In [1]:

```
class Card:
    suits = ["Clubs", "Diamonds", "Hearts", "Spades"]
    ranks = ["None","Ace","2","3","4","5","6","7","8","9","10","Jack","Queen","King"]

    def __init__(self, suit, rank):
        self.suit = suit
        self.rank = rank

    def __str__(self):
        return f" {Card.ranks[self.rank]}{Card.suits[self.suit]}"

    def __lt__(self, other):
        if self.rank == other.rank:
            return self.suit < other.suit
        else:
            return self.rank < other.rank

    def __gt__(self,other):
        if self.suit > other.suit:
            return True
        elif self.suit == other.suit:
            if self.rank > other.rank:
                return True
        return False

    def __eq__(self, other):
        return (self.rank == other.rank and self.suit == other.suit)

c1 = Card(1,3)
print(c1)
```

```
3Diamonds
```

In [2]:

```
import random

class Deck:
    def __init__(self):
        self.deck = []
        for suit in range(4):
            for rank in range(1,14):
                self.deck.append(Card(suit, rank))
        self.shuffle()


    def __str__(self):
        s = ""
        for i in range(len(self.deck)):               #You can put 52 but you can have multiple decks
            s += i * " " + str(self.deck[i]) + "\n"
        return s

    def __len__(self):
        return len(self.deck)

    def add_card(self, card):
        self.deck.append(card)

    def pop_card(self):
        return self.deck.pop()

    def shuffle(self):
        n_cards = len(self.deck)
        for i in range(n_cards):
            j = random.randrange(0,n_cards)
            self.deck[i],self.deck[j] = self.deck[j],self.deck[i]

    def is_empty(self):
        return len(self.cards) == 0


    def deal(self, hands, n_cards = 52):
        n_players = len(hands)
        for i in range(n_cards):
            if self.is_empty():                             #self is the deck
                break
            card = self.pop_card()
            current_player = i % n_players
            hands[current_player].add_card(card)
```

In [3]:

```
class Hand(Deck):
    def __init__(self,name):
        self.deck = []
        self.name = name
        self.win_count = 0


    def __str__(self):
        #input()
        return self.name + ' -> ' + ' '.join([str(card) for card in self.deck])

        #return self.label.join([str(card) for card in self.deck])


    #def __str__(self):
        #s = "Hand of " + self.label
        #if self.is_empty():
            #return s + " is empty"
        #s += " Contains \n" + Deck.__str__(self)        # override the class deck str
        #return s

    def label(self):
        return self.name

    def wincount(self):
        return self.win_count

    def roundwinner(self):
        self.win_count = self.win_count + 1
```

In [19]:

```
import time
import pandas as pd
import numpy as np


deck = Deck()

hands = []
players=['Deep','Dilip','Dhruv','Nancy']
for i in players:
    hands.append(Hand(f'{i}'))


while len(deck) > 0:
    for hand in hands:
        hand.add_card(deck.pop_card())

#print(hands[0])
fd=[]

for i in range(12):                 #range can be 13 if you want to complete all the cards
    #input()
    #time.sleep(1)
    pcards = []                     #append the list with playerd cards
    for hand in hands:
        pcards.append(hand.pop_card())
        #print(pcards)

    wincard = max(pcards)

    whand = hands[pcards.index(wincard)]
    print(whand)
    whand.roundwinner()
    whand.wincount()



    print(f"\nRound{i}: " + ' '.join([str(card) for card in pcards]) + f' Winner:{whand.label()} {str(wincard)}'"\n")



    d1=[]
    for hand in hands:
        print(f"Score for {hand.label()}: {hand.wincount()}")
        d1.append(hand.wincount())



    #print(d1)
    fd.append(d1)
    #df=pd.DataFrame.from_dict(fd, orient='index', columns=['total win'])
    #print(df)
    #print(fd)


data = np.array(fd)

d_dataset={}

for d in enumerate(players):
    d_dataset[d[1]]=data[:, d[0]]


dataset = pd.DataFrame(d_dataset)
#print(fd[-1])
```

```
Nancy ->  7Spades  QueenHearts  8Hearts  2Hearts  10Hearts  6Hearts  QueenDiamonds  JackDiamonds  8Diamonds  King
Diamonds  8Clubs  5Hearts

Round0:  10Diamonds  9Hearts  10Clubs  JackSpades Winner:Nancy  JackSpades

Score for Deep: 0
Score for Dilip: 0
Score for Dhruv: 0
Score for Nancy: 1
Deep ->  4Diamonds  KingHearts  10Spades  3Spades  9Clubs  JackClubs  5Clubs  6Diamonds  2Clubs  QueenSpades  9Sp
ades

Round1:  5Spades  4Spades  6Clubs  5Hearts Winner:Deep  5Spades

Score for Deep: 1
Score for Dilip: 0
Score for Dhruv: 0
Score for Nancy: 1
Deep ->  4Diamonds  KingHearts  10Spades  3Spades  9Clubs  JackClubs  5Clubs  6Diamonds  2Clubs  QueenSpades

Round2:  9Spades  QueenClubs  8Spades  8Clubs Winner:Deep  9Spades

Score for Deep: 2
Score for Dilip: 0
Score for Dhruv: 0
Score for Nancy: 1
Deep ->  4Diamonds  KingHearts  10Spades  3Spades  9Clubs  JackClubs  5Clubs  6Diamonds  2Clubs

Round3:  QueenSpades  4Hearts  KingClubs  KingDiamonds Winner:Deep  QueenSpades

Score for Deep: 3
Score for Dilip: 0
Score for Dhruv: 0
Score for Nancy: 1
Nancy ->  7Spades  QueenHearts  8Hearts  2Hearts  10Hearts  6Hearts  QueenDiamonds  JackDiamonds

Round4:  2Clubs  7Diamonds  AceDiamonds  8Diamonds Winner:Nancy  8Diamonds

Score for Deep: 3
Score for Dilip: 0
Score for Dhruv: 0
Score for Nancy: 2
Dilip ->  AceHearts  3Diamonds  9Diamonds  3Hearts  4Clubs  3Clubs  JackHearts

Round5:  6Diamonds  AceSpades  2Diamonds  JackDiamonds Winner:Dilip  AceSpades

Score for Deep: 3
Score for Dilip: 1
Score for Dhruv: 0
Score for Nancy: 2
Dhruv ->  AceClubs  7Hearts  7Clubs  5Diamonds  KingSpades  6Spades

Round6:  5Clubs  JackHearts  2Spades  QueenDiamonds Winner:Dhruv  2Spades

Score for Deep: 3
Score for Dilip: 1
Score for Dhruv: 1
Score for Nancy: 2
Dhruv ->  AceClubs  7Hearts  7Clubs  5Diamonds  KingSpades

Round7:  JackClubs  3Clubs  6Spades  6Hearts Winner:Dhruv  6Spades

Score for Deep: 3
Score for Dilip: 1
Score for Dhruv: 2
Score for Nancy: 2
Dhruv ->  AceClubs  7Hearts  7Clubs  5Diamonds

Round8:  9Clubs  4Clubs  KingSpades  10Hearts Winner:Dhruv  KingSpades

Score for Deep: 3
Score for Dilip: 1
Score for Dhruv: 3
Score for Nancy: 2
Deep ->  4Diamonds  KingHearts  10Spades

Round9:  3Spades  3Hearts  5Diamonds  2Hearts Winner:Deep  3Spades

Score for Deep: 4
Score for Dilip: 1
Score for Dhruv: 3
Score for Nancy: 2
Deep ->  4Diamonds  KingHearts

Round10:  10Spades  9Diamonds  7Clubs  8Hearts Winner:Deep  10Spades

Score for Deep: 5
Score for Dilip: 1
Score for Dhruv: 3
Score for Nancy: 2
Deep ->  4Diamonds
```

```
Round11:  KingHearts  3Diamonds  7Hearts  QueenHearts Winner:Deep  KingHearts

Score for Deep: 6
Score for Dilip: 1
Score for Dhruv: 3
Score for Nancy: 2
```

In [20]:

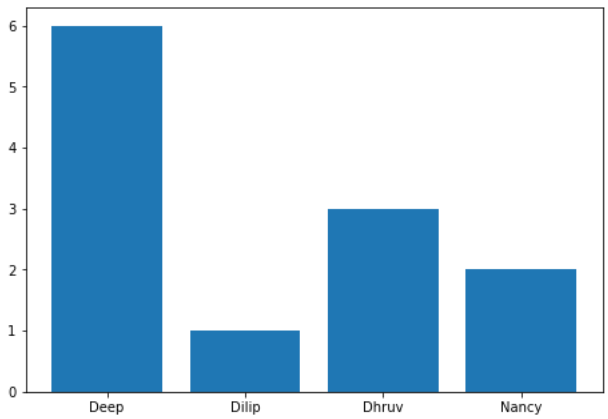
```
display(dataset)
```

	Deep	Dilip	Dhruv	Nancy
0	0	0	0	1
1	1	0	0	1
2	2	0	0	1
3	3	0	0	1
4	3	0	0	2
5	3	1	0	2
6	3	1	1	2
7	3	1	2	2
8	3	1	3	2
9	4	1	3	2
10	5	1	3	2
11	6	1	3	2

In [21]:

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
students =fd[-1]
ax.bar(players,students)
plt.show()
```

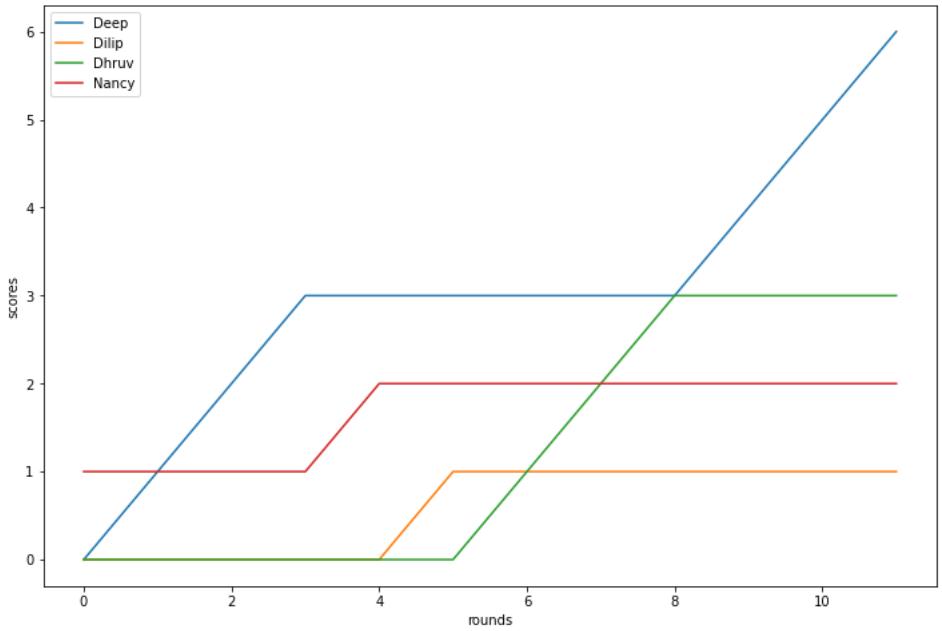


In [22]:

```
%matplotlib inline
dataset.plot(xlabel='rounds',ylabel='scores',figsize=(12,8))
```

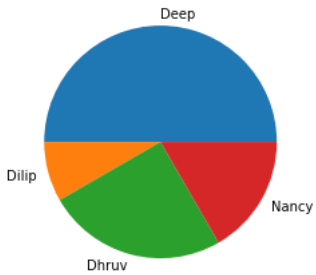
Out[22]:

```
<AxesSubplot:xlabel='rounds', ylabel='scores'>
```



In [23]:

```
plt.pie(fd[-1],labels = players)
plt.show()
```



In [ ]: