Python for Math and Stat Fall 2022
Final Exam

1. (12 pts) Let

```
arr = np.array([[ 7, 13,  3,  2],
                [12,  6,  9,  5]])
```

For the following 4 problems, write down what each code block would display if executed in a Jupyter cell.

(a) `arr[1, ::-1]`

(b) `arr[:, 3] ** 2`

(c) `arr[arr // 10 > 0]`

(d) `(lambda x: x+10)(arr[1, 2:])`

Solution:

(a) `array([ 5, 9, 6, 12])`

(b) `array([ 4, 25])`

(c) `array([13, 12])`

(d) `array([19, 15])`

2. (8 pts) Write a function **`digit_in_num(digit, num)`** that returns `True` if the integer `num` contains `digit` and returns `False` otherwise. Assume that `num` is a positive `int` and that `digit` is an `int` between 0 and 9 inclusive. **Use arithmetic operations.** DO NOT use string operations.

Examples:
`digit_in_num(8, 56180)` returns `True`.
`digit_in_num(4, 5618073)` returns `False`.

Solution:

```
def digit_in_num(digit, num):
    currnum = num

    while currnum > 0:
        if currnum % 10 == digit:
            return True
        else:
            currnum //= 10

    return False
```

3. (8 pts) Write a function **`translate(words, lang_dict)`** that takes a string of `words`, separated by spaces, and returns a new string, replacing each word found in `lang_dict` with its equivalent. If a word is not found in the dictionary, it appears unchanged in the new string.

   Example: To convert Spanish words into English, create a Spanish-English dictionary like the one below. Suppose the word `'piton'` (which means python) is not included.

   ```
   span_dict = { 'casa': 'house', 'azul': 'blue', ... }
   ```

   Then `translate('casa azul piton', span_dict)` returns `'house blue piton'`.

   Solution:

```
def translate(words, lang_dict):
    word_list = words.split()
    new_words = ''

    for w in word_list:
        if w in lang_dict:
            new_words += lang_dict[w] + ' '
        else:
            new_words += w + ' '

    return new_words[:-1]

def translate(words, lang_dict):
    return ' '.join([lang_dict[w] if w in lang_dict else w
                     for w in words.split()])
```

4. (8 pts)

```
def func(n):
    return ...
```

   Suppose `func` is an increasing function and you wish to find a value of `n` such that `func(n)` is greater than a threshold value. Write a function **`exceed(thresh)`** that checks the integers `n=1, 2, 4, 8, ...`, one at a time, and stops when `func(n)` is greater than `thresh`, returning the successful value of `n`. Each iteration doubles the previous value of `n`. (Assume that the domain and range of `func` include all positive real numbers.)

   Example:
   Suppose `func(n)` returns `n + 2**n`. Then `exceed(25)` returns `8` because $4 + 2^4 < 25$ and $8 + 2^8 > 25$.

   Solution:

```
def exceed(thresh):
    n = 1
    while func(n) <= thresh:
        n *= 2

    return n
```

5. (8 pts) Consider the nested square root expression

$$\sqrt{a_1 + \sqrt{a_2 + \sqrt{\cdots + \sqrt{a_n}}}}.$$

Write a **recursive** function **`roots(nums)`** that takes a non-empty list of positive numbers a_i and returns the value of the corresponding nested square root expression.

Example: `roots([7,2,4])` returns `3.0` because $\sqrt{7 + \sqrt{2 + \sqrt{4}}} = 3$.

Solution:

```
def roots(nums):
    if len(nums) == 1:
        return math.sqrt(nums[0])
    else:
        return math.sqrt(nums[0] + roots(nums[1:]))
```

6. (12 pts) The *Dow Jones Industrial Average* is a stock market index of 30 prominent companies. Consider the DataFrame **`dfdow`**, shown below, which contains information about these 30 companies, one per row, including each company's ticker symbol, January 1 stock price, and current stock price per share.

Ticker	Company	JanPrice	CurrPrice
AAPL	Apple	177.57	142.27
AMGN	Amgen	224.97	284.73
AXP	American Express	163.65	154.22
	...		
WMT	Walmart	144.72	148.91

Write code to do the following:

(a) Find the current stock price for Walt Disney in `dfdow`. The company's ticker symbol is DIS.

(b) Select the rows for companies that have increased in stock price since January 1 (as a DataFrame).

(c) Add a column called **`'YTD'`** which equals the year-to-date percentage change in stock price for each company. For example, Apple's stock price has dropped from 177.57 to 142.27, which is a drop of 19.9 percent, so Apple's `YTD` entry will be `-19.9`. (It is not necessary to round the values.)

(d) Identify the company (by ticker symbol) with the greatest `YTD` change.

Solution:

(a) `dfdow.CurrPrice['DIS']`

(b) `dfdow[dfdow.JanPrice < dfdow.CurrPrice]`

(c) `dfdow['YTD'] = 100 * (dfdow.CurrPrice - dfdow.JanPrice) / dfdow.JanPrice`

(d) `dfdow.YTD.idxmax()`

7. (19 pts)

(a) Create a class called **`Point`**. Each object in the class represents a point in the xy-plane. It has two attributes:

- **`x`**: the x-coordinate of the point
- **`y`**: the y-coordinate of the point

and the following methods:

- **`dist(pt)`**: returns the distance between the given `Point` and a second `Point`. (The distance between (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.)
- **`__repr__()`**: returns a string in the form `'(x, y)'` which will be displayed as the value of the `Point` object. (It is not necessary to round the coordinates.)

Examples:
`Point(2.3, -9).y` returns `-9`.
`Point(1, 2).dist(Point(-2, 6))` returns `5.0`.
`Point(2.3, -5)` displays `(2.3, -5)`, which is the output of the `__repr__()` method.

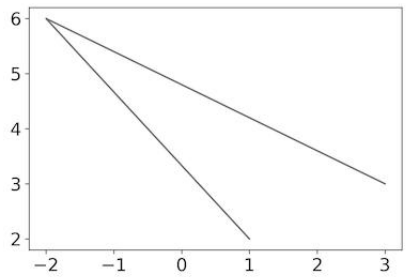
(b) Add this `Point` method:

- **`connect(pt_list)`**: given a list of `Point`s, draws line segments connecting the given point to the other points, in order.

Example:

```
pt1 = Point(-2, 6)
pt2 = Point(3, 3)
Point(1, 2).connect([pt1, pt2])
```

draws line segments from $(1, 2)$ to $(-2, 6)$ to $(3, 3)$.



Solution:

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def dist(self, pt):
        return math.sqrt((self.x - pt.x)**2 + (self.y - pt.y)**2)

    def __repr__(self):
        return f'({self.x}, {self.y})'

    def connect(self, pt_list):
        xcoords = [self.x] + [pt.x for pt in pt_list]
        ycoords = [self.y] + [pt.y for pt in pt_list]
```

```
plt.plot(xcoords, ycoords)
return
```