MENEDŻERSKA AKADEMIA NAUK STOSOWANYCH W WARSZAWIE

51 DPH COMPUTER ENGINEERING

PREPARED BY

ENFAL SEVİNÇ, 77789

PROGRAMMING IN SCRIPTING LANGUAGES

SUPERVISOR

KUMAR NALINAKSH

2023 POLAND, EU

Table of Content

Question 01

Write a program that creates a list of numbers and uses list comprehension to create a new list that contains only the even numbers from the original list.

Solution 01

```python
#1
numbers = [12, 22, 33, 54, 65, 76, 87, 98, 49, 10]
even_numbers = [num for num in numbers if num % 2 == 0]
print("Original list:", numbers)
print("Even numbers:", even_numbers)
```

```
"C:\Users\enfal\OneDrive\Masaüstü\laboratory homework3\Scripts\python.exe" "C:\Users\enfal\PycharmProjects\laboratory homework3\main.py"

Original list: [12, 22, 33, 54, 65, 76, 87, 98, 49, 10]

Even numbers: [12, 22, 54, 76, 98, 10]
```

Question 02

Write a program that creates a dictionary of book titles and their authors andprints the titles in alphabetical order

Solution 02

```
#2
books = {
    "The Great Gatsby": "F. Scott Fitzgerald",
    "Les Misérables": "Victor Hugo",
    "Crime Anda Punishment": "Fyodor Dostoyevski",
    "İnce Memed": "Yaşar Kemal",
    "The First Man": "Albert Camus "
        }
titles = sorted(books.keys())
for title in titles:
    print(title)
```

```
Crime Anda Punishment
Les Misérables
The First Man
The Great Gatsby
İnce Memed
```

Question 03

Write a program that uses a stack to reverse the words in a sentence

Solution 03

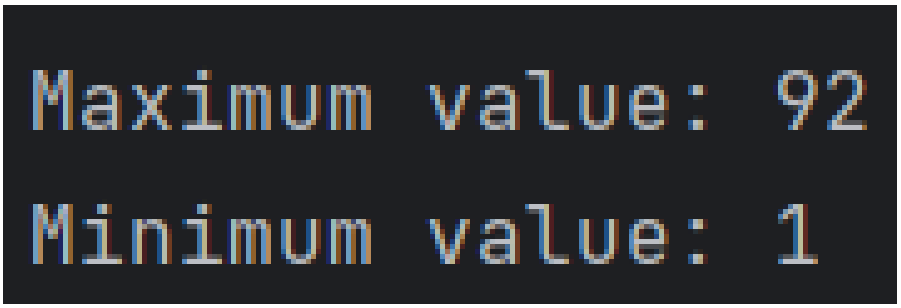
```
#3
def reverse_sentence(sentence):
    words = sentence.split()
    stack = []
    for word in words:
        stack.append(word)

    reversed_words = []
    while stack:
        reversed_words.append(stack.pop())
    reversed_sentence = " ".join(reversed_words)

    return reversed_sentence
sentence = "Hello world"
reversed_sentence = reverse_sentence(sentence)
print(reversed_sentence)
```

Question 04

Write a program that creates a set of numbers and prints the maximum and minimum values in the set

Solution 04

```
#4
numbers = {36, 66, 92, 19, 1, 45, 78}
max_number = max(numbers)
min_number = min(numbers)
print("Maximum value:", max_number)
print("Minimum value:", min_number)
```

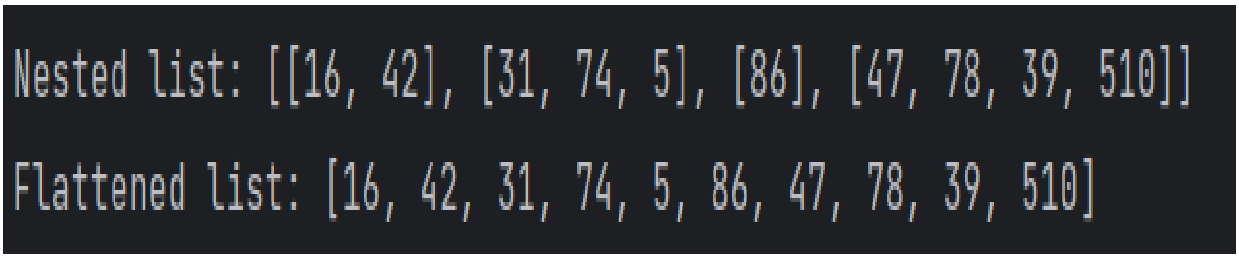


Question 05

Write a program that creates a nested list of numbers and uses nested list comprehension to create a flattened list.

Solution 05

```
#5
nested_list = [[16, 42], [31, 74, 5], [86], [47, 78, 39, 510]]
flattened_list = [num for sublist in nested_list for num in sublist]
print("Nested list:", nested_list)
print("Flattened list:", flattened_list)
```



Question 06

Write a program that uses a deque to implement a queue.

Solution 06

```python
#6
from collections import deque

class Queue:
    def __init__(self):
        self.items = deque()

    def is_empty(self):
        return len(self.items) == 0

    def enqueue(self, item):
        self.items.append(item)

    def dequeue(self):
        if self.is_empty():
            raise Exception("Queue is empty")
        return self.items.popleft()

    def size(self):
        return len(self.items)

q = Queue()
q.enqueue(1)
q.enqueue(2)
q.enqueue(3)
print(q.size())
print(q.dequeue())
print(q.dequeue())
print(q.size())
q.enqueue(4)
print(q.dequeue())
print(q.dequeue())
print(q.is_empty())
```

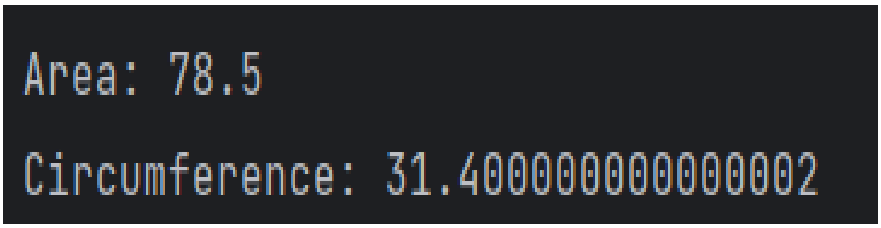
```
3
1
2
1
3
4
True
```

Question 07

Write a program that creates a class to representa circle, with methods to calculate the area and circumference of the circle.

Solution 07

```python
#7
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def calculate_area(self):
        return 3.14 * (self.radius ** 2)

    def calculate_circumference(self):
        return 2 * 3.14 * self.radius
circle = Circle(5)
print("Area:", circle.calculate_area())
print("Circumference:", circle.calculate_circumference())
```



Question 08

Write a program that creates a class to representa bank account, with methods to deposit and withdraw money, and to display the current balance.

Solution 08

```python
#8
class BankAccount:
    def __init__(self, initial_balance=0):
        self.balance = initial_balance
    def deposit(self, amount):
        self.balance += amount
    def withdraw(self, amount):
        if self.balance >= amount:
            self.balance -= amount
        else:
            print("Insufficient balance")
    def display_balance(self):
        print("Current balance:", self.balance)
my_account = BankAccount(2000)
my_account . deposit(500)
my_account . display_balance()
my_account . withdraw(500)
my_account . display_balance()
my_account . withdraw(1500)
```

```
Current balance: 2500
Current balance: 2000
```

Question 09

Write a program that creates a class to represent a student, with methods to calculate their GPA and to add or remove courses from their schedule

Solution 09

```python
#9
class Student:
    def __init__(self, name):
        self.name = name
        self.courses = {}
    def add_course(self, course, grade):
        self.courses[course] = grade
    def remove_course(self, course):
        if course in self.courses:
            del self.courses[course]
        else:
            print(f"{course} is not in {self.name}'s schedule")
    def calculate_gpa(self):
        if not self.courses:
            return None
        total_grade_points = sum(self.courses.values())
        num_courses = len(self.courses)
        return total_grade_points / num_courses

student = Student("Alice")
student.add_course("Math", 3.8)
student.add_course("English", 4.0)
print(student.calculate_gpa())
student.remove_course("Chemistry")
student.remove_course("Math")
print(student.calculate_gpa())
```

```
3.9
Chemistry is not in Alice's schedule
4.0
```

Question 10

Write a program that creates a class to represent a car, with methods to start the car, accelerate, and brake

Solution 10

```
#10
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
        self.speed = 0
        self.is_running = False

    def start(self):
        self.is_running = True
        print(f"{self.make} {self.model} {self.year} is now
running.")

    def accelerate(self, mph):
        if not self.is_running:
            print(f"{self.make} {self.model} {self.year} is not
running.")
        else:
            self.speed += mph
            print(f"{self.make} {self.model} {self.year} accelerated
to {self.speed} mph.")

    def brake(self, mph):
        if not self.is_running:
            print(f"{self.make} {self.model} {self.year} is not
running.")
        else:
            self.speed -= mph
            if self.speed < 0:
                self.speed = 0
            print(f"{self.make} {self.model} {self.year} slowed down
to {self.speed} mph.")

car = Car("BMW", "X5M", 2022)
car.start()
car.accelerate(30)
car.accelerate(20)
car.brake(10)
car.brake(60)
```

```
BMW X5M 2022 is now running.
BMW X5M 2022 accelerated to 30 mph.
BMW X5M 2022 accelerated to 50 mph.
BMW X5M 2022 slowed down to 40 mph.
BMW X5M 2022 slowed down to 0 mph.
```

Question 11

Write a program that creates a class to represent a game of tic-tac-toe, with methods to display the board, make a move, and determine the winner.

Solution 11

```
#11
class TicTacToe:
    def __init__(self):
        self.board = [[' ']*3 for _ in range(3)]
        self.current_player = 'X'

    def display_board(self):
        print('  0 1 2')
        for row in range(3):
            print(f"{row} {'|'.join(self.board[row])}")
            if row < 2:
                print('  -----')

    def make_move(self, row, col):
        if self.board[row][col] == ' ':
            self.board[row][col] = self.current_player
            self.current_player = 'O' if self.current_player == 'X'
else 'X'
        else:
            print('Invalid move: position is already on the table.')

    def check_winner(self):
        for i in range(3):
            if self.board[i][0] == self.board[i][1] ==
self.board[i][2] != ' ':
                return self.board[i][0]
            if self.board[0][i] == self.board[1][i] ==
self.board[2][i] != ' ':
                return self.board[0][i]
        if self.board[0][0] == self.board[1][1] == self.board[2][2]
!= ' ':
            return self.board[0][0]
        if self.board[0][2] == self.board[1][1] == self.board[2][0]
!= ' ':
            return self.board[0][2]
        if all(self.board[i][j] != ' ' for i in range(3) for j in
range(3)):
            return 'Tie'
        return None

game = TicTacToe()
game.make_move(0, 0)
game.make_move(1, 1)
game.make_move(0, 1)
game.make_move(1, 0)
game.make_move(2, 2)
game.display_board()
winner = game.check_winner()
```

```python
if winner:
    print(f"{winner} wins!")
else:
    print("It's a tie!")
```

```
   0 1 2
 0 X|X|
   -----
 1 O|O|
   -----
 2  | |X
 It's a tie!
```

Question 12

Write a program that uses inheritance to create a subclass of the circle class that represents a sphere, with methods to calculate the volume and surface area of the sphere

Solution 12

```python
#12
import math
class Circle:
    def __init__(self, radius):
        self.radius = radius
    def area(self):
        return math.pi * self.radius ** 2
    def circumference(self):
        return 2 * math.pi * self.radius
class Sphere(Circle):
    def volume(self):
        return 4/3 * math.pi * self.radius ** 3
    def surface_area(self):
        return 4 * math.pi * self.radius ** 2
c = Circle(3)
print(c.area())
print(c.circumference())

s = Sphere(3)
print(s.volume())
print(s.surface_area())
```

```
28.274333882308138
18.84955592153876
113.09733552923254
113.09733552923255


Process finished with exit code 0
```