**Python Day 5: Methods and Mutability**

**1. String Methods**

Methods are functions that belong to an object. Strings have many useful methods.

* **len(string)**
  + A built-in function that returns the total number of characters in a string (including spaces, commas, etc.).
  + **Example:**

a = '1,2,3'

print(len(a))

# Output: 5

* **.split(delimiter)**
  + Splits a single string into a **list** of smaller strings, using the delimiter you provide as the breaking point.
  + **Example:**

a = '1,2,3'

b = a.split(",")

print(b)

# Output: ['1', '2', '3']

print(type(b))

# Output: <class 'list'>

* **delimiter.join(iterable)**
  + Joins all the string items from an iterable (like a list) into a single **string**. The delimiter is placed *between* each item.
  + **Note:** This method only works if all items in the iterable are strings.
  + **Example 1 (List of strings):**

b = ['1', '2', '3']

c = "@".join(b)

print(c) # Output: 1@2@3

print(type(c)) # Output: <class 'str'>

* + **Example 2 (List of numbers):** You must convert numbers to strings first.

x = [1, 2]

# Convert integers to strings using a list comprehension

z = "@".join([str(item) for item in x])

print(z) # Output: 1@2

* **F-Strings (Formatted Strings)**
  + A way to easily embed variables and expressions inside a string. Start the string with an f and put variables in curly braces {}.
  + **Example:**

m = ['My Self']

n = f"I love {m}"

print(n) # Output: I love ['My Self']

**2. Tuple Methods**

Tuples are **immutable** (cannot be changed), so their methods are for reading data, not modifying it.

* **.count(value)**
  + Counts how many times a specific value appears in the tuple.
  + **Example:**

s = (1, 9, 7, 5, 6, 4, 1)

print(s.count(1)) # Output: 2

* **.index(value)**
  + Returns the position (index) of the *first* time a value is found.
  + **Example:**

s = (1, 9, 7, 5, 6, 4, 1)

print(s.index(6)) # Output: 4

**3. List Methods (Mutable)**

**Adding Items**

* **.append(item)**
  + Adds one single item to the very end of the list. If you append a list, it becomes a nested list.
  + **Example:**

a = [1, 2, 3]

a.append([4, 5])

print(a) # Output: [1, 2, 3, [4, 5]]

* **.extend(iterable)**
  + Adds *each element* from an iterable (like a list or string) to the end of the list individually.
  + **Example:**

a = [1, 2, 3]

a.extend([4, 5])

print(a) # Output: [1, 2, 3, 4, 5]

a.extend("Ram")

print(a) # Output: [1, 2, 3, 4, 5, 'R', 'a', 'm']

* **.insert(index, item)**
  + Inserts an item at a specific index, pushing other items to the right.
  + **Example:**

g = [6, 1, 7, 8]

g.insert(3, 9)

print(g) # Output: [6, 1, 7, 9, 8]

**Removing Items**

* **.pop(index)**
  + Removes and returns the item at the specified index. If no index is given, it removes the last item.
  + **Example:**

a = [1, 2, 3, [4, 5]]

a.pop(3) # Removes the item at index 3, which is [4, 5]

print(a) # Output: [1, 2, 3]

* **.remove(value)**
  + Removes the *first occurrence* of the specified value.
  + **Note:** This method takes exactly one argument. g.remove(7, 8) will cause a TypeError.
  + **Example:**

g = [1, 6, 1, 7, 8]

g.remove(1) # Removes the first '1' it finds

print(g) # Output: [6, 1, 7, 8]

* **.clear()**
  + Removes *all* items from the list, making it empty.
  + **Example:**

o = [1, 2, 3, 4, 5, 6, 7]

o.clear()

print(o) # Output: []

**Organizing Lists**

* **.sort()**
  + Sorts the list in-place (mutates it) in ascending order (smallest to largest).
  + **Example:**

j = [3, 4, 9, 5, 7, 6, 8]

j.sort()

print(j) # Output: [3, 4, 5, 6, 7, 8, 9]

* **.sort(reverse=True)**
  + Sorts the list in-place in descending order (largest to smallest).
  + **Example:**

h = [4, 6, 8, 2, 5, 7, 12, 14, 22]

h.sort(reverse=True)

print(h) # Output: [22, 14, 12, 8, 7, 6, 5, 4, 2]

* **.reverse()**
  + Reverses the order of the elements in the list in-place.
  + **Example:**

q = [4, 6, 8, 2, 5, 7, 12, 14, 22]

q.reverse()

print(q) # Output: [22, 14, 12, 8, 7, 6, 5, 4, 2]

**4. Mini-Projects & Core Concepts**

Your file also included examples of combining these concepts to solve problems.

* **Calculate Area of Rectangle**
  + **Concept:** Basic variable assignment and arithmetic.
  + **Code:**

L = 25

B = 15

A = L \* B

print(A) # Output: 375

* **Calculate Area of Triangle**
  + **Concepts:**
    1. **Defining Functions:** Using def to create a reusable block of code.
    2. **User Input:** Using input() to get data from the user.
    3. **Type Casting:** Using float() to convert the user's string input into a number for math.
    4. **Returning Values:** Using return to send a value back from a function.
  + **Code:**

# Function to calculate area of a triangle

def calculate\_triangle\_area(base, height):

area = 0.5 \* base \* height

return area

# Input from user

base = float(input("Enter the base of the triangle: "))

height = float(input("Enter the height of the triangle: "))

# Calculate and display the area

area = calculate\_triangle\_area(base, height)

print(f"The area of the triangle is: {area}")