# Python Codes

**Ques-1:**

- num  = [1,2,3,4,5,6]
- Write a python code to create a new list that contains,Only the even numbers from the original list.
- Each even number should be multiplied by 2.

**Ans-1:**

```
num = [1, 2, 3, 4, 5, 6]

new_list = []

for x in num:

    if x % 2 == 0:

        new_list.append(x * 2)

print(new_list)
```

**Ans-2:**

```
num = [1, 2, 3, 4, 5, 6]

# Using list comprehension

new_list = [x * 2 for x in num if x % 2 == 0]

print(new_list)
```

---

**Ques-2:**

- You have a list of numbers:
- numbers = [10, 25, 30, 45, 25, 60]

**Write Python code to:**

- Find the index of the first occurrence of `25`
- Insert `35` at index 2
- Remove the last item
- Copy the list
- Clear all items from the original list

**Ans:**

```
numbers = [10, 25, 30, 45, 25, 60]

# 1. Find the index of the first occurrence of 25
```

## Python Codes

```
index_25 = -1
for i in range(len(numbers)):
    if numbers[i] == 25:
        index_25 = i
        break
print("Index of first 25:", index_25)
# 2. Insert 35 at index 2 (manual reconstruction)
new_list = []
for i in range(len(numbers)):
    if i == 2:
        new_list.append(35)  # insert before index 2
    new_list.append(numbers[i])
numbers = new_list
print("After inserting 35 at index 2:", numbers)
# 3. Remove the last item (manual reconstruction)
new_list = []
for i in range(len(numbers) - 1):  # skip the last element
    new_list.append(numbers[i])
numbers = new_list
print("After removing the last item:", numbers)
# 4. Copy the list (manual copy)
numbers_copy = []
for i in range(len(numbers)):
    numbers_copy.append(numbers[i])
print("Copied list:", numbers_copy)
# 5. Clear all items from the original list (manual clearing)
numbers = []
print("Original list after clearing:", numbers)
```

# Python Codes

**Ques-3:**

- You have a list of numbers and need to find the first even number greater than 10:
- numbers = [3, 7, 12, 5, 18, 9, 20]
- Write Python code to find and print the first even number greater than 10.

**Ans:**

```
numbers = [3, 7, 12, 5, 18, 9, 20]

first_even_gt_10 = None   # placeholder

for num in numbers:

    if num % 2 == 0 and num > 10:   # check even and > 10

        first_even_gt_10 = num

        break   # stop at the first match

print("First even number greater than 10:", first_even_gt_10)
```

---

**Ques-4 Generate the random strings using python.**

**Ans:**

```
# Manual random string generator without built-in random/string modules

# Step 1: Define our "alphabet"

alphabet =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"

# Step 2: Simple linear congruential generator (LCG)

def lcg(seed, length):

    a = 1664525

    c = 1013904223

    m = 2**32

    numbers = []

    for _ in range(length):

        seed = (a * seed + c) % m

        numbers.append(seed)

    return numbers
```

## Python Codes

```
# Step 3: Map generated numbers to characters in our alphabet
def random_string(seed, size):
    nums = lcg(seed, size)
    result = "".join(alphabet[n % len(alphabet)] for n in nums)
    return result
# Example usage
print("Random string:", random_string(seed=12345, size=8))
```

✅ **Output example:**

Random string: kJ8pZ2bA

**Notes:**

- This produces **deterministic "random" strings** based on the seed.
- No built-in random or string modules are used.
- The LCG constants used are standard for demonstration purposes.

---

**Ques-5:**

**You have a list of fruits and their prices:**

- fruits = ['apple', 'banana', 'cherry']
- prices = [100, 200, 300]
- Create a dictionary where keys are fruits and values are prices multiplied by 1.1 (adding 10% tax).

**Expected Output:**

{'apple': 110.0, 'banana': 220.0, 'cherry': 330.0}

**Ans:**

```
fruits = ['apple', 'banana', 'cherry']
prices = [100, 200, 300]
# Manual dictionary creation (without built-ins like zip or dict comprehension)
fruit_price_dict = {}
for i in range(len(fruits)):
    # calculate price with 10% tax
```

```
    taxed_price = prices[i] * 1.1
    # assign manually
    fruit_price_dict[fruits[i]] = taxed_price
print(fruit_price_dict)
```

---

**6. How do we check if a string is a polindrome?**

* A **palindrome** is a string that reads the same forwards and backwards.

* You can check it manually using a **two-pointer approach** without built-in reverse functions.

**Example:**

```
def is_palindrome_manual_reverse(s):
    rev = ""
    i = len(s) - 1
    while i >= 0:
        rev += s[i]
        i -= 1
    if s == rev:
        return True
    else:
        return False
print(is_palindrome_manual_reverse("madam"))  # True
print(is_palindrome_manual_reverse("hello"))  # False
```

**Key point:**

“Compare characters from both ends moving towards the center; if all match, it’s a palindrome.”

---

**7. How do you reverse a dictionary in python?**

**Ans-1 Swap keys and values:**

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
# Create an empty dictionary
reversed_dict = {}
```

## Python Codes

```
# Iterate manually
for key in my_dict:
    value = my_dict[key]
    reversed_dict[value] = key
print(reversed_dict)   # {1: 'a', 2: 'b', 3: 'c'}
```

**Ans-2 Reverse the order of items:**

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
# Step 1: Collect items into a list
items = []
for key in my_dict:
    items.append((key, my_dict[key]))
# Step 2: Walk backwards and build new dict
reversed_dict = {}
index = len(items) - 1
while index >= 0:
    pair = items[index]
    reversed_dict[pair[0]] = pair[1]
    index -= 1
print(reversed_dict)   # {'c': 3, 'b': 2, 'a': 1}
```

---

**8. Write a program to Count the no. of words in a string in python?**

```
def count_words(s):
    count = 0
    in_word = False
    for ch in s:
        if ch != " " and not in_word:
            # Starting a new word
            count += 1
            in_word = True
```

# Python Codes

```
        elif ch == " ":
            in_word = False
    return count
text = "Hello world this is python"
print(count_words(text))   # 5
```

---

**9. Count the no.of vowels in a string?**

```
def count_vowels(s):
    vowels = "aeiouAEIOU"
    count = 0
    for ch in s:
        # manually check if ch is in vowels
        for v in vowels:
            if ch == v:
                count += 1
                break
    return count
text = "Hello World"
print(count_vowels(text))   # 3
```

---

**10. Find the first Non-Repeating Characters?**

```
def first_non_repeating(s):
    freq = {}   # manual frequency dict
    # Step 1: Count characters manually
    for ch in s:
        if ch not in freq:
            freq[ch] = 1
        else:
            freq[ch] += 1
```

## Python Codes

```
    # Step 2: Find the first character with count = 1
    for ch in s:
        if freq[ch] == 1:
            return ch
    return None  # if no non-repeating character
text = "aabbcdeff"
print(first_non_repeating(text))  # c
```

---

**11. Find the missing number from a list(1 to N)?**

```
def find_missing_number(arr, N):
    total = N * (N + 1) // 2
    sum_arr = 0
    for num in arr:
        sum_arr += num
    return total - sum_arr
numbers = [1, 2, 4, 5, 6]  # Missing 3
N = 6
print(find_missing_number(numbers, N))  # 3
```

---

**12. Count the frequency of elements in list?**

```
def count_frequency(lst):
    freq = {}   # empty dictionary
    for item in lst:
        if item not in freq:
            freq[item] = 1
        else:
            freq[item] += 1
    return freq
# Example
```

## Python Codes

```
my_list = [1, 2, 2, 3, 1, 4, 2]
print(count_frequency(my_list))
# Output: {1: 2, 2: 3, 3: 1, 4: 1}
```

---

**13. Check if two strings are anagrams?**

```
def are_anagrams(str1, str2):
    # Step 1: Check length
    length1 = length2 = 0
    for _ in str1:
        length1 += 1
    for _ in str2:
        length2 += 1
    if length1 != length2:
        return False
    # Step 2: Count frequency for str1
    freq1 = {}
    for ch in str1:
        if ch not in freq1:
            freq1[ch] = 1
        else:
            freq1[ch] += 1
    # Step 3: Count frequency for str2
    freq2 = {}
    for ch in str2:
        if ch not in freq2:
            freq2[ch] = 1
        else:
            freq2[ch] += 1
    # Step 4: Compare frequencies
```

## Python Codes

```
    for ch in freq1:
        if ch not in freq2 or freq1[ch] != freq2[ch]:
            return False
    return True
print(are_anagrams("listen", "silent"))  # True
print(are_anagrams("hello", "world"))    # False
```

---

**14. Remove duplicates from a list?**

```
def remove_duplicates(lst):
    unique = []
    for item in lst:
        is_present = False
        # Check if item already in unique list
        for u in unique:
            if item == u:
                is_present = True
                break
        if not is_present:
            unique.append(item)
    return unique
my_list = [1, 2, 2, 3, 1, 4, 2]
print(remove_duplicates(my_list))  # [1, 2, 3, 4]
```

---

**15. Find the max value in a list?**

```
def find_max(lst):
    if not lst:  # handle empty list
        return None
    max_val = lst[0]
    for item in lst:
```

## Python Codes

```
        if item > max_val:
            max_val = item
    return max_val
my_list = [3, 7, 2, 9, 4]
print(find_max(my_list))  # 9
```

---

**16. Convert numeric strings to corresponding numeric words and integers**

```
def num_to_words(num):
    ones = ["Zero","One","Two","Three","Four","Five","Six","Seven","Eight","Nine"]
    tens = ["", "", "Twenty","Thirty","Forty","Fifty","Sixty","Seventy","Eighty","Ninety"]
    teens =
["Ten","Eleven","Twelve","Thirteen","Fourteen","Fifteen","Sixteen","Seventeen","Eight
een","Nineteen"]


    num = int(num)  # Convert string to integer


    if 0 <= num < 10:
        return ones[num]
    elif 10 <= num < 20:
        return teens[num-10]
    elif 20 <= num < 100:
        return tens[num // 10] + ("" if num % 10 == 0 else "-" + ones[num % 10])
    else:
        return str(num)  # For numbers >= 100, return as string (can extend logic)
numeric_strings = ["5", "12", "45", "99"]
for s in numeric_strings:
    integer_value = int(s)
    word_value = num_to_words(s)
    print(f"String: {s} -> Integer: {integer_value}, Word: {word_value}")
```

# Python Codes

---

### 17. Fibonacci Series

```
def fibonacci(n):
  fib_sequence = [0, 1]
  while len(fib_sequence) < n:
    fib_sequence.append(fib_sequence[-1] + fib_sequence[-2])
  return fib_sequence
print(fibonacci(10))
```

---

### 18. Count the number of strings in input

```
def count_strings(lst):
   count = 0
   for item in lst:
      if type(item) == str:
         count += 1
   return count
data = [10, "hello", 3.5, "world", True, "python"]
print("Number of strings:", count_strings(data))
```

---

### 19. Repeated digit summation

**55555=5+5+5+5+5=25=2+5=7 (Loop has to continue till the outcome becomes a single digit)**

```
num = 55555
while num >= 10:          # repeat until single digit
   total = 0
   while num > 0:
      digit = num % 10  # get last digit
      total += digit
      num //= 10        # remove last digit
```

## Python Codes

```
    num = total
print(num)
```

**Output:** 7

**20. Reverse a String Using Function (Without Built-in Functions)**

```
def reverse_string(s):
    rev = ""
    i = len(s) - 1
    while i >= 0:
        rev += s[i]
        i -= 1
    return rev
print(reverse_string("python"))
```