Here are the answers to your Python programming questions:

**1) What are the types of Applications?**

* **Web Applications:** Applications that run on web servers and are accessed via browsers (e.g., Gmail, Facebook).
* **Mobile Applications:** Apps developed for mobile devices (e.g., WhatsApp, Instagram).
* **Desktop Applications:** Software that runs on personal computers (e.g., Microsoft Word, VLC Media Player).
* **Embedded Systems Applications:** Programs that run on hardware systems designed for specific tasks (e.g., smartwatches, medical devices).
* **Console Applications:** Applications that run in a console or command-line interface (e.g., terminal commands, scripts).
* **Gaming Applications:** Designed for entertainment and interactive experiences (e.g., Minecraft, PUBG).

**2) What is programming?**

* Programming is the process of creating instructions that a computer can execute. These instructions are written in programming languages like Python, Java, or C++.

**3) What is Python?**

* Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple programming paradigms like procedural, object-oriented, and functional programming.

**4) Write a Python program to check if a number is positive, negative, or zero.**

python

Copy code

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

if num > 0:

print("The number is positive")

elif num < 0:

print("The number is negative")

else:

print("The number is zero")

**5) Write a Python program to get the factorial of a given number.**

python

Copy code

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

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

print("Factorial of", num, "is", factorial(num))

**6) Write a Python program to get the Fibonacci series of a given range.**

python

Copy code

def fibonacci(n):

fib\_seq = [0, 1]

while len(fib\_seq) < n:

fib\_seq.append(fib\_seq[-1] + fib\_seq[-2])

return fib\_seq

n = int(input("Enter the number of terms: "))

print(fibonacci(n))

**7) How is memory managed in Python?**

* Python uses dynamic memory allocation and an internal memory manager to handle memory. Python’s memory management involves:
  + **Private Heap Space:** Python objects and data structures are stored here.
  + **Garbage Collection:** Python has an automatic garbage collector that reclaims memory by removing unused objects.
  + **Memory Pools:** Python uses memory pools to efficiently manage small objects.

**8) What is the purpose of the continue statement in Python?**

* The continue statement in Python is used inside loops to skip the current iteration and move to the next iteration without executing the remaining code in the loop body for that iteration.

**9) Write a Python program to swap two numbers with and without using a temporary variable.**

Python

Copy code

# With temp variable

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

b = int(input("Enter second number: "))

temp = a

a = b

b = temp

print("After swapping: a =", a, "b =", b)

# Without temp variable

a, b = b, a

print("After swapping without temp: a =", a, "b =", b)

**10) Write a Python program to find whether a given number is even or odd.**

Python

Copy code

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

if num % 2 == 0:

print(f"{num} is an even number")

else:

print(f"{num} is an odd number")

**11) Write a Python program to test whether a passed letter is a vowel or not.**

Python

Copy code

letter = input ("Enter a letter: "). lower()

if letter in 'aeiou':

print(f"{letter} is a vowel")

else:

print(f"{letter} is not a vowel")

**12) Write a Python program to sum three given integers. If two values are equal, the sum will be zero.**

Python

Copy code

def sum\_of\_integers (a, b, c):

if a == b or b == c or a == c:

return 0

return a + b + c

a, b, c = int (input ()), int (input ()), int (input ())

print ("Sum:", sum\_of\_integers (a, b, c))

**13) Write a Python program that will return True if two given integers are equal or their sum or difference is 5.**

Python

Copy code

def check\_values (a, b):

return a == b or (a + b == 5) or (abs (a - b) == 5)

a, b = int (input ()), int (input ())

print (check\_values (a, b))

**14) Write a Python program to sum the first n positive integers.**

Python

Copy code

n = int (input ("Enter a positive integer: "))

sum\_n = n \* (n + 1) // 2

print ("Sum of first", n, "positive integers is", sum\_n)

**15) Write a Python program to calculate the length of a string.**

Python

Copy code

string = input ("Enter a string: ")

print ("Length of the string is:", len(string))

**16) Write a Python program to count the number of characters (character frequency) in a string.**

Python

Copy code

string = input ("Enter a string: ")

char\_freq = {}

for char in string:

char\_freq[char] = char\_freq.get (char, 0) + 1

print ("Character frequency:", char\_freq)

**17) What are negative indexes, and why are they used?**

* Negative indexes in Python are used to access elements from the end of a list or sequence. For example, list [-1] will return the last element of the list. This makes accessing elements relative to the sequence's end easier without calculating the exact index.

**18) Write a Python program to count occurrences of a substring in a string.**

Python

Copy code

string = input ("Enter the main string: ")

substring = input ("Enter the substring: ")

count = string. count(substring)

print(f"'{substring}' occurs {count} times in the main string.")

**19) Write a Python program to count the occurrences of each word in a given sentence.**

Python

Copy code

sentence = input("Enter a sentence: ")

words = sentence.split()

word\_count = {}

for word in words:

word\_count[word] = word\_count.get(word, 0) + 1

print("Word occurrences:", word\_count)

**20) Write a Python program to get a single string from two given strings, separated by a space, and swap the first two characters of each string.**

Python

Copy code

def swap\_first\_two(str1, str2):

new\_str1 = str2[:2] + str1[2:]

new\_str2 = str1[:2] + str2[2:]

return new\_str1 + " " + new\_str2

str1 = input("Enter first string: ")

str2 = input("Enter second string: ")

print(swap\_first\_two(str1, str2))

**21) Write a Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing', then add 'ly'. If the string length is less than 3, leave it unchanged.**

Python

Copy code

def modify\_string(s):

if len(s) < 3:

return s

if s.endswith('ing'):

return s + 'ly'

return s + 'ing'

s = input("Enter a string: ")

print(modify\_string(s))

**22) Write a Python function to reverse a string if its length is a multiple of 4.**

Python

Copy code

def reverse\_if\_multiple\_of\_four(s):

if len(s) % 4 == 0:

return s[::-1]

return s

s = input("Enter a string: ")

print(reverse\_if\_multiple\_of\_four(s))

**23) Write a Python program to get a string made of the first 2 and the last 2 characters from a given string. If the string length is less than 2, return an empty string.**

Python

Copy code

def first\_last\_two(s):

if len(s) < 2:

return ''

return s[:2] + s[-2:]

s = input("Enter a string: ")

print(first\_last\_two(s))

**24) Write a Python function to insert a string in the middle of a string.**

Python

Copy code

def insert\_in\_middle(s, word):

mid = len(s) // 2

return s[:mid] + word + s[mid:]

s = input("Enter a string: ")

word = input("Enter a word to insert: ")

print(insert\_in\_middle(s, word))

**25) What is List? How will you reverse a list?**

* **List**: A list is a mutable, ordered collection of items in Python, which can be of different data types.
* **Reverse a list**:

Python

Copy code

my\_list = [1, 2, 3, 4, 5]

my\_list.reverse() # Method 1

print(my\_list)

# Method 2

my\_list = my\_list[::-1]

print(my\_list)

**26) How will you remove the last object from a list?**

Python

Copy code

my\_list = [1, 2, 3, 4, 5]

my\_list.pop() # Removes last object

print(my\_list)

**27) Suppose list1 is [2, 33, 222, 14, 25], what is list1[-1]?**

* list1[-1] is 25, as negative indexing starts from the end of the list.

**28) Differentiate between append() and extend() methods.**

* **append()**: Adds its argument as a single element to the end of the list.
* **extend()**: Adds each element of its argument (an iterable) to the list.

**29) Write a Python function to get the largest number, smallest number, and sum of all elements from a list.**

Python

Copy code

def list\_stats(lst):

return max(lst), min(lst), sum(lst)

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

print(list\_stats(lst))

**30) How will you compare two lists?**

* Two lists can be compared using ==, which checks if both lists have the same elements in the same order:

Python

Copy code

list1 = [1, 2, 3]

list2 = [1, 2, 3]

print(list1 == list2) # True

**31) Write a Python program to count the number of strings where the string length is 2 or more and the first and last character are the same from a given list of strings.**

Python

Copy code

def count\_special\_strings(strings):

return sum(1 for s in strings if len(s) >= 2 and s[0] == s[-1])

strings = ['abc', 'xyz', 'aba', '1221']

print(count\_special\_strings(strings))

**32) Write a Python program to remove duplicates from a list.**

Python

Copy code

my\_list = [1, 2, 2, 3, 4, 4, 5]

unique\_list = list(set(my\_list))

print(unique\_list)

**33) Write a Python program to check if a list is empty or not.**

Python

Copy code

my\_list = []

if not my\_list:

print("List is empty")

else:

print("List is not empty")

**34) Write a Python function that takes two lists and returns True if they have at least one common member.**

Python

Copy code

def common\_member(list1, list2):

return any(item in list1 for item in list2)

list1 = [1, 2, 3]

list2 = [3, 4, 5]

print(common\_member(list1, list2))

**35) Write a Python program to generate and print a list of the first and last 5 elements where the values are squares of numbers between 1 and 30.**

Python

Copy code

squares = [x\*\*2 for x in range(1, 31)]

print(squares[:5] + squares[-5:])

**36) Write a Python function that takes a list and returns a new list with unique elements of the first list.**

Python

Copy code

def unique\_elements(lst):

return list(set(lst))

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

print(unique\_elements(lst))

**37) Write a Python program to convert a list of characters into a string.**

Python

Copy code

char\_list = ['a', 'b', 'c', 'd']

string = ''.join(char\_list)

print(string)

**38) Write a Python program to select an item randomly from a list.**

Python

Copy code

import random

my\_list = [1, 2, 3, 4, 5]

print(random.choice(my\_list))

**39) Write a Python program to find the second smallest number in a list.**

Python

Copy code

def second\_smallest(numbers):

unique\_numbers = list(set(numbers))

unique\_numbers.sort()

return unique\_numbers[1]

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

print(second\_smallest(numbers))

**40) Write a Python program to get unique values from a list.**

Python

Copy code

my\_list = [1, 2, 2, 3, 4, 4, 5]

unique\_values = list(set(my\_list))

print(unique\_values)

**41) Write a Python program to check whether a list contains a sub-list.**

Python

Copy code

def is\_sublist(lst, sublist):

return str(sublist)[1:-1] in str(lst)

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

sublist = [2, 3]

print(is\_sublist(lst, sublist))

**42) Write a Python program to split a list into different variables.**

Python

Copy code

my\_list = [1, 2, 3]

a, b, c = my\_list

print(a, b, c)

**43) What is a tuple? Difference between list and tuple.**

* A **tuple** is an immutable, ordered collection of items.
* **Difference**:
  + **Lists** are mutable (can be changed).
  + **Tuples** are immutable (cannot be changed).

**44) Write a Python program to create a tuple with different data types.**

Python

Copy code

my\_tuple = (1, "Hello", 3.14, True)

print(my\_tuple)

**45) Write a Python program to unzip a list of tuples into individual lists.**

Python

Copy code

tuples\_list = [(1, 'a'), (2, 'b'), (3, 'c')]

numbers, letters = zip(\*tuples\_list)

print(list(numbers))

print(list(letters))

**47) How will you create a dictionary using tuples in Python?**

Python

Copy code

# Creating a dictionary using a list of tuples

tuples\_list = [('a', 1), ('b', 2), ('c', 3)]

dictionary = dict(tuples\_list)

print(dictionary)

* **Explanation**: You can use the dict() function to convert a list of tuples into a dictionary.

**48) Write a Python script to sort (ascending and descending) a dictionary by value.**

Python

Copy code

# Ascending order

my\_dict = {'a': 3, 'b': 1, 'c': 2}

sorted\_dict\_asc = dict(sorted(my\_dict.items(), key=lambda item: item[1]))

print("Ascending:", sorted\_dict\_asc)

# Descending order

sorted\_dict\_desc = dict(sorted(my\_dict.items(), key=lambda item: item[1], reverse=True))

print("Descending:", sorted\_dict\_desc)

**49) Write a Python script to concatenate the following dictionaries to create a new one.**

Python

Copy code

dict1 = {'a': 1, 'b': 2}

dict2 = {'c': 3, 'd': 4}

dict3 = {'e': 5, 'f': 6}

# Merging dictionaries

result = {\*\*dict1, \*\*dict2, \*\*dict3}

print(result)

**50) Write a Python script to check if a given key already exists in a dictionary.**

Python

Copy code

my\_dict = {'a': 1, 'b': 2, 'c': 3}

key = 'b'

if key in my\_dict:

print(f"{key} exists in the dictionary")

else:

print(f"{key} does not exist in the dictionary")

**51) How do you traverse through a dictionary object in Python?**

* You can traverse a dictionary using a for loop:

Python

Copy code

my\_dict = {'a': 1, 'b': 2, 'c': 3}

# Traverse keys and values

for key, value in my\_dict.items():

print(f"Key: {key}, Value: {value}")

**52) How do you check the presence of a key in a dictionary?**

* Use the in keyword:

Python

Copy code

my\_dict = {'a': 1, 'b': 2}

print('a' in my\_dict) # Output: True

**53) Write a Python script to print a dictionary where the keys are numbers between 1 and 15.**

Python

Copy code

my\_dict = {i: i\*\*2 for i in range(1, 16)}

print(my\_dict)

**54) Write a Python program to check if multiple keys exist in a dictionary.**

Python

Copy code

my\_dict = {'a': 1, 'b': 2, 'c': 3}

keys\_to\_check = ['a', 'b']

if all(key in my\_dict for key in keys\_to\_check):

print("All keys exist in the dictionary")

else:

print("Not all keys exist")

**55) Write a Python script to merge two Python dictionaries.**

Python

Copy code

dict1 = {'a': 1, 'b': 2}

dict2 = {'c': 3, 'd': 4}

# Merging

merged\_dict = {\*\*dict1, \*\*dict2}

print(merged\_dict)

**56) Write a Python program to map two lists into a dictionary.**

Python

Copy code

keys = ['a', 'b', 'c']

values = [1, 2, 3]

# Mapping two lists into a dictionary

mapped\_dict = dict(zip(keys, values))

print(mapped\_dict)

**57) Write a Python program to find the highest 3 values in a dictionary.**

Python

Copy code

my\_dict = {'a': 300, 'b': 400, 'c': 200, 'd': 500}

highest\_values = sorted(my\_dict.values(), reverse=True)[:3]

print("Highest 3 values:", highest\_values)

**58) Write a Python program to combine values in a Python list of dictionaries.**

Python

Copy code

from collections import Counter

data = [{'item': 'item1', 'amount': 400}, {'item': 'item2', 'amount': 300}, {'item': 'item1', 'amount': 750}]

result = Counter()

for d in data:

result[d['item']] += d['amount']

print(result)

**59) Write a Python program to create a dictionary from a string (track the count of letters).**

Python

Copy code

from collections import Counter

s = "w3resource"

letter\_count = Counter(s)

print(letter\_count)

**60) Write a Python function to calculate the factorial of a number (a nonnegative integer).**

Python

Copy code

def factorial(n):

if n == 0 or n == 1:

return 1

else:

return n \* factorial(n-1)

print(factorial(5)) # Output: 120

**61) Write a Python function to check whether a number is in a given range.**

Python

Copy code

def is\_in\_range(n, start, end):

return start <= n <= end

print(is\_in\_range(5, 1, 10)) # Output: True

**62) Write a Python function to check whether a number is perfect or not.**

Python

Copy code

def is\_perfect(n):

divisors\_sum = sum(i for i in range(1, n) if n % i == 0)

return divisors\_sum == n

print(is\_perfect(6)) # Output: True (6 is a perfect number)

**63) Write a Python function that checks whether a passed string is a palindrome or not.**

Python

Copy code

def is\_palindrome(s):

return s == s[::-1]

print(is\_palindrome("madam")) # Output: True

**63) Write a Python function to check whether a number is perfect or not.** A perfect number is a number that is equal to the sum of its divisors, excluding itself.

Python

Copy code

def is\_perfect(n):

divisors\_sum = sum(i for i in range(1, n) if n % i == 0)

return divisors\_sum == n

# Test the function

print(is\_perfect(6)) # Output: True (6 is a perfect number: 1 + 2 + 3 = 6)

**64) Write a Python function that checks whether a passed string is a palindrome or not.** A palindrome is a string that reads the same forwards and backward.

Python

Copy code

def is\_palindrome(s):

return s == s[::-1]

# Test the function

print(is\_palindrome("madam")) # Output: True

print(is\_palindrome("hello")) # Output: False

**65) How many basic types of functions are available in Python?** There are **two basic types of functions** in Python:

1. **Built-in functions**: These are functions that are already present in Python (e.g., print(), len(), input()).
2. **User-defined functions**: These are functions that are created by the user using the def keyword.

**66) How can you pick a random item from a list or tuple?** You can use the random.choice() method from the random module.

python

Copy code

import random

my\_list = [1, 2, 3, 4, 5]

print(random.choice(my\_list)) # Picks a random item from the list

**67) How can you pick a random item from a range?** You can use the random.randrange() method to pick a random number from a specified range.

python

Copy code

import random

print(random.randrange(1, 10)) # Random number between 1 and 9

**68) How can you get a random number in Python?** You can use random.randint() for an inclusive random integer within a range or random.random() for a random float between 0 and 1.

python

Copy code

import random

print(random.randint(1, 100)) # Random integer between 1 and 100

print(random.random()) # Random float between 0 and 1

**69) How will you set the starting value in generating random numbers?** You can set the starting value (seed) using random.seed(). It ensures the reproducibility of the random numbers.

Python

Copy code

import random

random.seed(10)

print(random.randint(1, 100)) # Will give the same result if run with the same seed

**70) How will you randomize the items of a list in place?** Use random.shuffle() to randomize the items in a list in place.

Python

Copy code

import random

my\_list = [1, 2, 3, 4, 5]

random.shuffle(my\_list)

print(my\_list) # The list is shuffled in place

**71) What is the file function in Python? What are the keywords to create and write a file?** The file function in Python is used to work with files (reading, writing, or appending). The common modes for opening a file are:

* 'r' : Read (default mode)
* 'w' : Write (creates a new file or truncates if exists)
* 'a' : Append (adds data to the end of the file)

To create a file and write to it:

Python

Copy code

with open('example.txt', 'w') as f:

f.write("Hello, world!")

**72) Write a Python program to read an entire text file.**

Python

Copy code

with open('example.txt', 'r') as f:

content = f.read()

print(content)

**73) Write a Python program to append text to a file and display the text.**

Python

Copy code

with open('example.txt', 'a') as f:

f.write("\nAppended text.")

with open('example.txt', 'r') as f:

content = f.read()

print(content)

**74) Write a Python program to read the first n lines of a file.**

Python

Copy code

def read\_first\_n\_lines(file\_name, n):

with open(file\_name, 'r') as f:

for \_ in range(n):

print(f.readline(), end='')

# Example usage

read\_first\_n\_lines('example.txt', 2)

**75) Write a Python program to read the last n lines of a file.**

Python

Copy code

def read\_last\_n\_lines(file\_name, n):

with open(file\_name, 'r') as f:

lines = f.readlines()

for line in lines[-n:]:

print(line, end='')

# Example usage

read\_last\_n\_lines('example.txt', 2)

**76) Write a Python program to read a file line by line and store it into a list.**

Python

Copy code

def file\_to\_list(file\_name):

with open(file\_name, 'r') as f:

return f.readlines()

# Example usage

lines\_list = file\_to\_list('example.txt')

print(lines\_list)

**77) Write a Python program to read a file line by line and store it into a variable.**

Python

Copy code

def file\_to\_string(file\_name):

with open(file\_name, 'r') as f:

return f.read()

# Example usage

file\_content = file\_to\_string('example.txt')

print(file\_content)

**78) Write a Python program to find the longest word in a file.**

Python

Copy code

def find\_longest\_word(file\_name):

with open(file\_name, 'r') as f:

words = f.read().split()

longest\_word = max(words, key=len)

return longest\_word

# Example usage

print(find\_longest\_word('example.txt'))

**79) Write a Python program to count the number of lines in a text file.**

Python

Copy code

def count\_lines(file\_name):

with open(file\_name, 'r') as f:

return len(f.readlines())

# Example usage

print(count\_lines('example.txt'))

**80) Write a Python program to count the frequency of words in a file.**

Python

Copy code

from collections import Counter

def word\_frequency(file\_name):

with open(file\_name, 'r') as f:

words = f.read().split()

return Counter(words)

# Example usage

print(word\_frequency('example.txt'))

**81) Write a Python program to write a list to a file.**

Python

Copy code

def write\_list\_to\_file(file\_name, my\_list):

with open(file\_name, 'w') as f:

for item in my\_list:

f.write(f"{item}\n")

# Example usage

my\_list = ['apple', 'banana', 'cherry']

write\_list\_to\_file('output.txt', my\_list)

**82) Write a Python program to copy the contents of a file to another file.**

Python

Copy code

def copy\_file(source\_file, destination\_file):

with open(source\_file, 'r') as src:

content = src.read()

with open(destination\_file, 'w') as dest:

dest.write(content)

# Example usage

copy\_file('example.txt', 'copy\_example.txt')

**83) Explain Exception handling? What is an Error in Python?**

* **Exception Handling** in Python is the process of responding to the occurrence of exceptions—runtime errors that disrupt the normal flow of a program. Python uses the try-except blocks to handle exceptions.
* An **Error** in Python can be a **syntax error** (detected before code execution) or an **exception** (runtime error).

**Example of exception handling:**

Python

Copy code

try:

a = 10 / 0

except ZeroDivisionError:

print("Cannot divide by zero!")

**84) How many except statements can a try-except block have? Name some built-in exception classes.**

* A try-except block can have **multiple except statements** to handle different exceptions.

**Some built-in exception classes:**

1. ZeroDivisionError
2. ValueError
3. TypeError
4. IndexError

**85) When will the else part of try-except-else be executed?**

* The **else** block is executed **only if no exceptions** are raised in the try block.

Python

Copy code

try:

result = 10 / 2

except ZeroDivisionError:

print("Cannot divide by zero.")

else:

print("Division successful:", result)

**86) Can one block of except statements handle multiple exceptions?**

* Yes, you can handle multiple exceptions by listing them as a tuple in a single except block.

Python

Copy code

try:

result = int("abc")

except (ValueError, TypeError):

print("Caught an exception!")

**87) When is the finally block executed?**

* The **finally** block is always executed, regardless of whether an exception occurred or not.

Python

Copy code

try:

result = 10 / 0

except ZeroDivisionError:

print("Error occurred!")

finally:

print("This will always run.")

**88) What happens when '1' == 1 is executed?**

* This comparison returns False because it compares a **string** ('1') with an **integer** (1), and they are of different types.

**89) How do you handle exceptions with try-except-finally in Python? Explain with coding snippets.**

* You can use try to run a block of code, except to catch specific errors, and finally to execute cleanup code that will run regardless of an error.

Python

Copy code

try:

result = 10 / 0

except ZeroDivisionError:

print("Cannot divide by zero!")

finally:

print("Execution finished.")

**90) Write a Python program that asks the user to enter only odd numbers, else raise an exception.**

Python

Copy code

def check\_odd\_number():

try:

num = int(input("Enter an odd number: "))

if num % 2 == 0:

raise ValueError("This is not an odd number.")

except ValueError as e:

print(e)

# Example usage

check\_odd\_number()