arestes que surten del node (per cada aresta guardem l'identificador del node veí i el pes associat a l'aresta).

A Caronte teniu un fitxer `graf.py` que declara una classe per guardar un graf utilitzant llistes d'adjacència per guardar la informació de les aretes. Tal com està definida la classe, tota la informació del graf es guarda en una única llista. Cada element d'aquesta llista correspon a un node del graf i guarda, a la primera posició, l'etiqueta associada al graf, i a la segona, la llista d'arestes que surten del graf. Per cada aresta es guarda una altra llista amb dos valors: l'identificador del node veí i el pes associat a l'aresta. Així, si tenim aquest graf no dirigit:



la representació del graf dins de la llista serà la següent:

```
[['A', [[1, 1], [2, 4]]],
 ['B', [[0, 1], [3, 2]]],
 ['C', [[0, 4], [3, 3]]],
 ['D', [[1, 2], [2, 3]]]]
```

La classe `Graf` que us donem ja té un constructor per inicialitzar un graf i un mètode per mostrar-lo per pantalla. Sobre aquesta classe volem que afegiu els mètodes següents:

a) Un mètode `nodesAillats` que retorni una llista amb les etiquetes dels nodes aïllats del graf (aquells nodes que no tenen cap aresta).

b) Un mètode `nodeMaxArestes` que retorni l'etiqueta del node amb més arestes del graf.
c) Un mètode `etiquetesAdjacents` que donada l'etiqueta d'un node (que es passa com a paràmetre), retorni una llista amb les etiquetes de tots els seus nodes adjacents.
d) Un mètode `cicles` que retorni tots els cicles de 3 nodes del graf (igual que a l'exercici 2 del tema de grafs), sense repetir-ne cap.

EXERCICIS DE DICCIONARIS

Exercici 10 (nivell bàsic)

Escriure una funció que rebi com a paràmetre el nom d'un fitxer i retorni un diccionari que contingui el nº de vegades que apareix cada paraula al fitxer. A més a més ha de mostrar per pantalla la llista de paraules i el nº de cops que apareix cadascuna al fitxer ordenades alfabèticament.

Exercici 11 (nivell mig)

a) Escriure una funció que rebi com a paràmetre el nom d'un fitxer i retorni un diccionari que contingui, per cada caràcter de l'alfabet, el nº de vegades que apareix al fitxer i una llista amb totes les paraules del fitxer que contenen aquest caràcter.

b) Escriure una funció que, a partir del diccionari creat a l'apartat anterior, mostri per cada caràcter (ordenats alfabèticament), la paraula més llarga en què apareix.

Exercici 12 (nivell mig) – EXERCICI AVALUABLE

Volem crear un diccionari de sinònims on, per cada paraula (clau) del diccionari, el valor que hi guardem és una llista de tots els seus sinònims.

a) Implementeu una funció per afegir un sinònim al diccionari amb la capçalera següent:

```
afegeixSinonim(diccionari, paraula, sinonim)
```

La funció ha d'afegir el sinònim que es passa com a paràmetre al diccionari. Si la paraula no existeix al diccionari, s'hi ha d'afegir amb el seu sinònim. Si la paraula ja existeix al diccionari, el nou sinònim s'afegeix al final de la llista de sinònims actual de la paraula. Si el sinònim ja existeix dins de la llista de sinònims de la paraula, no s'ha de tornar a afegir.

b) Implementeu una funció que agafi com a entrada un diccionari de sinònims com el que es crea a l'apartat anterior i una llista amb paraules i retorni una nova llista transformada utilitzant el diccionari de sinònims de la forma següent:

- Per les paraules de la llista original que apareixen al diccionari hem de seleccionar aleatòriament un dels sinònims de la paraula que hi hagi al diccionari i substituir a la nova llista la paraula original pel seu sinònim.
- Les paraules de la llista original que no apareixen al diccionari han d'afegir-se directament a la nova llista sense modificar-les.

La funció tindrà aquesta capçalera:

```
def conversioSinonims(frase, diccionari)
```

Per fer aquesta funció podeu utilitzar la funció `random.randrange(n)`, que retorna un número aleatori entre 0 i n-1.

EXERCICIS D'ARRAYS

Exercici 13 (nivell mig)

Donada una matriu, que es passa com a paràmetre en un array de *numpy*, escriure funcions per:

a) Retornar una nova matriu en què tots els valors fora del rang *[n,m*], que es passa com a paràmetre, es substitueix pel valor *x,* que també es passa com a paràmetre.
b) Retornar una llista que contingui, per cada fila, el producte de tots els elements parells de la fila.
c) Suposant que la matriu és quadrada, retornar la suma de tots els elements que no estan a la diagonal.
d) Retornar una nova matriu amb els elements de la matriu original que són més grans que la mitjana de tots els elements de la seva fila. La resta d'elements de la matriu s'han de posar a 0.
e) Suposant que la matriu és quadrada, retornar si la matriu és 'diagonal dominant', és a dir, si tots els elements de la diagonal són més grans que la resta dels elements de la seva fila.
f) Retornar una llista que contingui, per cada fila, el valor més gran i la posició que ocupa dins de la fila.
g) Retornar quantes vegades apareix a la matriu un valor *n* que es passa com a paràmetre.
h) Retornar una nova matriu amb les files *i* i *j* (que es passen com a paràmetres) intercanviades.

Exercici 14 (nivell mig)

Suposem que tenim una matriu *mxn* on cada fila guarda els valors de les notes obtingudes pels *m* estudiants d'una assignatura en cadascun dels *n* lliuraments d'exercicis obligatoris de l'assignatura. Escriure funcions per:

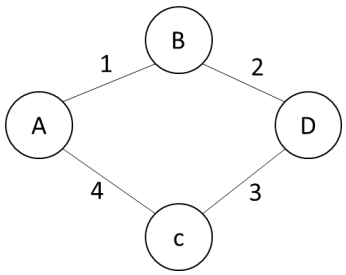
a) Retornar la mitjana de tots els lliuraments de l'estudiant que correspon a la fila *k* (que es passa com a paràmetre), si ha fet tots els lliuraments. Si no els ha fet tots, retornarà 0.
b) Retornar quants alumnes han lliurat tots els exercicis i la mitjana dels seus lliuraments és superior a 5.
c) Donat un número d'exercici que es passa com a paràmetre, retornar el número d'estudiants que l'han lliurat, i la nota mínima, mitjana i màxima de tots els lliuraments fets.
d) Suposant que els lliuraments són setmanals, retornar en una llista el nº d'estudiants que han abandonat l'assignatura en cada setmana. Suposarem que un estudiant abandona l'assignatura a la setmana *k* si a partir d'aquella setmana ja no ha fet cap més lliurament (si un estudiant no ha fet mai cap lliurament, considerarem que abandona a la setmana zero).

Exercici 15 (nivell mig) – EXERCICI AVALUABLE

En aquest exercici heu de fer el mateix que a l'exercici 9, però utilitzant una matriu d'adjacència per representar les arestes del graf, enlloc de llistes d'adjacència.

Com ja sabem en la representació en forma de matriu d'adjacència guardem els nodes en una llista amb les seves etiquetes i les arestes en una matriu on cada element `i`, `j` de la matriu representa si hi ha una aresta entre els nodes amb índex `i` i `j` a la llista de nodes. El valor que es guarda a la matriu és `0` si no hi ha aresta o el valor del pes associat a l'aresta si hi ha una aresta entre els dos nodes.

Igual que a l'exercici 9, a Caronte teniu un fitxer `graf.py` que declara una classe per guardar un graf utilitzant una matriu d'adjacència per guardar la informació de les aretes. Tal com està definida la classe, la informació dels nodes es guarda en una llista on cada element guarda l'etiqueta d'un node. La informació de les arestes es guarda en un array tal com s'ha explicat abans. Així, si tenim aquest graf no dirigit:



la representació del graf dins de la llista serà la següent:

```
nodes: ['A', 'B', 'C', 'D']
arestes: array([[0, 1, 4, 0],
       [1, 0, 0, 2],
       [4, 0, 0, 3],
       [0, 2, 3, 0]])
```

La classe `Graf` que us donem ja té un constructor per inicialitzar un graf i un mètode per mostrar-lo per pantalla. Sobre aquesta classe volem que afegiu els mètodes següents:

a) Un mètode `nodesAillats` que retorni una llista amb les etiquetes dels nodes aïllats del graf (aquells nodes que no tenen cap aresta).
b) Un mètode `nodeMaxArestes` que retorni l'etiqueta del node amb més arestes del graf.
c) Un mètode `etiquetesAdjacents` que donada l'etiqueta d'un node (que es passa com a paràmetre), retorni una llista amb les etiquetes de tots els seus nodes adjacents.
d) Un mètode `cicles` que retorni tots els cicles de 3 nodes del graf (igual que a l'exercici 2 del tema de grafs), sense repetir-ne cap.