**ARDHI UNIVERSITY**

****

**SCHOOL OF EARTH SCIENCE, REAL ESTATE, BUSINESS STUDIES AND INFORMATICS**

**DEPARTMENT OF COMPUTER SYSTEMS AND MATHEMATICS**

**BSC. COMPUTER SYSTEMS AND NETWORKS**

**IS 122; DATA STRACTURE AND ALGORITHM**

**YEAR I 2023/2024**

**PREPARED BY GROUP 10**

**SUBMITTED TO DR NKOTAGU**

|  |  |  |  |
| --- | --- | --- | --- |
| **S\N** | **NAME** | **REG. NUMBER** | **SIGNATURE** |
| **1** | **JACOB LUCAS BULLUGU** | **32285/T.2023** |  |
| **2** | **JULIETH JAPHET RUTAIHWA** | **30784/T.2023** |  |
| **3** | **KABWE KASIM RASHID** | **31432/T.2023** |  |
| **4** | **AUGUSTINO WILLIAM CHAULA** | **31671/T.2023** |  |
| **5** | **HANS RUMISHA KIMAMBO** | **31851/T.2023** |  |

1. Write a Python function that returns the smallest integer in a list. You can assume that the list will always contain integers. The function signature is as follows *def get\_smallest\_integer(my\_list)*

**File name:** 1-smallest\_int.py, **Folder**: 000-programming\_in\_python**, Repository:** data\_structure\_and\_algorithms

Solution:

**Task:** Write a Python function that returns the smallest integer in a list.  
**File name:** 1-smallest\_int.py  
**Folder:** 000-programming\_in\_python  
**Repository:** data\_structure\_and\_algorithms

python

def get\_smallest\_integer(my\_list):

if not my\_list: # check if the list is empty

return None

smallest = my\_list[0]

for num in my\_list:

if num < smallest:

smallest = num

return smallest

if(\_\_name\_\_==\_\_main\_\_):

print(get\_smallest\_integer([3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5])) # Output: 1

2. Write a Python function that searches for the first occurrence of an integer in a list and returns its index. You can assume that the list will always contain integers.

*def find\_first\_occurrence(my\_list, num)*

**File name:** 2-search \_int.py, **Folder**: 000-programming\_in\_python**, Repository:** data\_structure\_and\_algorithms

Solution:

**Task:** Write a Python function that searches for the first occurrence of an integer in a list and returns its index.  
**File name:** 2-search\_int.py  
**Folder:** 000-programming\_in\_python  
**Repository:** data\_structure\_and\_algorithms

python

Copy code

# 2-search\_int.py

def find\_first\_occurrence(my\_list, num):

for index, value in enumerate(my\_list):

if value == num:

return index

return -1 # return -1 if the number is not found

if(\_\_name\_\_==\_\_main\_\_):

print(find\_first\_occurrence([3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5], 5)) # Output: 4

print(find\_first\_occurrence([3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5], 7)) # Output: -1

3. Write a Python function that prints a right-angled triangle with a given height using (\*). For example, if a height of 5 is given the output should be as follows:

**\***

**\*\***

**\*\*\***

**\*\*\*\***

**\*\*\*\*\***

The function signature is as follows *def print\_right\_triangle(height)*

**File name:** 3-print\_triangle.py, **Folder**: 000-programming\_in\_python**, Repository:** data\_structure\_and\_algorithms

Solution:

**Task:** Write a Python function that prints a right-angled triangle with a given height using (\*).  
**File name:** 3-print\_triangle.py  
**Folder:** 000-programming\_in\_python  
**Repository:** data\_structure\_and\_algorithms

python

Copy code

# 3-print\_triangle.py

def print\_right\_triangle(height):

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

print('\*' \* i)

if(\_\_name\_\_==\_\_main\_\_):

print\_right\_triangle(5)

# Output:

# \*

# \*\*

# \*\*\*

# \*\*\*\*

# \*\*\*\*\*

4. Write a Python function that factorizes an input number into its prime factors. The prime factors should be returned as a list of numbers.

The function signature is as follows *def factorize(number)*

**File name:** 4-factorize.py, **Folder**: 000-programming\_in\_python**, Repository:** data\_structure\_and\_algorithms

Solution:

**Task:** Write a Python function that factorizes an input number into its prime factors.  
**File name:** 4-factorize.py  
**Folder:** 000-programming\_in\_python  
**Repository:** data\_structure\_and\_algorithms

python

Copy code

# 4-factorize.py

def factorize(number):

factors = []

# Start with the smallest prime factor

divisor = 2

while number >= divisor:

while number % divisor == 0:

factors.append(divisor)

number //= divisor

divisor += 1

return factors

if(\_\_name\_\_==\_\_main\_\_):

print(factorize(100)) # Output: [2, 2, 5, 5]

print(factorize(37)) # Output: [37]

5. Write a Python function that takes a list of integers as input and returns the sum of all the even numbers in the list. The function signature is as follows: *def sum\_even\_numbers(my\_list)*

**File name:** 5-sum\_even.py, **Folder**: 000-programming\_in\_python**, Repository:** data\_structure\_and\_algorithms

### Solution:

**Task:** Write a Python function that takes a list of integers as input and returns the sum of all the even numbers in the list.  
**File name:** 5-sum\_even.py  
**Folder:** 000-programming\_in\_python  
**Repository:** data\_structure\_and\_algorithms

python

Copy code

# 5-sum\_even.py

def sum\_even\_numbers(my\_list):

return sum(num for num in my\_list if num % 2 == 0)

if(\_\_name\_\_==\_\_main\_\_):

print(sum\_even\_numbers([1, 2, 3, 4, 5, 6])) # Output: 12

6. Write a Python function that takes a string as input and returns a dictionary containing the frequency of each character (case-insensitive) in the string. Ignore non-alphabetic characters. The function signature is as follows: *def character\_frequency(string)*

**File name:** 6-char\_frequency.py, **Folder**: 000-programming\_in\_python**, Repository:** data\_structure\_and\_algorithms

### Solution

### **Task:** Write a Python function that takes a string as input and returns a dictionary containing the frequency of each character (case-insensitive) in the string. Ignore non-alphabetic characters. **File name:** 6-char\_frequency.py **Folder:** 000-programming\_in\_python **Repository:** data\_structure\_and\_algorithms

python

Copy code

# 6-char\_frequency.py

def character\_frequency(string):

frequency = {}

for char in string.lower():

if char.isalpha():

if char in frequency:

frequency[char] += 1

else:

frequency[char] = 1

return frequency

if(\_\_name\_\_==\_\_main\_\_):

print(character\_frequency("Hello, World!")) # Output: {'h': 1, 'e': 1, 'l': 3, 'o': 2, 'w': 1, 'r': 1, 'd': 1}

**7.** Write a Python function that checks whether a given integer is Prime or not. The function should return True is the integer is prime and return False if the integer is not prime. The signature of the function is as follows: def is\_prime(number):

**File name:** 7-is\_prime.py, **Folder**: 000-programming\_in\_python**, Repository:** data\_structure\_and\_algorithms

### solution

**Task:** Write a Python function that checks whether a given integer is prime or not. The function should return True if the integer is prime and False if the integer is not prime.  
**File name:** 7-is\_prime.py  
**Folder:** 000-programming\_in\_python  
**Repository:** data\_structure\_and\_algorithms

python

Copy code

# 7-is\_prime.py

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

if(\_\_name\_\_==\_\_main\_\_):

print(is\_prime(29)) # Output: True

print(is\_prime(15)) # Output: False

8. Write a Python function in which when given a list of integers and an integer target, returns indices of the two numbers such that they add up to target. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order. The function signature is as follows: *def two\_indices(nums, target)*

**Example:**

**Input:** nums = [2, 7, 11, 15], target = 9

**Output:** [0,1]

**Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1].

**File name:** 8-two\_indices.py, **Folder**: 000-programming\_in\_python**, Repository:** data\_structure\_and\_algorithms

### Solution:

**Task:** Write a Python function that, when given a list of integers and an integer target, returns indices of the two numbers such that they add up to the target. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order.  
**File name:** 8-two\_indices.py  
**Folder:** 000-programming\_in\_python  
**Repository:** data\_structure\_and\_algorithms

python

Copy code

# 8-two\_indices.py

def two\_indices(nums, target):

num\_dict = {}

for index, num in enumerate(nums):

complement = target - num

if complement in num\_dict:

return [num\_dict[complement], index]

num\_dict[num] = index

if(\_\_name\_\_==\_\_main\_\_):

print(two\_indices([2, 7, 11, 15], 9)) # Output: [0, 1]

print(two\_indices([3, 2, 4], 6)) # Output: [1, 2]

Question 9:

|  |  |
| --- | --- |
| Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M. **SYMBOL** | **VALUE** |
| I | 1 |
| V | 5 |
| X | 10 |
| L | 50 |
| C | 100 |
| D | 500 |
| M | 1000 |

### Solution:

**Task:** Write a Python function that accepts an integer and converts it to a Roman numeral string. The function should only return the string.  
**File name:** 9-int\_to\_roman.py  
**Folder:** 000-programming\_in\_python  
**Repository:** data\_structure\_and\_algorithms

python

Copy code

# 9-int\_to\_roman.py

def int\_to\_roman(n):

val = [

1000, 900, 500, 400,

100, 90, 50, 40,

10, 9, 5, 4,

1

]

syms = [

"M", "CM", "D", "CD",

"C", "XC", "L", "XL",

"X", "IX", "V", "IV",

"I"

]

roman\_numeral = ""

for i in range(len(val)):

count = n // val[i]

roman\_numeral += syms[i] \* count

n -= val[i] \* count

return roman\_numeral

if(\_\_name\_\_==\_\_main\_\_):

print(int\_to\_roman(27)) # Output: XXVII

print(int\_to\_roman(1994)) # Output: MCMXCIV