Your Name:	TJ Sipin
Perm #:	343050-1

This homework and its due date are posted on the course website: ucsb-cs8.github.io/w20/hwk. Submit a PDF file of this assignment following the guidelines posted on the website.

h03: Perkovic Ch2.3 (Lists and Tuples), 2.4 (Objects and Classes)

0. (5 points) Use the provided homework template (without any blank pages), filling out all requested fields, and correctly submitting a legible electronic document on Gradescope before the due date. By submitting this document you agree that this is your own work and that you have read the policies regarding Academic Integrity: https://studentconduct.sa.ucsb.edu/academic-integrity.

1. **Section 2.3 describes lists and tuples in Python. Assuming the following assignment statements have been entered at the Python prompt:**

```
work_days = ("Mon", "Tues", "Wed", "Thu", "Fri")
weekend = ("Sat", "Sun")
week = work_days + weekend
courses = [ ("CMPSC 8","MW"), ("CMPSC 8","TR"), ("PSTAT 120","MWF")]
exams = [[1.27, 2.24], [1.28, 2.25], 3.14]
```

(2 pts each) **What would be the result of entering the following in IDLE (Python interactive shell)?** (Note: You are encouraged to check your answers at the Python prompt before turning in your work, but try this on paper first, just by reading the text and trying to predict what will happen. Then try typing in the results at the Python prompt. Change your answers if they were mistaken, but even more important, try to figure out why you were incorrect).

Be very precise. For example, `True` is not the same in Python as `true`: upper vs. lower case matters; "UCSB" and ["UCSB"] are not the same in Python—one is a string, and the other is a list of length one containing a single string. You will not get full credit for answers that are not precisely correct.

Expression	**Result**	**Expression**	**Result**
`len(work_days)`	5	`len(week)`	7
`len(courses)`	3	`len(exams)`	3

Expression	**Result**	**Expression**	**Result**
`len(courses[1])`	2	`len(exams[0])`	2
`"CMPSC 8" in courses`	False	`max(exams[1])`	2.25
`"CMPSC 8" in courses[2]`	False	`"Sun" in week`	True

2. (4 pts) As described in section 2.3, lists and tuples are similar, but there are two big differences. Briefly list those differences.

1. Lists are mutable, tuples are not
2. Tuples use parentheses ( ), while lists use brackets [ ]

3. (3 pts) Write a line of Python code that assigns the variable `my_course` to have a value that is a **tuple** of length 1, containing only the single string "`CS 8`".

my_course = ("CS 8")

4. (2 pts each) Assume that the following sequence of statements has been entered at the Python prompt. Note that subsequent statements may change the value of the variables (e.g., `fruits` is altered by the call to the `append` method)

```
fruits = ["pear","banana","apple","banana"]
fruits.append("orange")
fruits.reverse()
```

Expression	**Result**	**Expression**	**Result**
`len(fruits)`	5	`fruits[-1][-2]`	'a'
`fruits[0]`	'orange'	`fruits[2][0]`	'a'
`fruits.count("apple")`	1	`"plum" not in fruits`	True
`fruits.count("banana")`	2	`fruits.count(fruits[-2])`	2
`len(fruits[1])`	6	`"an" in fruits[2]`	False

5. (2 pts each) As discussed in Section 2.4, the `type()` function returns the type of a Python value. We are using the definition of the variable `week` from Question 1.

When you pass a variable such as `x`, `type(x)` returns the type of the *value* that the variable `x` currently refers to. Below, you need to figure out the type of the expression that is given to the `type()` function, which means you should figure out the result of the expression, before you can predict its type. (*Note: You are encouraged to check your answers at the Python prompt before turning in your work, but try this on paper first, just by reading the text and trying to predict what will happen. Then try typing in the results at the Python prompt. Change your answers if they were mistaken, but even more important, try to figure out* ***why*** *you were incorrect*).

What would be the result of entering the following at the Python interactive shell prompt?

Expression	`<class '_____'>`	**Expression**	`<class '_____'>`
`type("8")`	str	`type(3 * '8')`	str
`type(8)`	int	`type((8, 8))`	tuple
`type(abs(-8.0))`	float	`type([8]==[8, 8])`	bool
`type([8, 88])`	list	`type(week)`	tuple
`type('8.88')`	str	`type(week[-1][0])`	str
`type(8.88)`	float	`type(4//2/4%2)`	float

6. (3 pts) Briefly explain what would happen if we tried to sort a list containing numbers and strings? Why?

You would get an error because strings and integers cannot be compared.

7.1 (2 pts) Section 2.4 introduces the important concept of the *operator precedence* rules. Make sure to read and understand that section. Fill in the blanks:

The *order* in which operators are evaluated is defined either *explicitly* using parentheses

or *implicitly* using either the *operator precedence* rules or the left-to-right

evaluation rule if the operators have the same precedence.

7.2 (3 pts) Given what you just read, figure out the result of this expression. Explain the result.

```
(2 ** 3 ** 2) == ((2 ** 3) ** 2)
```

False

The left hand side's order is defined implicity due to the left-to-right evaluation rule, whereas the right hand side's order is defined explicitly using parentheses. First the 2 is exponentiated to the 3 (which equals 8), and then squared (equaling 64).