- If you have any scratch work, please circle your final answer.
- Any code you write should run in a Jupyter cell; every character counts!
- For all questions on this exam, assume that all necessary packages have been imported.

1. In the following problems, write what each code block would display if executed in a Jupyter cell. If it would display an error or create an infinite loop, write "error" or "infinite loop". In all of these problems, the object `arr` is given by

```
arr = np.arange(1, 7).reshape(3, 2)
```

(a)
```
arr.flatten().astype(float)
```

(b)
```
arr[2, :] - 1
```

(c)
```
arr[::2, 1:] = 0
arr
```

(d)
```
print(arr ** 2 == arr + 2)
arr[arr ** 2 == arr + 2]
```

Solution:

(a)
```
array([1., 2., 3., 4., 5., 6.])
```

(b)
```
array([4, 5])
```

(c)
```
array([[1, 0],
       [3, 4],
       [5, 0]])
```

(d)
```
[[False  True]
 [False False]
 [False False]]

array([2])
```

2. (a) Write a function `factors(n)` which returns a list of all positive factors of `n`. For example, `factors(6)` would return `[1, 2, 3, 6]`.

   (b) Write a function `most_factors(n)` which returns the number between 1 and n (inclusive) which has the most positive factors. If multiple numbers have the same number of factors, `most_factors(n)` should return the largest one.

Solution:

(a)
```
def factors(n):
    fac = []
    for i in range(1, n+1):
        if n % i == 0:
        fac.append(i)
    return fac
```

(b)
```
def most_factors(n):
curr = 0
    curr_count = 0
    for i in range(1, n + 1):
        i_count = len(factors(i))
        if i_count > curr_count:
            curr = i
            curr_count = i_count
    return curr
```

3. (a) Write a function `encode(word, shuffled)` which encodes the letters in a word by replacing them with the corresponding letter in shuffled. To give a simplified example, suppose that the alphabet consists of the characters `'abc'`, and `shuffled='bac'`. Then `encode('cab', shuffled)` would return `'cba'`, because `shuffled` swaps the letters `'a'` and `'b'`.

   You may fill in code here, if you prefer:

```
def encode(word, shuffled):
    alphabet = 'abcdefghijklmnopqrstuvwxyz'
```

   (b) Suppose you want to *decode* an output from the function `encode()`, to get the original word back. How would you do that? You may either explain in words what you would change about the function `encode()`, or you may also write the code.

Solution:

(a)
```
def encode(word, shuffled):
    alphabet = list('abcdefghijklmnopqrstuvwxyz')
    new_word = ''
    cypher = dict(zip(alphabet, shuffled))
    for l in word:
        print(cypher[l])
    return new_word
```

(b) A function which could decode an encoded word would be defined almost the same way as `encode()`, except with the objects `alphabet` and `shuffled` switched in the dictionary.

4. Consider the weather data below, indexed by days since a last April 19.

	Temp	Clouds
0	46	True
1	56	False
2	62	True
⋮	⋮	⋮

(a) Add today's weather records: the temperature is 65, and there are clouds.

(b) Create a column `'Diff'` which records the difference between the daily temperature and the overall mean temperature in the dataset.

(c) Among all cloudy days, write code which returns a list of the five indices that correspond to the largest difference from the mean temperature.

Solution:

(a) `df.loc[len(df)] = [65, True]`

(b) `df['Diff'] = df['Temp'] - df['Temp'].mean()`

(c) `df[df['Clouds'] == True].Diff.nlargest(5).index.tolist()`

5. Create a class called `Animal`. Each instance has the following attributes to record information about that type of animal:

   - `name`: the name of the animal,
   - `n_eyes`: the number of eyes the animal has,
   - `n_limbs`: the number of limbs (arms or legs) the animal has,
   - `digits_per_limb`: the number of digits (fingers or toes) per limb.

   And the methods:

   - `total_digits()`: returns the total number of digits an animal has
   - `more_eyes(animal)`: compares the number of eyes between the current instance and the object `animal`, and returns the name of the animal that has more eyes (you may assume one does have more)
   - `summary()`: prints a formatted short summary of the animal, and returns nothing. For example, `human.summary()` would print

     ```
     human
     2 eyes
     4 limbs
     ```

   Solution:

```
class Animal:
    def __init__(self, name, n_eyes, n_limbs, digits_per_limb):
        self.name = name
        self.n_eyes = n_eyes
        self.n_limbs = n_limbs
        self.digits_per_limb = digits_per_limb

    def total_digits(self):
        return self.n_limbs * self.digits_per_limb

    def more_eyes(self, animal):
        if self.n_eyes > animal.n_eyes:
            return self.name
        else:
            return animal.name

    def summary(self):
        print(f'{self.name}\n{self.n_eyes} eyes\n{self.digits_per_limb} digits')
```

More space on the next page (you may break your code up between pages).

6. Write a function `plot_row_sum(arr)` which plots the **sums** of the rows in the Numpy array `arr` as a scatterplot. If the entries of the array are integers, the dots should be red; if the entries are floats, they dots should be blue. Use **try except** to catch invalid input, in which case no plot is created; instead, display the message "invalid input".

   Solution:

```
def plot_row_sum(arr):
    try:
        row_sums = arr.sum(axis=1)
        if arr.dtype == int:
            color = 'red'
        elif arr.dtype == float:
            color = 'blue'

        plt.scatter(range(len(row_sums)), row_sums, color=color)
        plt.show()

    except:
        print('invalid input')
```