Global Knowledge®

Python the Language Exercises

Estimated time for completion: 45 minutes

Overview:

In this lab, you will practice writing basic Python code and working with the core language features. You will work with the major language features covered in the lecture in by creating a very basic tic-tac-toe game. We will make many shortcuts or hacks to avoid jumping ahead into topics not necessarily yet covered including function, classes, and more.

These instructions assume you have PyCharm and **Python 3** installed. You can use any OS you choose (OS X, Windows, or Linux).

Goals:

- Create a basic game
- Gain experience with the major language features of Python
- Display output / receive input from the command prompt / terminal

Part 1 – Create a project folder and configure your interpreter

There is no 'starter code' for this lab. Just create a new project or folder to hold your files. We will be using PyCharm for this but you may use any editor you like provided you can translate the instructions to that environment.

- Open PyCharm
- Choose new project (empty)
- Right-click the project and add a new Python file (name it game.py)

- If necessary, set the project interpreter to Python3
- Test your project configuration by adding this single statement to your game.py file, right-click it and choose run:

  ```
  print("testing 1, 2, 3…")
  ```

Part 2 – Build the logical structure of your game

Now let's sketch out the logical structure and just add some comments to be filled in later. Be aware, we will have **significant** code duplication until we cover functions, collections, and classes. This will probably make you appreciate just how useful each of those are.

We will model our 3x3 tic-tac-toe board with a 3x3 2D array. Define it as follows:

```
x = 1   # value for X's entry

o = -1  # value for O's entry

e = 0   #value for empty cell

game = [

    [e, e, e],

    [e, e, e],

    [e, e, e],

]
```

Here is the basic structure of the code. Feel free to copy this to your game.py file and fill it in step by step.

```
# 1. Print out cheesy welcome
```

```
# 2. setup game data structures

x = 1

o = -1

e = 0

game = [

    [e, e, e],

    [e, e, e],

    [e, e, e],

]


# 3. Print the empty board (will require two for loops)

#     Convert value of x => 'x'

#     Convert value of o => 'o'

#     Convert value of e (empty) => '.'



# 4. Setup game loop variables

# turn, winner, and hasWon.


# 5. Game loop (while loop until someone has won):

#   print who's turn it is (X or O)

#   ask user for row (via input('prompt text'), convert to int via int(text))

#   ask user for column (same)

#   verify row and column are in bounds

#   verify that entry is not already chosen
#   set row/column to correct value (e.g. game[row][column] = x)

#   show the board again (now that it's updated)

#   check for a winner (any row or diagonal sums to 3 => x's win, -3 => o's win), note without collections or functions, this is not pretty

#   switch players, repeat.


# 6. Announce winner and show the winning board.
```

Here is the output from a standard game. Use the structure listed above along with this output to implement the tic-tac-toe game.

```
Welcome to tic-tac-toe profession v0.1


The board is set:

. . .

. . .

. . .


It's X's turn.

Choose row (zero based): 0

Choose column (zero based): 0

Excellent choice:

x . .

. . .

. . .
```

```
We're checking for a winner...
No winner yet, keep going!

It's O's turn.
Choose row (zero based): 0
Choose column (zero based): 1
Excellent choice:
x o .
. . .
. . .

We're checking for a winner...
No winner yet, keep going!

It's X's turn.
Choose row (zero based): 2
Choose column (zero based): 0
Excellent choice:
x o .
. . .
x . .

We're checking for a winner...
No winner yet, keep going!
```

```
It's O's turn.
Choose row (zero based): 1
Choose column (zero based): 0
Excellent choice:
x o .
o . .
x . .

We're checking for a winner...
No winner yet, keep going!

It's X's turn.
Choose row (zero based): 1
Choose column (zero based): 1
Excellent choice:
x o .
o x .
x . .

We're checking for a winner...
No winner yet, keep going!

It's O's turn.
Choose row (zero based): 0
Choose column (zero based): 2

Excellent choice:

x o o

o x .

x . .


We're checking for a winner...

No winner yet, keep going!


It's X's turn.

Choose row (zero based): 2

Choose column (zero based): 2

Excellent choice:

x o o

o x .

x . x


We're checking for a winner...

We have a winner! It's the X's!

Way to win the day


x o o

o x .

x . x
```

If you get stuck, you can find the solution in the after code of this lab. Try to get it working without peeking though!