CIRCUITS AND ELECTRONICS : ANALOG AND DIGITAL

Names: ___Parker and Lauryn

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

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:
- What is Python
- How to collaborate on Python code
- Printing
- Commenting
- Some common variables
- If else statements
- Comparison operators

---

1. What is Python?

In a couple of sentences, describe why the Python language was created and a few things people like about it.

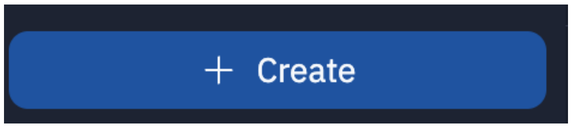
- People like Python because they find it simple to use and it is very intuitive.
- Python language was created due to the desire for a programming language that was both simple to interpret and visually beautiful to look at.

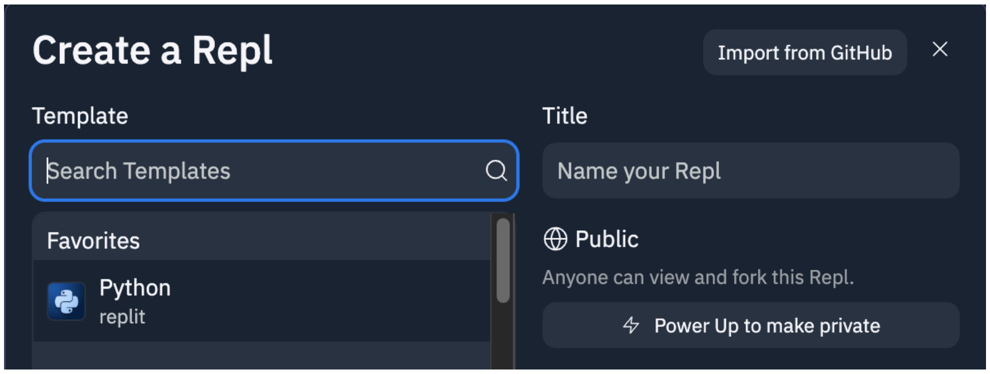
---

2. Easy tool for writing Python collaboratively: https://replit.com/

REPL = Read-Eval-Print-Loop

- Start by creating an account
- Once you're logged in, use the create button and choose python





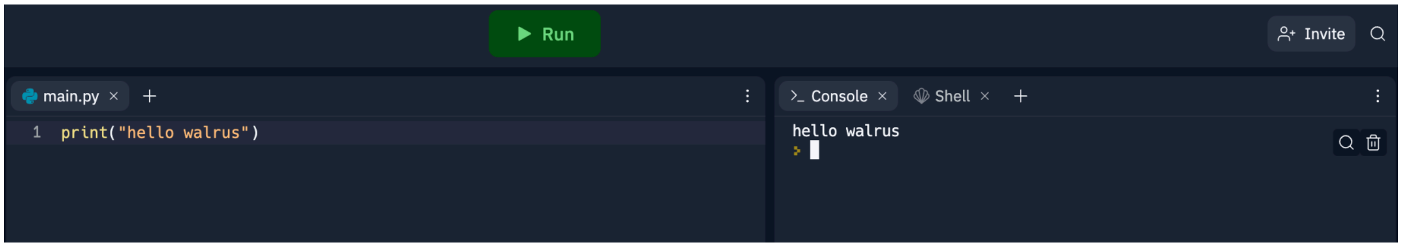
- Then invite your partner by sending them a link or by adding them via username to the Repl
- Try typing anything in the window until both of you can see the other user.

---

3. Printing in Python

Printing in Python is easy! Here's an example.

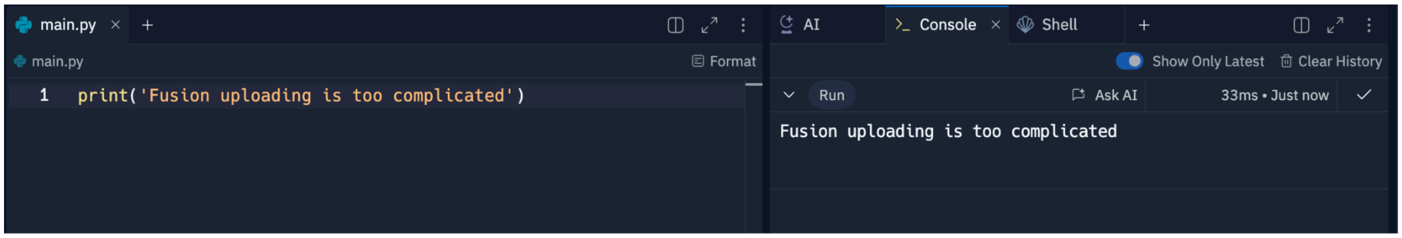
- Make sure your console is open (look to the right of the screen)
- Type what you want to print with the syntax shown
- Press the run button



```python
print("hello walrus")
```

```
hello walrus
>
```

Add a screenshot of what you and your partner printed:



```python
print('Fusion uploading is too complicated')
```

```
Fusion uploading is too complicated
```

---

4. Commenting

When coding, what is the purpose of commenting? Examples here

Commenting can be used to help explain the coding in Python. Comments can also make the code easier to read. Additionally, it can be used to prevent execution when testing the code.

How do you make a single-line comment in python?
Just write the # symbol and simply say that "this is a comment."

How do you make a multiline comment in python?
You can add a # to each line of the comments to make it multiline or you can add the number of lines of the comment with quotation marks around it.

5. Some common variables

Explain, in one phrase, each of these variable types:

int - represents whole numbers

Float - represents decimal values

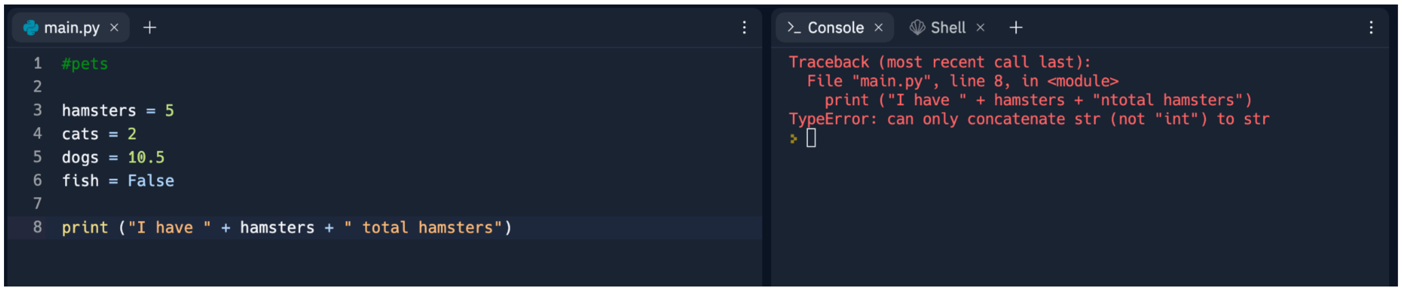
str - Used to convert the data in to a string type data

bool - Used to store two data types as True or False

---

6. Use some variables and data types

Here's an example where we say how many pets we have (too many BTW). We have an error because we're trying to concatenate two different data types (a string and an Int).



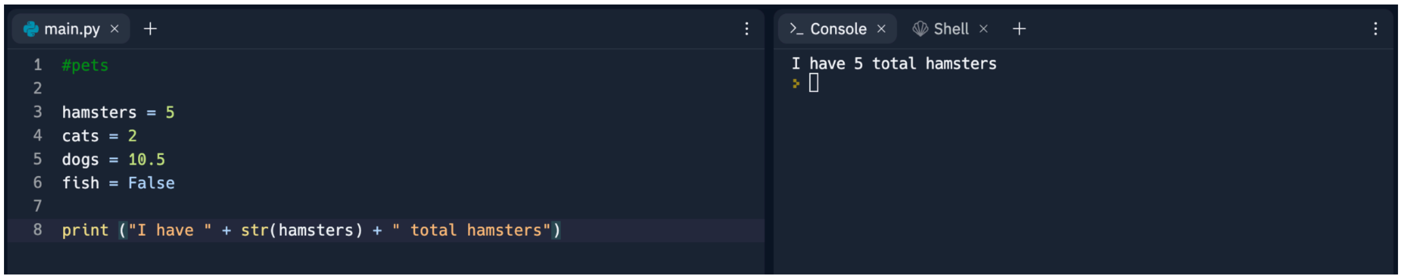
```
#pets

hamsters = 5
cats = 2
dogs = 10.5
fish = False

print ("I have " + hamsters + " total hamsters")
```

```
Traceback (most recent call last):
  File "main.py", line 8, in <module>
    print ("I have " + hamsters + "ntotal hamsters")
TypeError: can only concatenate str (not "int") to str
```

We fix this by letting python know that we want to change the number of hamsters to a string (str).


main.py

```python
#pets

hamsters = 5
cats = 2
dogs = 10.5
fish = False

print ("I have " + str(hamsters) + " total hamsters")
```

Console Shell

```
I have 5 total hamsters
```

This is called casting. Explain casting below:

Now continue this code by printing all the following:

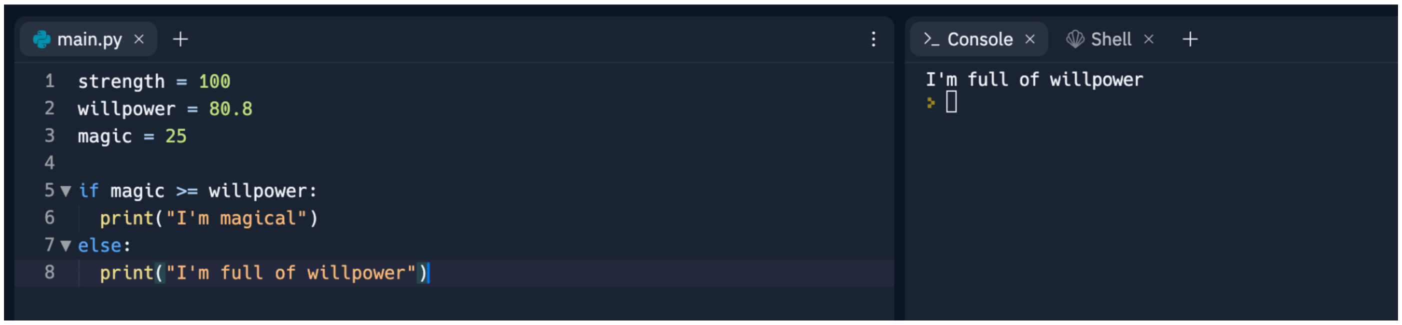
- How many cats do you have?
- How many dogs do you have?
- Do you have any fish? Think of a sentence structure that will accommodate the answer by referencing the variable.
- Add one more pet type and number to your collection
- Create a sentence that adds all your pets together.

Screenshot your code here:

7A. Comparison operators and if statements

In Python, indentations mean something. We indent to show that a command is part of a loop or a conditional statement. See an example here

Look at this example. Notice how the print commands are indented, and only one is executed based on the True or False nature of the if statement.


main.py

```python
strength = 100
willpower = 80.8
magic = 25

if magic >= willpower:
  print("I'm magical")
else:
  print("I'm full of willpower")
```

Console Shell

```
I'm full of willpower
```

Create code that does the following:

- You are a character with: Persuasion, Luck, Influence, and Stealth
- Set those variables to any numeric value you want
- Create two "if else" statements that compare at least two of the characteristics and print something about you.
- Use two different comparison operators. Your choices are (== , >= , <= , >, <)

Screenshot your code here:

7B: Using "and" and "or" in if statements

Modify the example code from the last exercise to use a "and" and "or" in some if statement. Create code that does the following:

- Check to see if strength OR magic is greater than willpower. Print a statement if True.
- Check to see if both willpower AND strength are greater than magic. Print a statement if True.

Find examples here.

Screenshot your code here:

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. Randomize a set of characteristics

3. Executing the code will trigger a “battle” where two characters with randomly generated values are compared in multiple ways; then, an outcome is printed explaining which character won and why.

Screenshot your code here:

Alternative Experienced Python Exercises

1. What are the key features that have contributed to its popularity in various fields?

2. Personal Report

   Objective:
   Create a Python script that generates a fun, personalized report based on user input. Make the report look like it's from a "Secret Agent" mission briefing, with some lighthearted additions!

   Sections to Include:

A. Mission Title: A cool, mysterious title provided by the user, displayed in all caps with a dynamic, centered border.
B. Agent Details:
   a. Agent's Code Name (string), e.g., "Agent Thunderbolt."
   b. Today's Date in a "Mission Initiated" format, e.g., "Mission Initiated: YYYY-MM-DD."
C. Mission Summary:
   a. Brief description of the mission (string), like "Retrieve the Golden Artifact."
   b. Total number of "mission objectives" (integer), validated to ensure it's a positive integer.
   c. Success rate percentage (float) to two decimal places, representing your "Mission Success Probability."

Requirements:

- Use f-strings and/or `format()` to align and style each line creatively.
- Include error handling to ensure valid data entries for all fields. If a user enters an invalid input (like a string for an integer field), prompt them to "Rethink and enter the correct data."
- Display a fun "Mission Complete" message at the end, with a rating based on the success probability.

3. Potion Shop Inventory Tracker

Objective:
Create a Python program that simulates an inventory system for a fantasy potion shop. Your program should use all four variable types (`int`, `float`, `str`, `bool`) in a fun, meaningful way.

Details:

1. Potion Name (`str`): Store the name of a magical potion, like "Elixir of Swiftness."
2. Stock Quantity (`int`): Track how many of these potions are left in stock.
3. Potion Price (`float`): Store the price per potion in gold coins.
4. Availability (`bool`): Indicate whether the potion is currently available for sale (e.g., `True` for available, `False` for sold out).

Functions to Include:

- `add_stock()`: Add potions to the stock.
- `sell_potion()`: Sell a potion, decrease stock, and print a fun success message if available.
- `check_availability()`: Print a message indicating if the potion is available.
- `display_info()`: Display all potion details in a user-friendly way.

4. Turn-based game

Objective:
Create a simple turn-based battle game using Python classes to model characters and actions. Each character will have randomized stats, and the game will proceed in rounds until one character wins.

Steps:

A. Import Required Modules
   a. Use the **`random`** module to generate random attributes.
B. Create a **`Character`** Class
   Define a **`Character`** class that includes:
   a. Attributes:
      i. **`name`** (**`str`**): The character's name.
      ii. **`health`** (**`int`**): Initial health, randomized between 50 and 100.
      iii. **`strength`** (**`int`**): Physical attack power, randomized between 10 and 20.
      iv. **`magic`** (**`int`**): Magical attack power, randomized between 5 and 15.
      v. **`defense`** (**`int`**): Defense value, reducing damage taken, randomized between 5 and 10.
   b. Methods:
      i. **`attack()`**: Randomly choose between a strength or magic attack.

        ii. **`take_damage(amount)`**: Reduce health by **`amount`**, considering the defense attribute.

C. Game Loop Logic
    a. Use a loop to alternate turns between two characters, each choosing to either attack or defend.
    b. For each round:
        i. Display each character’s stats.
        ii. Prompt the player to choose an action (attack or defend).
        iii. Calculate damage, taking into account the attacker’s attribute and defender’s defense.
        iv. Display the outcome, including health updates.
    c. End the game when one character's health reaches 0, declaring the other character as the winner.