**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 3](#_bookmark0)

[Solution 01 3](#_bookmark1)

[Question 02 4](#_bookmark2)

[Solution 02 5](#_bookmark3)

[Question 03 6](#_bookmark4)

[Solution 03 6](#_bookmark5)

[Question 04 7](#_bookmark6)

[Solution 04 7](#_bookmark7)

[Question 05 8](#_bookmark8)

[Solution 05 8](#_bookmark9)

[Question 06 9](#_bookmark10)

[Solution 06 9](#_bookmark11)

[Question 07 10](#_bookmark10)

[Solution 07 10](#_bookmark10)

[Question 08 11](#_bookmark10)

[Solution 08](#_bookmark10) 11

[Question 09 12](#_bookmark10)

[Solution 09](#_bookmark10) 12

[Question 10 13](#_bookmark10)

[Solution 10 14](#_bookmark10)

[Question 11 15](#_bookmark10)

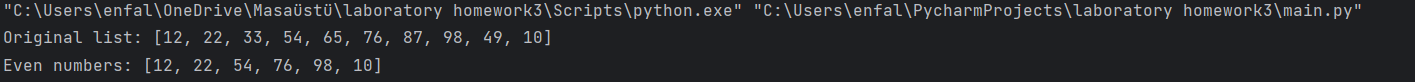
[Solution 11 15/16](#_bookmark10)

[Solution 12 17](#_bookmark10)

## 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

#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)



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

#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)

metin, yazı tipi, ekran görüntüsü, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

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)



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)

metin, yazı tipi, ekran görüntüsü, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

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

#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())

ekran görüntüsü, yazı tipi, metin, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 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

#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())

## metin, yazı tipi, ekran görüntüsü, grafik içeren bir resim Açıklama otomatik olarak oluşturuldu

## 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

#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)

metin, yazı tipi, ekran görüntüsü, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 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

#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())

metin, yazı tipi, ekran görüntüsü, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 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)

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

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

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

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

#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())

## metin, yazı tipi, ekran görüntüsü içeren bir resim Açıklama otomatik olarak oluşturuldu