پروژه عملی 1: بازی دوز - راهنمای گام به گام ¶

در این بخش، راهنمای گام به گام بازی دوز یا Tic Tac Toe را در اختیار شما قرار می دهیم.

مرور برخی از ابزارهایی که در بخش دست گرمی استفاده کردیم:

دریافت ورودی کاربر:

```
player1 = input("Please pick a marker 'X' or 'O'")
```

توجه نمایید که تابع `input` ورودی کاربر را در قالب رشته برمیگرداند، به همین دلیل برای تبدیل به عدد صحیح از روش زیر برای `cast` استفاده نمایید:

```
position = int(input('Please enter a number'))
```

برای پاک کردن صفحه نمایش میان تعویض نوبت بازی بازیکنان از روش زیر استفاده نمایید:

```
from IPython.display import clear_output
clear_output()
```

توجه نمایید که دستور `clear_output()` تنها در محیط Jupyter Notebook قابل اجرا می باشد و برای IDE های دیگر از روش زیر استفاده نمایید.:

```
print('\n'*100)
```

دستور بالا خروجی کنسول قبلی را از محدوده نمایش خارج می نماید. حالا بریم سراغ کد:

مرحله 1: تابعی بنویسید که بتواند یک تخته را چاپ کند. تخته خود را به عنوان یک لیست تنظیم کنید، به طوری که هر شاخص 1-9 با یک شماره در پد شماره مطابقت داشته باشد، بنابراین نمایش 3 در 3 تخته را دریافت می کنید.

```
   |   |
 X | O | X
   |   |
-----------
   |   |
 O | X | O
   |   |
-----------
   |   |
 X | O | X
   |   |
```

In [ ]:

```
from IPython.display import clear_output

def display_board(board):

    pass
```

تست مرحله 1: تابع خود را با ورودی آزمایش زیر از نقشه بورد مورد آزمایش قرار دهید و در صورت نیاز فاصله ها را تنظیم نمایید.

In [ ]:

```
test_board = ['#','X','O','X','O','X','O','X','O','X']
display_board(test_board)
```

مرحله 2: تابع بنویسید تا ورودی کاربران در قالب یک عدد که مکان علامت X یا O را بیان می کند، دریافت نماید. می توانید از یک حلقه while برای انجام پرسش تا زمان دریافت ورودی صحیح استفاده نمایید.

In [ ]:

```
def player_input():

    pass
```

تست گام دوم: تابع را اجرا نمایید تا از دریافت خروجی مورد نظر اطمینان حاصل نمایید.

In [ ]:

```
player_input()
```

گام سوم: تابعی بنویسید که لیست علامت های روی بورد بازی X یا O و مکان علامت گذاری در بازه صفر تا 9 را دریافت کرده و علامت مورد نظر را در مکان مناسب انتخاب شده قرار دهد.

In [ ]:

```
def place_marker(board, marker, position):

    pass
```

تست مرحله 3: تابع فوق را با ورودی بورد و مکان انتخاب شده آزمایش نمایید.

In [ ]:

```
place_marker(test_board,'$',8)
display_board(test_board)
```

مرحله 4: تابعی بنویسید تا نشانه X یا O را بعنوان ورودی دریافت کرده تشخیص دهد که آیا کاربر برنده شده است یا نه

In [ ]:

```
def win_check(board, mark):

    pass
```

تست مرحله 4: تابع `win_check` را برروی بورد آزمایش خودمان تست کنید - باید `True` را برگرداند.

In [ ]:

```
win_check(test_board,'X')
```

مرحله 5: تابعی بنویسید تا با استفاده از ماژول `random` یک بازیکن را بعنوان نفر اول انتخاب کرده و یک رشته برای این کار چاپ کند. می توانید از تابع `random.randint()` استفاده نمایید.

In [ ]:

```
import random

def choose_first():
    pass
```

مرحله 6:تابعی بنویسید که چک کند آیا فضای خالی در بورد بازی در موقعیت مشخص شده، وجود دارد یا نه

In [ ]:

```
def space_check(board, position):

    pass
```

مرحله 7: تابعی بنویسید تا بررسی کند یا بورد بازی توسط علامت های بازیکن ها پر شده است یا نه؟ `True` برای حالت پر و `False` برای حالت دیگر

In [ ]:

```
def full_board_check(board):

    pass
```

مرحله 8: تابعی بنویسید تا از کاربر موقعیت حرکت بعدی بازیکن را بپرسد (1-9) و سپس از تابع مرحله 6 برای بررسی خالی بودن موقعیت انتخاب شده استفاده نماید و در صورتیکه موقعیت خالی باشد مقدار موقعیت انتخاب شده را برای ادامه بازی برگرداند.

In [ ]:

```
def player_choice(board):

    pass
```

مرحله 9: تابعی بنویسید تا از بازیکن بپرسد که تمایل به انجام بازی مجدد دارد یا نه

In [ ]:

```
def replay():

    pass
```

مرحله 10: حالا به مرحله سخت رسیدیم 😈 سعی کنین تمام منطق بازی را در یک حلقه `while` با شرط همواره درست قرار دهید.

In [ ]:

```
print('Welcome to Tic Tac Toe!')

#while True:
    # Set the game up here
    #pass

    #while game_on:
        #Player 1 Turn


        # Player2's turn.

            #pass

    #if not replay():
        #break
```

دست مریزاد!