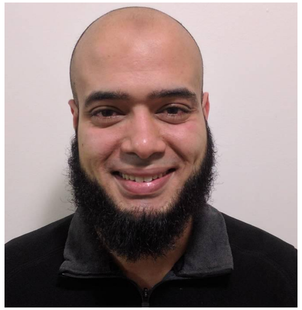
Python Programming
Classes Homework 3

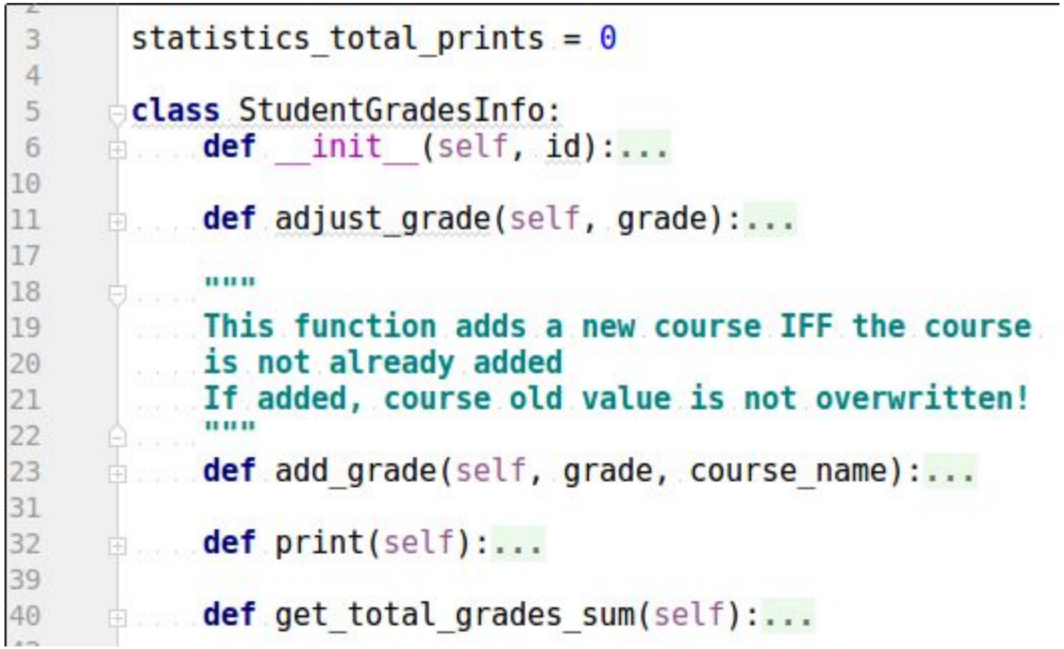
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Problem #1: Students Grades - Code Review

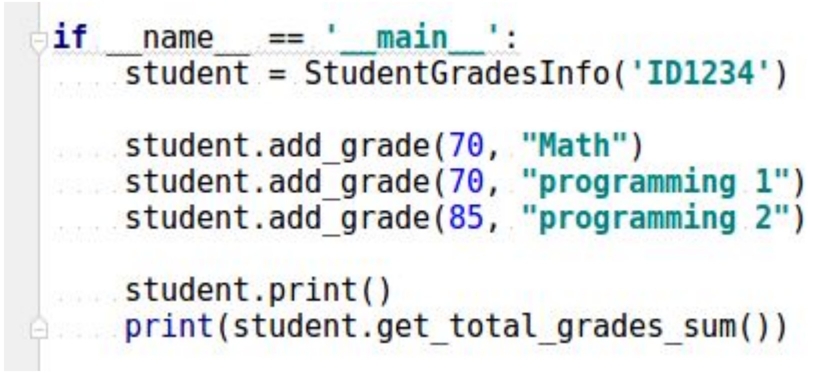
- **Requirements:**
- Class for a student and his grades per course
- Add grade
  - Don’t update if exists
- Grade max is 100
  - e.g. 76.5/100
- Printing functionality
  - Track # of calls



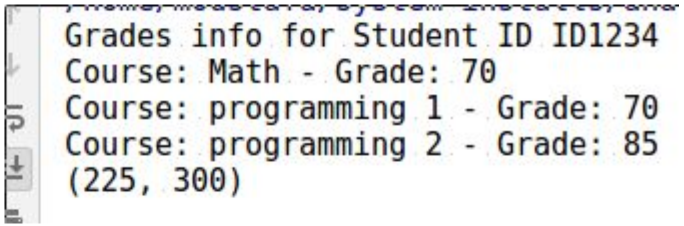
```
statistics_total_prints = 0

class StudentGradesInfo:
    def __init__(self, id):...

    def adjust_grade(self, grade):...

    """
    This function adds a new course IFF the course
    is not already added
    If added, course old value is not overwritten!
    """
    def add_grade(self, grade, course_name):...

    def print(self):...

    def get_total_grades_sum(self):...
```

Problem #1: Students Grades - Code Review

- The main usage

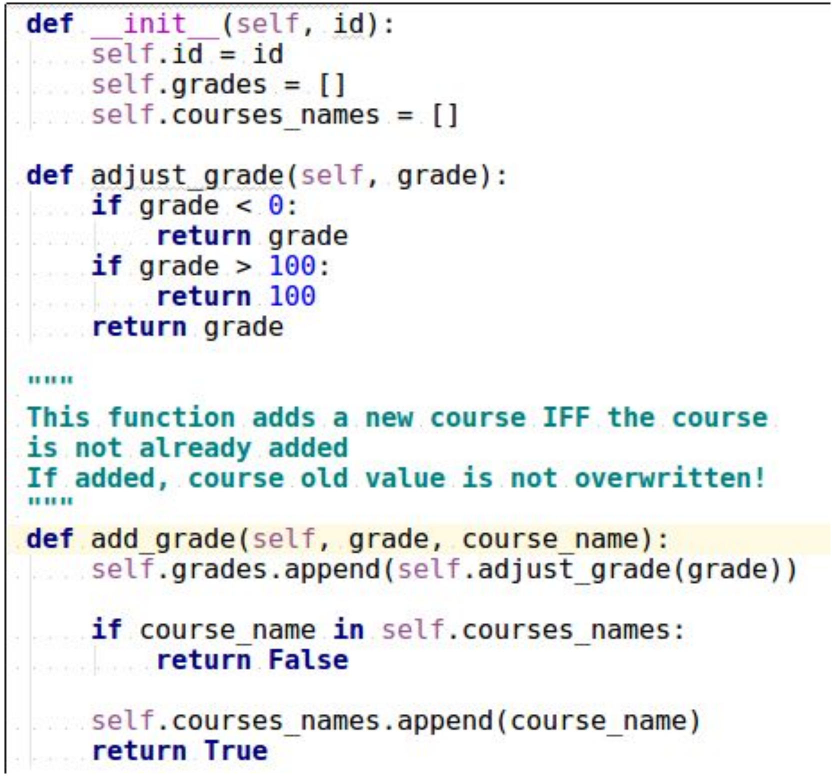


```python
if __name__ == '__main__':
    student = StudentGradesInfo('ID1234')

    student.add_grade(70, "Math")
    student.add_grade(70, "programming 1")
    student.add_grade(85, "programming 2")

    student.print()
    print(student.get_total_grades_sum())
```

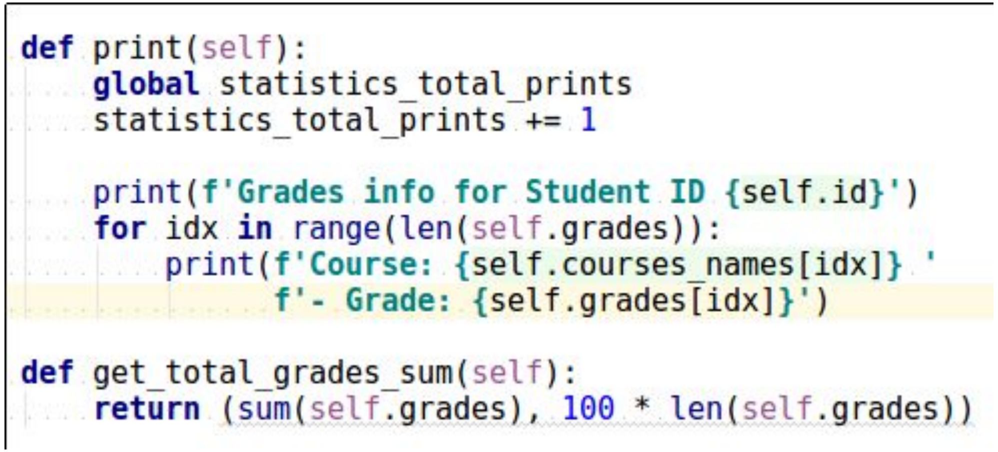


```
Grades info for Student ID ID1234
Course: Math - Grade: 70
Course: programming 1 - Grade: 70
Course: programming 2 - Grade: 85
(225, 300)
```

Problem #1: Students Grades - Code Review



```python
def __init__(self, id):
    self.id = id
    self.grades = []
    self.courses_names = []

def adjust_grade(self, grade):
    if grade < 0:
        return grade
    if grade > 100:
        return 100
    return grade

"""
This function adds a new course IFF the course
is not already added
If added, course old value is not overwritten!
"""
def add_grade(self, grade, course_name):
    self.grades.append(self.adjust_grade(grade))

    if course_name in self.courses_names:
        return False

    self.courses_names.append(course_name)
    return True
```



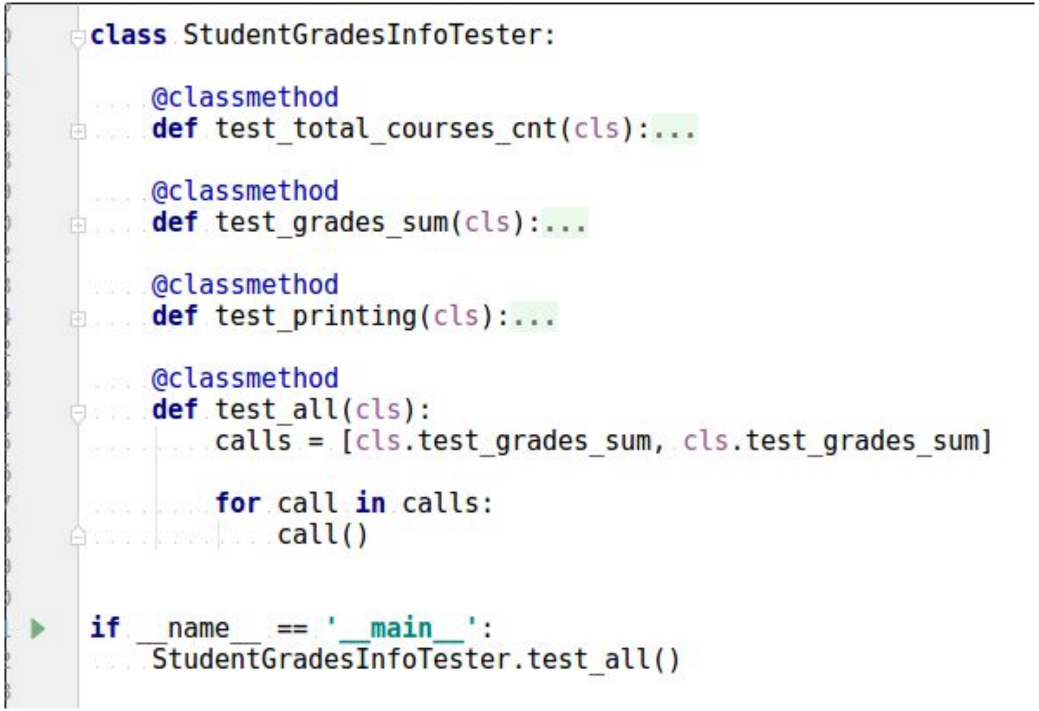
```python
def print(self):
    global statistics_total_prints
    statistics_total_prints += 1

    print(f'Grades info for Student ID {self.id}')
    for idx in range(len(self.grades)):
        print(f'Course: {self.courses_names[idx]} '
              f'- Grade: {self.grades[idx]}')

def get_total_grades_sum(self):
    return (sum(self.grades), 100 * len(self.grades))
```

Software Testing: Background

- *“**Software testing** proves the existence of bugs not their absence.”* – *Anonymous*
- *“If you don’t like **unit testing** your product, most likely your customers won’t like to test it either.”* – *Anonymous*
- Blackbox testing:
    - we test the public functionality of a class Focus on **what not how** No care of internals
- Whitebox testing:
    - we care about really what happens internally inside our methods.
- Let’s do some testing :)

Problem #2: Students Grades - Testing

- Develop a class that test our previous class
  - Try the **old** code
  - Then the **fixed** code
- You may go beyond these tests
  - For print: feel free to only sketch the approach and don’t implement

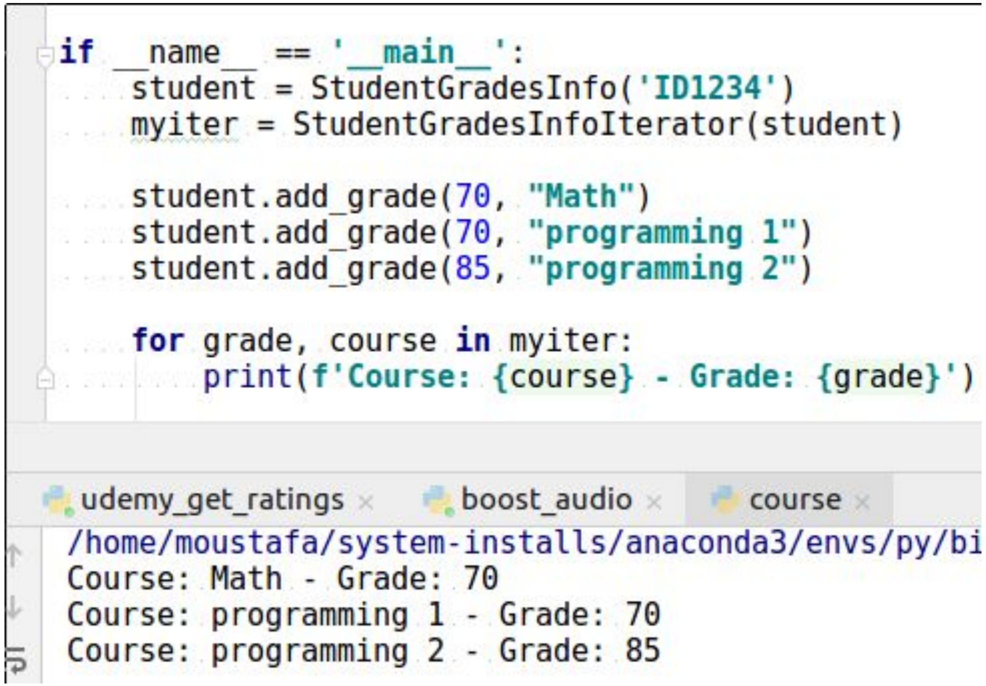


```
class StudentGradesInfoTester:

    @classmethod
    def test_total_courses_cnt(cls):...

    @classmethod
    def test_grades_sum(cls):...

    @classmethod
    def test_printing(cls):...

    @classmethod
    def test_all(cls):
        calls = [cls.test_grades_sum, cls.test_grades_sum]

        for call in calls:
            call()


if __name__ == '__main__':
    StudentGradesInfoTester.test_all()
```

Problem #3: Students Grades - Code Extension

- We would like to support iterations functionality, which is more practical than the limited print functions
  - Force a print / Print only to a console / Print all content! .. Bad design!
- For some reasons, we can’t change the code
  - Another idea is to **extend** its functionality!
- Your team lead asked to develop a class that satisfy the following main
  - Mainly a new class that works on an object from StudentGradesInfo
  - The new class allows us to iterate over an info object
  - See screenshoot

Problem #3: Students Grades - Code Extension



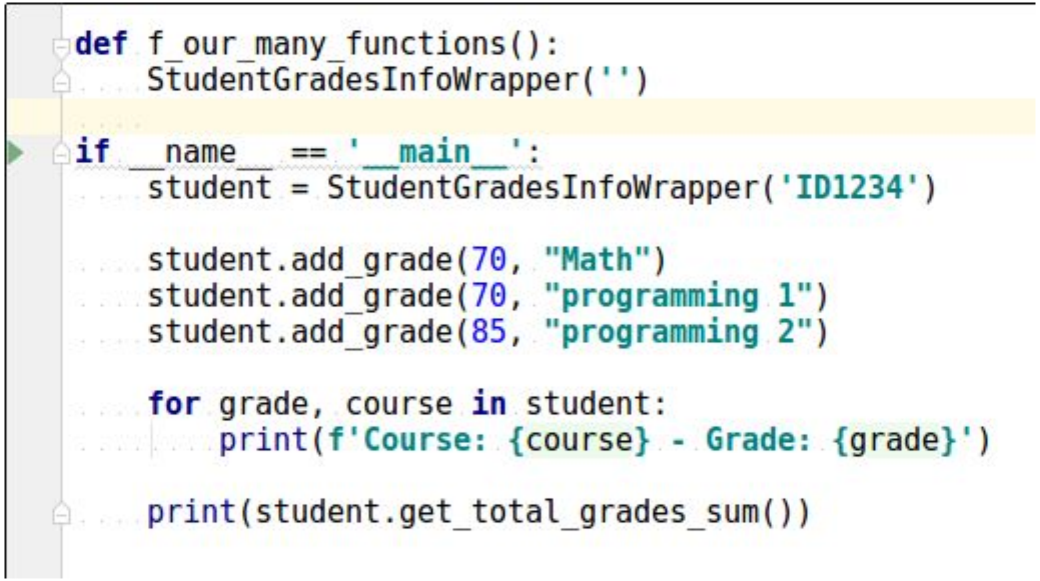
```python
if __name__ == '__main__':
    student = StudentGradesInfo('ID1234')
    myiter = StudentGradesInfoIterator(student)

    student.add_grade(70, "Math")
    student.add_grade(70, "programming 1")
    student.add_grade(85, "programming 2")

    for grade, course in myiter:
        print(f'Course: {course} - Grade: {grade}')
```

```
/home/moustafa/system-installs/anaconda3/envs/py/bi
Course: Math - Grade: 70
Course: programming 1 - Grade: 70
Course: programming 2 - Grade: 85
```

Problem #4: Students Grades - Wrapper

- StudentGradesInfo is from an open source library. *Good to save time*
    - Your team lead is afraid from hidden bugs or maintenance stop
    - What if we have 20 classes that use it and then we decided to replace or write our own!
        - Any change in this StudentGradesInfo => change in all of them!
- Your team lead suggested building a wrapper
    - The idea is create another class StudentGradesInfoWrapper
        - It provides the same functionality as StudentGradesInfo
        - It is based on a StudentGradesInfo object
    - With every call to StudentGradesInfoWrapper, just call same method in ur local object
    - Now all your code depends on the wrapper not on the open source code that may change
    - Provide also iteration cabailities

Problem #4: Students Grades - Wrapper



```python
def f_our_many_functions():
    StudentGradesInfoWrapper('')

if __name__ == '__main__':
    student = StudentGradesInfoWrapper('ID1234')

    student.add_grade(70, "Math")
    student.add_grade(70, "programming 1")
    student.add_grade(85, "programming 2")

    for grade, course in student:
        print(f'Course: {course} - Grade: {grade}')

    print(student.get_total_grades_sum())
```

“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”