CIRCUITS AND ELECTRONICS : ANALOG AND DIGITAL

Names: _____Gaby and Avery_____

Instructions: Copy and Complete this document and post a link to it from your class site. (One form per group)
Use https://replit.com/

While learning these basic concepts, I ask you to not use generative AI at first. If you must use it, use the device_blob chatbot which will give you hints before giving you the answer. (custom made for this class :))

Value: 5 points

- 3 pts : Exercises are correct or at least attempted for full credit.
- 2 pt : Relatively equal participation from both partners

Learning Goals:

- Review!
- Lists
- For loops
- Dictionaries
- While loops
- Python Errors

---

1. Review

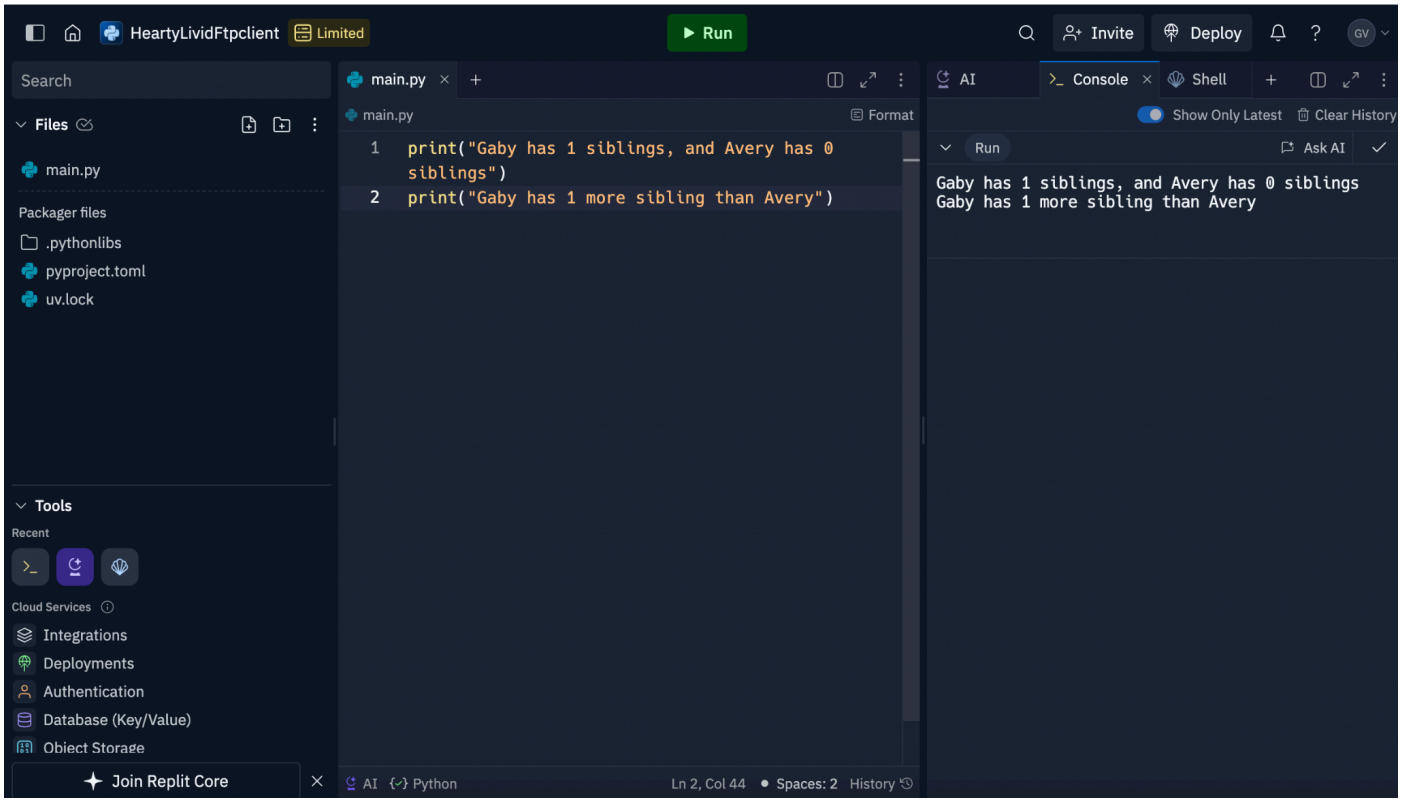
Using repl.it create the following:
(Refer back to the first set of exercises if you don't remember how to do this)

Create four variables

- Your name: Gaby's
- Your partner's name: Avery
- How many siblings you have: 1 sibling
- Your partner's total siblings: 0 siblings

Create a statement that prints each person's name, how many siblings and who has more by how many.
Example: Dusty has 4 siblings, and Zuma has 1 sibling. Dusty has 3 more siblings than Zuma.

Screenshot your code here:



When do you indent a line in python?

You need a to indent a line in python when you're working with structures like loops you need to indent your code blocks

---

2. Lists

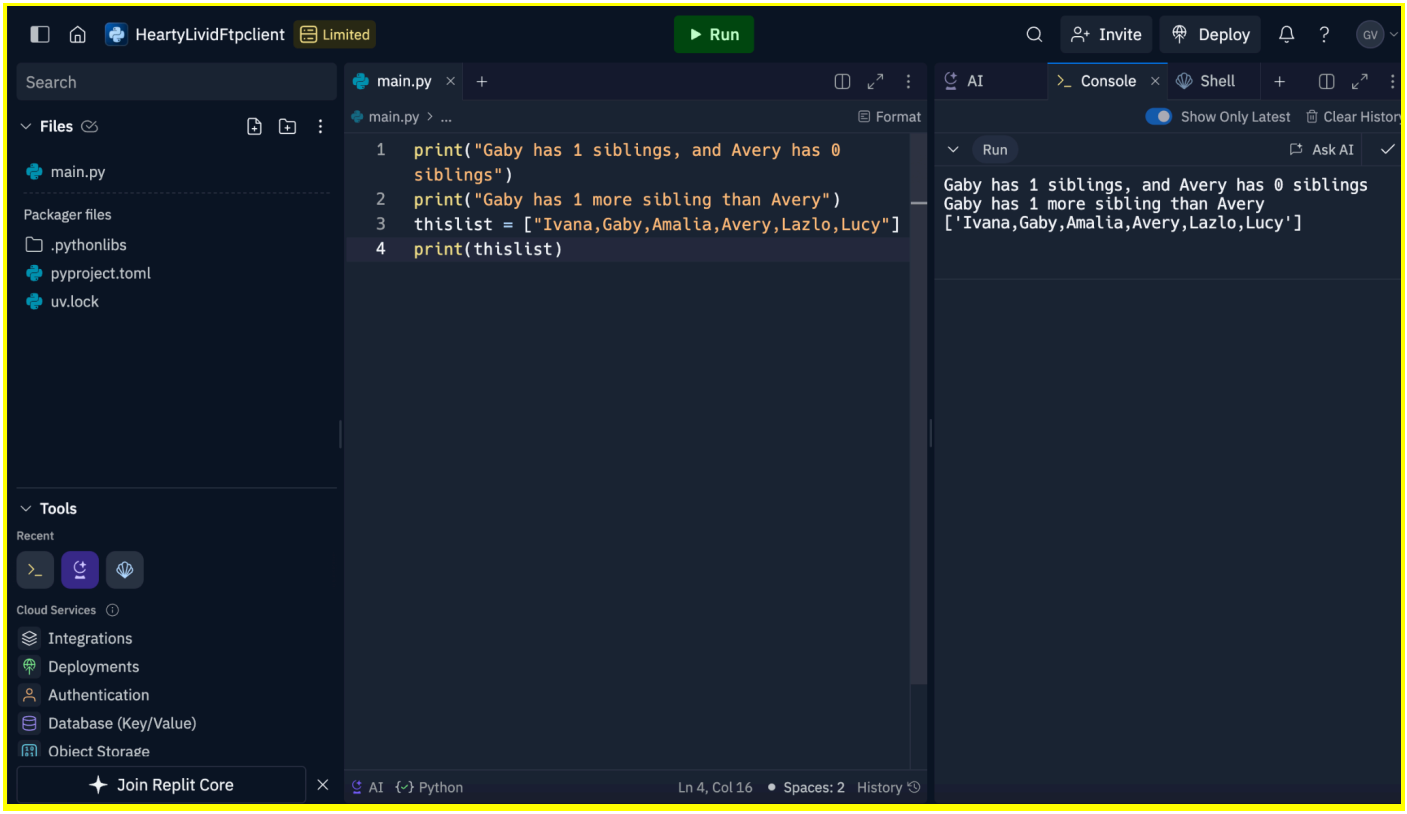
A list is a variable with multiple values - see an example

```
my_schedule = ["D&T", "English", "Biology"]
```

Using repl.it create the following:

- Make a list with six people's names in this class.
- Print the list

Screenshot your code here:



Next: Lists allow you to access specific data items. Example

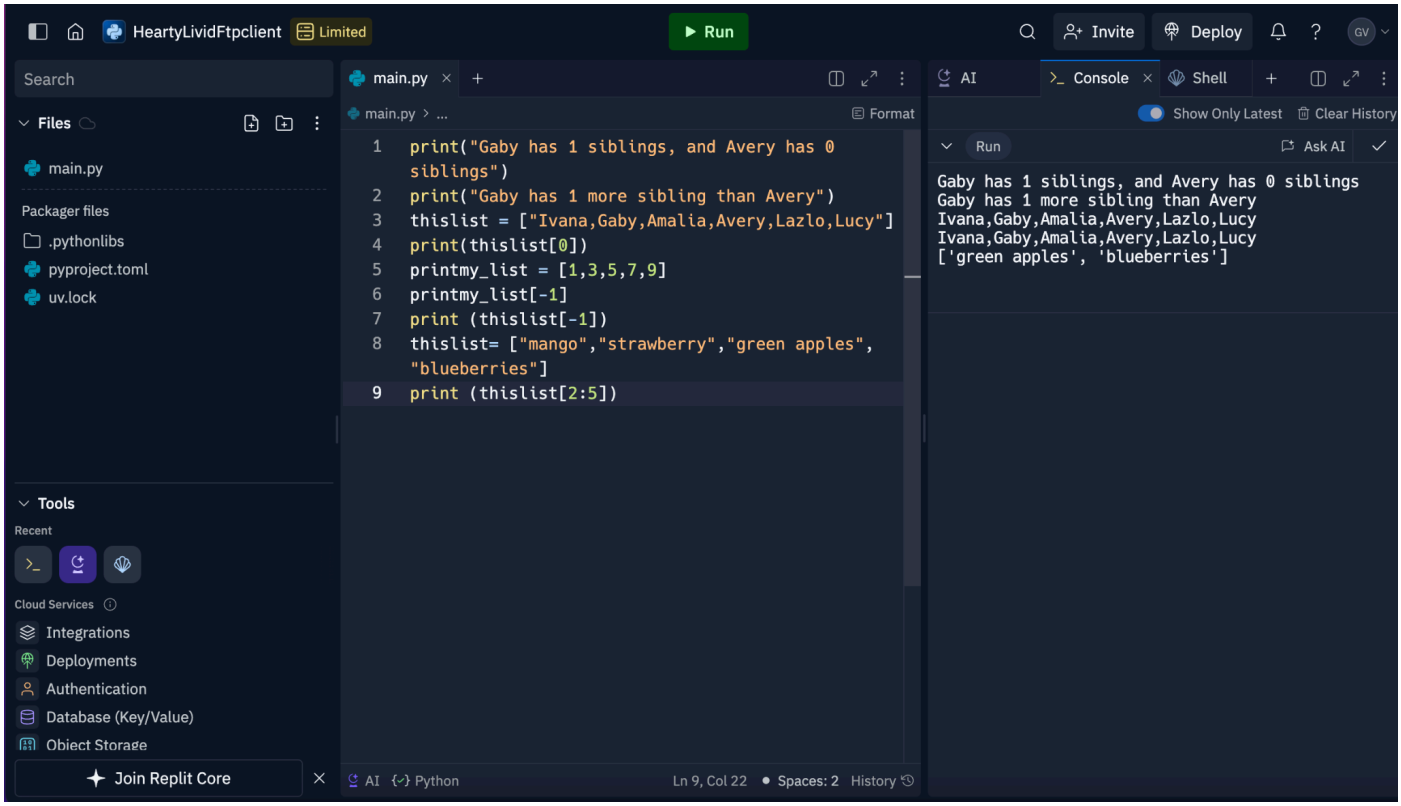
- Print only the odd number of items in your list. (Keep in mind that the first item is 0, not 1 no need to use a loop yet)

- In another line, print only the last item in your list by negative indexing.

Sometimes, you might need to know how long a list is. See how to print the length of a list

- Add a third line to print the length of your list.

Screenshot your code here:



---This is as far as we got (11/2)---

You can change items in a list. See how here
You can also add or insert items to a list. See how here

Using repl.it create the following:

- Keep using the list of names you had in the last sketch.
- Create a NEW list with two new names from the class
- Using list indexing, CHANGE the center of the original list to include the 2nd of the two new names and print the revised list

Screenshot your code here:

There's so much more you can do with lists. Look at the common list methods

- Create a list of favorite shows
- Create a sketch that uses reverse, sort, and one other method.
- Print out the results

Screenshot your code here:

---

3. For Loops

Look at these examples using a for loop to print everything in a list.

Using repl.it create the following:

- Use the list of names you used in the last exercise
- Create a 2nd list with present tense verbs that these people might engage in. (example: “jumps”)
- Use a for loop to print your list of first names
- Use the range function command to print only names indexed at 2 - 5
- Use a nested loop to print each first name along with each of the verbs.

Screenshot your code here:

---

4. Dictionaries

Dictionaries allow for more complex data objects in which data is represented in key:value pairs. Read here

```
city_worker = {
  "name": "Elmer",
  "career": "Plumber",
  "age": 99,
  "Avaliable": True
}
```

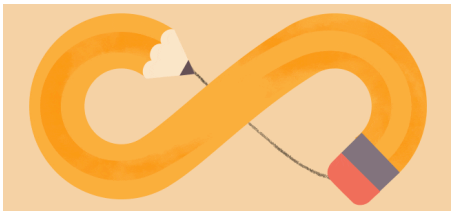
Create a dictionary that has data about a movie you and your partner like. Create at least five key:value pairs for the dictionary.

- Print the entire dictionary.
- Print only two Key Values from your dictionary. example

Screenshot your code here:

---

5. While Loops



"While loops" do something as long as a condition is true. See examples here
We will frequently use the statement "while True" to repeat something forever.

Try this:

```
import time
hello=1

while True:
  print(hello)
  hello=hello+1
  time.sleep(1)
```

hello+=1 is a shorter and equivalent way to write which statement above?

- Create a similar sketch where you count up by fives. When the number is over 50,000 you use the break command to end the loop.

Screenshot your code here:

- How can you do the same thing with even less code?

6. Python Errors

Python errors can be helpful [if you know what they mean](#).

- Paste this code into replit.

```
city_worker = {
  "name": "Elmer",
  "career": "Plumber",
  "age": 99,
  "Available": True
}

print(city_worker["mood"])
```

What error do you get and what does it mean:

- Paste this code into replit. What error do you get and what does it mean:

```
rat=100
cats = 52
print(dogs)
```

What error do you get and what does it mean:

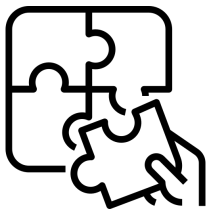
- Paste this code into replit. What error do you get and what does it mean:

```
word = "hello"
for x in range(0,6):
  print(word[x])
```

What error do you get and what does it mean:

For those who have some extra time:

Can you figure out any part of this puzzle?



1. Find an online resource that shows you how to randomize values.
2. Create a list of seven kinds of cereal.
3. Create a list of seven people.
4. Create a loop that grabs a random name and a random cereal to make a sentence that say “This [person] ate [cereal name] today.”
5. Have this loop run every 0.5 seconds
6. Add a couple of statements at the end that keep track of how many times the loop has run, and print it in a statement of some kind.

Optional: Keep track of how many times a cereal is eaten. If a cereal comes up 5 times, can you remove it from the list because the box is empty? Hint: This means your random number will need to be based on the length of the list and not on a static number.

Screenshot your code here:

Alternative Experienced Python Exercises

1. Enhanced List Manipulation

Create a program that does the following:

- Initialize a list of 10 random integers between 1 and 100.
- Use a list comprehension to create a new list containing only the even numbers from the original list.
- Sort the new list in descending order.
- Use a lambda function with the `filter()` method to remove numbers divisible by 3 from the sorted list.
- Print the final result.

2. Dictionary and List Combination

Create a program that:

- Defines a dictionary where keys are names of countries and values are lists of their top 3 cities by population.
- Implements a function that takes a country name as input and returns a formatted string listing its cities.
- Uses a try-except block to handle cases where the input country is not in the dictionary.
- Incorporates a while loop to allow multiple queries until the user decides to exit.

3. Advanced For Loop with Enumerate

Write a program that:

- Creates a list of tuples, where each tuple contains a student's name and their grade (0-100).
- Uses a for loop with `enumerate()` to print each student's name, grade, and their position in the class (assuming the list is sorted by grade in descending order).
- Calculates and prints the class average grade.

4. File I/O and Dictionary Manipulation

Create a program that:

- Reads a text file containing words and their frequencies (one word-frequency pair per line).
- Creates a dictionary from this data.
- Implements functions to:
    - Add new words or update the frequencies of existing words.
    - Remove words below a certain frequency threshold.
    - Find the top N most frequent words.
- Writes the updated dictionary back to a new file in a formatted manner.

5. Object-Oriented Programming Challenge

Design a simple bank account system:

- Create a `BankAccount` class with attributes for account number, holder name, and balance.
- Implement methods for deposit, withdrawal, and balance inquiry.
- Create a `SavingsAccount` subclass that inherits from `BankAccount` and includes an interest rate attribute.
- Override the withdrawal method in `SavingsAccount` to enforce a minimum balance.
- Create a few accounts and demonstrate the functionality of your classes.

6. Error Handling and Debugging

Provide a piece of code with multiple intentional errors (syntax errors, logical errors, and runtime errors).

- Identify each error type.
- Explain the cause of each error.
- Fix the errors and make the code run correctly.

This exercise will enhance debugging skills and deepen understanding of Python's error messages.