Exam 2 Solutions

Question 2 - Implementing zip

```
class Zip:
  def __init__(self, l1, l2):
    self._l1 = l1
    self._l2 = l2
    self._length = min(len(l1), len(l2))
    self._curr_index = 0

  def has_next(self):
    return self._curr_index < self._length

  def next(self):
    if self.has_next():
       val = (self._l1[self.curr_index], self._l2[self._curr_index])
       self._curr_index += 1
       return val
    else:
       return None

  def reset(self):
    self._curr_index = 0
```

Question 3 - Miscellaneous Topics

Question 3.1 + 3.2 - Hashing

This hash function **does not** work.

Explanation (and a good reference answer for 3.2):

> This hash function does not work because it is not consistent with the `__eq__` function, meaning it does not return the same hash value even if they are equal objects. Two `IceCream`s could be equal according to `__eq__` (same `brand` and `flavor`), but

> could hash to different places if they had different scoops since `__hash__` also uses the `scoops` field.

Question 3.3

There is no single "right answer" here so we accepted any answer that answered the prompts and demonstrated a clear understanding of one of the ethical concerns we discussed in class and how it applies to the provided situation. As the criteria shows, we graded on

- Picking 1 case study to compare to
- Summarizing an ethical concern from the case study
- Comparing that case study to the provided situation
- Explanation shows depth of understanding of the problem discussed in class and how it relates to this provided situation

Question 4 - Machine Learning

Recall that a **hyper-parameter** is something you specify before training the model (the choices of which impact the quality of the model that is eventually learned). The **parameters** of a model are the specific values learned by the model during the process of training.

Q4.1 - Number of Hidden Layers

This is a **hyper-parameter** since it's something that you decide before training the model.

Q4.2 - Number of Hidden Nodes

This is a **hyper-parameter** since it's something that you decide before training the model.

Q4.3 - Weights

This is a **parameter** since it is learned by the learning algorithm to make the network more accurate.

Q4.4 - Bias

This is a **parameter**, much like the weights. This is learned by the learning-algorithm to be tuned to the specific value that works for the target task.

Q4.5 - Activation Function

This is a **hyper-parameter** since it's something you specify about the network, much like the architecture (number of hidden layers / nodes). We saw a few examples of different activation functions, and which one you choose would likely lead to different models learned.

Question 5 - Geospatial

Q5.1 - Join

Below, we show the result as a table for readability, but the specification stated we wanted your answer written as a CSV. The order of the rows/columns does not matter

Solution

name	continent	geometry	city	country	population	GDP
W	C1	Polygon1	A	W	200	20
W	C1	Polygon1	B	W	100	50
X	C2	Polygon3	C	X	300	60
Z	C1	Polygon2	NaN	NaN	NaN	NaN

Q5.2 - Plot GDP and Population

You don't need to explicitly `fillna` here since `dissolve` (like most other pandas) functions ignores missing-values in the computation (the same effect of it being a 0 for this computation)

```
fig, [[ax1, ax2], [ax3, ax4]] = plt.subplots(2, 2)

merged_country = gdf.merge(df, left_on='name', right_on='country', how='left')


grouped_country = merged_country.dissolve(by='name', aggfunc='sum')
grouped_continent = merged_country.dissolve(by='continent', aggfunc='sum')

grouped_country.plot(column='population', legend=True, ax=ax1)
grouped_country.plot(column='GDP', legend=True, ax=ax2)
grouped_continent.plot(column='population', legend=True, ax=ax3)
grouped_continent.plot(column='GDP', legend=True,ax=ax4)
```

Question 6 - Images

Q6.1 - `a * b`

Note: There was a typo on the exam that said `a + b` in one place, but this doesn't have an impact on the answer since these both don't work for the same reason.

> Error. Following the rules of broadcasting, `b` will be padded to the left with ones to become a `(1, 4)`. The problem then comes from a mismatch in the second dimension where `a` has value `3` and `b` has value `4` since neither of them are `1` meaning neither can be stretched to match the other.

Q6.2 - Mystery 1

Either of the following shapes work

- `(5, 4)`
- `(5, 1)`

Q6.3 - Mystery 3

> Error. To make a 3D result, `d` would need to have 3 dimensions. When adding `a` (a `(4, 3)`) to a 3D array, it will be

> padded on the left to a `(1, 4, 3)` which cannot be broadcasted since the second and third dimensions disagree with the result shape and neither are 1.

Question 7 - Convolution

Two common solutions are shown below

```
def color_convolution(image, kernel):
    kernel_height, kernel_width = kernel.shape
    image_height, image_width, dim = image.shape

    result_height = image_height - kernel_height + 1
    result_width = image_width - kernel_width + 1
    result = np.zeros((result_height, result_width, dim))

    for i in range(result_height):
        for j in range(result_width):
            red = image[i:i+kernel_height, j:j+kernel_width, 0]
            green = image[i:i+kernel_height, j:j+kernel_width, 1]
            blue = image[i:i+kernel_height, j:j+kernel_width, 2]
            result[i, j, 0] = np.sum(red * kernel)
            result[i, j, 1] = np.sum(green * kernel)
            result[i, j, 2] = np.sum(blue * kernel)

    return result

def color_convolution(image, kernel):
    kernel_height, kernel_width = kernel.shape
    image_height, image_width, dim = image.shape

    result_height = image_height - kernel_height + 1
    result_width = image_width - kernel_width + 1
    result = np.zeros((result_height, result_width, dim))

    for i in range(result_height):
        for j in range(result_width):
            for k in range(dim):
                curr = image[i:i+kernel_height, j:j+kernel_width, k]
                result[i, j, k] = np.sum(curr * kernel)

    return result
```