

Forum:[Hack Forums](#)

Thread: [Arrays & Grid Movement.](#)

Skulduggery Pleasant - February 22, 2016, 11:53 pm

If I were to setup a grid, so an array with a 10 by 10; and have `pos[0][9] == "T"` (T = Test).

How would I get a working movement system?

So like, "+ve = Right -ve = Left" and you were to input the integer "5". Since it is positive it would move to the right, 5 places. And then the same with up and down: "+ve = Up -ve = Down". 3 = Move up 3 places. How would I resassign the position of "T", and remove the `[0][9]`?

Goodies - February 23, 2016, 1:14 am

Here's a quick and dirty way of doing it. Use "l" for left "r" for right "u" for up and "d" for down. The Y axis must be inverted because we are printing from top down.

PHP Code

```
<?php
class Grid():
    class __Position:
        def __init__(self, x, y):
            self.x = x
            self.y = y

        def __call__(self):
            return self.x, self.y

    def __init__(self, size):
        if len(size) != 2:
            raise ValueError("Size should be a tuple/list of (x, y).")
        x, y = list(size)
        self.__x, self.__y = x, y
        if not all([type(_) is int for _ in [x, y]]):
            raise TypeError("Parameters must be integers.")
        self.__grid = [["X" for __ in range(x)] for _ in range(y)]
        self.__pos = self.__Position(x//2, y//2)
        self.__directions = {
            "l": (-1, 0),
            "r": (1, 0),
            "u": (0, 1),
            "d": (0, -1)
        }

    @property
    def pos(self):
        return self.__pos

    def __print(self):
        x, y = self.__pos()
        self.__grid[y][x] = " "
```

```

for line in self.__grid:
    print ' '.join(line)

def move(self, direction):
    if direction.lower() in self.__directions:
        xdiff, ydiff = self.__directions[direction]
        x, y = self.__pos()
        if x + xdiff < 0 or x + xdiff >= self.__x\
            or y - ydiff < 0 or y - ydiff >= self.__y:
            print("Invalid Move")
        else:
            self.__grid[y][x] = "X"
            self.__pos.x = x + xdiff
            self.__pos.y = y - ydiff
            print(self.__pos.x, self.__pos.y)

def interact(self):
    ui = ""
    while ui not in ['quit', 'exit']:
        self.move(ui)
        self.__print()
        ui = raw_input(": ")

grid = Grid((10, 10))
grid.interact()

```

Skulduggery Pleasant - February 23, 2016, 7:32 am

(Feb 23, 2016 02:14 AM)Goodies Wrote: [\[CITE\]](#) Here's a quick and dirty way of doing it. Use "l" for left "r" for right "u" for up and "d" for down. The Y axis must be inverted because we are printing from top down.

PHP Code

```

<?php
class Grid():
    class __Position:
        def __init__(self, x, y):
            self.x = x
            self.y = y

        def __call__(self):
            return self.x, self.y

    def __init__(self, size):
        if len(size) != 2:
            raise ValueError("Size should be a tuple/list of (x, y).")
        x, y = list(size)
        self.__x, self.__y = x, y
        if not all([type(_) is int for _ in [x, y]]):
            raise TypeError("Parameters must be integers.")
        self.__grid = [["X" for __ in range(x)] for _ in range(y)]
        self.__pos = self.__Position(x//2, y//2)
        self.__directions = {

```

```

    "l": (-1, 0),
    "r": (1, 0),
    "u": (0, 1),
    "d": (0, -1)
}

```

```
@property
```

```
def pos(self):
    return self.__pos
```

```
def __print(self):
    x, y = self.__pos()
    self.__grid[y][x] = " "
    for line in self.__grid:
        print ' '.join(line)
```

```
def move(self, direction):
    if direction.lower() in self.__directions:
        xdiff, ydiff = self.__directions[direction]
        x, y = self.__pos()
        if x + xdiff < 0 or x + xdiff >= self.__x\
            or y - ydiff < 0 or y - ydiff >= self.__y:
            print("Invalid Move")
        else:
            self.__grid[y][x] = "X"
            self.__pos.x = x + xdiff
            self.__pos.y = y - ydiff
            print(self.__pos.x, self.__pos.y)
```

```
def interact(self):
    ui = ""
    while ui not in ['quit', 'exit']:
        self.move(ui)
        self.__print()
        ui = raw_input(": ")
```

```
grid = Grid((10, 10))
grid.interact()
```

Thanks for this, however, it's a little too complicated for me!! Alongside, I currently have something done however, it's not working. If you like I can upload what I currently have, and describe what my objective is, if you would be kind enough to help?

Goodies - February 23, 2016, 11:47 am

Yeah just post it. I'll try and do what I can.

Feanor - February 23, 2016, 12:01 pm

Yeah goodies gave you a good example here, here's some more code;

PHP Code

```
<?php
def make_matrix(num_rows, num_cols, entry_fn):
```

```

return [[entry_fn(i,j)
        for j in range(num_cols)]
        for i in range(num_rows)]

def is_diagonal(i, j):
    return 1 if i == j else 0

def show_matrix(matrix):
    for row in matrix:
        print row

matrix = make_matrix(5, 5, is_diagonal)

show_matrix(matrix)
??

```

Skulduggery Pleasant - February 23, 2016, 1:50 pm
pastebin.com/QMF0UfQz

This is what I have.

Skulduggery Pleasant - February 25, 2016, 10:43 am
 Help?? Please. I still need help!

Goodies - February 25, 2016, 3:30 pm
 We don't understand what it is that you want... what is the end goal here?

EwilDave - February 27, 2016, 12:44 pm
 I've just started reading your code and immediately I see a few errors.

First of all, look at the moveplayer() function:

```

PHP Code
<?php
def moveplayer():
    validmove = False
#Allows the "While" loop to have an un-met condition, meaning it will run into said condition is changed.
    while not validmove:
        directionlr = int(input("-ve Left +ve Right: "))
        movement = prow - directionlr
        directionud = int(input("-ve Down +ve Up: "))
        movement2 = pcol - directionud
        validmove = True
#The condition has been changed, and the While loop will no longer run, due to the condition being met..0

```

validmove has no purpose in this code. Your while loop runs once, every single time. There is no condition that checks whether the movement is valid (i.e. whether the movement has passed the grid and is < 0 or > 9 -- assuming your grid goes from 0 to 9)

In the current state you might as well write PHP Code
 <?php

```

directionlr = int(input("-ve Left +ve Right: "))
movement = prow - directionlr
directionud = int(input("-ve Down +ve Up: "))
movement2 = pcol - directionud
-----

```

I'm also going to point out the `setup_grid()`, but this time mostly in order to make your code more Pythonic - although there is a slight mistake in it.

This is it in it's current state:

PHP Code

```

<?php
def setup_grid():
    #Create a blank array
    for counter in range(0, grid_size):
        grid.append(["0"] * grid_size)
    grid[0][9] = "P"

```

This will setup a blank grid, and by default, `[0][9]` is P(Player). The other "cells" will be displayed as 0.

The mistake is as such: In your loop you are only appending to grid - this means that if grid had data in it, you'd just continue to add to it - what you'd have to do is empty the list at the beginning by calling `grid = []`

However, it could be re-written using list comprehensions to be in the form:

PHP Code

```

<?php
def setup_grid():
    grid = [["0"] * grid_size for _ in range(grid_size)]
    grid[0][9] = "P"

```

This will solve both making a blank array, as well as filling it with 0s and placing the player in the position (0,9)

To finish off, I just quickly skimmed through your code and wrote a response on issues I picked up on from first glance, if you have any more inquiries - don't hesitate to PM me. Always willing to help!

- Ewil

Skulduggery Pleasant - February 28, 2016, 11:50 pm

(Feb 27, 2016 01:44 PM)EwilDave Wrote: [\[CITE\]](#) I've just started reading your code and immediately I see a few errors.

First of all, look at the `moveplayer()` function:

PHP Code

```

<?php
def moveplayer():
    validmove = False
#Allows the "While" loop to have an un-met condition, meaning it will run into said condition is chang

```

ed.

while not validmove:

```
directionlr = int(input("-ve Left +ve Right: "))
movement = prow - directionlr
directionud = int(input("-ve Down +ve Up: "))
movement2 = pcol - directionud
validmove = True
```

#The condition has been changed, and the While loop will no longer run, due to the condition being met..0

validmove has no purpose in this code. Your while loop runs once, every single time. There is no condition that checks whether the movement is valid (i.e. whether the movement has passed the grid and is < 0 or > 9 -- assuming your grid goes from 0 to 9)

In the current state you might as well write PHP Code

```
<?php
directionlr = int(input("-ve Left +ve Right: "))
movement = prow - directionlr
directionud = int(input("-ve Down +ve Up: "))
movement2 = pcol - directionud
-----
```

I'm also going to point out the setup_grid(), but this time mostly in order to make your code more Pythonic - although there is a slight mistake in it.

This is it in it's current state:

PHP Code

```
<?php
def setup_grid():
    #Create a blank array
    for counter in range(0, grid_size):
        grid.append(["0"] * grid_size)
    grid[0][9] = "P"
```

This will setup a blank grid, and by default, [0][9] is P(Player). The other "cells" will be displayed as 0.

The mistake is as such: In your loop you are only appending to grid - this means that if grid had data in it, you'd just continue to add to it - what you'd have to do is empty the list at the beginning by calling grid = []

However, it could be re-written using list comprehensions to be in the form:

PHP Code

```
<?php
def setup_grid():
    grid = [["0"] * grid_size for _ in range(grid_size)]
    grid[0][9] = "P"
```

This will solve both making a blank array, as well as filling it with 0s and placing the player in the position (0,9)

To finish off, I just quickly skimmed through your code and wrote a response on issues I picked up on from first glance, if you have any more inquiries - don't hesitate to PM me. Always willing to help!

- Ewil

This is brilliant, thanks for the help, however, I was just wondering; the player is on [0][9], and if I were to move the player say, 2 right 2 down, [2][7], how would I remove the P from 09 and place a new one on 27

EwilDave - February 29, 2016, 2:11 am

PHP Code

```
<?php
grid[0][9] = "O"
grid[2][7] = "P"
```

In my humble opinion, the grid architecture is relatively bad - I'd much rather to have a dictionary with a set of co-ordinates and have a tuple as the key and the value as the character/enemy on that location.

Skulduggery Pleasant - February 29, 2016, 7:35 am

What I mean is, that, if you are to have a user input, say, move -3 left (+3 = right).

So like, grid displays, asks user, move to what co-ordinates, then inputs say 5 for left/right, and 3 for up/down, means, 5 right, 3 up. How would you remove [x][y] = P. and change it to say [5][3] = P

Stealth™ - February 29, 2016, 7:39 am

If you still need more code, I have some Matrix classes from my computer science course last year.

Let me know!

Skulduggery Pleasant - February 29, 2016, 7:58 am

@Stealth they would be greatly appreciated.

Stealth™ - February 29, 2016, 8:02 am

Matrix.py

Code """

This is a partial implementation of the Matrix ADT using a 2-D array. Students will have to complete this class before other lab exercises.

The code is from the textbook "Data Structures and Algorithms Using Python" by Rance D. Nicaise

CSCI 204
Fall 2013

Students need to modify this file according to the lab description
"""

```
from Array import Array2D
```

```
class Matrix:
```

```
    # Create a matrix of size numRows X numCols initialized to 0
```

```
    def __init__( self, numRows, numCols ):
```

```
        self._theGrid = Array2D( numRows, numCols )
```

```
        self._theGrid.clear( 0 )
```

```

# Return the number of rows in the matrix
def numRows( self ):
    return self._theGrid.numRows()

# Return the number of columns in the matrix
def numCols( self ):
    return self._theGrid.numCols()

# Return the value of element at (i, j): x[i, j]
def __getitem__( self, ndxTuple ):
    assert len( ndxTuple ) == 2, "Invalid number of array subscripts."
    return self._theGrid[ ndxTuple[0], ndxTuple[1] ]

# Set the value of element [i,j] to the value of v
def __setitem__( self, ndxTuple, v ):
    assert len( ndxTuple ) == 2, "Invalid number of array subscripts."
    self._theGrid[ ndxTuple[0], ndxTuple[1] ] = v

# Scale the matrix by the given scalar
def scaleBy( self, scalar ):
    for r in range( self.numRows() ):
        for c in range( self.numCols() ):
            self[ r, c ] *= scalar

# Create and returns a new matrix that is the result from matrix addition
def __add__( self, rhsMatrix ):
    assert rhsMatrix.numRows() == self.numRows() and \
        rhsMatrix.numCols() == self.numCols(), \
        "Matrix size not compatible in the two matrices."

    newMatrix = Matrix( self.numRows(), self.numCols() )
    for r in range( self.numRows() ):
        for c in range( self.numCols() ):
            newMatrix[ r, c ] = self[ r, c ] + rhsMatrix[ r, c ]

    return newMatrix

```

The following operations will be implemented by students #####
 ##### as a set of lab exercises. #####

```

# Create and return a new matrix that is the transpose of the original
def transpose( self ):
    transposedMatrix = Matrix(self.numCols(), self.numRows())
    for col in range(self.numCols()):
        for row in range(self.numRows()):
            transposedMatrix.__setitem__((col, row), self.__getitem__((row, col)))
    return transposedMatrix

# Return a string form
def __str__( self ):

```

```

tempString = "
for row in range(self.numRows()):
    for col in range(self.numCols()):
        tempString += (str(self.__getitem__([row, col])) + ' ')
    tempString += '\n'
return tempString

# Create and return a new matrix that results from matrix subtraction
def __sub__( self, rhsMatrix ):
    assert rhsMatrix.numRows() == self.numRows() and rhsMatrix.numCols() == self.numCols(),
'Matrices must share dimensions to be subtracted!'
    newMatrix = Matrix(self.numRows(), self.numCols())
    for row in range(self.numRows()):
        for col in range(self.numCols()):
            newMatrix[row, col] = self[row, col] - rhsMatrix[row, col]
    return newMatrix

# Create and return a new matrix that results from matrix multiplication
def __mul__( self, rhsMatrix ):
    assert rhsMatrix.numRows() == self.numCols(), 'The rating matrix should have the same
amount of columns as the second matrix has rows!'
    newMatrix = Matrix(self.numRows(), rhsMatrix.numCols())
    for row in range(newMatrix.numRows()):
        for col in range(newMatrix.numCols()):
            newMatrix.__setitem__([row, col], 0)
    for row in range(self.numRows()):
        for col in range(rhsMatrix.numCols()):
            for row2 in range(rhsMatrix.numRows()):
                newMatrix[row, col] += (int(self[row, row2]) * int(rhsMatrix[row2, col]))
    return newMatrix

```

Array

Code """

Implement the Array ADT using array capabilities of the ctype module

The code is from the textbook, "Data Structures and Algorithms Using Python"
by Rance D. Necaise

CSCI 204
Fall 2013

Students need to examine this code but don't need to make modifications (for this lab)

"""

import ctypes

class Array:

```

# Create an array with 'size' elements
def __init__( self, size ):
    assert size > 0, "Array size must be > 0"
    self._size = size
    # Create the array structure using the ctypes module
    PyArrayType = ctypes.py_object * size

```

```

self._elements = PyArrayType()
# Initialize each element
self.clear( None )

# Return the size of the array
def __len__( self ):
    return self._size

# Get the content of the indexed element
def __getitem__( self, index ):
    assert index >=0 and index < len( self ), \
        "Array subscript out of range"
    return self._elements[ index ]

# Set the value of the array element at the given index
def __setitem__( self, index, value ):
    assert index >=0 and index < len( self ), \
        "Array subscript out of range"
    self._elements[ index ] = value

# Clear the array by setting each element to the given value
def clear( self, value ):
    for i in range( len( self ) ):
        self._elements[ i ] = value

# Return the array's iterator for traversing the elements
def __iter__( self ):
    return _ArrayIterator( self._elements )

.....

An iterator for the Array ADT that allows the iteration of the array
object.
.....

class _ArrayIterator:

    def __init__( self, theArray ):
        self._arrayRef = theArray
        self._curNdx = 0

    def __iter__( self ):
        return self

    def __next__( self ):
        if self._curNdx < len( self._arrayRef ):
            entry = self._arrayRef[ self._curNdx ]
            self._curNdx += 1
            return entry
        else:
            raise StopIteration

.....

Implementation of the Array2D ADT using an Array of Arrays
.....

```

```

class Array:
    # Create a 2-D array of size numRows X numCols
    def __init__( self, numRows, numCols ):
        # Create a 1-D array to store an array reference for each row
        self._theRows = Array( numRows )

        # Create the 1-D arrays for each rows of the 2-D array
        for i in range( numRows ):
            self._theRows[ i ] = Array( numCols )

    # Return the number of rows in the 2-D array
    def numRows( self ):
        return len( self._theRows )

    # Return the number of cols in the 2-D array
    def numCols( self ):
        return len( self._theRows[0] )

    # Clear the array by setting every element to the given value
    def clear( self, value ):
        for row in range( self.numRows() ):
            self._theRows[ row ].clear( value )

    # Get the content of the element at position [i, j]
    def __getitem__( self, ndxTuple ):
        assert len( ndxTuple ) == 2, "Invalid number of array subscriptts."
        row = ndxTuple[ 0 ]
        col = ndxTuple[ 1 ]
        assert row >= 0 and row < self.numRows() \
            and col >= 0 and col < self.numCols(), \
            "Array subscripts out of range."

        the1dArray = self._theRows[ row ]
        return the1dArray[ col ]

    # Set the content of the element at position [i, j]
    def __setitem__( self, ndxTuple, value ):
        assert len( ndxTuple ) == 2, "Invalid number of array subscriptts."
        row = ndxTuple[ 0 ]
        col = ndxTuple[ 1 ]
        assert row >= 0 and row < self.numRows() \
            and col >= 0 and col < self.numCols(), \
            "Array subscripts out of range."

        the1dArray = self._theRows[ row ]
        the1dArray[ col ] = value

```

Skulduggery Pleasant - February 29, 2016, 9:16 am

Thanks! For this. Will read it all! :D