**Practice Questions: Type Casting (Type Conversion)**

1. **String to Integer:**
   * **Task: Given the string price\_str = "150", convert it to an integer. Add 50 to the result and print the final value.**
2. **Integer to Float and String:**
   * **Task: Given the integer count = 25. Convert it to a float and print the result. Then, convert the original integer to a string and concatenate it with the text " items".**
3. **Float to Integer (Truncation):**
   * **Task: Given the float pi\_val = 3.14159. Use the int() function to convert it to an integer. Explain in a comment what happens to the decimal part during this conversion.**
4. **List to Tuple and Set:**
   * **Task: Given the list data\_list = [1, 2, 2, 3, 4, 4]. Convert this list first into a tuple, and then into a set. Print both the resulting tuple and set.**
5. **String to List of Characters:**
   * **Task: Given the string word = "PYTHON". Use a type casting function to convert the string into a list where each character is a separate element.**
6. **Tuple to List and Modification:**
   * **Task: Given the tuple config\_tup = ('host', 'user', 'port', 80). Convert it to a list. Change the last element (80) to 8080. Finally, print the modified list.**
7. **Set to List and Sorting:**
   * **Task: Given the set unordered\_set = {50, 10, 30, 20}. Convert it to a list, and then use the built-in sorted() function to sort the list. Print the sorted list.**
8. **Handling Invalid Integer Conversion:**
   * **Task: Attempt to convert the string invalid\_num = "12a" into an integer using int().**
   * **Challenge: Instead of letting the program crash, explain in a comment what specific ValueError would be raised and why this type conversion fails. (You do not need to use try/except, just explain the error).**
9. **Converting Dictionary Keys/Values:**
   * **Task: Given the dictionary data\_dict = {'a': 1, 'b': 2, 'c': 3}. Use type casting functions to convert all the keys into a list and all the values into a tuple. Print both the list of keys and the tuple of values.**
10. **Boolean to Integer/String:**
    * **Task: Given the Boolean values is\_on = True and is\_off = False. Convert is\_on to an integer and is\_off to a string. Print the integer result and the string result.**

**Practice Questions: Input and Print Functions**

1. **Basic Input and Output: Write a program that asks the user, "What is your favorite color?" and then prints the response back using the format: "Ah, [Color] is a lovely choice!"**
2. **Printing Multiple Values (Separator): Given the variables city = "London", country = "UK", and population = 8982000, use a single print() statement to output the data separated by a comma and a space. (Hint: use the sep argument).**
3. **Concatenation and Type Conversion: Ask the user for their age using input(). The result will be a string. Print a message saying, "Next year you will be [Age + 1] years old." (You must perform the necessary type conversion for the addition.)**
4. **Formatting End Characters: Write three separate print() statements that each print one of the words: "Hello", "World", and "Python". Modify the first two calls to use a space as the end character instead of a newline, so the final output appears on a single line.**
5. **Formatted String (f-string): Store your name and a hobby in two separate variables. Use an f-string in a print() statement to clearly display both pieces of information in one sentence.**
6. **Mathematical Input: Ask the user to enter two numbers. Print the result of multiplying the two numbers. Ensure that the program handles the input values as numbers before performing the multiplication.**
7. **Print to a File: Write a program that opens a file named log.txt in write mode ('w') and uses a print() function call to write the string "Application started successfully." into that file. (Hint: use the file argument).**
8. **Empty Input Handling (Conceptual): Explain in one sentence what the data type of the variable user\_input will be if the user just presses Enter without typing anything when prompted by user\_input = input("Enter text: ").**
9. **Multiple Prompts: Write a program that uses three sequential input() calls to get the user's First Name, Last Name, and Email Address. Then, print all three pieces of information, one per line.**
10. **Print with Escape Sequences: Use a single print() statement to display the following text exactly as shown (including the quotation marks and newline):**

**The developer said: "Use the '\n' for a new line."**

**Practice Questions: Built-in Functions**

1. **Finding Maximum and Minimum (max(), min()):**
   * **Task: Given the list of temperatures temps = [25.5, 30.1, 18.7, 35.0, 22.3], use built-in functions to find and print the highest and lowest temperatures recorded.**
2. **Rounding Numbers (round()):**
   * **Task: Given the number π≈3.14159265. Use the built-in round() function to print the value rounded to two decimal places and also the value rounded to the nearest whole integer.**
3. **Getting Length and Type (len(), type()):**
   * **Task: Given the dictionary config = {'timeout': 60, 'retries': 3, 'log\_level': 'INFO'}. Print the total number of key-value pairs in the dictionary. Then, print the data type of the entire config variable.**
4. **Absolute Value (abs()):**
   * **Task: Given the negative number debt = -450.75. Use the built-in function to find and print the positive magnitude of the debt.**
5. **Converting to Boolean (bool()):**
   * **Task: Write a single line of code that uses the built-in bool() function to test the "truthiness" of three different values: the integer 0, the empty string "", and the string "hello". Print the three resulting Boolean values.**
6. **Checking All True (all()):**
   * **Task: Given the list of results checks = [True, True, False, True]. Use the built-in all() function to determine if all checks in the list passed (are True). Print the result.**
7. **Checking Any True (any()):**
   * **Task: Given the list of error codes errors = [0, 0, 404, 0]. Use the built-in any() function combined with a generator expression or list comprehension to check if any non-zero error code is present in the list. Print the result.**
8. **Type Conversion (int(), str()):**
   * **Task: Given the string age\_str = "35" and the integer year\_int = 2025. Convert age\_str to an integer, add 5 to it, and then convert the result back into a string. Print the final string result.**
9. **Sorting Collections (sorted()):**
   * **Task: Given the list of mixed strings and numbers (representing scores and names) data = [5, 10, 'Bob', 2, 'Alice']. Use the sorted() function to sort only the numerical elements of the list and print the resulting sorted list of numbers.**
10. **Generating Range (list() and range()):**
    * **Task: Use the built-in range() function along with the list() constructor to create and print a list of all numbers from 10 down to 1 (inclusive), skipping every other number (i.e., 10,8,6,…).**

**Practice Questions: Function**

1. **Define and Call:** Write a Python function called greet that takes one argument, name, and prints the message "Hello, [name]! Welcome to Python."
2. **Return Value:** Write a function calculate\_area that takes length and width as parameters and returns the calculated area of a rectangle. Test the function with a sample input.
3. **Default Arguments:** Create a function power(base, exponent=2) that returns base raised to the power of exponent. Call the function once with only the base and once with both arguments.
4. **Keyword Arguments:** Write a function describe\_pet(pet\_name, animal\_type) that prints a sentence describing the pet. Call the function using only **keyword arguments** to show that argument order doesn't matter (e.g., describe\_pet(animal\_type='dog', pet\_name='Buddy')).
5. **Arbitrary Arguments (\*args):** Write a function sum\_all(\*numbers) that accepts an arbitrary number of numerical arguments and returns their sum.
6. **Arbitrary Keyword Arguments (\*\*kwargs):** Write a function build\_profile(first, last, \*\*user\_info) that accepts a first name, a last name, and an arbitrary number of keyword arguments, storing them all in a dictionary.
7. **Docstrings:** Define a function that calculates the volume of a cube and include a **docstring** explaining its purpose, parameters, and return value.
8. **Lambda Function:** Write a single **lambda function** that multiplies three numbers. Assign it to a variable and use it to calculate 2×3×4.
9. **Scope:** Predict the output of the following code and explain *why*:

x = 10

def change\_x():

x = 5

print(f"Inside: {x}")

change\_x()

print(f"Outside: {x}")

1. **Nested Logic:** Write a function is\_prime(number) that takes an integer and returns True if the number is a prime number, and False otherwise. (A number is prime if it is greater than 1 and has no positive divisors other than 1 and itself).

**Practice Questions: File Handling**

1. **Reading Entire File:** Write a program that opens a file named data.txt (assume it exists) and reads its entire contents into a single string variable, then prints the string.
2. **Writing/Overwriting:** Write a program that opens a file named output.txt in write mode ('w'), writes the line "This is the first line." to it, and then closes the file.
3. **Appending:** Write a program that opens the same output.txt file in append mode ('a') and adds a new line "This is the second line." after the first line.
4. **Reading Line by Line:** Write a program that opens data.txt and iterates through the file **line by line**, printing each line along with its line number (starting from 1).
5. **Using with Statement:** Rewrite the file reading task from Question 1 using the with statement to ensure the file is properly closed, even if errors occur.
6. **Reading Specific Lines:** Write a program that reads the first three lines of data.txt and stops, regardless of how long the file is.
7. **Data Processing:** Assume numbers.txt contains one integer on each line (e.g., 10, 5, 12). Write a program to read these numbers, convert them to integers, and print the **sum** of all numbers.
8. **Splitting and Storing:** Write a program that reads a line from a file where values are separated by commas (e.g., apple,banana,cherry) and stores these values in a Python **list**.
9. **Binary Mode:** Explain in a single sentence the purpose of opening a file in binary mode ('rb' or 'wb') compared to text mode ('r' or 'w').
10. **Error Simulation:** Describe what happens if you try to open a non-existent file using the 'r' mode. Which built-in exception is raised?

**Practice Questions: Error Handling and Exceptions**

1. **Basic try...except:** Write a program that prompts the user for two integers and prints their division result. Use a try-except block to gracefully handle the situation if the user enters a non-integer value.
2. **Handling ZeroDivisionError:** Modify the division program from Question 1 to specifically catch and handle a ZeroDivisionError if the user enters 0 as the second number.
3. **Using else Block:** Write a program with a try-except-else structure. Attempt to open and read a file in the try block. If successful, print a success message in the else block.
4. **Using finally Block:** Write a program that includes a try-finally block. The finally block should print the message "Execution complete." regardless of whether an exception occurred.
5. **Raising an Exception:** Write a function check\_positive(num) that raises a built-in Python ValueError if the input num is less than or equal to 0.
6. **Multiple Exceptions:** Write a single try-except structure that can handle both a TypeError (e.g., adding a string and an integer) and a NameError (e.g., using an undefined variable), printing a different message for each.
7. **Custom Exception:** Define your own custom exception class named InsufficientFundsError. Then, write a simple function that raises this exception if a transaction amount exceeds the current balance.
8. **Exception with continue:** Write a for loop that iterates over a list of strings and integers. Use a try-except block to safely convert each item to an integer, using **continue** to skip the items that fail the conversion and print the number converted.
9. **Catching All Errors:** Write a try-except block that catches **all** possible exceptions and prints "An unexpected error occurred." (Hint: Catch the base Exception class).
10. **File Error Combination:** Write a program that attempts to open a file settings.ini. Use a try-except block to catch both FileNotFoundError (print "File not found.") and PermissionError (print "Access denied.").