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 18

SUBMITTED TO DR MICHAEL NKOTAGU

S/N	STUDENT NAME	REGISTRATION NUMBER	SIGNATURE
1.	HERMAN PAULO	31863/T.2023	
2.	EZEKIEL KIPONYA	32242/T.2023	
3.	SHAFII HAMISI	30999/T.2023	
4.	FRANCIS LAITON	32279/T.2023	
5.	TUMAINI MBUNDA	31003/T.2023	

Question 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
```

Question 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
```

Question 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:
# *
# **
# ***
# ****
# *****
```

Question 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]
```

Question 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
```

Question 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}
```

Question 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
```

Question 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	**VALUE**

seven different
symbols: I, V,
X, L, C, D and
M. **SYMBOL**

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

```python
# 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
```