SI 506 Midterm

1.0 Dates

- Release date: Thursday, 21 October 2021, 4:00 PM Eastern
- Due date: on or before Saturday, 23 October 2021, 11:59 PM Eastern

2.0 Overview

The midterm exam is open network, open readings, and open notes. You may refer to code in previous lecture exercises, lab exercises, and problem sets for inspiration.

We recommend that at a minimum you bookmark the following w3schools Python pages and/or have them open in a set of browser tabs as you work on the midterm exam:

- Python keywords
- Python operators
- Python built-in functions
- Python `list` methods
- Python `str` methods

3.0 Points

The midterm is worth 1000 points and you accumulate points by passing a series of auto grader tests.

4.0 Solo effort

❗ You are prohibited from soliciting assistance or accepting assistance from any person while taking the exam. The midterm code that you submit *must* be your own work. Likewise, you are prohibited from assisting any other student required to take this exam. This includes those taking the exam during the regular exam period, as well as those who may take the exam at another time and/or place due to scheduling conflicts or other issues.

5.0 README and template files

In line with the weekly lab exercises and problem sets you will be provided with a number of files:

1. a `README.md` that contains the assignment instructions
2. a numbered `midterm_XX.py` template file for you to write your code.
3. one or more `*.txt` and/or `*.csv` files that contain exam data.

Please download the midterm files from Canvas Files as soon as they are released. This is a timed event and delays in acquiring the assignment files will shorten the time available to engage with the assignment. The clock is not your friend.

The template file will contain function definitions that you will implement along with a `main()` function from which you will call the other functions, make variable assignments, and perform other tasks in response to a series of challenges.

Implementing functions involves replacing the placeholder `pass` statement with working code. You may be asked to add missing parameters to function definitions. You may also be asked to define a function in its entirety (less the Docstring) as directed per the instructions.

You will call each function from the `main()` function passing arguments to the function and assigning the return value to a specified variable per the instructions.

❗ *DO NOT* modify or remove the scaffolded code that we provide in the template unless instructed to do so.

The template `*.py` file resembles the following skeletal scaffolding (verbose multi-line Docstring instructions are shortened to a single line in the example):

```python
import some_module


def some_function(some_parameter, another_parameter):
    """A description of expected behavior.

        Parameters:
            ...

         Returns:
            ...
    """

    pass # TODO Implement


def another_function(): # TODO Add missing parameter(s)
    """A description of expected behavior.

        Parameters:
            ...

        Returns:
            ...
    """

    pass # TODO Implement


 # TODO define and implement Challenge X function here


def main():
    """Entry point for the program.

        Parameters:
            ...

         Returns:
```

```
            . . .
        """

    # Manage execution flow below.

    # CHALLENGE 01

    var_01 = None # TODO Assign value

    # . . .

    # CHALLENGE 0X

    # TODO Implement loop(s), conditional statement, add filtered value(s)
to specified variable

    # . . .

    # CHALLENGE 10

    var_02 = None # TODO Call function; assign return value

    # . . .


# Check if the Python interepreter knows this file as "__main__" (i.e., a
script
# intended to be run from the command line). If True, call the main()
function
# which serves as the entry point to the program.
if __name__ == '__main__':
    main()
```

6.0 Challenges

The midterm comprises ten (10) challenges. The teaching team recommends that you complete each challenge in the order specified in the README.

Certain challenges can be solved with a single line of code. Others may involve writing several lines of code including implementing one or more functions in order to solve the challenge.

The challenges cover the following topics introduced between weeks 01 and 07:

- Basic syntax and semantics
  - Values (objects), variables, variable assignment
  - Expressions and statements
  - Data types
    - numeric values (int, float)
    - boolean values (True, False)
    - NoneType object (None)
    - sequences: list, range, str, tuple,

- Operators: arithmetic, assignment, comparison, logical, membership
- Sequence indexing and slicing
- List creation; adding, modifying, removing and unpacking list elements
- Tuple creation; adding items (packing) and unpacking tuple items
- Nested lists and tuples
- String, list, and tuple concatenation
- String formatting: formatted string literal (f-string)
- Control flow
  - Iteration
    - `for` loop, `for i in range()` loop, `while` loop
    - Accumulator pattern
    - Counter usage (e.g. `count += 1`)
    - `break` and `continue` statements
    - Nested loops
  - Conditional execution
    - `if`, `if-else`, `if-elif-else` statements
    - Truth value testing (`if < object >:`)
- Functions
  - Built-in functions discussed in class and featuring in lab exercises and problem sets
  - Type methods (e.g., `list` and `str` methods)
  - User-defined functions
    - Defining a function with/without parameters, parameters with default values, and with/without a return statement
    - Calling a function and passing to it arguments by position and/or keyword arguments
    - Calling a function or functions from within another function's code block
    - Assigning a function's return value to a variable
  - `main()` function (included in midterm template scaffolding)
- Files read / write
  - `with open()` statement
  - Reading from and writing to *.txt files
  - `csv` module import (included in scaffolded code)
  - `csv.reader()`, `csv.writer()`
  - Reading from and writing to *.csv files

7.0 A note on code styling

The auto grader will include tests that check whether or not your code adheres to Python styling guidelines relative to the use of whitespace in certain expressions and statements. The goal is to encourage you to write code that adheres to the Python community's styling practices. Doing so enhances code readability and aligns you with other Python programmers.

In particular, always surround the following operators on either side with a single space:

- assignment (`=`)
- augmented assignment (`+=`, `-=`, etc.)
- comparisons (`==`, `<`, `>`, `!=`, `<>`, `<=`, `>=`, `in`, `not in`, `is`, `is not`)
- Booleans (`and`, `or`, `not`).

```python
# Correct
var = search_entities(entities, search_term)

# Incorrect
var=search_entities(entities, search_term)

# Correct
count += 1

# Incorrect
count+=1
```

Note however that an exception exists with respect to function parameters and arguments. Do *not* surround the assignment operator with spaces when either:

1. defining a parameter with a default value
2. passing a keyword argument

```python
# Correct
def create_email_address(uniqname, domain='umich.edu'):
    """TODO"""
    return f"{uniqname}@{domain}"

# Incorrect
def create_email_address(uniqname, domain = 'umich.edu'):
    """TODO"""
    return f"{uniqname}@{domain}"

# Correct
email_address = create_email_address(uniqname='anthwhyte',
domain='gmail.com')

# Incorrect
email_address = create_email_address(uniqname = 'anthwhyte', domain =
'gmail.com')
```

8.0 Gradescope submissions

You may submit your problem solution to Gradescope as many times as needed before the expiration of the exam time. Your **final** submission will constitute your exam submission.

❗ You *must* submit your solution file to *Gradescope* before the expiration of exam time. Solution files submitted after the expiration of exam time will receive a zero (0) score.

9.0 auto grader / manual scoring

If the auto grader is unable to grade your submission successfully with a score of 1000 points the teaching team will review your submission. Partial credit **may** be awarded for submissions that fail one or more auto

grader tests if the teaching team (at our sole discretion) deem a score adjustment warranted.

If you submit a partial solution, please include (if you have time) comments that explain what you were attempting to accomplish in the area(s) of the program that are not working properly. We will review your comments when determining partial credit.