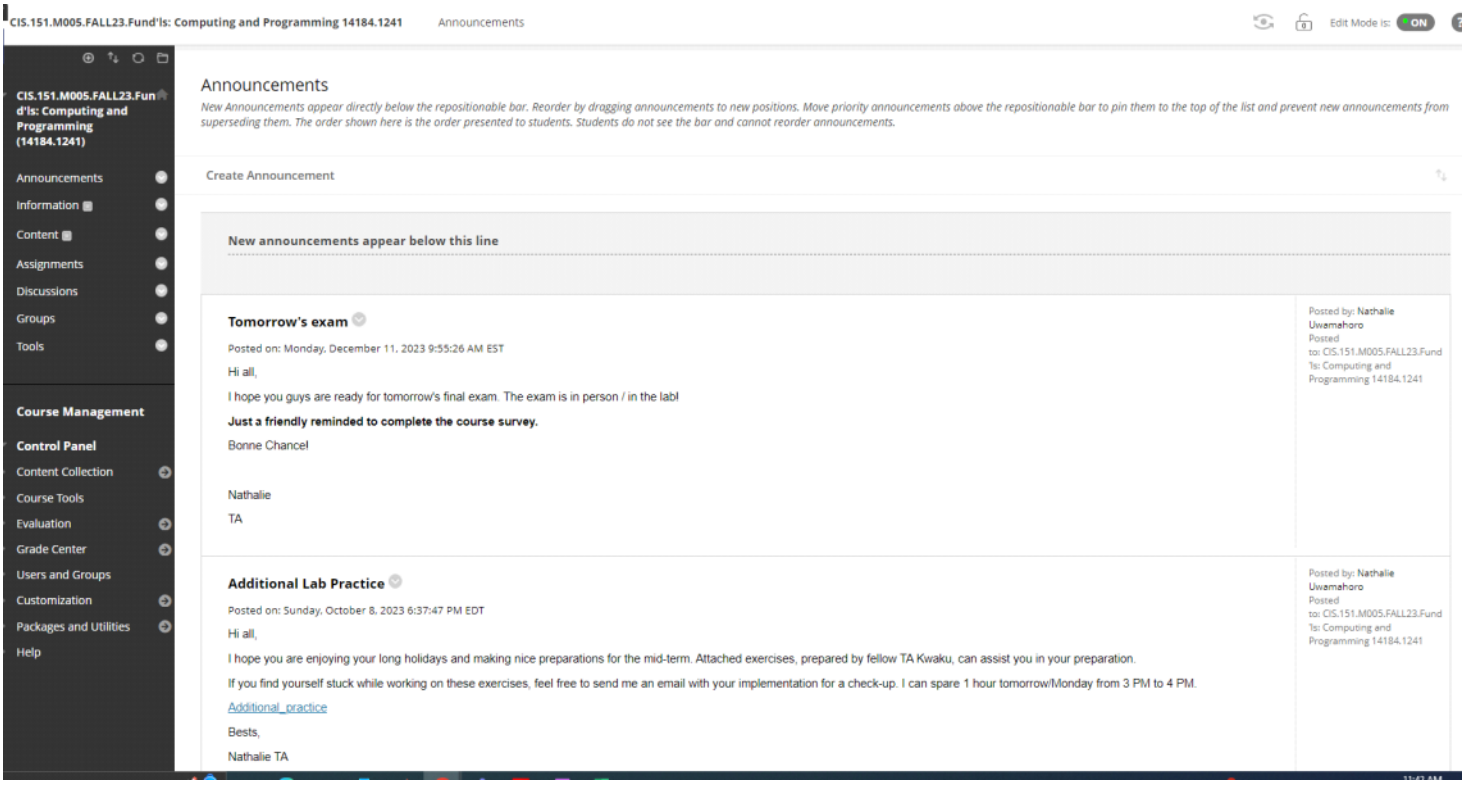
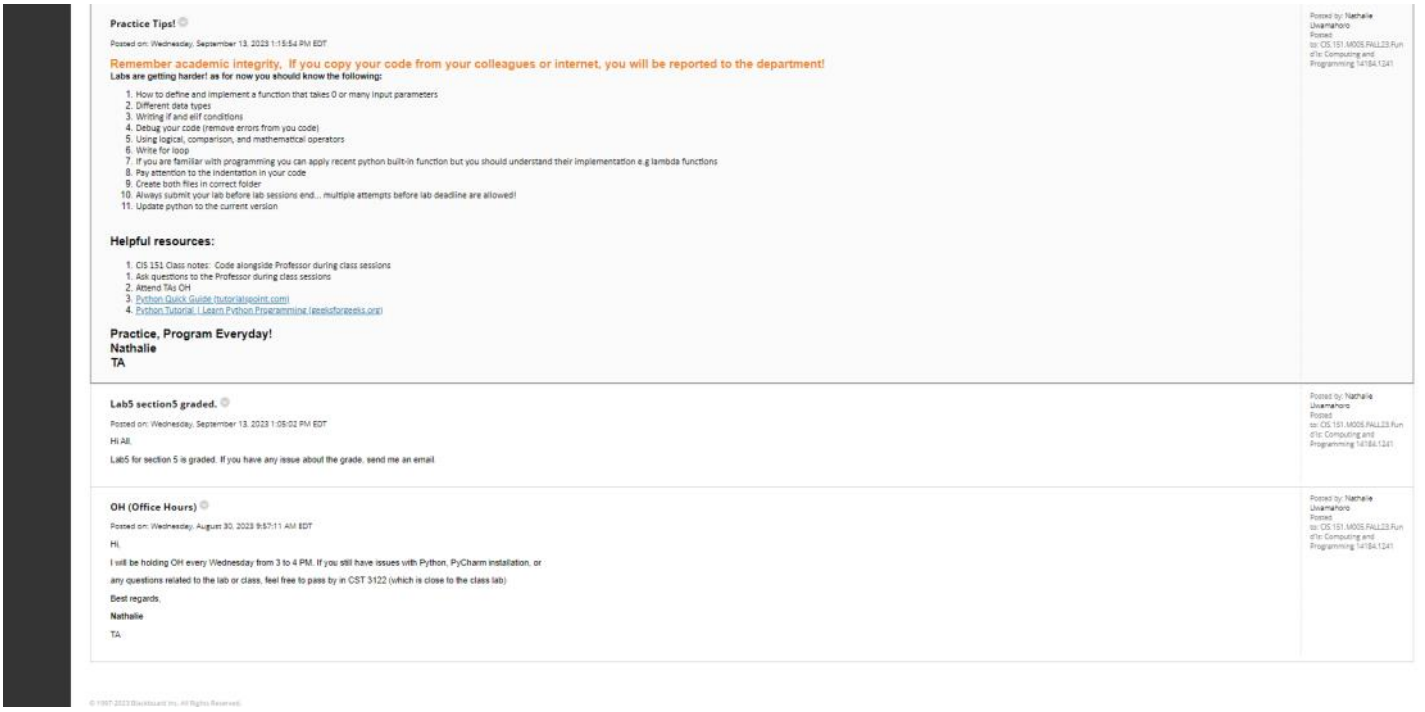
Outstanding TA

Wednesday, December 20, 2023 11:40 AM

Announcements from session 5 and 6 (60 students)





Graded labs 24 and Final Exam.

Sample lab1

Description

Instructions start PyCharm
Create a new project "lab1" make sure 'Base interpreter' is using a Python 3.x version
Create a new python file named "lab1" (file -> new -> python file -> python file).
copy-paste the code shown below into lab1.py:

```
def sayHi() -> str:
    return "hi"

def sayHiTo(name: str) -> str:
    return "hi" + " " + name

def greet(greeting: str, name: str) -> str:
    return greeting + " " + name

def func():
    print("hi")
```

Create a new unit test names "lab1test" (file -> new -> python file -> python unit test)
copy-paste the code shown below into lab1test.py

```
import unittest
from lab1 import *
class MyTestCase(unittest.TestCase):
    def test_sayHi(self):
        result = sayHi()
        self.assertEqual("hi", result)
    def test_sayHiTo(self):
        result = sayHiTo("john")
        self.assertEqual("hi john", result)
        result = sayHiTo("jane")
        self.assertEqual("hi jane", result)
    def test_greet(self):
        result = greet("Hello", "Carrie")
        self.assertEqual("Hello Carrie", result)
        result = greet("Yo", "Charlie")
        self.assertEqual("Yo Charlie", result)

if __name__ == '__main__':
    unittest.main()
```

Run lab1test.py. All the tests should pass. If not, ask for help.
1. Change the name of the function "sayHi" to "sayHello".
Run lab1test.py, one of the tests should fail.
Fix the failing test.
2. Now change the body of the function "sayHello" so it returns the string "Hello".
Run lab1test.py, one of the tests should fail.
Fix the failing test.
3. Change the body of the function "sayHiTo" so it returns a string with a period (".") at the end.
Run lab1test.py, one of the tests should fail.
Fix the failing test.
4. Change the parameter names of the function "greet":
Change "greeting" to "salutation".
Change "name" to "target".
Run lab1test.py, and fix any failing tests.
Submit lab1.py and lab1test.py

Sample lab 20

QUESTION 1

1. Start PyCharm, create a project named lab20.
Create a new python file called lab20, and copy paste this code into it.

```
def populate_dict(keyNames: list, fname: str) -> dict:
    pass
def dict_from_file(fname: str) -> dict:
    pass
def write_dict_to_file(score: dict, fname: str) -> None:
    pass
def num_failing_scores(gradebook: dict) -> int:
    pass
```

Create a new python unit test: lab20test.
Copy-paste this code into it:

```
import unittest
import lab20
import os
class MyTestCase(unittest.TestCase):
    files = ["scores.txt", "tuples.txt", "tmp.txt"]
    def setUp(self) -> None:
        with open(self.files[0], "w") as f:
            print("jen, 88, 47, 90", file=f)
        tuples = "(jen 66), (joe 77), (jane 88)"
        with open(self.files[1], "w") as f:
            print(tuples, file=f)
    def tearDown(self) -> None:
        for f in self.files:
            if os.path.exists(f):
                os.remove(f)
    def test_populate_dict(self):
        keyNames = ["name", "lab1", "lab2", "midterm"]
        result = lab20.populate_dict(keyNames, self.files[0])
        expected = {"name": "jen", "lab1": "88", "lab2": "47", "midterm": "90"}
        self.assertDictEqual(expected, result)
    def test_dict_from_file(self):
        result = lab20.dict_from_file(self.files[1])
        expected = {"jen": 66, "joe": 77, "jane": 88}
        self.assertDictEqual(expected, result)
    def test_write_to_dict(self):
        input = {"jen": 66, "joe": 77, "jane": 88}
        lab20.write_dict_to_file(input, self.files[2])
        self.assertTrue(self.files[2])
        with open(self.files[2], "r") as fh:
            lines = fh.readlines()
        expected = ["jen: 66\n", "joe: 77\n", "jane: 88\n"]
        for i in range(len(expected)):
            self.assertEqual(expected[i], lines[i])
    def test_num_failing_scores(self):
        input = {"jen": 46, "joe": 77, "jane": 88}
        result = lab20.num_failing_scores(input)
        self.assertEqual(1, result)
        input["joe"] = 39
        result = lab20.num_failing_scores(input)
        self.assertEqual(2, result)
        input["jane"] = 29
        result = lab20.num_failing_scores(input)
        self.assertEqual(3, result)

if __name__ == '__main__':
    unittest.main()
```

1. Implement the function populate_dict which
takes two input parameters keyNames which is a list, and fname which is a str
fname will have one line with a name and scores separated by comma space
returns a dictionary with keys from keyNames and corresponding values from the file
2. Implement the function dict_from_file which
takes one input parameter fname which is a str
opens and reads the file pointed to by fname which will contain one line
creates an item in a dict for each (name val) pair read from fname
returns the dictionary created
3. Implement the function write_dict_to_file which
takes two parameters score, whcih is a dict, and fname which is a string
writes each item from score to a line in fname
returns nothing
4. Implement the function num_failing_scores which

takes one input parameter gradebook, which is a dict
each item in gradebook is a name key and a score values
returns an int matching the number of scores in gradebook that are under 50

Lab22

QUESTION 1

1. Start PyCharm, create a project named lab22.
   Create a new python file called lab22, and copy paste this code into it.

```
def package_dicts(d1: dict, d2: dict) -> list:
    pass
def cumulative_grade(d1: dict) -> dict:
    pass
def gradebook(key_names: list, fname: str) -> list:
    pass
def dict_from_file3(fname: str) -> dict:
    pass
```

   Create a new python unit test: lab22test.
   Copy-paste this code into it:

```
import unittest
import lab22
import os
class MyTestCase(unittest.TestCase):
    files = ["gb.txt", "scores.txt"]
    def setUp(self) -> None:
        with open(self.files[0], "w") as fh:
            line1 = "jen 88:47:90"
            print(line1, file=fh)
            line2 = "joe 87:46:91"
            print(line2, file=fh)
        with open(self.files[1], "w") as fh:
            print("jen 46:88", file=fh)
            print("joe 77", file=fh)
            print("jane 88:66", file=fh)
    def tearDown(self) -> None:
        for file in self.files:
            if os.path.exists(file):
                os.remove(file)
    def test_package_dicts(self):
        d1 = {"k1": 12, "k2": 22}
        d2 = {"k3": 12, "k4": 22}
        result = lab22.package_dicts(d1, d2)
        self.assertListEqual([d1, d2], result)
    def test_cumulative_grade(self):
        input = {"name": "joe", "lab1": 88, "lab3": 92, "hw1": 8, "hw3": 6, "midterm": 85, "final": 95}
        result = lab22.cumulative_grade(input)
        self.assertEqual(result["name"], "joe")
        self.assertEqual(result["cumulative_score"], 90)
    def test_gradebook(self):
        key_names = ["name", "lab1", "lab2", "midterm"]
        result = lab22.gradebook(key_names, self.files[0])
        self.assertIsInstance(result, list)
        self.assertEqual(2, len(result))
        self.assertIsInstance(result[0], dict)
        self.assertIsInstance(result[1], dict)
        e1 = dict(zip(key_names, ["jen", "88", "47", "90"]))
        self.assertDictEqual(e1, result[0])
        e2 = dict(zip(key_names, ["joe", "87", "46", "91"]))
        self.assertDictEqual(e2, result[1])
    def test_dict_from_file3(self):
        result = lab22.dict_from_file3(self.files[1])
        expected = {"jen": ["46", "88"], "joe":["77"], "jane":["88", "66"]}
        self.assertDictEqual(expected, result)

if __name__ == '__main__':
    unittest.main()
```

   1. Implement the function package_dicts which
   takes two input params d1 and d2 both dictionaries
   returns a list whose first item is d1 and second is d2
   2. Implement the function cumulative_grade which
   takes one input parameter d1, which is a dictionary
   d1 is guaranteed to have keys "name", "midter" and "final"
   d1 may have one or many keys for labs
   each key for a lab will have the word lab follwed by a number
   computes a cumulative score = 0.6 (average of lab scores) + 0.2 midterm score + 0.2 final score
   returns a dictionary with a key "name" and value of name from d1
   and a key "cumulative_score" and value the computed cumulative score
   3. Implement the function gradebook which
   takes two input parameters key_names, which is a list of strings and fname which is a string
   each line in fname has a name, a space and a set of scores separated by colon
   forms a dictionary from keys_names and the contents of each line
   returns a list of dictionaries, one dictionary for each line
   4. Implement the function dict_from_file which
   takes one input parameter fname, which is a string
   each line in fname has a name, a space and a set of scores separated by colon
   returns a dictionary whose keys are all the names and values are lists of scores

Lab 24:

QUESTION 1

1. Start PyCharm, create a project named lab24.
   Create a new python file called lab24, and copy paste this code into it.

```
def string_to_dict(p1: str) -> dict:
    pass
def list_of_lists_to_dict(p1: list) -> dict:
    pass
def union(d1: dict, d2: dict) -> dict:
    pass
def intersection(d1: dict, d2: dict) -> dict:
    pass
```

   Create a new python unit test: lab24test.
   Copy-paste this code into it:

```
import unittest
import lab24
class MyTestCase(unittest.TestCase):
    def test_string_to_dict(self):
        input ="[[jen, 98], [joe, 77], [jill, 85]]"
        result = lab24.string_to_dict(input)
        self.assertIsInstance(result, dict)
        self.assertDictEqual({"jen":98, "joe":77, "jill":85}, result)
    def test_list_of_lists_to_dict(self):
        input = [["jen", 98, 82], ["joe", 77], ["jill", 95, 72, 88]]
        result = lab24.list_of_lists_to_dict(input)
```

```
        self.assertIsInstance(result, dict)
        self.assertDictEqual({"jen":[98, 82], "joe":[77], "jill":[95, 72, 88]}, result)
    def test_union(self):
        one = {"k1":1, "k11":11, "k101":101}
        two = {"k2":1, "k11":12, "k101":102}
        result = lab24.union(one, two)
        self.assertIsInstance(result, dict)
        self.assertDictEqual({"k1":[1], "k2":[1], "k11":[11,12], "k101":[101, 102]}, result)
    def test_intersection(self):
        one = {"k1":1, "k11":11, "k101":101}
        two = {"k2":1, "k11":12, "k101":102}
        result = lab24.intersection(one, two)
        self.assertIsInstance(result, dict)
        self.assertDictEqual({"k11":[11,12], "k101":[101, 102]}, result)

if __name__ == '__main__':
    unittest.main()
```

1. Implement the function string_to_dict which
takes one input parameter p1 which is a str
p1 is a string representation of a list
each item in the list is a two item list
each two item list is a name followed by a score
returns a dictionary with an item for each two item list, with the value an int
2. Implement the function list_of_lists_to_dict which
takes one input parameter p1 which is a list of lists
each item in p1 is a list which has at least two items a name and at least one score
returns a dictionary with names as keys and list of scores as values
3. Implement the function union which
takes two input parameters d1 and d2, both dictionaries
both d1 and d2 have string keys and int values
returns a dictionary with string keys and list values, including any key that is in either d1 or d2 or both
4. Implement the function intersection which
takes two input parameters d1 and d2, both dictionaries
both d1 and d2 have string keys and int values
returns a dictionary with string keys and list values, including any key that is in both d1 and d2

1. Attach File