# Python for Math and Stat Fall 2024
# Final Exam

Assume that all necessary packages have been imported.

1. (9 pts) For the following 3 problems, write down what each code block would display if executed in a Jupyter cell. If the code generates an error or infinite loop, write `Error`.

   (a)
   ```
   arr = np.array([[ 3, 14, 10,  6],
                   [ 8,  2, 12,  1],
                   [ 9, 13,  7,  5]])
   arr[:2, ::2]
   ```

   (b)
   ```
   go = 'gobuffs'
   f'{go[-1]}go'
   ```

   (c)
   ```
   (lambda x: 2*x)('go')
   ```

   **Solution:**

   (a)
   ```
   array([[ 3, 10],
          [ 8, 12]])
   ```
   (b) `'sgo'`

   (c) `'gogo'`

2. (13 pts)

   (a) Write a recursive function **`omit_end8(nums)`** that takes a list of positive integers and returns a copy of the list, excluding all numbers that end with `8` as the ones digit.

   Examples:
   `omit_end8([5, 18, 4, 800])` returns `[5, 4, 800]`.
   `omit_end8([18])` returns `[]`.

   (b) Write another version of the same function **`omit_end8(arr)`** that takes a numpy array as input and uses vectorization (without looping) to return the same result as before but as an array.

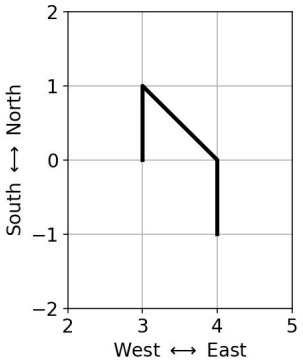
   **Solution:**

   (a)
   ```
   def omit_end8(nums):
       if len(nums) == 0:
           return []

       if nums[0] % 10 == 8:
           return omit_end8(nums[1:])
       else:
           return [nums[0]] + omit_end8(nums[1:])
   ```
   (b)
   ```
   def omit_end8(arr):
       return arr[arr % 10 != 8]
   ```

3. (24 pts) A robot is moving around a coordinate plane. Its position is always a *lattice point* $(x, y)$ with $x$ and $y$ integers. The robot can move in any of 8 directions (N, NE, E, SE, S, SW, W, or NW) to reach a neighboring lattice point. Assume that the positive $y$-axis points North and the positive $x$-axis points East.

   Example: If a robot starts at position $(3, 0)$ and is given the commands `['N', 'SE', 'S']`, it will move to $(3, 1)$, then $(4, 0)$, then $(4, -1)$.




   (a) Write a function **`robot_one_move(pos, cmd)`** that takes a robot's `[x,y]` position and a single `cmd` corresponding to one of the 8 directions. It returns the robot's new position. The function uses the already defined **`dir_dict`**, a dictionary with the 8 directions as keys and their corresponding changes in $x, y$ coordinates as the values.

```
dir_dict = {
    'N': [0, 1], 'NE': [ 1, 1], 'E': [ 1,0], 'SE': [ 1,-1],
    'S': [0,-1], 'SW': [-1,-1], 'W': [-1,0], 'NW': [-1, 1] }
```

   Example: `robot_one_move([3, 0], 'N')` returns `[3, 1]`.

   (b) Write a function **`robot_moves(pos, cmds)`** that takes a robot's `[x,y]` position and a list of `cmds` (directions) as input. It returns a list of all the positions visited by the robot, including the starting position. The function calls `robot_one_move()`.

   Example: `robot_moves([3, 0], ['N', 'SE', 'S'])` returns
   `[[3, 0], [3, 1], [4, 0], [4, -1]]`.

   (c) Write a function **`robot_path(pos, cmds)`** that takes a robot's `[x,y]` position and a list of `cmds` (directions) as input. The function calls `robot_moves()`, then displays the robot's path. (See the sample plot on the previous page. It is not necessary to add axis descriptions or adjust the plot size.)

**Solution:**

(a)
```
def robot_one_move(pos, cmd):
    dir_dict = {
        'N': [0, 1], 'NE': [ 1, 1], 'E': [ 1,0], 'SE': [ 1,-1],
        'S': [0,-1], 'SW': [-1,-1], 'W': [-1,0], 'NW': [-1, 1] }

    x, y = pos
    xchg, ychg = dir_dict[cmd]

    return [x + xchg, y + ychg]
```

(b)
```
def robot_moves(pos, cmds):
    pts = [pos]

    for cmd in cmds:
        pts.append(robot_one_move(pts[-1], cmd))

    return pts
```

(c)
```
def robot_path(pos, cmds):
    pts = robot_moves(pos, cmds)
    xvals, yvals = zip(*pts)

    plt.plot(xvals, yvals)
    plt.show()
```

4. (17 pts) Create a class called **`Robot`**. Each instance of the class corresponds to one robot and has one attribute:

   - **`pos`**: the robot's current position stored as `[x, y]`.

   Example: `vars(Robot([3, 0]))` returns `{'pos': [3, 0]}`.

   The class includes these methods:

   - **`moves(cmds)`**: calls `robot_moves()` to move the robot based on a list of commands. It updates the robot's `pos` to correspond to the final position.
     Example: `Robot([3, 0]).moves(['N', 'SE', 'S'])` will change `pos` to `[4, -1]`.
   - **`teleport()`**: chooses a random lattice point to move the robot to, updating `pos` to the new position. The maximum change for each coordinate is 10 units in either the positive or negative direction. (The random move may leave the robot in the same position.)
     Example: `Robot([100, -20]).teleport()` might change `pos` to `[95, -12]`.

   **Solution:**

```
class Robot:

    def __init__(self, pos):
        self.pos = pos

    def moves(self, cmds):
        pts = robot_moves(self.pos, cmds)
        self.pos = pts[-1]

    def teleport(self):
        x, y = self.pos
        newx = random.randint(x-10, x+10)
        newy = random.randint(y-10, y+10)
        self.pos = [newx, newy]
```

5. (12 pts) The DataFrame **`dfrobot`** contains specifications for more than 50 robots in a research lab. The DataFrame has an index column `Name` and columns for the color, motion type, and height (in inches) for each robot.

| Name | Color | Motion | Hgt |
|---|---|---|---|
| **RoboBee** | Silver | Fly | 1 |
| **R2-D2** | White | Roll | 43 |
| **C-3PO** | Gold | Walk | 69 |
| | ... | | |

Write code to do the following:

(a) Determine the number of robots in `dfrobot` that have a height of 1 inch.

(b) Select the names of all white, rolling robots. The result should be a pandas index or a list of strings.

(c) Among the walking robots, one is the shortest. Identify the name of that robot as a string.

(d) A yellow 40-inch rolling robot named `WALL-E` has just arrived. Add its information to `dfrobot`.

**Solution:**

(a) `len(dfrobot[(dfrobot.Hgt == 1)])` OR `(dfrobot.Hgt == 1).sum()`

(b)
```
dfrobot[(dfrobot.Motion == 'Roll') & (dfrobot.Color == 'White')].index
```

(c)
```
dfrobot[(dfrobot.Motion == 'Walk')].Hgt.idxmin()
```

(d)
```
dfrobot.loc['WALL-E'] = ['Yellow', 'Roll', 40]
```