МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКО”

Факультет прикладної математики

Кафедра системного програмування і спеціальних комп’ютерних систем

Лабораторна робота №1

З дисципліни «Організація баз даних»

«Ознайомлення з базовими конструкціями мови Python. Спрощена база даних»

Виконав:

студент ІІІ-го курсу

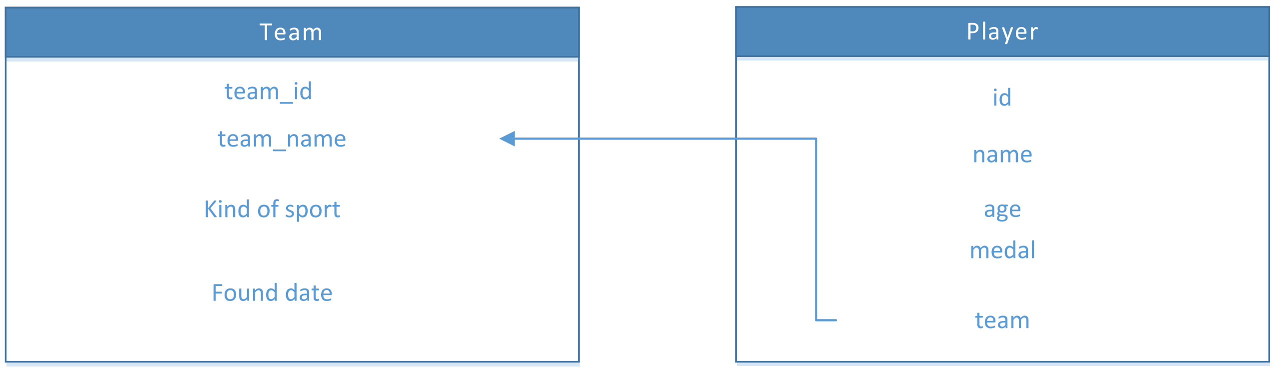
групи КВ-41

Горпинич-Радуженко Іван

Київ 2016

Варіант:

5.	Команда-Спортсмен	Вивести найкращих спортсменів кожної команди



Текст програми:

class Main:

```
    def __init__(self, fileProduct, fileOrder):
        self.fileProduct = fileProduct
        self.fileOrder = fileOrder

    def main(self):
        ui = GUI.GUI()
        main = Engine.Engine(self.fileProduct, self.fileOrder)
        choice = ui.menu()
        while choice != 7:
            if choice == 1:
                self.show_database(ui, main)
            elif choice == 2:
                self.show_table(ui, main)
            elif choice == 3:
                self.insert(ui, main)
            elif choice == 4:
                self.delete(ui, main)
            elif choice == 5:
                self.update(ui, main)
            elif choice == 6:
                self.select(ui, main)
            choice = ui.menu()
        main.pack()
        sys.exit(0)

    def show_database(self, ui, main):
        ui.show_table_athlete('athlete', main.get_athlete())
    ui.show_table_team('team', main.get_team())

def show_table(self, ui, main):
    table = self.what_table(ui)
    if table == 3:
        return
    if table == 1:
        ui.show_table_athlete('athlete', main.get_athlete())
    else:
        ui.show_table_team('team', main.get_team())

def insert(self, ui, main):
    table = self.what_table(ui)
    if table == 3:
        return
    if table == 1:
        self.insert_into_athlete(ui, main)
    else:
        self.insert_into_team(ui, main)

def delete(self, ui, main):
    table = self.what_table(ui)
    if table == 3:
        return
    if table == 1:
        self.delete_from_athlete(ui, main)
    else:
        self.delete_from_team(ui, main)

def update(self, ui, main):
    table = self.what_table(ui)
    if table == 3:
        return
    if table == 1:
        self.update_athlete(ui, main)
    else:
        self.update_team(ui, main)

def select(self, ui, main):
    main.select_variant()

def insert_into_athlete(self, ui, main):
    info = ui.insert_athlete_info()
    if not info:
        return
    if not (info[0] and info[1]and info[2]and info[3]):
        ui.error('Invalid input')
        return
    is_error = main.insert_into_athlete(info[0], info[1], info[2], info[3])
    ui.is_successful(is_error)
```

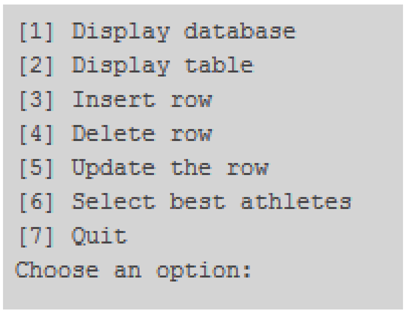
```
def insert_into_team(self, ui, main):
    info = ui.insert_team_info()
    if not info:
        return
    if not (info[0] and info[1] and info[2]):
        ui.error('Invalid input')
        return
    is_error = main. insert_into_team(info[0], info[1], info[2].strftime("%d/%m/%y"))
    ui.is_successful(is_error)

def delete_from_athlete(self, ui, main):
    info = ui.delete_athlete_info()
    if info:
        is_error = main.delete_from_athlete(info[0])
        ui.is_successful(is_error)

def delete_from_team(self, ui, main):
    info = ui.delete_team_info()
    if info:
        is_error = main.delete_from_team(info[0], info[1])
        ui.is_successful(is_error)

def update_athlete(self, ui, main):
    old_info = ui.update__info_athlete()
    if not old_info:
        ui.error('Invalid input')
        return
    existing_athlete = main.athlete_to_update(old_info[0])
    if type(existing_athlete) == str:
        ui.error(existing_athlete)
        return
    new_info = ui.update_new_info_athlete()
    if new_info:
        is_error = main.update_athlete(old_info[0], new_info[0], new_info[1], new_info[2], new_info[3])
        ui.is_successful(is_error)

def update_team(self, ui, main):
    old_info = ui.update__info_team()
    if not (old_info and old_info[0] and old_info[1]):
        ui.error('Invalid input')
        return
    existing_team = main.team_to_change(old_info[0], old_info[1])
    if type(existing_team) == str:
        ui.error(existing_team)
        return
    new_info = ui.update_new_info_team()
    if new_info:
        is_error = main.update_team(old_info[0], old_info[1], new_info[0], new_info[1], new_info[2])
        ui.is_successful(is_error)
    def what_table(self, ui):
        table = ui.what_table()
        while not table:
            table = ui.what_table()
        return table

if __name__ == '__main__':
    c = Main('athlete.txt', 'team.txt')
    c.main()
```

```
import pickle
```

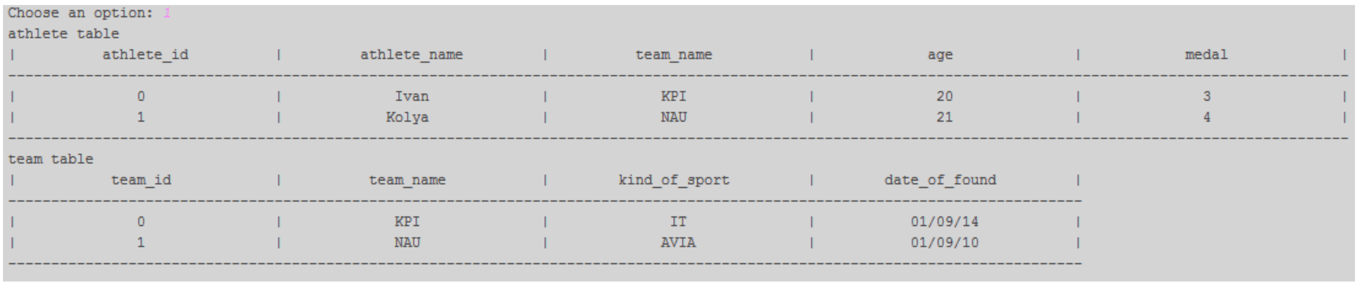
class Engine:

```
    def __init__(self, fileteam, fileAthlete):
        try:
            self.fileteam = fileteam
            self.fileAthlete = fileAthlete
            DB = open(fileteam, 'rb')
            self.athlete = pickle.load(DB)
            DB.close()
            DB = open(fileAthlete, 'rb')
            self.team = pickle.load(DB)
            DB.close()
        except:
            self.athlete = list()
            self.team = list()

    def get_athlete(self):
        return self.athlete

    def get_team(self):
        return self.team

    def insert_into_athlete(self, athlete_name, age, medal, team):
        if not self.team_name_in_table(team):
            return 'No such team'
        if not self.athlete:
            athlete_id = 0
        else:
            athlete_id = len(self.athlete)
        if self.athlete_name_in_table(athlete_name):
            return 'Such athlete already exists'
        self.athlete.append({'athlete_id': athlete_id, 'athlete_name': athlete_name, 'age': age, 'medal': medal,
'team_name': team})

    def insert_into_team(self, team,kind_of_sport, found_date):

        if self.team_name_in_table(team):
            return 'Such team already exists'
        if not self.team:
            team_id = 0
        else:
            team_id = len(self.team)
        self.team.append({'team_id': team_id, 'team_name': team, 'kind_of_sport': kind_of_sport,
'date_of_found': found_date})

    def delete_from_athlete (self, athlete_id):
        existing_athlete = self.athlete_id_in_table(athlete_id)
        if not existing_athlete:
            return 'No such athlete'
        self.athlete.remove(existing_athlete[0])

    def delete_from_team (self, team, kind_of_sport):
        if filter(lambda x: x['team_name'] == team, self.athlete):
            return 'Cannot delete an team'
        team = self.team_to_change(team, kind_of_sport)
        '''
        if type(team) == str:
            return team
            '''
        self.team.remove(team)

    def athlete_to_update(self, athlete_id):
        existing_athlete = self.athlete_id_in_table(athlete_id)
        if not existing_athlete:
            return 'No such athlete'

    def update_athlete(self, athlete_id, new_athlete_name, new_age, new_medal, new_team):

        existing_athlete = self.athlete_id_in_table(athlete_id)
        if new_athlete_name:
            existing_athlete[0]['athlete_name'] = new_athlete_name
        if new_age:
            existing_athlete[0]['age'] = new_age
        if new_medal:
            existing_athlete[0]['medal'] = new_medal
        if new_team:
            existing_athlete[0]['team_name'] = new_team

    def team_to_change(self, team, kind_of_sport):
        existing_team = self.team_name_in_table(team)
        if not existing_team:
            return 'No such team'
        if filter(lambda x: x['team_name'] == team, self.athlete):
            return 'Cannot update an team'
        return existing_team[0]

    def update_team(self, team, kind_of_sport, new_team, new_kind, new_found):
        if new_team and new_kind and self.team_in_table(new_team, new_kind):
            return 'Such team already exists'
        existing_team = self.team_in_table(team, kind_of_sport)
        if new_team:
            existing_team[0]['team_name'] = new_team
        if new_kind:
          existing_team[0]['kind_of_sport'] = new_kind
        if new_found:
            existing_team[0]['date_of_found'] = new_found.strftime("%d/%m/%y")

    def select_variant(self):
        for team in self.team:
            print "team: ", team['team_name']
            team_players = filter(lambda x: x['team_name'] == team['team_name'], self.athlete)
            max_medals, best_athlete = 0, None
            for athlete in team_players:
                if athlete['medal'] > max_medals:
                    max_medals = athlete['medal']
                    best_athlete = athlete
            if best_athlete is not None:
                print "Best athlete: ", best_athlete['athlete_name'], best_athlete['medal']
                print '-'*30

    def pack(self):
        DB = open(self.fileteam, 'wb')
        pickle.dump(self.athlete, DB)
        DB.close()
        DB = open(self.fileAthlete, 'wb')
        pickle.dump(self.team, DB)
        DB.close()

    def team_name_in_table(self, team_name):
        return filter(lambda x: x['team_name'] == team_name, self.team)

    def athlete_name_in_table(self, athlete_name):
        return filter(lambda x: x['athlete_name'] == athlete_name, self.athlete)

    def athlete_id_in_table(self, athlete_id):
        return filter(lambda x: x['athlete_id'] == athlete_id, self.athlete)

    def team_in_table(self,team, kind_of_sport):
        return filter(lambda x: x['team_name'] == team and x['kind_of_sport'] == kind_of_sport, self.team)

class GUI:
    def menu(self):
        print '\n[1] Display database'
        print '[2] Display table'
        print '[3] Insert row'
        print '[4] Delete row'
        print '[5] Update the row'
        print '[6] Select best athletes'
        print '[7] Quit'
        try:
            selection = int(raw_input('Choose an option: '))
            if not 1 <= selection <= 7:
                raise ValueError
            return selection
        except ValueError:
            self.error('Invalid input')
            return None

    def show_table_team(self, table_name, table):
        print '{:^10}'.format(table_name + ' table')
        if not table:
            print '{:^10}'.format('empty')
        else:
            columns = table[0].keys()
            print '|{:^30}|{:^30}|{:^30}|{:^30}|'.format(columns[1] ,columns[3], columns[0], columns[2])
            print '-' * 125
            for row in table:
                print '|{:^30}|{:^30}|{:^30}|{:^30}|'.format(row[columns[1]], row[columns[3]], row[columns[0]],
row[columns[2]])
            print '-' * 125
    def show_table_athlete(self, table_name, table):
        print '{:^10}'.format(table_name + ' table')
        if not table:
            print '{:^10}'.format('empty')
        else:
            columns = table[0].keys()
            print '|{:^30}|{:^30}|{:^30}|{:^30}|{:^30}|'.format(columns[2], columns[4], columns[3],
columns[0],columns[1])
            print '-' * 156
            for row in table:
                print '|{:^30}|{:^30}|{:^30}|{:^30}|{:^30}|'.format(row[columns[2]], row[columns[4]],
row[columns[3]], row[columns[0]], row[columns[1]])
            print '-' * 156

    def delete_athlete_info(self):
        row = list()
        print '\nDeleting athlete'
        athlete_id = int(raw_input("Enter athlete_id: "))
        row.append(athlete_id)
        return row

    def delete_team_info(self):
        row = list()
        print '\nDeleting team'
        try:
            row.append(raw_input("Enter team_name: "))
            row.append(raw_input("Enter kind_of_sport: "))
            return row
        except ValueError:
            self.error('Invalid input')
            return None

    def insert_athlete_info(self):
        row = list()
        print '\nInserting athlete'
        try:
            row.append(raw_input("Enter athlete_name: "))
            row.append(raw_input('Enter age: '))
            row.append(raw_input('Enter medal: '))
            row.append(raw_input('Enter team: '))
            return row
        except ValueError:
            self.error('Invalid input')
            return None

    def insert_team_info(self):
        row = list()
        print '\nInserting team'
        try:
            row.append(raw_input("Enter team_name: "))
            row.append(raw_input("Enter kind of sport: "))
            date_str = raw_input("Enter found date (dd/mm/yy): ")
            if not date_str:
                raise ValueError
            row.append(datetime.datetime.strptime(date_str, "%d/%m/%y").date())
            return row
        except ValueError:
            self.error('Invalid input')
            return None

    def update__info_athlete(self):
        row = list()
        print '\nUpdating athlete'
        row.append(int(raw_input("Enter athlete_id to update: ")))
        return row

    def update__info_team(self):
        row = list()
        print '\nUpdating team'
        try:
            row.append(raw_input("Enter team_name to update: "))
            row.append(raw_input("Enter kind of sport to update: "))
            return row
        except ValueError:
            self.error('Invalid input')
            return None

    def update_new_info_athlete(self):
        row = list()
        try:
            row.append(raw_input("Enter new athlete_name (press Enter if you don't want to update this
attribute): "))
            row.append(int(raw_input("Enter new age (press '0' if you don't want to update this attribute): ")))
            row.append(int(raw_input("Enter new medal (press '0' if you don't want to update this attribute):
")))
            row.append(raw_input("Enter new team (press Enter if you don't want to update this attribute): "))
            return row
        except ValueError:
            self.error('Invalid input')
            return None

    def update_new_info_team(self):
        row = list()
        try:
            row.append(raw_input("Enter new team_name (press Enter if you don't want to update this
attribute): "))
            row.append(raw_input("Enter new kind_of_sport (press Enter if you don't want to update this
attribute): "))
            date_str = raw_input(
                "Enter new found date (dd/mm/yy) (press Enter if you don't want to update this attribute): ")
            if not date_str:
                row.append(date_str)
            else:
                row.append(datetime.datetime.strptime(date_str, "%d/%m/%y").date())
            return row
        except ValueError:
            self.error('Invalid input')
            return None

    def is_successful(self, error_message):
        if not error_message:
            print '\nSuccess'
        else:
            self.error(error_message)

    def what_table(self):
        print '\nChoose the table: '
        print '[1] athlete'
        print '[2] team'
        print '[3] Back to menu'
        try:
            selection = int(raw_input('Choose an option: '))
            if not 1 <= selection <= 3:
                raise ValueError
            return selection
        except ValueError:
            self.error('Invalid input')
            return None
    def error(self, message):
        print '\n'+message
```

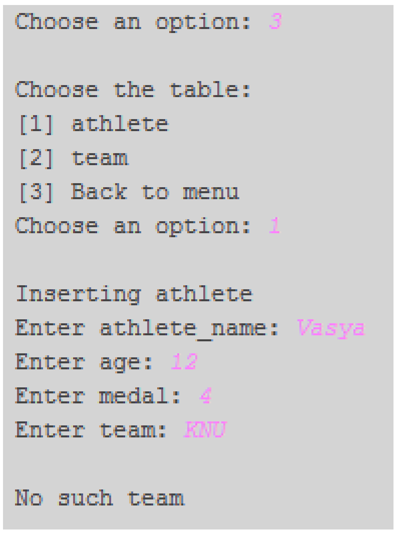
Скріншоти:



```
[1] Display database
[2] Display table
[3] Insert row
[4] Delete row
[5] Update the row
[6] Select best athletes
[7] Quit
Choose an option:
```



```
Choose an option: 1
athlete table
|      athlete_id      |     athlete_name     |      team_name      |         age         |        medal        |
------------------------------------------------------------------------------------------------------------------
|          0           |         Ivan         |         KPI         |         20          |          3          |
|          1           |        Kolya         |         NAU         |         21          |          4          |
------------------------------------------------------------------------------------------------------------------
team table
|        team_id       |       team_name      |    kind_of_sport    |    date_of_found    |
-------------------------------------------------------------------------------------------
|          0           |         KPI          |          IT         |      01/09/14       |
|          1           |         NAU          |         AVIA        |      01/09/10       |
-------------------------------------------------------------------------------------------
```



```
Choose an option: 3

Choose the table:
[1] athlete
[2] team
[3] Back to menu
Choose an option: 1

Inserting athlete
Enter athlete_name: Vasya
Enter age: 12
Enter medal: 4
Enter team: KNU

No such team
```