Python for Math and Stat Fall 2024
Exam 2

Assume that all necessary packages have been imported.

1. (15 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)
   ```
   primes = [2, 3, 5, 7, 11]
   list(zip(primes[:3], primes[2:]))
   ```

   (b)
   ```
   m = 2
   while m < 20:
       print(f'#{m}', end=' ')
       m *= 3
   ```

   (c)
   ```
   def func(n):
       print(n, end=' ')
       if n < 5:
           return n
       else:
           return func(n-3) + n

   func(10)
   ```

   Solution:

   (a) `[(2, 5), (3, 7), (5, 11)]`

   (b) `#2 #6 #18`

   (c)
   ```
   10 7 4
   21
   ```

2. (23 pts) The list below has 12 tuples, arranged in month order, with each tuple containing the name and number of days for a month of the year. (Assume the `'Feb'` tuple has either 28 or 29 days, depending on the year.)

```
months = [('Jan', 31), ..., ('Nov', 30), ('Dec', 31)]
```

(a) Write a function **`date_to_tuple(strdate)`** that takes a date in `mm/dd/yy` string format and returns the corresponding `(month, day, year)` tuple of `int`s. The 2-digit year `yy` should be converted into a 4-digit integer `20yy`.

Example: `date_to_tuple('07/04/24')` returns `(7, 4, 2024)`.

(b) Write a function **`day_of_year(strdate, months)`** that takes a date in `mm/dd/yy` string format and calculates the number of days since the start of the year. It calls `date_to_tuple()`. Assume the list `months` contains the information shown above.

Example: `day_of_year('02/01/24', months)` returns `32` because Feb 1 is the 32nd day of the year.

(c) **Write code** to create a dictionary called `month_dict`, using the information in the list `months`. Each key is a month name and each value is a tuple containing the number of the month (1-12) and number of days in the month:

{`'Jan': (1, 31), ..., 'Nov': (11, 30), 'Dec': (12, 31)`}.

Solution:

(a)
```
def date_to_tuple(strdate):
    mm, dd, yy = strdate.split('/')
    return (int(mm), int(dd), 2000 + int(yy))
```

(b)
```
def day_of_year(strdate, months):
    month, day, year = date_to_tuple(strdate)
    dayct = 0

    for m in range(month-1):
        dayct += months[m][1]

    return dayct + day
```

(c)
```
month_dict = {}
for index, tup in enumerate(months):
    name, numdays = tup
    month_dict[name] = (index+1, numdays)
```

OR

```
month_dict = {}
for i in range(len(months)):
    name, numdays = months[i]
    month_dict[name] = (i+1, numdays)
```

OR

```
month_dict = {tup[0]:
              (index+1, tup[1]) for index, tup in enumerate(months)}
```

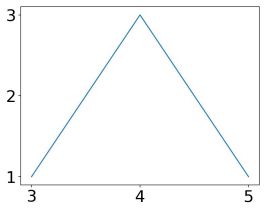
OR

```
month_dict = {months[i][0]:
              (i+1, months[i][1]) for i in range(12)}
```

3. (12 pts)

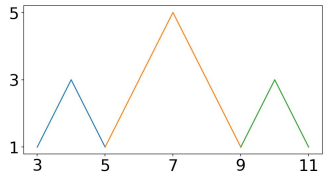
   (a) Write a function **`mtn(pos, size)`** which displays a single "mountain" with lower left corner at `pos`, an (x, y) tuple. The width and height of the mountain are equal to the given `size`.

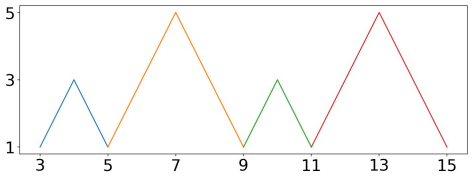
   Example: `mtn((3, 1), 2)` produces the following result.



   (b) Write a function **`mtn_range(pos, size, mtn_ct)`** which displays `mtn_ct` side-by-side "mountains" at the given `pos`, alternating between mountains of the given `size` and larger mountains twice the `size`. The function should call `mtn()`. (Use the default colors and aspect ratio.)

   Examples: If `pos=(3, 1)` and `size=2`, the results for `mtn_ct=3` and `mtn_ct=4` are shown.





Solution:

(a)
```
def mtn(pos, size):
    x, y = pos
    xvals = [x, x + 0.5*size, x + size]
    yvals = [y, y + size, y]
    plt.plot(xvals, yvals)
```

OR

```
def mtn(pos, size):
    xvals = [pos[0], pos[0] + 0.5*size, pos[0] + size]
    yvals = [pos[1], pos[1] + size, pos[1]]
    plt.plot(xvals, yvals)
```

(b)
```
def mtn_range(pos, size, mtn_ct):
    x, y = pos
    for i in range(mtn_ct):
        if i % 2 == 0:
            mtn((x, y), size)
            x += size
        else:
            mtn((x, y), 2*size)
            x += 2*size
```

OR

```
def mtn_range(pos, size, mtn_ct):
    x, y = pos
    for i in range(mtn_ct):
        width = size if i % 2 == 0 else 2*size
        mtn((x, y), width)
        x += width
```