```
FUNCTION add_numbers
    LOOP forever
        TRY
            PROMPT user: "Enter two or more numbers separated by spaces"
            numbers input("egevbhnjeytcftvb).split()
            READ input and SPLIT by spaces into list 'numbers'

            IF length of 'numbers' < 2 THEN
                RAISE error "Please enter at least two numbers."

            CONVERT each element in 'numbers' to float
            CALCULATE total = sum of 'numbers'

            PRINT "Sum:", total
            BREAK loop   // Exit since valid input is given

        CATCH ValueError AS ve
            PRINT "Error:", ve

        CATCH any other Exception
            PRINT "Error: Make sure you are entering only numbers."
    END LOOP
END FUNCTION
```

```
import os

FUNCTION is_valid_sentence(sentence)
    RETURN True if sentence contains at least one alphabetic character
    ELSE False
END FUNCTION


FUNCTION punishment
    SET attempts = 0

    WHILE attempts < 3
        PROMPT user: "Enter the sentence"
        READ sentence

        IF is_valid_sentence(sentence) IS True
            BREAK loop   // valid sentence found
        ELSE
```

```
            PRINT "Don't be dumb. Enter a proper sentence."
            INCREMENT attempts by 1
    END WHILE

    IF attempts == 3
        PRINT "Too many incorrect attempts. Exiting program."
        RETURN   // exit function

    PROMPT user: "Enter the number of times to repeat the sentence"
    READ times as integer

    SET file_path = "CompletedPunishment.txt"

    TRY
        OPEN file at file_path in write mode
        FOR i FROM 1 TO times
            WRITE sentence followed by newline into file
        CLOSE file
        PRINT sentence + " has been written " + times + " times to " +
file_path
    CATCH IOError AS e
        PRINT "An error occurred while writing to the file:", e
END FUNCTION


CALL punishment
```

```
FUNCTION word_count
    PROMPT user: "Enter a word to search"
    READ input word and CONVERT to lowercase

    SET file_name = "PythonSummary.txt"

    IF file_name does not exist in current directory
        PRINT "Error: The file 'PythonSummary.txt' was not found in the
current directory."
        RETURN   // exit function

    TRY
        with open(file_nname, "r") as file:
        content = file.read().lower()
        #Remove punctuation
        content = content.translate(str.maketrans("","",sring.punctuation))
```

```
        word_count = content.split().count(word)
        CLOSE file

        SPLIT content into list of words
        CALCULATE word_count = number of times input word appears in list
        PRINT "The word '<word>' occurs <word_count> times."

    CATCH any Exception as e
        PRINT "An error occurred while reading the file:", e
END FUNCTION


CALL word_count
```

```
class Course:

    def __init__(

        self, department, number, name, credits, days, start_time, end_time,
avg_grade

    ):

        self.department = department



    def format(self):

        return (

            f"COURSE: {self.department}{self.number}: {self.name}\n"

            f"Number of Credits: {self.credits}\n"

            f"Days of Lectures: {self.days}\n"

            f"Lecture Time: {self.start_time} - {self.end_time}\n"

            f"Stat: on average, students get {self.avg_grade}% in this
course\n"

        )
```

```
def class_schedule():
    with open("classesInput.txt", "r") as file:
        lines = file.readlines()

    num_courses = int(lines[0].strip())
    courses = []

    index = 1
    for i in range(num_courses):
        department = lines[index].strip()
        number = lines[index + 1].strip()
        index += 8

        course = Course(
            department, number, name, credits, days, start_time, end_time,
avg_grade
        )
        courses.append(course)

    with open("formatted_schedule.txt", "w") as file:
        for course in courses:
            file.write(course.format() + "\n")
```

```
class_schedule()
```

```
FUNCTION load_grades
    IF file "grades.txt" exists
        OPEN file in read mode
        READ JSON data into dictionary 'grades'
        RETURN grades
    ELSE
        RETURN empty dictionary
END FUNCTION


FUNCTION save_grades(grades)
    OPEN "grades.txt" in write mode
    WRITE grades dictionary as JSON into file
END FUNCTION


FUNCTION create_grade(grades)
    PROMPT user: "Enter student's full name"
    READ name
    PROMPT user: "Enter the grade"
    READ grade
    SET grades[name] = grade
    CALL save_grades(grades)
    PRINT "Grade for <name> added."
END FUNCTION


FUNCTION get_grade(grades)
    PROMPT user: "Enter student's full name to get the grade"
    READ name
    IF name exists in grades
        PRINT "<name>'s grade: <grade>"
    ELSE
        PRINT "Student not found."
END FUNCTION


FUNCTION edit_grade(grades)
    PROMPT user: "Enter student's full name to edit the grade"
    READ name
```

```
    IF name exists in grades
        PROMPT user: "Enter the new grade for <name>"
        READ new_grade
        UPDATE grades[name] = new_grade
        CALL save_grades(grades)
        PRINT "<name>'s grade updated."
    ELSE
        PRINT "Student not found."
END FUNCTION


FUNCTION delete_grade(grades)
    PROMPT user: "Enter student's full name to delete the grade"
    READ name
    IF name exists in grades
        DELETE grades[name]
        CALL save_grades(grades)
        PRINT "<name>'s grade deleted."
    ELSE
        PRINT "Student not found."
END FUNCTION


FUNCTION grade_program
    SET grades = load_grades()

    LOOP forever
        DISPLAY menu:
            1. Add a grade
            2. Get a grade
            3. Edit a grade
            4. Delete a grade
            5. Exit

        PROMPT user: "Choose an option"
        READ choice

        IF choice == "1" THEN
            CALL create_grade(grades)
        ELSE IF choice == "2" THEN
            CALL get_grade(grades)
        ELSE IF choice == "3" THEN
            CALL edit_grade(grades)
        ELSE IF choice == "4" THEN
            CALL delete_grade(grades)
        ELSE IF choice == "5" THEN
```

```
            BREAK loop
        ELSE
            PRINT "Invalid option."
    END LOOP
END FUNCTION


CALL grade_program
```