Þátttaka í kennslustund, viku 5. kennslustund A

Ylfa - ylfas23@ru.is
Donna - donn24@ru.is
Bjarki - bjarkik24@ru.is
Victor - victor23@ru.is

Umræða í krossaspurningum

1. **In programming, the act of refactoring refers to:**
   - Svar C = Restructuring code such that the functionality is the same, but the structure of the code is better and more readable.
2. **Which answers specifically apply to the term reusable function?**
   - Við vorum ekki alveg viss hérna en svöruðum fyrst D.:
   - c) The function is concise enough that it makes sense to call it from different parts in the program
   - d) The function is self-contained and has a clear role
3. **Decomposition, in the context of functions, means:**
   - svar B
   - To break up a program into reusable units
4. Svöruðum D.
5. Svöruðum B.
6. Svörðum B, svo D - sem var ekki rétt. Rétta svarið er D.
   Fengum útskýringu frá kennara.
   Svarið er D. Af því að = Það er hægt að senda föll inn í fallið. - func er nafnið á falli
7. Svöruðum fyrst B en rétta svarið er D - það er enginn return á def reset - og það er bara verið að prenta út x - sem er 4
8. Svörðum A.
9. Svöruðum fyrst 13(D), en svo 23(E) → en rétta svarið er 18(C) → rugluðumst aðeins á að það var bara verið að kalla á increase_3 - með x ekki double_2(number)
10. **Which statement is true about functions in Python?**
    - Svöruðum fyrst B, en það er ekki rétt → rétta svarið er C.

Umræða í forritunarverkefnum

Problem A. | Tile Traveller

Stutt lýsing á verkefni:
Hvað á forritið að gera?

The assignment is to develop a computer game in which a player is located in a grid of tiles, one of which is occupied by the player. At each iteration, the program displays the directions for which there are adjacent tiles that the player can travel to.

Input:
Input consists of one line for each move made by the player, in order, where each move consists of any of the characters n, e, s, w, N, E, S, or W. These lines repeat until the player has won the game.

Output:
First, the program should print "You can travel: {Valid directions}.", without quotations, where valid directions can be **(N)orth**, **(E)as**t, **(S)outh, and (W)est.**
The directions should be separated by spaces and the word "or".
The following **repeats** until the user wins the game, at which point the program **exits** after outputting a victory message:

The program prompts the user with "**Direction**: ", without quotations.

If the player inputs an invalid direction, the program should output **"Not a valid direction!"**, without quotations.

I**f the player inputs a valid direction,** the program should output the new valid directions as stated above, **"You can travel: {Valid directions}."**, without quotations.

If the player wins, the program should output **"Victory!"**, without quotations.

Umræða: (hvaða vandamál komu upp og hvernig var því leyst?)

Hugmynd að lausn #1

1. Skipta forritið niður.
   Ákváðum að skipta þessu niður í nokkur föll svo main fall.
   Print_directions: fall sem prentar út directions - þar inni eru þau líka define-uð
   """Prints the available directions to the player"""
   - (N)orth.
   - (S)outh.
   - (E)ast.
   - (W)est.

   Move(position, direction): fall sem hreyfir notenda eftir input.
   """Moves the player to a new position based on direction"""
   get_Available_directions: fall sem sýnir notenda hvaða directions hann getur valið út frá hvar
   hann er í leiknum
   """Returns the available directions based on the players current position"""
   Main: Aðall fallið
   Leikmaðurinn byrjar á reit (1, 1) og markmiðið er að komast á reit (3, 1)

Vandamál með hugmynd að lausn #1

- Þegar við keyrðum kóðan þá virðist þetta hafa gert allt rétt - en Kattis vildi ekki samþykkja lausninn okkar.
- Mögulega vegna þess að við notuðum [dir] fyrir directions
- Töluðum við dæmatímakennara - kóðinn okkar er rétt en þar sem outputtið var ekki í nákvæmlega sömu röðun og Kattis testinn voru þá var kattis ekki að samþykkja það.
- Prófum að endurraða outputt-ið okkar fyrir tilraun #2 og það virkaði!!!!

Tilraun #1

```
def print_directions(available_directions):
    """Prints the available directions to the player."""
    directions = {
        'N': "north (N)",
        'S': "south (S)",
        'E': "east (E)",
        'W': "west (W)"
    }
    print("You can travel:", ' or '.join([directions[dir] for dir in
available_directions]) + ".")

def move(position, direction):
    """Moves the player to a new position based on the direction."""
    x, y = position
    if direction == 'N':
        return x, y + 1
    elif direction == 'S':
        return x, y - 1
    elif direction == 'E':
        return x + 1, y
    elif direction == 'W':
        return x - 1, y
    return position

def get_available_directions(position):
    """Returns the available directions based on the player's current position."""
    x, y = position
    if (x, y) == (1, 1):
        return 'N'
    elif (x, y) == (1, 2):
        return 'NSE'
    elif (x, y) == (1, 3):
```

```
        return 'ES'
    elif (x, y) == (2, 1):
        return 'N'
    elif (x, y) == (2, 2):
        return 'SW'
    elif (x, y) == (2, 3):
        return 'EW'
    elif (x, y) == (3, 1):
        return ''
    elif (x, y) == (3, 2):
        return 'NS'
    elif (x, y) == (3, 3):
        return 'SW'
    return ''

def main():
    position = (1, 1)  # Start at position (1, 1)

    while position != (3, 1):  # The goal is to reach (3, 1)
        available_directions = get_available_directions(position)
        print_directions(available_directions)

        move_direction = input("Direction: ").upper()

        if move_direction in available_directions:
            position = move(position, move_direction)
        else:
            print("Not a valid direction!")

    print("Victory!")

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

Tilraun #2

```
def print_directions(available_directions):
  """Prints the available directions to the player."""
  directions = {
      'N': "(N)orth",
      'S': "(S)outh",
```

```
        'E': "(E)ast",
        'W': "(W)est"
    }
    print("You can travel:", ' or '.join([directions[dir] for dir
in available_directions]) + ".")


def move(position, direction):
    """Moves the player to a new position based on the
direction."""
    x, y = position
    if direction == 'N':
        return x, y + 1
    elif direction == 'S':
        return x, y - 1
    elif direction == 'E':
        return x + 1, y
    elif direction == 'W':
        return x - 1, y
    return position


def get_available_directions(position):
    """Returns the available directions based on the player's
current position."""
    x, y = position
    if (x, y) == (1, 1):
        return 'N'
    elif (x, y) == (1, 2):
        return 'NES'
    elif (x, y) == (1, 3):
        return 'ES'
```

```
  elif (x, y) == (2, 1):
      return 'N'
  elif (x, y) == (2, 2):
      return 'SW'
  elif (x, y) == (2, 3):
      return 'EW'
  elif (x, y) == (3, 1):
      return ''
  elif (x, y) == (3, 2):
      return 'NS'
  elif (x, y) == (3, 3):
      return 'SW'
  return ''


def main():
  position = (1, 1)  # Start at position (1, 1)
   while position != (3, 1):  # The goal is to reach (3, 1)
      available_directions = get_available_directions(position)
      print_directions(available_directions)
#         print(position)

      move_direction = input("Direction: ").upper()

      if move_direction in available_directions:
          position = move(position, move_direction)
      else:
          print("Not a valid direction!")


  print("Victory!")
```

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