Exercices Python

Morpion

Réaliser un jeu de morpion en mode texte

Morpion

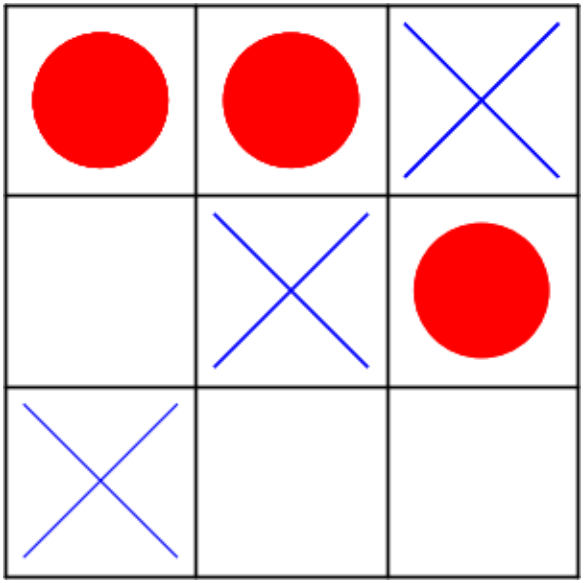
Réaliser un jeu de morpion en mode texte

```
T=[[0,0,0],[0,0,0],[0,0,0]]

joueur_courant=1


def affiche_jeu():
    print()
    for kx in range(3):
        for ky in range(3):
            v=T[kx][ky]
            print(v,end="")
        print()
    print()

def check_gagnant(j):
    for kx in range(3):
        if all(T[kx][k]==j for k in range(3))==True:
            return True
    for ky in range(3):
        if all(T[k][ky]==j for k in range(3))==True:
            return True

    if all(T[k][k]==j for k in range(3))==True:
        return True

    if all(T[2-k][k]==j for k in range(3))==True:
        return True
```

```
fin=False
while fin!=True:

    print("Joueur",joueur_courant)

    x=int(input("x="))
    y=int(input("y="))

    if x>=0 and x<=2 and y>=0 and y<=2:

        if T[x][y]==0:
            T[x][y]=joueur_courant

            affiche_jeu()
            fin=check_gagnant(joueur_courant)

            if fin==True:
                print("Joueur",joueur_courant,"a gagne")
            else:
                if joueur_courant==1:
                    joueur_courant=2
                else:
                    joueur_courant=1

        else:
            print("Case",x,",",y,"deja occupee")


    else:
        print("Coordonnees incorrectes")
```

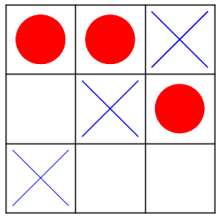
Morpion

Réaliser un jeu de morpion en mode graphique



Morpion

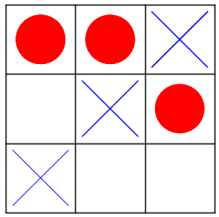
Réaliser un jeu de morpion en mode graphique



```
class window:

    def __init__(self,figure):
        self.T=[[0,0,0],[0,0,0],[0,0,0]]
        self.current=1
        self.figure=figure
        self.draw()
    def connect(self):
        self.press=self.figure.canvas.mpl_connect("button_press_event",self.click_so
        uris)

    def click_souris(self,evenement):
        x,y=math.floor(evenement.xdata),math.floor(evenement.ydata)
        if x>=0 and x<=2 and y>=0 and y<=2:
            if self.T[x][y]==0:
                self.T[x][y]=self.current

                if self.check_winner()==True:
                    print("Joueur",self.current,"a gagne !")


                if self.current==1:
                    self.current=2
                else:
                    self.current=1

                self.draw()

            else:
                print("Case deja prise")
        else:
            print("Clique hors champ")
```

Morpion

Réaliser un jeu de morpion en mode graphique

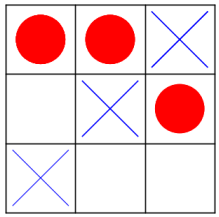


```
class window:

    def __init__(self,figure):
        self.T=[[0,0,0],[0,0,0],[0,0,0]]
        self.current=1
        self.figure=figure
        self.draw()
    def connect(self):
        self.press=self.figure.canvas.mpl_con
        uris)

    def click_souris(self,evenement):
        x,y=math.floor(evenement.xdata),math.
        if x>=0 and x<=2 and y>=0 and y<=2:
            if self.T[x][y]==0:
                self.T[x][y]=self.current

                if self.check_winner()==True:
                    print("Joueur",self.curre


                if self.current==1:
                    self.current=2
                else:
                    self.current=1

                self.draw()

            else:
                print("Case deja prise")
        else:
            print("Clique hors champ")
```

```
def draw(self):
    for k in range(4):
        plt.plot([0,3],[k,k],'k-')
        plt.plot([k,k],[0,3],'k-')

    for kx in range(3):
        for ky in range(3):
            if self.T[kx][ky]==1:
                self.affiche_cercle(kx+0.5,ky+0.5)
            elif self.T[kx][ky]==2:
                self.affiche_croix(kx,ky)

    plt.axis("equal")
    plt.axis([-1,4,-1,4])
    plt.draw()


def affiche_cercle(self,x,y):
    circle=plt.Circle((x,y),radius=0.35,color="r")
    self.figure.gca().add_patch(circle)

def affiche_croix(self,x,y):
    e=0.1
    plt.plot([x+e,x+1-e],[y+e,y+1-e],"b")
    plt.plot([x+e,x+1-e],[y+1-e,y+e],"b")
```

Morpion

Réaliser un jeu de morpion en mode graphique



```
class window:

    def __init__(self,figure):
        self.T=[[0,0,0],[0,0,0],[0,0,0]]
        self.current=1
        self.figure=figure
        self.draw()
    def connect(self):
        self.press=self.figure.canvas.mpl_con
        uris)

    def click_souris(self,evenement):
        x,y=math.floor(evenement.xdata),math.
        if x>=0 and x<=2 and y>=0 and y<=2:
            if self.T[x][y]==0:
                self.T[x][y]=self.current

                if self.check_winner()==True:
                    print("Joueur",self.curre


                if self.current==1:
                    self.current=2
                else:
                    self.current=1

                self.draw()

            else:
                print("Case deja prise")
        else:
            print("Clique hors champ")
```

```
def draw(self):
    for k in range(4):
        plt.plot([0,3],[k,k],'k-')
        plt.plot([k,k],[0,3],'k-')

    for kx in range(3):
        for ky in range(3):
            if self.T[kx][ky]==1:
```

```
def check_winner(self):
    for kx in range(3):
        if all(self.T[kx][k]==self.current for k in range(3))==True:
            return True
    for ky in range(3):
        if all(self.T[k][ky]==self.current for k in range(3))==True:
            return True

    if all(self.T[k][k]==self.current for k in range(3))==True:
        return True

    if all(self.T[2-k][k]==self.current for k in range(3))==True:
        return True
```

```
fig=plt.figure()
win=window(fig)

win.connect()

plt.show()
```

Application

Créez une classe de polynome creux qui

- Ne sauvegarde que les valeurs des coefficients indiqués (les autres sont à 0)

=> Méthode append() permet d'ajouter un coefficient

- Est évalué en un point x avec la méthode eval(x)

- Permet l'addition de polynomes creux
- Permet la multiplication entre polynomes creux

Application

Créez une classe de polynome creux

```
class sparse_polynomial:
    def __init__(self):
        self.data={}
    def append(self,value):
        self.data[value[0]]=value[1]
    def eval(self,x):
        return sum([pow(x,k)*w for k,w in zip(self.data.keys(),self.data.values())])
    def __str__(self):
        s=""
        for k in self.data.keys():
            s+=str(self.data[k])+"x^"+str(k)+" + "
        s=s.replace("+ -","- ")
        return s[:-2]

    def __add__(self,polynomial):
        p=sparse_polynomial()
        p.data=self.data.copy()
        for coeff in polynomial.data.keys():
            w=polynomial.data[coeff]
            if coeff in p.data:
                p.data[coeff] += w
            else:
                p.append((coeff,w))
        return p
    def __mul__(self,polynomial):
        p=sparse_polynomial()
        for coeff1 in self.data.keys():
            for coeff2 in polynomial.data.keys():
                coeff=coeff1+coeff2
                w=self.data[coeff1]*polynomial.data[coeff2]
                if coeff in p.data:
                    p.data[coeff] += w
                else:
                    p.append((coeff,w))
        return p
```

```
p1=sparse_polynomial()
p1.append((4,6))
p1.append((152,-0.04))
p1.append((78,-1.1))

p2=sparse_polynomial()
p2.append((7,4.5))
p2.append((2,2))

p3=p1*p2
print(p3)
```