

D002 Python for Everyone

Lesson 5: 2D List and Assemble your functions

Dr. Kevin Wang

Department of Computer Science and Engineering

What we have covered so far

- Variables
- Basic Operation `= + - * / ** %`
- Comparison symbol `== != > < >= <=`
- Logical Operator `and or not`
- Branching `if if else if elif elif elif else`
- Loop `while for`
- List
- Function
 - void functions vs functions with return value

Warm up exercise

- Write a function that computes $x^2 + 2x + 1$. Then, write a loop to print the value of the function for $x \in [1, 10]$ (This is read as x in the range of 1 to 10)
- Save it as [Q1.py](#)

Warm up example - Hangman

Recall the function `checker` and `printer` that you have written yesterday.

Work out the logic of Hangman:

1. Pick a secret word

- 2.

- 3.

- 4.

....

100. Game over

Pick a secret word

```
from random import randint  
dictionary = ["apple", "banana", "carrot", "dog"]  
secret = dictionary[randint(0,3)]
```

do once only

User input

```
x = input("Please enter a letter") # do many time
```

Check the input

We should use the function checker. But that needs a list to record which position has been opened.

```
opened = [] # do once only

...

new_opened = checker(secret, x)
opened = opened + new_opened
```

Check if gameover

Let says we only check the winning condition. (Never lose)

Tell me which of the following wins?

case	secret	opened
1	dog	[0,2,1]
2	dog	[0,1,1]
3	apple	[1,2,4,1,2,0]

What is the winning condition???

Gameover checker

```
gameover = True
for i in range(0, len(secret)):
    if not i in opened:
        gameover = False
```


Complete Hangman Program #1

```
def checker(sentence, letter):
    result = []
    index = 0
    while index < len(sentence):
        if sentence[index] == letter:
            result.append(index)
        index = index + 1
    return result

def printer(secret, opened):
    i = 0
    while i < len(secret):
        if i in opened:
            print(secret[i] , end="")
        else:
            print("_", end="")

        i = i + 1
    print()
```

Complete Hangman Program #2

```
from random import randint
dictionary = ["apple", "banana", "carrot", "dog"]
secret = dictionary[randint(0,3)]
opened = []

while True:
    x = input("Please enter a letter")
    opened = opened + checker(secret, x)
    printer(secret, opened)
    gameover = True
    for i in range(0, len(secret)):
        if not i in opened: #some letter is still needed to open
            gameover = False
    if gameover == True:
        break
```

2D List

Assume we have a coordinate $p = (3, 4)$, which can be expressed in python as a list

```
p = [3, 4]
```

To records more points, we can name more variables,

```
q = [2, 4]  
r = [3, 5]  
s = [7, 4]
```

or, store them as a list.

```
points = [p, q, r, s]  
# or  
pts = [[3,4], [2,4], [3,5], [7,4]]
```

Example: Using this 2D list

Find the point that is nearest to the origin

```
from math import sqrt
pts = [[3,4], [2,4], [3,5], [7,4]]
dist = 0
nearest = -1
i = 0
while i < len(pts):
    t = sqrt(pts[i][0] ** 2 + pts[i][1] ** 2)
    if t < dist or nearest == -1:
        dist = t
        nearest = i
    i = i + 1

print("The point nearest to origin is ", pts[nearest])
print("The distance is %.3f" % dist)
```

Example: Tic-Tac-Toe

	0	1	2
0	X		O
1		X	O
2			X

How to picture a 2D
array of a tic-tac-toe
board.

Example: Tic-Tac-Toe

```
cells = [[' ', ' ', ' '], [' ', ' ', ' '], [' ', ' ', ' ']]

col = int(input("Please enter column"))
row = int(input("Please enter row"))

cells[row][col] = 'X'
```

It is very important to use row-major convention. You can change it `cells[col][row]` but not suggested. The reason is rather complicate.

Example - Print the cells

We print the cells row-by-row

```
print("-" * 13)
for i in range(0, 3):
    for j in range(0, 3):
        print("| %s " % cells[i][j], end="")
    print("|")
print("-" * 13)
```

Example - Print the cells

Since this part may be executed many times, make it a function

```
def printcell(cells):  
    print("-" * 13)  
    for i in range(0, 3):  
        for j in range(0, 3):  
            print("| %s " % cells[i][j], end="")  
        print("|")  
    print("-" * 13)
```


Example - Check the column

Recall our row major convention `cells[row][col]`. And, this task would perform many times, make it a function

```
def check_col(cells):  
    if cells[0][0] == cells[1][0] == cells[2][0]:  
        return True  
    if cells[0][1] == cells[1][1] == cells[2][1]:  
        return True  
    if cells[0][2] == cells[1][2] == cells[2][2]:  
        return True  
    return False
```

or simply

```
def check_col(cells):  
    for i in range(0, 2):  
        if cells[0][i] == cells[1][i] == cells[2][i]:  
            return True  
    return False
```

Check the others

Assume you can work out the function to check rows and check diagonals.

```
def check(cells):  
    if check_col(cells) or check_row(cells) or check_diagonal(cells):  
        return True  
    return False
```

Q2: Write a Tic-Tac-Toe if you are ready.

Bonus for L4

Question: Write a function to determine if two rectangles are overlapping.

There are many solutions...

```
def is_overlapping(a1,c1,a2,c2):  
    # if R2 is on R1's left  
    if c2[0] < a1[0]:  
        return False  
    # if R1 is on R2's left  
    if c1[0] < a2[0]:  
        return False  
    # if R2 is above R1  
    if c2[1] > a1[1]:  
        return False  
    # if R1 is above R2  
    if c1[1] > a2[1]:  
        return False  
    return True
```