Python Practice Worksheet 2

Focus: Lists, Loops, NoneType, Errors, Stacks & Functions

# Actionable Steps to Learn Each Concept (with Examples)

## 1. Lists

1. Read about lists in the official Python documentation or a beginner’s Python tutorial.

Example search: "Python lists tutorial"

1. Create a new Python file and declare a list with several elements.

Example: my\_list = [10, 20, 30, 40]

1. Practice accessing elements by index, including trying an out-of-range index to see the error.

Example: print(my\_list[2]) print(my\_list[10]) (will cause IndexError)

1. Use list methods like append(), pop(), and remove() to modify your list.

Example: my\_list.append(50) my\_list.pop() my\_list.remove(20)

1. Try creating lists with different data types (numbers, strings, lists).

Example: mixed\_list = [1, "apple", [2, 3]]

1. Question: What happens if you try to access an index that doesn’t exist in a list?

## 2. Loops

1. Study the basic for loop and while loop syntax in Python.

Example: for i in range(5): print(i) count = 0

while count < 5:  print(count)  count += 1

1. Write a for loop that iterates through your list and prints each item.

Example: for item in my\_list: print(item)

1. Modify your loop to add conditions (e.g., print only even numbers from a list).

Example: for num in my\_list:  if num % 2 == 0:    print(num)

1. Write a while loop that continues until a certain condition is met (e.g., user inputs 'quit').

Example: user\_input = ""

while user\_input != "quit":  user\_input = input("Type something (or 'quit' to exit): ")

1. Question: How do you use a loop to process each item in a list?

## 3. NoneType

1. Research what None and NoneType mean in Python.

Example search: "Python NoneType explanation"

1. Write a function that returns None if a condition isn't met (for example, negative inputs).

Example: def double\_or\_none(n):  if n < 0:    return None  return n \* 2

1. Print the result of calling your function with different inputs (positive, zero, negative).

Example: print(double\_or\_none(10)) print(double\_or\_none(-5))

1. Check if a variable is None using is or ==.

Example: result = double\_or\_none(-1)

if result is None:  print("Result is None")

1. Question: What is NoneType, and how is the value None used in Python?

## 4. Errors

1. Look up common Python errors like IndexError, ValueError, and ZeroDivisionError.

Example search: "Common Python errors"

1. Intentionally cause each error in code to see what error message appears.

Example: print(my\_list[10]) int("hello") print(10 / 0)

1. Learn the syntax for try and except blocks.

Example: try:  print(10 / 0)

except ZeroDivisionError:  print("Cannot divide by zero!")

1. Write code that handles at least two different types of errors gracefully.

Example: try:  print(int("abc"))

except ValueError:  print("That's not a number!")

1. Question: What happens when you encounter an error in your code? How can Python handle errors?

## 5. Stacks

1. Review how lists can be used as stacks (LIFO – Last In, First Out).

Example: stack = []

1. Implement push (append()) and pop (pop()) operations on a list.

Example: stack.append(1)stack.append(2)stack.pop()

1. Write a small program that simulates a stack (e.g., reversing a word, managing browser history).

Example: word = "python"

stack = list(word)

reversed\_word = ""

while stack:  reversed\_word += stack.pop()

print(reversed\_word)

1. Print the stack after each operation to understand changes.

Example: print(stack)

1. Question: How can you add or remove elements in a stack using a list?

## 6. Function Basics

1. Read about function definitions and parameters in Python.

Example search: "Python function tutorial"

1. Write a simple function (e.g., def greet(name):) and call it with different arguments.

Example: def greet(name):  print("Hello,", name)

greet("Sam")

greet("Alex")

1. Create a function that returns a value and another that prints directly.

Example: def add(a, b):  return a + b

print(add(2, 3))

1. Combine functions with lists and loops for more advanced practice.

Example: def square(x):  return x\*x

numbers = [1, 2, 3, 4]

for n in numbers:  print(square(n))

1. Question: What are the steps to define and use a function in Python?

# Challenge Section (with Example Inputs)

1. Create a list of numbers and use a loop to print their sum.

Example: nums = [3, 7, 2] total = 0 for n in nums: total += n print(total)

2. Write a loop that prints all even numbers from 1 to 20.

Example: for i in range(1, 21):  if i % 2 == 0: print(i)

3. Write a loop that keeps asking for user input until the user types 'quit'.

Example: while True:  txt = input()  if txt == "quit": break

4. Write a function that takes a number as input and returns None if the number is negative; otherwise, returns double the number. Print results for several test cases, including a negative number.

Example: def double\_or\_none(n):  if n < 0: return None  return n\*2

print(double\_or\_none(5))

print(double\_or\_none(-3))

5. Write code that tries to open a file that does not exist, then catches and prints a message when an error occurs.

Example: try:  open("nofile.txt") except FileNotFoundError:  print("File not found!")

6. Write code that divides two numbers and uses try-except to handle division by zero.

Example: try:  a = 5  b = 0  print(a / b) except ZeroDivisionError:  print("Can't divide by zero!")

7. Create a stack using a list. Push three numbers, pop one, and print the stack.

Example: stack = [] stack.append(1) stack.append(2) stack.append(3) stack.pop() print(stack)

8. Use a stack to reverse a word input by the user.

Example: word = input("Enter word: ") stack = list(word) rev = "" while stack:  rev += stack.pop() print(rev)

9. Define a function greet\_user that takes a name and prints "Hello, [name]!" Call it with three names.

Example: def greet\_user(name):  print("Hello,", name) greet\_user("Sam") greet\_user("Alex") greet\_user("Taylor")

10. Make a simple calculator using functions to add, subtract, multiply, and divide two numbers.

Example: def add(a, b): return a+b def sub(a, b): return a-b def mul(a, b): return a\*b def div(a, b): return a/b print(add(3, 2)) print(div(3, 0))

# Reflection

1. Which activity did you find the most challenging?
2. What new concepts did you discover about lists, loops, NoneType, errors, stacks, or functions?
3. How do these concepts help you write bigger programs?