**Lab#6**

**QUESTIONS**

**QUESTION # 1**

**Write a program that generates a list of odd numbers from 1 to 1000, prints numbers and then prints the sum of numbers.**

**Input:**

For num in range(1, 1000):

If num % 2 != 0:

Print(num)

Sum\_natural = 0

For I in range(1, num + 1):

Sum\_natural += i

Print(f”The sum of natural numbers up to {num} is: {sum\_natural}”)

**Output:**

Odd numbers from 1 to 1000: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155, 157, 159, 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 181, 183, 185, 187, 189, 191, 193, 195, 197, 199, 201, 203, 205, 207, 209, 211, 213, 215, 217, 219, 221, 223, 225, 227, 229, 231, 233, 235, 237, 239, 241, 243, 245, 247, 249, 251, 253, 255, 257, 259, 261, 263, 265, 267, 269, 271, 273, 275, 277, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 299, 301, 303, 305, 307, 309, 311, 313, 315, 317, 319, 321, 323, 325, 327, 329, 331, 333, 335, 337, 339, 341, 343, 345, 347, 349, 351, 353, 355, 357, 359, 361, 363, 365, 367, 369, 371, 373, 375, 377, 379, 381, 383, 385, 387, 389, 391, 393,395, 397, 399, 401, 403, 405, 407, 409, 411, 413, 415, 417, 419, 421, 423, 425, 427, 429, 431, 433, 435, 437, 439, 441, 443, 445, 447, 449, 451, 453, 455, 457, 459, 461, 463, 465, 467, 469, 471, 473, 475, 477, 479, 481, 483, 485, 487, 489, 491, 493, 495, 497, 499, 501, 503, 505, 507, 509, 511, 513, 515, 517, 519, 521, 523, 525, 527, 529, 531, 533, 535, 537, 539, 541, 543, 545, 547, 549, 551, 553, 555, 557, 559, 561, 563, 565, 567, 569, 571, 573, 575, 577, 579, 581, 583, 585, 587, 589, 591, 593, 595, 597, 599, 601, 603, 605, 607, 609, 611, 613, 615, 617, 619, 621, 623, 625, 627, 629, 631, 633, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 663, 665, 667, 669, 671, 673, 675, 677, 679, 681, 683, 685, 687, 689, 691, 693, 695, 697, 699, 701, 703, 705, 707, 709, 711, 713, 715, 717, 719, 721, 723, 725, 727, 729, 731, 733, 735, 737, 739, 741, 743, 745, 747, 749, 751, 753, 755, 757, 759, 761, 763, 765, 767, 769, 771, 773, 775, 777, 779, 781, 783, 785, 787, 789, 791, 793, 795, 797, 799, 801, 803, 805, 807, 809, 811, 813, 815, 817, 819, 821, 823, 825, 827, 829, 831, 833, 835, 837, 839, 841, 843, 845, 847, 849, 851, 853, 855, 857, 859, 861, 863, 865, 867, 869, 871, 873, 875, 877, 879, 881, 883, 885, 887, 889, 891, 893, 895, 897, 899, 901, 903, 905, 907, 909, 911, 913, 915, 917, 919, 921, 923, 925, 927, 929, 931, 933, 935, 937, 939,941, 943, 945, 947, 949, 951, 953, 955, 957, 959, 961, 963, 965, 967, 969, 971, 973, 975, 977, 979, 981, 983, 985, 987, 989, 991, 993, 995, 997, 999] Sum of odd number: 250000

[Program finished]

**QUESTION#2**

**Write a program that generates a list of 20 random numbers, prints the list and then finds the (1) largest number, and (2) smallest number using linear search method.**

**Input:**

Import random

Random\_numbers = [random. Randint(1, 100) for \_ in range (20)]

Print (“List of random numbers:”)

Print(random\_numbers)

Largest = max(random\_numbers)

Smallest = min(random\_numbers)

Print (“Largest number:”, largest)

Print (“Smallest number:”, smallest)

**Output:**

List of random numbers:

[18, 97, 66, 57, 64, 83, 96, 45, 16, 99, 87, 97, 46, 3, 74, 73, 78, 44, 37, 47]

Largest number: 99

Smallest number: 3

[Program finished]

**QUESTION#3**

**Write a Python program that prompts the user to enter 5 numbers, stores them in a list, and then prints the sum of all even numbers in the list.**

**Input:**

numbers = []

for i in range (5):

num = int(input (f"Enter number {i+1}: "))

numbers.append(num)

even\_sum = sum(num for num in numbers if num % 2 == 0)

print("Sum of even numbers:", even\_sum)

**Output:**

Enter number 1: 5

Enter number 2: 6

Enter number 3: 7

Enter number 4: 8

Enter number 5: 9

Sum of even numbers: 14

[Program finished]

**QUESTION#4**

**Create a tuple containing the names of 5 countries. Write a program that iterates**

**through the tuple and prints only the countries with a length greater than 7 characters.**

**Input:**

Countries = (‘United States’, ‘Canada’, ‘Australia’, ‘Netherlands’, ‘Switzerland’)

Print (“Countries with a length greater than 7 characters:”)

For country in countries:

If len(country) > 7:

Print(country)

**Output:**

Countries with a length greater than 7 characters:

United States

Australia

Netherlands

Switzerland

[Program finished]

**QUESTION#5**

**Generate a list of 50 integers. Write a program that iterates through the list and prints the square of each number if it's even, and the cube if it's odd.**

**Input:**

integer\_list = list(range(1, 51))

for num in integer\_list:

if num % 2 == 0:

print(f"Square of {num}: {num\*\*2}")

else:

print(f"Cube of {num}: {num\*\*3}")

**Output:**

Cube of 1: 1

Square of 2: 4

Cube of 3: 27

Square of 4: 16

Cube of 5: 125

Square of 6: 36

Cube of 7: 343

Square of 8: 64

Cube of 9: 729

Square of 10: 100

Cube of 11: 1331

Square of 12: 144

Cube of 13: 2197

Square of 14: 196

Cube of 15: 3375

Square of 16: 256

Cube of 17: 4913

Square of 18: 324

Cube of 19: 6859

Square of 20: 400

Cube of 21: 9261

Square of 22: 484

Cube of 23: 12167

Square of 24: 576

Cube of 25: 15625

Square of 26: 676

Cube of 27: 19683

Square of 28: 784

Cube of 29: 24389

Square of 30: 900

Cube of 31: 29791

Square of 32: 1024

Cube of 33: 35937

Square of 34: 1156

Cube of 35: 42875

Square of 36: 1296

Cube of 37: 50653

Square of 38: 1444

Cube of 39: 59319

Square of 40: 1600

Cube of 41: 68921

Square of 42: 1764

Cube of 43: 79507

Square of 44: 1936

Cube of 45: 91125

Square of 46: 2116

Cube of 47: 103823

Square of 48: 2304

Cube of 49: 117649

Square of 50: 2500

[Program finished]

**QUESTION#6**

**Write a program that takes two sets as input and prints the union of the two sets.**

**Input:**

Set1\_input = input (“Enter elements of the first set separated by spaces: “)

Set1 = set1\_input.split()

Set2\_input = input (“Enter elements of the second set separated by spaces: “)

Set2 = set2\_input.split()

Set1 = set(set1)

Set2 = set(set2)

Union\_set = set1.union(set2)

Print (“Union of the two sets:”, union\_set)

**Output:**

Enter elements of the first set separated by spaces: 2

Enter elements of the second set separated by spaces: 3

Union of the two sets: {‘3’, ‘2’}

[Program finished]

**QUESTION#7**

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

**Input:**

Char\_list = [‘a’, ‘b’, ‘c’, ‘d’, ‘e’]

String\_result = ‘’.join(char\_list)

Print(“Resulting string:”, string\_result)

**Output:**

Resulting string: abcde

[Program finished]

**QUESTION#8**

**Write a program that randomly selects an item from list\_1 and then maps it to a**

**randomly selected item of list\_2.**

**Input:**

Import random

List\_1 = [‘apple’, ‘banana’, ‘cherry’, ‘date’, ‘elderberry’]

List\_2 = [‘cat’, ‘dog’, ‘elephant’, ‘fox’, ‘giraffe’]

Random\_item\_1 = random.choice(list\_1)

Random\_item\_2 = random.choice(list\_2)

Print(“Randomly selected item from list\_1:”, random\_item\_1)

Print(“Randomly selected item from list\_2:”, random\_item\_2)

**Output:**

Randomly selected item from list\_1: cherry

Randomly selected item from list\_2: cat

[Program finished]

**QUESTION#9**

**Write a Python program that takes a list of student names and their corresponding grades as key-value pairs in a dictionary. The program should then prompt the user to enter a student name, and it should output the grade associated with that student. If the student is not found in the dictionary, the program should print a message saying "Student not found.**

**Input:**

Student\_grades = {

‘Alice’: 85,

‘Bob’: 92,

‘Charlie’: 78,

‘David’: 88,

‘Emma’: 95

}

Student\_name = input(“Enter student name: “)

If student\_name in student\_grades:

Print(f”Grade of {student\_name}: {student\_grades[student\_name]}”)

Else:

Print(“Student not found.”)

**Output:**

Enter student name: Alice

Grade of Alice: 85

[Program finished]

**QUESTIONS**

**QUESTION#1**

**Three lists i.e., list\_1, list\_2 and list\_3 are initialized with 10 items each. Write Python code that will print items from the lists using zip method.**

**Output**

list\_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

list\_2 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

list\_3 = ['apple', 'banana', 'cherry', 'date', 'elderberry', 'fig', 'grape', 'honeydew', 'kiwi', 'lemon']

for item\_1, item\_2, item\_3 in zip(list\_1, list\_2, list\_3):

print(item\_1, item\_2, item\_3)

**QUESTION#2**

**What is the difference between append and extend methods of a list?**

**Answer:**

The `append` and `extend` methods in Python are both used to add elements to a list, but they differ in how they add those elements:

1. `append`: This method adds a single element to the end of the list. It takes a single argument, which is the element to be added. If the element is a list, it will be added as a single element, not as individual elements.

Example:

```python

my\_list = [1, 2, 3]

my\_list.append(4)

print(my\_list) # Output: [1, 2, 3, 4]

```

2. `extend`: This method adds all the elements from an iterable (such as a list, tuple, or another iterable) to the end of the list. It takes a single argument, which is the iterable whose elements are to be added to the list.

Example:

```python

my\_list = [1, 2, 3]

another\_list = [4, 5, 6]

my\_list.extend(another\_list)

print(my\_list) # Output: [1, 2, 3, 4, 5, 6]

In summary, `append` adds a single element to the end of the list, while `extend` adds all the elements from an iterable to the end of the list.

**QUESTION#3**

**Predict the output**

grocery\_list = ['flour','cheese','carrots']

for idx,val in enumerate(grocery\_list):

print("%s: %s" % (idx, val)).

**Answer:**

0: flour

1: cheese

2: carrots

**QUESTION#4**

**Consider the code below:**

items=[3,45,66,5,90,101]

items2=[55,77]

items.remove(5)

print(items)

items.extend(items2)

print(items.pop(6))

What will be the output of the above code?

**Output:**

77