

# Lab 3 Task 8 Solution

## 8) Additional Tasks

### 8.1) A user player

```
1  ...
2  class UserPlayer(Player):
3
4  def move(self, current: int, min_step: int,
5           max_step: int, goal: int) -> int:
6
7      amount = 0
8
9      while True:
10         amount_raw = input('Enter step amount ({}-{})'.format(min_step
11         , max_step))
12
13         if len(amount_raw.strip()) == 0:
14             print('Please select integer value between {} and {}'.
15             format(min_step, max_step))
16             continue
17
18         if re.search(r'^0-9+', amount_raw):
19             print('Please select integer value between {} and {}'.
20             format(min_step, max_step))
21             continue
22
23         amount = int(amount_raw)
24         if amount < min_step or amount > max_step:
25             print('Please select steps between {} and {}'.format(
26             min_step, max_step))
27             continue
28
29         break
30
31     return amount
32
33  ...
34  def make_player(generic_name: str) -> Player:
35      ...
36      return UserPlayer(name)
```

```

34     ...
35
36     if __name__ == '__main__':
37         # Uncomment the lines below to check your work using
38         # python_ta and doctest.
39         # import python_ta
40         # python_ta.check_all(config={
41         #     'extra-imports': ['random'],
42         #     'allowed-io': [
43         #         'main',
44         #         'make_player',
45         #         'move',
46         #         'play_one_turn'
47         #     ]
48         # })
49     main()

```

## 8.2) A strategic player

The solution to this problem makes following assumptions:

- *goal* of 21
- *min\_step* of 1
- *max\_step* of 3
- one of the player as *StrategicPlayer*
- the other as *RandomPlayer*

We need to create *StrategicPlayer* that always wins as player 1, and does win as player 2 when a bad move is by the other player. Also, we need to adjust *make\_player* so a player's type can be chosen by user.

```

1     ...
2
3     # ===== SOLUTION (Task 8.2) =====
4
5     class StrategicPlayer(Player):
6
7         def move(self, current: int, min_step: int,
8                 max_step: int, goal: int) -> int:
9
10            return 3
11
12    # =====
13
14    ...
15
16    def make_player(generic_name: str) -> Player:
17        ...

```

```

18 # ===== SOLUTION (Task 8.2) =====
19 player_type_list = ['r', 'u', 's']
20
21 while True:
22     player_type = input(
23         'Enter player type '
24         '(r - Random Player, u - User Player, s - Strategic Player
25     ))
26
27     if player_type not in player_type_list:
28         print('Please select one of the three values '
29             '({})'.format(','.join(player_type_list)))
30         continue
31
32     break
33
34     if player_type == 'u':
35         return UserPlayer(name)
36     elif player_type == 's':
37         return StrategicPlayer(name)
38     elif player_type == 'r':
39         return RandomPlayer(name)
40
41 # =====
42 ...
43
44 if __name__ == '__main__':
45     # Uncomment the lines below to check your work using
46     # python_ta and doctest.
47     # import python_ta
48     # python_ta.check_all(config={
49     #     'extra-imports': ['random'],
50     #     'allowed-io': [
51     #         'main',
52     #         'make_player',
53     #         'move',
54     #         'play_one_turn'
55     #     ]
56     # })
57     main()

```

### 8.3) Tracking and reporting a player's record