WORKSHEET

PYTHON – WORKSHEET 1

# Q1 to Q8 have only one correct answer. Choose the correct option to answer your question.

1. Which of the following operators is used to calculate remainder in a division?
   1. # B) &

**C) %** D) $

1. In python 2//3 is equal to?
   1. 0.666 **B) 0**

C) 1 D) 0.67

1. In python, 6<<2 is equal to?
   1. 36 B) 10

**C) 24** D) 45

1. In python, 6&2 will give which of the following as output?
   1. **2** B) True

C) False D) 0

1. In python, 6|2 will give which of the following as output?
   1. 2 B) 4

C) 0 **D) 6**

1. What does the finally keyword denotes in python?
   1. It is used to mark the end of the code
   2. It encloses the lines of code which will be executed if any error occurs while executing the lines of code in the try block.
   3. **The finally block will be executed no matter if the try block raises an error or not.**
   4. None of the above
2. What does raise keyword is used for in python?
   1. **It is used to raise an exception.** B) It is used to define lambda function

C) it's not a keyword in python. D) None of the above

1. Which of the following is a common use case of yield keyword in python?
   1. in defining an iterator B) while defining a lambda function

C) **In defining a generator** D) in for loop.

# Q9 and Q10 have multiple correct answers. Choose all the correct options to answer your question.

1. Which of the following are the valid variable names?
   1. **\_abc** B) 1abc

**C) abc2** D) None of the above

1. Which of the following are the keywords in python?
   1. **yield** B) raise

C) look-in D) all of the above

# Q11 to Q15 are programming questions. Answer them in Jupyter Notebook.

1. **Write a python program to find the factorial of a number.**

Certainly! Here is a simple Python program to find the factorial of a number using both iterative and recursive methods:

### Iterative Method

python

Copy code

def factorial\_iterative(n):

if n < 0:

return "Factorial is not defined for negative numbers."

result = 1

for i in range(1, n + 1):

result \*= i

return result

# Example usage

num = int(input("Enter a number: "))

print("Factorial (iterative) of", num, "is", factorial\_iterative(num))

### Recursive Method

python

Copy code

def factorial\_recursive(n):

if n < 0:

return "Factorial is not defined for negative numbers."

elif n == 0 or n == 1:

return 1

else:

return n \* factorial\_recursive(n - 1)

# Example usage

num = int(input("Enter a number: "))

print("Factorial (recursive) of", num, "is", factorial\_recursive(num))

### Explanation:

* **Iterative Method**: Uses a loop to multiply numbers from 1 to n.
* **Recursive Method**: Uses a recursive function to multiply n by the factorial of n-1, with a base case for n equal to 0 or 1.

You can use either method based on your preference or specific requirements.

1. **Write a python program to find whether a number is prime or composite.**

Certainly! Here’s a Python program to determine whether a number is prime or composite.

### Prime Number Check Program

A prime number is a number greater than 1 that has no positive divisors other than 1 and itself. A composite number is a positive integer greater than 1 that is not prime, meaning it has more than two positive divisors.

Here’s how you can write the program:

python

Copy code

def is\_prime(number):

if number <= 1:

return False

if number <= 3:

return True

if number % 2 == 0 or number % 3 == 0:

return False

i = 5

while i \* i <= number:

if number % i == 0 or number % (i + 2) == 0:

return False

i += 6

return True

def main():

num = int(input("Enter a number: "))

if num < 0:

print("Negative numbers are neither prime nor composite.")

elif is\_prime(num):

print(num, "is a prime number.")

else:

if num > 1:

print(num, "is a composite number.")

else:

print(num, "is neither prime nor composite.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

### Explanation:

1. **is\_prime function**:
   * Returns False for numbers less than or equal to 1.
   * Uses simple checks to quickly identify small prime numbers and even numbers.
   * Uses the 6k ± 1 optimization to test divisibility, which reduces the number of checks needed.
2. **main function**:
   * Takes user input, checks if the number is prime or composite, and handles negative numbers and zero.

You can run this program to check if a given number is prime or composite based on the input.

1. **Write a python program to check whether a given string is palindrome or not.**

Certainly! A palindrome is a string that reads the same forwards and backwards. Here’s a Python program to check whether a given string is a palindrome:

python

Copy code

def is\_palindrome(s):

# Convert string to lowercase and remove spaces

s = s.lower().replace(" ", "")

# Compare the string with its reverse

return s == s[::-1]

def main():

input\_string = input("Enter a string: ")

if is\_palindrome(input\_string):

print(f'"{input\_string}" is a palindrome.')

else:

print(f'"{input\_string}" is not a palindrome.')

if \_\_name\_\_ == "\_\_main\_\_":

main()

### Explanation:

1. **is\_palindrome function**:
   * Converts the input string to lowercase to ensure case-insensitivity.
   * Removes any spaces from the string to focus only on the characters.
   * Compares the string with its reverse (s[::-1]).
2. **main function**:
   * Takes user input and checks if it’s a palindrome using the is\_palindrome function.
   * Prints the result accordingly.

This program will correctly determine if a string is a palindrome by ignoring spaces and case differences.

4o mini

1. **Write a Python program to get the third side of right-angled triangle from two given sides.**

To find the third side of a right-angled triangle when two sides are known, you can use the Pythagorean theorem. The Pythagorean theorem states that in a right-angled triangle, the square of the length of the hypotenuse (the side opposite the right angle) is equal to the sum of the squares of the lengths of the other two sides.

Given two sides, you can determine which is the hypotenuse or which side is missing, and then use the theorem to find the missing side.

Here’s a Python program to compute the third side of a right-angled triangle given two sides:

python

Copy code

import math

def find\_third\_side(a, b, c):

# Determine which side is the hypotenuse (largest side)

sides = sorted([a, b, c])

# If the largest side is the hypotenuse

if math.isclose(sides[0]\*\*2 + sides[1]\*\*2, sides[2]\*\*2):

return sides[2]

else:

# If not a right-angled triangle or wrong input

return None

def main():

print("Enter the sides of the right-angled triangle.")

a = float(input("Enter the length of the first side: "))

b = float(input("Enter the length of the second side: "))

c = float(input("Enter the length of the third side: "))

result = find\_third\_side(a, b, c)

if result:

print(f"The third side (hypotenuse) of the right-angled triangle is: {result:.2f}")

else:

print("The given sides do not form a right-angled triangle.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

### Explanation:

1. **find\_third\_side function**:
   * Sorts the three sides to identify the largest one, which is the hypotenuse.
   * Uses the Pythagorean theorem to check if the given sides can form a right-angled triangle. If they do, returns the hypotenuse.
2. **main function**:
   * Takes three side lengths as input.
   * Calls find\_third\_side to determine which side is the hypotenuse and prints the result.

This program helps find the hypotenuse if two sides are given, or checks if the sides form a valid right-angled triangle. If the sides don’t form a right-angled triangle, it indicates that.

1. **Write a python program to print the frequency of each of the characters present in a given string.**

To print the frequency of each character in a given string, you can use a dictionary to count occurrences. Here’s a Python program that does this:

python

Copy code

def character\_frequency(s):

# Create an empty dictionary to store character frequencies

freq\_dict = {}

# Iterate over each character in the string

for char in s:

# Update the frequency count for each character

if char in freq\_dict:

freq\_dict[char] += 1

else:

freq\_dict[char] = 1

return freq\_dict

def main():

input\_string = input("Enter a string: ")

# Get the frequency of each character

frequencies = character\_frequency(input\_string)

# Print the frequencies

print("Character frequencies:")

for char, freq in frequencies.items():

print(f"'{char}': {freq}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

### Explanation:

1. **character\_frequency function**:
   * Initializes an empty dictionary freq\_dict to store character counts.
   * Iterates over each character in the string. If the character is already in the dictionary, it increments the count; otherwise, it adds the character to the dictionary with a count of 1.
2. **main function**:
   * Takes user input for the string.
   * Calls character\_frequency to get the frequency of each character.
   * Prints the frequencies in a readable format.

This program will correctly count and display the frequency of each character in the input string, including spaces and punctuation.