Code Analysis - Week 1, Arch 1

1. Analyze the programming solutions given below and write down in one sentence: what do they do? What problems do they try to solve?

```python
# Code Analysis 1
# Inputs
a = int(input("enter value A: "))
b = int(input("enter value B: "))

# Processing
t = a
a = b
b = t

# Outputs
print("A =", a)
print("B =", b)
```

`Code Analysis 1` asks for input A and B which is imediatly converted to an integer after that the numbers A and B are switched and printed.

```python
# Code Analysis 2

# Inputs
num = int(input("Enter a number: "))

# Processing
dig = num % 10

# Outputs
print(dig)
```

`Code Analysis 2` also asks for an input which is imediatly converted to an integer. After the conversion it will calculate and print the remained of `num` with modulus 10.

1. Implementing a solution for a given problem is challenging for a beginner programmer. It is helpful to have a guideline with some steps. Check this guideline and apply it the Problem 5 of this week. Hint: A template with some examples provided here

```python
# Format and calculate the sum. Input 3141 should become 3+1+4+1=9

# Inputs
input = input()
input_to_str = str(input)
input_to_array = [char for char in input_to_str]

# Processing
```

```python
sum = 0
sum_str = ""
counter = 0
for c in input_to_array:
    if len(sum_str) % 2:
        sum_str += "+" + input_to_array[counter]
    else:
        sum_str += input_to_array[counter]
    sum += int(input_to_array[counter])
    counter += 1

# Outputs
print(sum_str + "=" + str(sum))
```

1. One of the students has tried to apply the guideline for a given problem. But, the code does not produce the expected results. Check the code and without executing the code try to find the mistake.

```python
# Ask the user for the name of item X.
# Then ask the user for the price of item X.
# Finally, ask the user for desired quantity of item X.
# Based on input, calculate how much you have to pay.
# Write the message in the form:
#   "To purchase N units of X you must pay M euros."


# Inputs
name = input("Input the name of item X: ")
price = input("What is the price of", name, "? ")
quantity = input("How many units of", name, "do you want to buy? ")

# Processing
total = price * quantity

# Outputs
print("To purchase", quantity, "units of", name, "you must pay", total,
"euros.")
```

The issue here is that there are multiple arguments given to the input of price and quantity but input() doesn't expect multiple arguments.
This could most likely be solved by doing this instead:

```python
name = input("Input the name of item X: ")
price = input(f"What is the price of {name}? ")
quantity = input(f"How many units of {name} do you want to buy? ")
```

Code Analysis: Input/Output - Week 5, Arch 2

1. Analyze the two given codes below without executing them. What will be the result of the programs?

```
a_tuple = ('Never', 'gonna', 'give', 'you', 'up')
counter = 0
for x in a_tuple:
    if x[0] ==  'g':
        counter = counter + 1
    else:
        counter = counter + 2
print(counter)
```

This will most likely result in `8` as it will increase the counter by `1` if the word starts with 'g', else it will increase the counter by 2.

```
def do_something(x):
    rtuple = x,
    for i in range(2,11):
        rtuple = rtuple + ((x*i),)
    return rtuple
print(do_something(6))
```

The function will use the following formula. Since it uses a range in a for loop it will start with `rtuple = 6 + ((6*2))` and end with `rtuple = 6 + ((6*11))`. So it is adding the current result to the previous result, but as a new item in the tuple. The tuple is complete when the end of the range is reached.

```
def process_strings(strings):
    processed_strings = []
    for string in strings:
        processed_string = ""
        for char in reversed(string):
            processed_string += char
        processed_strings.append(processed_string)
    return processed_strings

def main():
    names = ["Alice", "Bob", "Charlie", "Dave"]
    processed_names = process_strings(names)
    for name in processed_names:
        print(name)

main()
```

In short, we pass a list full of names into a function and then call a function we called to reverses each name. An example being that `Alice` would become `ecilA`.
When we did that to all names in the list we return the now full array with reversed names and print it in main().

Code Analysis: Exception Handling - Week 10, Arch 3

There is a code provided below by a programmer. Looking to the initializer of the class, it seems the code starts by reading data from a file. Unfortunately, we don't have access to the file, but we can analyze the code and understand how the content of the file is structured.

- Read the body of load_data(self). Try to guess how the data columns are structured.

It will read each line from the file and split them as new items in a list.
Then it from each list item it will get the temperature data by mapping it.

- Reading the method `load_data(self)` we may understand the structure of the data columns and data types, but still we don't know the meaning of data values. This is something we can extract from the method `construct_temprature_list(self)`. Read this method carefully and try to explain what will be result of this method?

It will return a result like this [(2022, {1: 32.5, 2: 28.0}), (2023, {3: 25.1, 4: 27.8})]
because of

```
temperature_list.append((year, {}))
```

and

```
if month not in temperature_list[-1][1]:
                    temperature_list[-1][1][month] = 0.0
                temperature_list[-1][1][month] = max(temperature ,
temperature_list[-1][1][month])
```

- Manually write some lines within a text file and try to run the code using your own data values. Does it print what you expect?

Yes, `12 19 2023 0` prints what I expected to be printed, namely `[(2023, {12: 0.0, 15: 0.0})]`

```
class TemperatureDataAnalyzer:
    def __init__(self, file_path):
        self.file_path = file_path
        self.temperature_data = []

    # Method to open the file and load lines as an attribute
    def load_data(self):
        with open(self.file_path, 'r') as file:
            data = [line.strip().split() for line in file]
            self.temperature_data =
[list(map(int,d[:-1]))+[float(d[len(d)-1])] for d in data]

    # Method to perform the analysis and construct the list
    def construct_temperature_list(self):
        temperature_list = []
        for data in self.temperature_data:
            month, day, year, temperature = data[:]
            if year not in [item[0] for item in temperature_list]:
                temperature_list.append((year, {}))
            if month not in temperature_list[-1][1]:
                temperature_list[-1][1][month] = 0.0
            temperature_list[-1][1][month] = max(temperature ,
temperature_list[-1][1][month])
        return temperature_list

def main():
    file_path = './temps.txt'
    analyzer = TemperatureDataAnalyzer(file_path)
    analyzer.load_data()
    temperature_list = analyzer.construct_temperature_list()
    print(temperature_list)

if __name__ == '__main__':
    main()
```