Aleksandra Zawadka i Martyna Sadowska

Projekt badawczy

Algebra gry SET

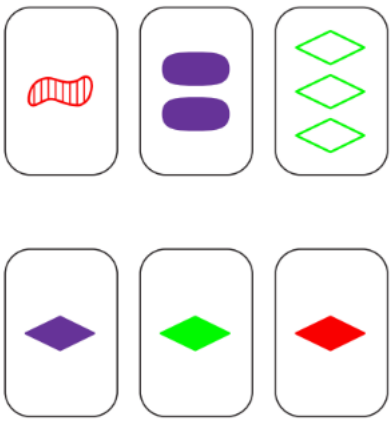
Zasady gry SET

SET to gra karciana, która polega na szukaniu trzech kart tworzących SET. Karty mają 4 cechy, a każda z cech 3 możliwe wartości:

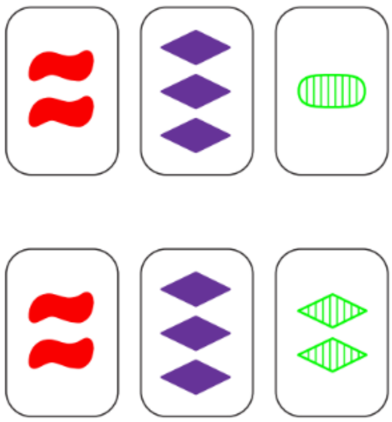
Cechy			
Kolor	Zielony	Czerwony	Fioletowy
Kształt	Zygzak	Romb	Owal
Liczba	1	2	3
Wypełnienie	Pełne	Puste	Paski

3 karty tworzą SET, jeśli dla danej cechy jej wartości są wszystkie różne lub wszystkie takie same.

Przykłady kart, które tworzą SET:



Przykłady kart, które nie tworzą SETa:



Jest dokładnie jedna karta dla każdej kombinacji wartości cech, więc kart w talii jest $3^4 = 81$. W związku z tym dla każdych dwóch kart z talii jest tylko jedna konkretna karta, z którą stworzą SET.

Typowa partia gry polega na rozłożeniu 12 kart na środku stołu. Każdy gracz szuka w tych kartach SETa, a gdy go znajdzie, mówi głośno *SET!* i zabiera karty ze stołu. W miejsca zabranych kart dokładane są nowe i gra trwa dalej. Jeżeli wśród wyłożonych kart nie ma żadnego SETa, to z puli dokładane są 3 karty aż będzie jakiś SET. Gra kończy się, gdy nie ma kart do dobrania, a wygrywa osoba, która znalazła najwięcej SETów.

Współrzędne w grze SET

Każdą kartę z talii można przedstawić jako wektor o czterech współrzędnych, które należą do ciała Z_3. Wtedy każda współrzędna opisuje jedną z 4 cech karty, a jej wartości odpowiadają jednej z 3 możliwych wartości cechy.

Jako nasze pierwsze zadanie udowodnimy, że korzystając z tych oznaczeń, punkty w płaszczyźnie tworzą linie wtedy i tylko wtedy, gdy suma ich wektorów jest równa 0.

<u>Dowód</u>

- →
  Weźmy dowolne trzy punkty z płaszczyzny SET, które tworzą SET.
  Jeżeli wartości pewnej ich cechy są wszystkie takie same, to suma w Z_3 dla tej współrzędnej jest równa $3 \cdot x$, gdzie x jest wartością ze zbioru {0, 1, 2}, co daje 0.
  Jeżeli wartości pewnej ich cechy są wszystkie różne, to jedyna możliwa suma to 0+1+2. W Z_3 jest ona równa 0.
  Wiemy, że dla każdej współrzędnej suma jest równa 0, więc cała suma jest równa 0.
- ←
  Weźmy dowolne trzy punkty z płaszczyzny SET, których suma w Z_3 jest równa 0. Jedyne możliwości otrzymania 0 w dodawaniu trzech elementów w Z_3, to 0+1+2 lub x + x + x, gdzie x jest wartością ze zbioru {0, 1, 2}. W pierwszym przypadku oznacza to, że każda wartość cechy jest inna, a w drugim, że wartość cechy dla każdej karty jest taka sama. Wtedy te trzy karty tworzą SET.

Co kończy dowód.

Operacja binarna na kartach SET

Zdefiniujmy binarną operację * na kartach z talii gry SET, która dla dowolnych dwóch odrębnych kart x, y, zwraca trzecią kartę z, z którą tworzą SET. x * y = z wtedy i tylko wtedy, gdy x, y i z tworzą SET.

Dla dowolnej karty x definiujemy x * x = x.

W artykule „Algebra From Geometry in the Card Game SET” zostało już pokazane, że:

- $x * y = y * x$
- $x * y = z \leftrightarrow z * x = y \leftrightarrow z * y = x$
- $x * y = x * z \rightarrow y = z$
- $(x * y) * z \neq x * (y * z)$

Przestrzeń SET

Przestrzeń gry SET możemy przedstawić w sposób algebraiczny. Karty będziemy oznaczać jako punkty, a karty tworzące razem SET będą połączone linią. Ponieważ dwie karty mają zawsze określoną trzecią, z którą tworzą SET, to dwa punkty wyznaczają już przebieg linii. W ten sposób cała przestrzeń SET to czterowymiarowa przestrzeń zawierająca 81 punktów.

<u>Płaszczyzna</u> w grze SET to minimalny zbiór kart, w którym:

- są trzy odrębne karty, które nie tworzą SETa,
- dla każdych dwóch kart trzecia tworząca z nimi SET jest w tym zbiorze.

Płaszczyzna jest więc minimalnym zbiorem punktów z przestrzeni SET, który zawiera przynajmniej trzy punkty nieleżące na jednej linii oraz dowolne dwa punkty są połączone linią. Płaszczyzna jest dwuwymiarowym podzbiorem przestrzeni SET.

<u>Hiperpłaszczyzna</u> w grze SET to minimalny zbiór kart, w którym:

- są cztery odrębne karty, które nie leżą na tej samej płaszczyźnie,
- dla każdych dwóch kart trzecia tworząca z nimi SET jest w tym zbiorze.

Hiperpłaszczyzna jest więc minimalnym zbiorem punktów z przestrzeni SET, który zawiera przynajmniej cztery punkty nieleżące w jednej płaszczyźnie oraz dowolne dwa punkty są połączone linią. Hiperpłaszczyzna jest trójwymiarowym podzbiorem przestrzeni SET.

Jako nasze kolejne zadanie udowodnimy, że płaszczyzna zawsze zawiera 9 punktów.

Dowód

Punkty naszej płaszczyzny to 4-elementowe wektory z ciała Z_3, więc ich elementy mają jedną z trzech wartości: 0, 1, lub 2. Płaszczyzna to dwuwymiarowy podzbiór przestrzeni SET. Jest ona rozpięta przez dwa niezależne punkty, ponieważ określają one linię. Liczba możliwych kombinacji liniowych tych wektorów będzie liczbą punktów na płaszczyźnie.

Oznaczmy te wektory przez v_1 i v_2. Kombinacją liniową będzie wektor $v = a \cdot v_1 + b \cdot v_2$, gdzie a i b to liczby z ciała Z_3. Każdą z liczb a i b można wybrać na 3 sposoby, stąd możliwych punktów jest $3 \cdot 3 = 9$.

Co kończy dowód.

Udowodnimy też, że hiperpłaszczyzna zawiera zawsze 27 punktów.

Dowód

Punkty naszej hiperpłaszczyzny to 4-elementowe wektory z ciała Z_3, więc ich elementy mają jedną z trzech wartości: 0, 1, lub 2. Hiperpłaszczyzna to trójwymiarowy podzbiór przestrzeni SET. Jest ona rozpięta przez trzy niezależne punkty. Liczba możliwych kombinacji liniowych tych wektorów będzie liczbą punktów na płaszczyźnie.

Oznaczmy te wektory przez v_1, v_2, v_3. Kombinacją liniową będzie wektor $v = a \cdot v_1 + b \cdot v_2 + c \cdot v_3$, gdzie a, b i c to liczby z ciała Z_3. Każdą z liczb a, b i c można wybrać na 3 sposoby, stąd możliwych punktów jest $3 \cdot 3 \cdot 3 = 27$.

Co kończy dowód.

Teraz udowodnimy, że na każdej płaszczyźnie jest 12 linii.

Dowód

Wiemy już, że płaszczyzna składa się z 9 punktów oraz że dla dowolnych dwóch punktów trzeci jest jednoznacznie wyznaczony.

Policzmy, na ile sposobów można wybrać pary punktów z tych 9:

$$\binom{9}{2} = 36$$

Każda para definiuje jedną linię. Jednakże w ten sposób wybieramy trzy razy ten sam SET, więc nasz wynik to:

$$\frac{36}{3} = 12$$

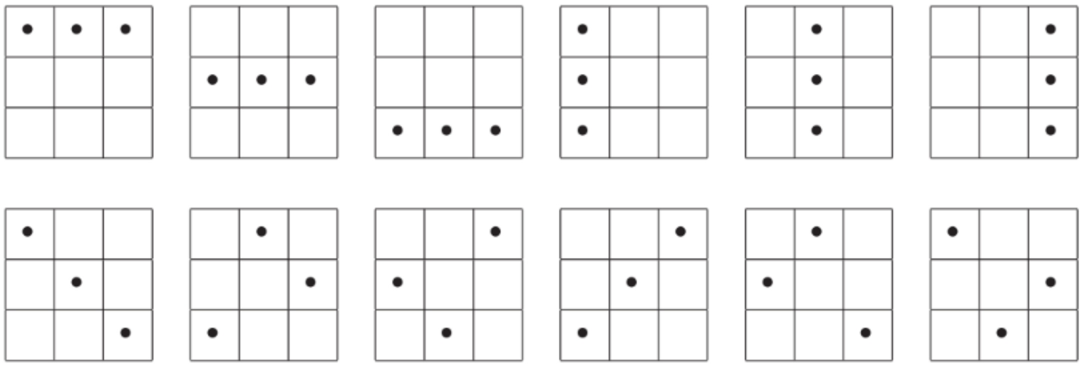
Co kończy dowód

Płaszczyzna SET

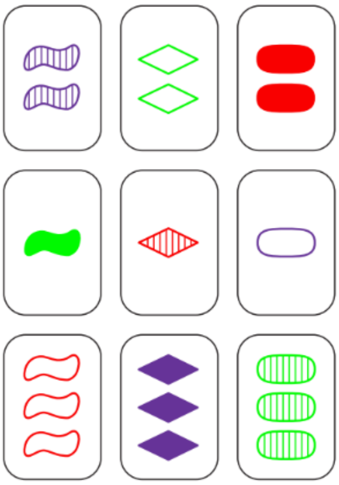
Elementy płaszczyzny SET możemy ułożyć w następujący sposób:

x	y	x*y
z	x*(y*z)	y*(x*z)
x*z	z*(x*y)	y*z

Wtedy linie łączące punkty układają się następująco:



Przykładowa płaszczyzna SET:



Policzymy teraz, ile jest wszystkich płaszczyzn i hiperpłaszczyzn w przestrzeni SET.

Wiemy, że płaszczyzna składa się z 9 kart, gdzie 3 z nich nie tworzą SETa.

Wylosujmy najpierw te trzy karty. Wszystkich kart jest 81, więc pierwszą kartę możemy wylosować na 81 sposobów. Drugą losujemy z pozostałych 80 kart. Zostaje nam 79 kart, ale jedna z nich tworzy z już wylosowanymi SET, więc trzecią kartę losujemy na 78 sposobów. Reszta kart w płaszczyźnie jest już określona. Mamy $81 \cdot 80 \cdot 78$.

Losując w ten sposób określamy miejsca dla trzech początkowych kart, więc wynik musimy podzielić przez ich możliwe ustawienia.

Pierwszą kartę możemy ułożyć na 9 sposobów. Dla drugiej karty zostaje 8 miejsc. Dla trzeciej karty zostaje 6 miejsc, ponieważ jedna, tworząca SET z kartami 1 i 2, jest już na płaszczyźnie.

Wychodzi więc, że mamy tyle płaszczyzn:

$$\frac{81 \cdot 80 \cdot 78}{9 \cdot 8 \cdot 6} = 1170$$

Wiemy już, że na hiperpłaszczyźnie jest 27 kart, gdzie 4 z nich nie leżą na jednej płaszczyźnie.

Wybieramy pierwsze trzy karty, tak jak do płaszczyzny, na $81 \cdot 80 \cdot 78$ sposobów. Czwarta karta będzie dowolną, która nie należy do płaszczyzny, którą tworzą wybrane przez nas karty, więc możemy ją wybrać na (81 - 9) = 72 sposoby.

Losując w ten sposób określamy miejsca dla czterech początkowych kart, więc wynik musimy podzielić przez ich możliwe ustawienia.

Pierwszą kartę możemy ułożyć na 27 sposobów. Dla drugiej karty zostaje 26 miejsc, dla trzeciej już 24, ponieważ karta, z którą tworzą SET pierwsze dwie, już jest na hiperpłaszczyźnie. Ułożenie kolejnych 5 kart jest określone przez płaszczyznę, którą tworzą pierwsze trzy karty, więc dla karty czwartej pozostaje (27 – 9) = 18 miejsc do wyboru.

Stąd wszystkich hiperpłaszczyzn jest:

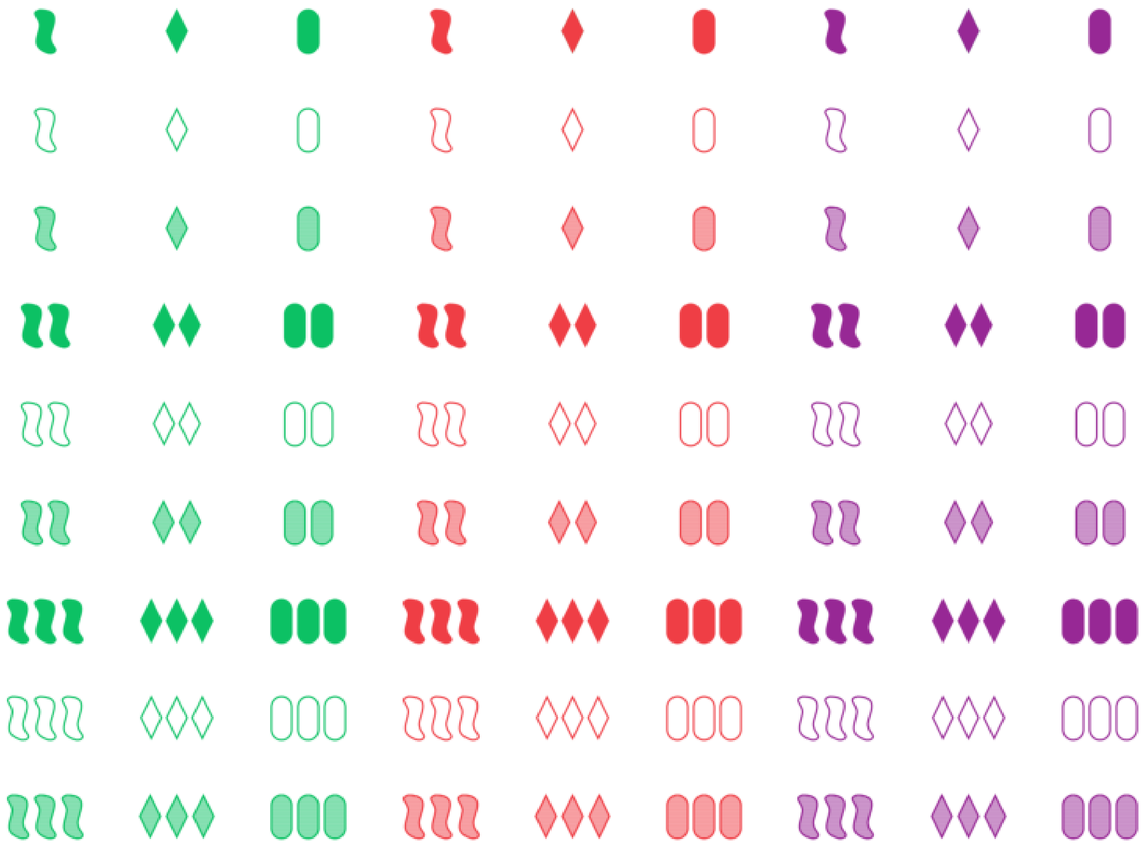
$$\frac{81 \cdot 80 \cdot 78 \cdot 72}{27 \cdot 26 \cdot 24 \cdot 18} = 120$$

Programy w Pythonie

Na początku wygenerujemy całą przestrzeń SET

```
def generate_cards():
    cards = [[0]*9 for i in range(9)]
    i = 0
    j = 0
    for color in range(3):
        for shape in range(3):
            for number in range(3):
                for shade in range(3):
                    card = [color,shape,number,shade]
                    cards[i][j] = card
                    i += 1
            i = 0
            j += 1
    return cards


1 usage
def show_images(cards):
    f, axarr = plt.subplots( nrows: 9,  ncols: 9)
    for i in range(9):
        for j in range(9):
            image_datas = ("img//" + str(colors[cards[i][j][0]]) + str(shape[cards[i][j][1]])
                           + str(shading[cards[i][j][3]]) + str(number[cards[i][j][2]]) + ".png")
            axarr[i][j].imshow(mpimg.imread(image_datas))
            axarr[i][j].axis("off")
    plt.show()


1 usage
def main():
    #random.seed(123)
    cards = generate_cards()
    show_images(cards)
```

Przestrzeń SET



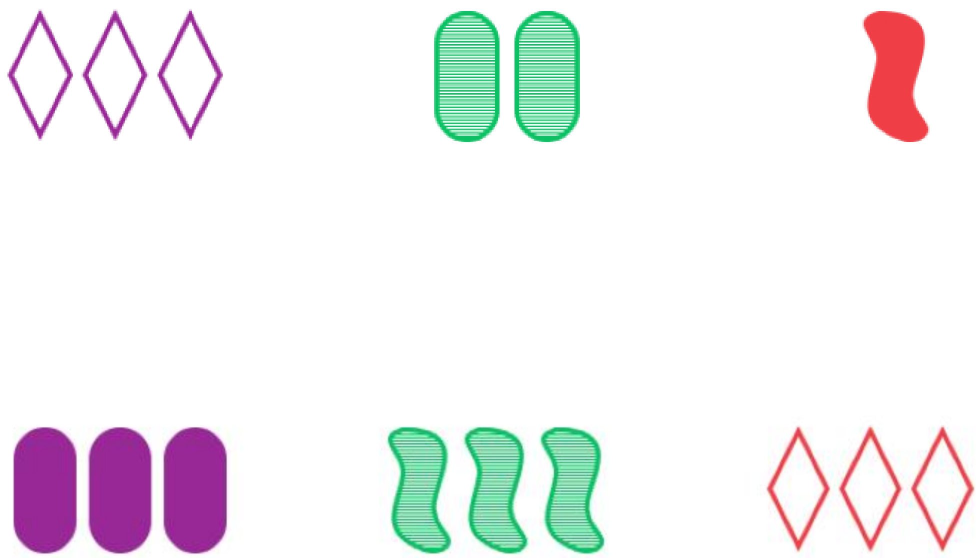
Korzystając ze współrzędnych w Z_3, możemy napisać program, który dla dwóch losowych kart znajduje trzecią, z którą tworzą SET.

```python
import random
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

colors = {
    0: "green",
    1: "red",
    2: "purple"
}
number = {
    0: 1,
    1: 2,
    2: 3
}
shape = {
    0: "squiggle",
    1: "diamond",
    2: "oval"
}
shading = {
    0: "filled",
    1: "empty",
    2: "shaded"
}
```

```python

1 usage
def random_cards():
    card_1 = [0, 0, 0, 0]
    card_2 = [0, 0, 0, 0]
    while card_1 == card_2:
        for i in range(4):
            card_1[i] = random.randint( a: 0,  b: 2)
            card_2[i] = random.randint( a: 0,  b: 2)
    return card_1, card_2


3 usages
def print_card(karta):
    print("Card:")
    print(f"- color: {colors[karta[0]]}")
    print(f"- shape: {shape[karta[1]]}")
    print(f"- number: {number[karta[2]]}")
    print(f"- shading: {shading[karta[3]]}")
    print()


```

```python
1 usage
def find_set(card_1, card_2):
    card_3 = [0, 0, 0, 0]
    for i in range(4):
        card_3[i] = (3 - card_1[i] - card_2[i]) % 3
    return card_3


1 usage
def show_images(cards):
    f, axarr = plt.subplots( nrows: 1,  ncols: 3)
    image_datas = [0, 0, 0]
    for i in range(3):
        image_datas[i] = ("img//" + str(colors[cards[i][0]])
                          + str(shape[cards[i][1]]) + str(shading[cards[i][3]])
                          + str(number[cards[i][2]]) + ".png")
        axarr[i].imshow(mpimg.imread(image_datas[i]))
        axarr[i].axis("off")
    plt.show()

```

```python
1 usage
def main():
    #random.seed(123)
    card_1, card_2 = random_cards()
    print("Two random cards:")
    print_card(card_1)
    print_card(card_2)
    card_3 = find_set(card_1, card_2)
    print("Card that creates a SET:")
    print_card(card_3)
    cards = [card_1, card_2, card_3]
    show_images(cards)


if __name__ == "__main__":
    main()
```



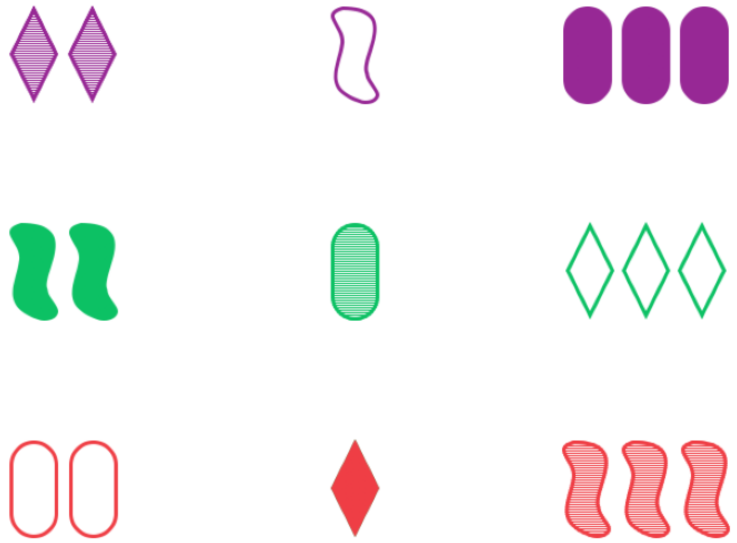
Korzystając teraz z poprzednio ustalonej tabelki możemy napisać program, który dla trzech losowych kart, które nie tworzą SETa, znajduje płaszczyznę.

```python
def check_set(cards):
    for i in range(4):
        if (cards[0][i] + cards[1][i] + cards[2][i]) % 3 != 0:
            return False
    return True


def random_cards():
    card_1 = [0, 0, 0, 0]
    card_2 = [0, 0, 0, 0]
    card_3 = [0, 0, 0, 0]
    cards = [card_1, card_2, card_3]
    while (card_1 == card_2) or (card_1 == card_3) or (card_2 == card_3) or check_set(cards):
        for i in range(4):
            card_1[i] = random.randint(0, 2)
            card_2[i] = random.randint(0, 2)
            card_3[i] = random.randint(0, 2)
    return cards
```

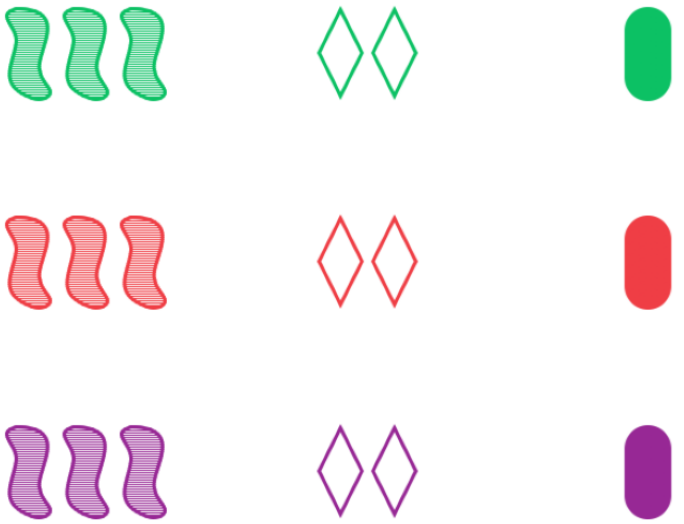
```python
def print_card(karta):
    print("Card:")
    print(f"- color: {colors[karta[0]]}")
    print(f"- shape: {shape[karta[1]]}")
    print(f"- number: {number[karta[2]]}")
    print(f"- shading: {shading[karta[3]]}")
    print()


def find_set(card_1, card_2):
    card_3 = [0, 0, 0, 0]
    for i in range(4):
        card_3[i] = (3 - card_1[i] - card_2[i]) % 3
    return card_3


def show_images(cards):
    f, axarr = plt.subplots(3, 3)
    for i in range(3):
        for j in range(3):
            image_datas = ("img//" + str(colors[cards[i][j][0]]) + str(shape[cards[i][j][1]])
                           + str(shading[cards[i][j][3]]) + str(number[cards[i][j][2]]) + ".png")
            axarr[i][j].imshow(mpimg.imread(image_datas))
            axarr[i][j].axis("off")
    plt.show()
```

```python
def find_plane(cards):
    plane = [[0]*3 for i in range(3)]
    plane[0][0] = cards[0]
    plane[0][1] = cards[1]
    plane[0][2] = find_set(cards[0], cards[1])
    plane[1][0] = cards[2]
    plane[2][0] = find_set(cards[0], cards[2])
    plane[2][2] = find_set(cards[1], cards[2])
    plane[1][1] = find_set(cards[0], plane[2][2])
    plane[1][2] = find_set(cards[1], plane[2][0])
    plane[2][1] = find_set(cards[2], plane[0][2])
    return plane


1 usage
def main():
    #random.seed(123)
    cards = random_cards()
    print("Three random cards:")
    print_card(cards[0])
    print_card(cards[1])
    print_card(cards[2])
    plane = find_plane(cards)
    print("Plane:")
    for i in range(3):
        for j in range(3):
            print_card(plane[i][j])
    show_images(plane)


if __name__ == "__main__":
    main()
```

Pierwsza przykładowa płaszczyzna:



Druga przykładowa płaszczyzna:



Źródła:

- Algebra From Geometry in the Card Game SET - Timothy E. Goldberg
- https://aperiodical.com/2018/07/the-mathematical-beauty-of-the-game-set/
- https://en.wikipedia.org/wiki/Set_(card_game)