In [6]:
```
a=[[1,2]]
for i in range(len(a)):
    a[i].reverse()
for i in a:
    print(*i)
```
```
2 1
```

In [3]:
```
pip install translate
```
```
Collecting translate
  Downloading translate-3.6.1-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: click in c:\users\murthy\anaconda3\lib\site-packages (from translate) (8.0.3)
Requirement already satisfied: lxml in c:\users\murthy\anaconda3\lib\site-packages (from translate) (4.6.3)
Collecting libretranslatepy==2.1.1
  Downloading libretranslatepy-2.1.1-py3-none-any.whl (3.2 kB)
Requirement already satisfied: requests in c:\users\murthy\anaconda3\lib\site-packages (from translate) (2.26.0)
Requirement already satisfied: colorama in c:\users\murthy\anaconda3\lib\site-packages (from click->translate) (0.4.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\murthy\anaconda3\lib\site-packages (from requests->translate) (3.2)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\murthy\anaconda3\lib\site-packages (from requests->translate) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\murthy\anaconda3\lib\site-packages (from requests->translate) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\murthy\anaconda3\lib\site-packages (from requests->translate) (2021.10.8)
Installing collected packages: libretranslatepy, translate
Successfully installed libretranslatepy-2.1.1 translate-3.6.1
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
from translate import Translator
translator=Translator(to_lang="Telugu")
translation=translator.translate("Good Morning")
print(translation)
```
```
శుభోదయం
```

In [6]:
```
def m1(a,b): # 1st m1 defined here
    print(a+b)
def m1(a,b,c): # later changed the m1 method
    print(a*b*c)
print(m1(10,20)) # same method name , diff signatures -->it is method overloading
print(m1(10,20,30))

# same method signature ,different classes --> method overriding
class A:
    def m1(self):
        print("in class A")
class B(A):
    def m1(self):
        print("in class B")
obj=B()
```
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_17072/3087974266.py in <module>
      3 def m1(a,b,c): # later changed the m1 method
      4     print(a*b*c)
----> 5 print(m1(10,20))
      6 print(m1(10,20,30))

TypeError: m1() missing 1 required positional argument: 'c'
```

In [10]:
```
class animal():
    def speaks(method):
        return "Animals speaks in their own language"
class dog(animal):
    def speaks(self):
        return "Dog barks"
class cat(animal):
    def spaeks(self):
        return "Cat meow"
class lion(animal):
    def speaks(self):
        return "Lion roars"
obj=lion()
print(obj.speaks())
```
```
Lion roars
```

In [11]:
```
#there is no key word for abstract class but it has abc package
from abc import ABC,abstractmethod
class Area(ABC): # area class is abstract coz inherited ABC
    @abstractmethod
    def calculate_area(self):
        pass
class Square(Area):
    def calculate_area(self):
        print("in square method")
class Rectabgle(Area):
    def calculate_area(self):
        print("in rectangle method")
ob=Square()
ob.calculate_area
```
Out[11]:
```
<bound method Square.calculate_area of <__main__.Square object at 0x0000022629E92BE0>>
```

In [6]:
```
print(1<<2) # left shift
print(16>>2) # rigth shift 16/2=8/2=4
print(7<<3) # 7*2=14*2=28*2=56
print(11<<3) # 11*2=22*2=44*2=88
print(7>>1) # 7/2=3
```
```
4
4
56
88
3
```

In [7]:
```
# 16 8 4 2 1
#  0 0 0 0 1
#a 0 0 1 0 0 = 2

# 1 0 0 0 0
# 0 0 1 0 0 = 4
```

In [ ]:
```
# TIC-TAC-TOE
def update_board(board,chance,marker,x,y):
  #player1 chance
  if chance==True:
    board[x][y]=marker
    if check_for_win(board):
      print('Player1 wins!!')
      return 'Game Over'
    chance=False
  else:
    board[x][y]=marker
    if check_for_win(board):
      print('Player2 wins!!')
      return 'Game Over'
    chance=True
def play_game():
  player1=0
  player2=0
  m1,m2=get_markers()
  print(f"player 1: {m1}")
  print(f"player 2: {m2}")
  chance=True
  while True:
    print_board(board)
    x,y=Get_coordinates()
    if chance:
      chance=update_board(board,chance,m1,x,y)
      if chance=='Game Over':
        break
    else:
      chance=update_board(board,chance,m2,x,y)
      if chance=='Game Over':
        break
play_game()
def check_for_win(board):
  for row in board:
    if row[0]==row[1] and row[1]==row[2] and row[1]!='':
      return True
  for i in range(len(board)):
    if board[0][i]==board[1][i] and board[1][i]==board[2][i] and board[2][i]!='':
      return True
  if board[0][0]==board[1][1] and board[1][1]==board[2][2] and board[2][2]!='':
    return True
  if board[0][-1]==board[1][1] and board[1][1]==board[2][0]:
    return True
```