

AS
COMPUTER SCIENCE
(7516/1D)

Paper 1 (Python)

Skeleton Program

Skeleton Program code for the AQA AS UNIT 1 SAM this code should be used in conjunction with the Preliminary Material written by the AQA Programmer Team developed in the Python 3.1 programming environment

```
import random
import pickle
import sys

NO_OF_TRAPS = 2
N_S_DISTANCE = 5
W_E_DISTANCE = 7

class Logical():
    # The Logical class is used to allow the value of a Boolean variable to be
    returned from a subroutine.
    def __init__(self):
        self.Is = False

class CellReference():
    # Creates object with accessible NoOfCellsSouth and NoOfCellsEast properties
    def __init__(self):
        self.NoOfCellsSouth = 0
        self.NoOfCellsEast = 0

class GameData():
    # Creates object with accessible TrapPositions, MonsterPosition, PlayerPosition,
    FlaskPosition and MonsterAwake properties
    def __init__(self):
        self.TrapPositions = []
        self.MonsterPosition = CellReference()
        self.PlayerPosition = CellReference()
        self.FlaskPosition = CellReference()
        self.MonsterAwake = Logical()

def SetUpCavern():
    Cavern = []
    for Count1 in range(N_S_DISTANCE):
        Cavern.append([])
        for Count2 in range(W_E_DISTANCE):
            Cavern[Count1].append('0')
    return Cavern

def SetUpTrapPositions(TrapPositions):
    for Trap in range(NO_OF_TRAPS):
        TrapPositions.append(CellReference())

def DisplayMenu():
    print('MAIN MENU')
    print('')
    print('1. Start new game')
    print('2. Load game')
    print('3. Save game')
    print('4. Play training game')
```

```
print('9. Quit')
print('')
print('Please enter your choice: ')

def GetMainMenuChoice():
    Choice = eval(input())
    print('')
    return Choice

def ResetCavern(Cavern):
    for Count1 in range(N_S_DISTANCE):
        for Count2 in range(W_E_DISTANCE):
            Cavern[Count1][Count2] = ' '

def GetNewRandomPosition():
    Position = CellReference()
    while Position.NoOfCellsSouth ==0 and Position.NoOfCellsEast ==0:
        #a random coordinate of (0,0) needs to be rejected as this is the starting
position of the player
        Position.NoOfCellsSouth = random.randint(0, N_S_DISTANCE-1)
        Position.NoOfCellsEast = random.randint(0, W_E_DISTANCE-1)
    return Position

def SetPositionOfItem(Cavern, ObjectPosition, Item, NewGame):
    Position = CellReference()
    if NewGame and (Item != '*'):
        while Cavern[Position.NoOfCellsSouth][Position.NoOfCellsEast] != ' ':
            Position = GetNewRandomPosition()
        ObjectPosition = Position
    Cavern[ObjectPosition.NoOfCellsSouth][ObjectPosition.NoOfCellsEast] = Item
    return ObjectPosition

def SetUpGame(Cavern, TrapPositions, MonsterPosition, PlayerPosition,
FlaskPosition, MonsterAwake, NewGame):
    ResetCavern(Cavern)
    if NewGame:
        PlayerPosition.NoOfCellsSouth = 0
        PlayerPosition.NoOfCellsEast = 0
        MonsterAwake.Is = False
    for Count in range(NO_OF_TRAPS):
        TrapPositions[Count] = SetPositionOfItem(Cavern, TrapPositions[Count], 'T',
NewGame)
    MonsterPosition = SetPositionOfItem(Cavern, MonsterPosition, 'M', NewGame)
    FlaskPosition = SetPositionOfItem(Cavern, FlaskPosition, 'F', NewGame)
    PlayerPosition = SetPositionOfItem(Cavern, PlayerPosition, '*', NewGame)
    return Cavern, TrapPositions, MonsterPosition, PlayerPosition, FlaskPosition,
MonsterAwake

def SetUpTrainingGame(Cavern, TrapPositions, MonsterPosition, PlayerPosition,
FlaskPosition, MonsterAwake):
    ResetCavern(Cavern)
    PlayerPosition.NoOfCellsSouth = 2
    PlayerPosition.NoOfCellsEast = 4
    MonsterAwake.Is = False
```

```
TrapPositions[0].NoOfCellsSouth = 1
TrapPositions[0].NoOfCellsEast = 6
TrapPositions[1].NoOfCellsSouth = 3
TrapPositions[1].NoOfCellsEast = 4
MonsterPosition.NoOfCellsSouth = 0
MonsterPosition.NoOfCellsEast = 3
FlaskPosition.NoOfCellsSouth = 4
FlaskPosition.NoOfCellsEast = 5
Cavern, TrapPositions, MonsterPosition, PlayerPosition, FlaskPosition,
MonsterAwake = SetUpGame(Cavern, TrapPositions, MonsterPosition, PlayerPosition,
FlaskPosition, MonsterAwake, False)
    return Cavern, TrapPositions, MonsterPosition, PlayerPosition, FlaskPosition,
MonsterAwake

def LoadGame(TrapPositions, MonsterPosition, PlayerPosition, FlaskPosition,
MonsterAwake):
    Filename = input('Enter the name of the file to load: ')
    print('')
    LoadedGameFile = open(Filename, 'rb')
    LoadedGameData = pickle.load(LoadedGameFile)
    LoadedGameFile.close()
    TrapPositions[0].NoOfCellsSouth = LoadedGameData.TrapPositions[0].NoOfCellsSouth
    TrapPositions[0].NoOfCellsEast = LoadedGameData.TrapPositions[0].NoOfCellsEast
    TrapPositions[1].NoOfCellsSouth = LoadedGameData.TrapPositions[1].NoOfCellsSouth
    TrapPositions[1].NoOfCellsEast = LoadedGameData.TrapPositions[1].NoOfCellsEast
    MonsterPosition.NoOfCellsSouth = LoadedGameData.MonsterPosition.NoOfCellsSouth
    MonsterPosition.NoOfCellsEast = LoadedGameData.MonsterPosition.NoOfCellsEast
    PlayerPosition.NoOfCellsSouth = LoadedGameData.PlayerPosition.NoOfCellsSouth
    PlayerPosition.NoOfCellsEast = LoadedGameData.PlayerPosition.NoOfCellsEast
    FlaskPosition.NoOfCellsSouth = LoadedGameData.FlaskPosition.NoOfCellsSouth
    FlaskPosition.NoOfCellsEast = LoadedGameData.FlaskPosition.NoOfCellsEast
    MonsterAwake.Is = LoadedGameData.MonsterAwake.Is

def SaveGame(TrapPositions, MonsterPosition, PlayerPosition, FlaskPosition,
MonsterAwake):
    CurrentGameData = GameData()
    CurrentGameData.TrapPositions = TrapPositions
    CurrentGameData.MonsterPosition = MonsterPosition
    CurrentGameData.PlayerPosition = PlayerPosition
    CurrentGameData.FlaskPosition = FlaskPosition
    CurrentGameData.MonsterAwake = MonsterAwake
    Filename = input('Enter new file name: ')
    print('')
    CurrentFile = open(Filename, 'wb')
    pickle.dump(CurrentGameData, CurrentFile)
    CurrentFile.close()

def DisplayCavern(Cavern, MonsterAwake):
    for Count1 in range(N_S_DISTANCE):
        print(' ----- ')
        for Count2 in range(W_E_DISTANCE):
            if Cavern[Count1][Count2] in [' ', '*'] or ((Cavern[Count1][Count2] == 'M')
and MonsterAwake.Is):
```

```
        print('|' + Cavern[Count1][Count2], end='')
    else:
        print('| ', end='')
    print('|')
print(' ----- ')
print('')

def DisplayMoveOptions():
    print('')
    print('Enter N to move NORTH')
    print('Enter E to move EAST')
    print('Enter S to move SOUTH')
    print('Enter W to move WEST')
    print('Enter M to return to the Main Menu')
    print('')

def GetMove():
    Move = input()
    print('')
    return Move

def MakeMove(Cavern, Direction, PlayerPosition):
    Cavern[PlayerPosition.NoOfCellsSouth][PlayerPosition.NoOfCellsEast] = ' '
    if Direction == 'N':
        PlayerPosition.NoOfCellsSouth -= 1
    elif Direction == 'S':
        PlayerPosition.NoOfCellsSouth += 1
    elif Direction == 'W':
        PlayerPosition.NoOfCellsEast -= 1
    elif Direction == 'E':
        PlayerPosition.NoOfCellsEast += 1
    Cavern[PlayerPosition.NoOfCellsSouth][PlayerPosition.NoOfCellsEast] = '*'

def CheckValidMove(PlayerPosition, Direction):
    ValidMove = True
    if not (Direction in ['N', 'S', 'W', 'E', 'M']):
        ValidMove = False
    return ValidMove

def CheckIfSameCell(FirstCellPosition, SecondCellPosition):
    InSameCell = False
    if (FirstCellPosition.NoOfCellsSouth == SecondCellPosition.NoOfCellsSouth) and
    (FirstCellPosition.NoOfCellsEast == SecondCellPosition.NoOfCellsEast):
        InSameCell = True
    return InSameCell

def DisplayWonGameMessage():
    print('Well Done! You have found the flask containing the Styxian potion.')
    print('You have won the game of MONSTER!')
    print('')

def DisplayTrapMessage():
    print('Oh no! You have set off a trap. Watch out, the monster is now awake!')
    print('')
```

```
def MoveFlask(Cavern, NewCellForFlask, FlaskPosition):
    Cavern[NewCellForFlask.NoOfCellsSouth][NewCellForFlask.NoOfCellsEast] = 'F'
    Cavern[FlaskPosition.NoOfCellsSouth][FlaskPosition.NoOfCellsEast] = ' '
    FlaskPosition = NewCellForFlask
    return Cavern, FlaskPosition

def MakeMonsterMove(Cavern, MonsterPosition, FlaskPosition, PlayerPosition):
    OriginalMonsterPosition = CellReference()
    MonsterMovedToSameCellAsFlask = False
    OriginalMonsterPosition.NoOfCellsSouth = MonsterPosition.NoOfCellsSouth
    OriginalMonsterPosition.NoOfCellsEast = MonsterPosition.NoOfCellsEast
    Cavern[MonsterPosition.NoOfCellsSouth][MonsterPosition.NoOfCellsEast] = ' '
    if MonsterPosition.NoOfCellsSouth < PlayerPosition.NoOfCellsSouth:
        MonsterPosition.NoOfCellsSouth += 1
    elif MonsterPosition.NoOfCellsSouth > PlayerPosition.NoOfCellsSouth:
        MonsterPosition.NoOfCellsSouth -= 1
    elif MonsterPosition.NoOfCellsEast < PlayerPosition.NoOfCellsEast:
        MonsterPosition.NoOfCellsEast += 1
    elif MonsterPosition.NoOfCellsEast > PlayerPosition.NoOfCellsEast:
        MonsterPosition.NoOfCellsEast -= 1
    MonsterMovedToSameCellAsFlask = CheckIfSameCell(MonsterPosition, FlaskPosition)
    if MonsterMovedToSameCellAsFlask:
        Cavern, FlaskPosition = MoveFlask(Cavern, OriginalMonsterPosition,
FlaskPosition)
    Cavern[MonsterPosition.NoOfCellsSouth][MonsterPosition.NoOfCellsEast] = 'M'

def DisplayLostGameMessage():
    print('ARGHHHHHH! The monster has eaten you. GAME OVER.')
    print('Maybe you will have better luck the next time you play MONSTER!')
    print('')

def PlayGame(Cavern, TrapPositions, MonsterPosition, PlayerPosition,
FlaskPosition, MonsterAwake):
    Eaten = False
    FlaskFound = False
    MoveDirection = ' '
    DisplayCavern(Cavern, MonsterAwake)
    while not (Eaten or FlaskFound or (MoveDirection == 'M')):
        ValidMove = False
        while not ValidMove:
            DisplayMoveOptions()
            MoveDirection = GetMove()
            ValidMove = CheckValidMove(PlayerPosition, MoveDirection)
        if MoveDirection != 'M':
            MakeMove(Cavern, MoveDirection, PlayerPosition)
            DisplayCavern(Cavern, MonsterAwake)
            FlaskFound = CheckIfSameCell(PlayerPosition, FlaskPosition)
            if FlaskFound:
                DisplayWonGameMessage()
            Eaten = CheckIfSameCell(MonsterPosition, PlayerPosition)
            if not MonsterAwake.Is and not FlaskFound and not Eaten:
                MonsterAwake.Is = CheckIfSameCell(PlayerPosition, TrapPositions[0])
```

```
    if not MonsterAwake.Is:
        MonsterAwake.Is = CheckIfSameCell(PlayerPosition, TrapPositions[1])
    if MonsterAwake.Is:
        DisplayTrapMessage()
        DisplayCavern(Cavern, MonsterAwake)
if MonsterAwake.Is and not Eaten and not FlaskFound:
    Count = 0
    while Count < 2 and not Eaten:
        MakeMonsterMove(Cavern, MonsterPosition, FlaskPosition, PlayerPosition)
        Eaten = CheckIfSameCell(MonsterPosition, PlayerPosition)
        print('')
        input("Press Enter key to continue")
        DisplayCavern(Cavern, MonsterAwake)
        Count += 1
if Eaten:
    DisplayLostGameMessage()

if __name__ == "__main__":
    Cavern = SetUpCavern()
    Choice = 0
    FlaskPosition = CellReference()
    MonsterAwake = Logical()
    MonsterPosition = CellReference()
    PlayerPosition = CellReference()
    TrapPositions = [] # list of trap positions
    SetUpTrapPositions(TrapPositions)
    while Choice != 9:
        DisplayMenu()
        Choice = GetMainMenuChoice()
        if Choice == 1:
            Cavern, TrapPositions, MonsterPosition, PlayerPosition, FlaskPosition,
            MonsterAwake = SetUpGame(Cavern, TrapPositions, MonsterPosition, PlayerPosition,
            FlaskPosition, MonsterAwake, True)
            PlayGame(Cavern, TrapPositions, MonsterPosition, PlayerPosition,
            FlaskPosition, MonsterAwake)
        elif Choice == 2:
            LoadGame(TrapPositions, MonsterPosition, PlayerPosition, FlaskPosition,
            MonsterAwake)
            SetUpGame(Cavern, TrapPositions, MonsterPosition, PlayerPosition,
            FlaskPosition, MonsterAwake, False)
            PlayGame(Cavern, TrapPositions, MonsterPosition, PlayerPosition,
            FlaskPosition, MonsterAwake)
        elif Choice == 3:
            SaveGame(TrapPositions, MonsterPosition, PlayerPosition, FlaskPosition,
            MonsterAwake)
        elif Choice == 4:
            Cavern, TrapPositions, MonsterPosition, PlayerPosition, FlaskPosition,
            MonsterAwake = SetUpTrainingGame(Cavern, TrapPositions, MonsterPosition,
            PlayerPosition, FlaskPosition, MonsterAwake)
            PlayGame(Cavern, TrapPositions, MonsterPosition, PlayerPosition,
            FlaskPosition, MonsterAwake)
```