

## EECS 1015 – Fall 2021: Final programming exam

**Topics covered:** *flow-control, functions, strings, formatted printing, lists, dictionaries, assertions, classes, multi-file*

**Assigned:** Dec 7, 2021 (Tuesday)

**Due date:** Dec 9, 2021 (Thursday) [11.59 pm Eastern Time – No extensions or late submissions]

### #Important for the final programming exam

- 1) Submit your final exam via web-submit to the "final" folder (see instructions on the last pages).
- 2) Make sure you correctly submit your files (**two files** – `final.py`, `utilities.py` [please do not zip them]).
- 3) Read the task descriptions carefully to understand everything you need to do. There are five (5) tasks [task 0-task4].

## 1. INSTRUCTIONS

- 1) You have three days to complete this take-home final exam. There are no extensions or late submissions.
- 2) The final programming questions incorporate components from topics 2 to 10.
- 3) The final has five (5) tasks – (task 0, task 1, task 2, task 3, task 4). Please complete each task.
- 4) This final is graded out of 100 points.
- 5) IMPORTANT: Your program must use the starting code provided below (see trinket.io link)
- 6) Please name your submission: – `final.py`, `utilities.py`

### STARTING – two files provided with starting code.

(1) `final.py` – the main program with the five tasks functions.

(2) `utilities.py` – variables and class needed for the tasks (see details for Task 3 and Task 4)

You can cut-and-paste from here: <https://trinket.io/python/576c440592>

(On the trinket site, there is a tab to access the two files – remember to rename "main.py" as "final.py")

A video describing the final is available here: [https://www.eecs.yorku.ca/~mbrown/EECS1015\\_final.mp4](https://www.eecs.yorku.ca/~mbrown/EECS1015_final.mp4)

**TASK 0 (0 points correct, -10 points deducted if not done correctly)**

1) Place your information in the comments at the beginning of your python file. Please include the standard information in the comments as follows:

```
#####  
# Name: Your name  
# Student ID: XXXXXXX  
# email: XXXXX  
# Section A or B  
#####
```

2) Function `task0()` should print out the following:

```
Final Exam - EECS1015  
Name: Your Name  
Student ID: XXXXXXX  
email: youremail@aol.com  
Section A or B
```

Why the redundant information? We have software that helps us with grading. That software looks only in the comments. The function `task0()` helps the instructor and TA verify your name visually while grading.

## Task 1: Rock, Paper, Scissors (Topics: Variables, user-input, flow-control, function) [20pnts]

Task 1 implements the "rock, paper, scissors" game. This is a two-player game, where each player simultaneously selects from one of three options: *rock*, *paper*, *scissors*.

**The game is typically played with hand gestures and has the following rules:**

Rock wins against scissors, loses to paper, ties with rock.

Paper wins against rock, loses to scissors, ties with paper.

Scissors win against paper, lose to rock, tie with scissors.

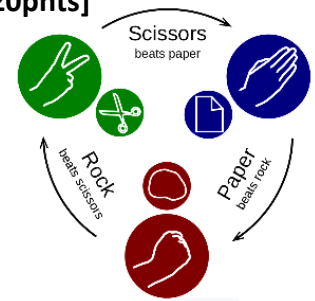


Image credit: Enzoklop (CC)

In a real game, both players would make their selection simultaneously.

In our version, the user will select first, and then the computer opponent named "HAL" will make a random selection.

### Require function

In addition to the `task1()` function, you need to implement the following function:

`printOutcome(params: userSelection, computerSelection) -> no return`

The parameters are the user's and computer's selection. The function prints the outcomes depending on the selections:

(1) "You win!"; (2) "HAL wins!"; (3) "A tie!"

You decide what data type the parameters should be (e.g., integers representing the selection or strings representing "rock", "paper", "scissors"). Just ensure your function prints the correct outcome.

### Task 1 should behave as follows:

- (1) Print "Rock, Paper, Scissors!"
- (2) Print "Make your selection. . ."
- (3) Get input from user – 1 (rock), 2 (paper), 3 (scissors).
- (4) If the input is not 1, 2, or 3, print "Invalid selection. Try again." and ask the user for input again.
- (5) Randomly select HAL's choice – rock, paper, scissors.
- (6) Print out the user's selection and HAL's selection.
- (7) Pass the user's and HAL's selections to the function `printOutcome()` to print the outcome of the game.
- (8) Ask the user if they want to play again (Y). If the input is 'Y' or 'y', go back to (2).

**Example of task 1 running (user input in red) – see video link for more examples.**

----- Task1: Rock, Paper, Scissors -----

Rock, Paper, Scissors!

Make your selection. . .

(1) rock, (2) paper, (3) scissors? **1**

You: rock

HAL: rock

A tie!

Play again (Y/N)? **y**

Make your selection. . .

(1) rock, (2) paper, (3) scissors? **2**

You: paper

HAL: scissors

HAL wins!

Play again (Y/N)? **y**

Make your selection. . .

(1) rock, (2) paper, (3) scissors? **4**

Invalid selection. Try again.

Make your selection. . .

(1) rock, (2) paper, (3) scissors? **3**

You: scissors

HAL: paper

You win!

Play again (Y/N)? **n**

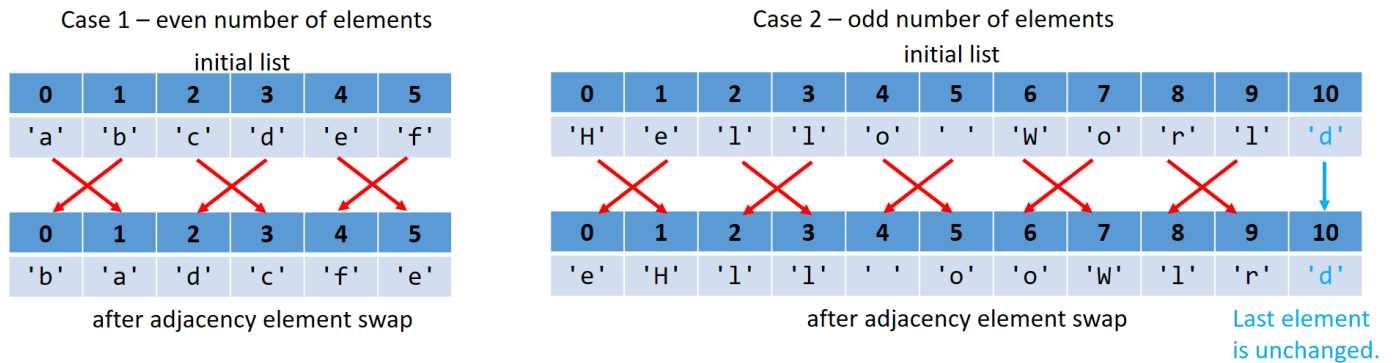
You can assume the input will be an integer. Check if it is 1, 2 or 3. If not, inform user of invalid selection as shown.

## Task 2 – Adjacent element swap for a list (Topics: list, loops, functions, string manipulation, assert) [20pts]

The user will input a string as a series of characters separated by spaces. The input string should be converted to a list of characters. Task 2 is to swap the values of every two items in the list; for example – items at positions 0 and 1 are swapped; items 2 and 3 are swapped; items 4 and 5 are swapped, and so on, until the end of the list. You can assume that the list has at least two items – if not, your swap function should raise an `AssertionError` (see details below).

Note that there are two cases: (1) the list has an even number of elements; (2) the list has an odd number of elements. For case 1, swap all elements. For case 2, swap all elements but leave the *last* element unchanged.

See the diagram below for examples.



### Require function

In addition to the `task2()` function, you need to implement the following function:

`swapAdjacentElements(params: alist) -> no return`

The parameter `alist` is a list of characters. The function should modify the list as described above. Note that you should modify the list passed to the function and not create a new list. The function has no return.

**ASSERT:** This function should assert with the following error if the list has less than two (2) characters:

"Must enter two or more characters!"

### Task 2 should behave as follows:

- (1) Prompt the user to input a sequence of two or more characters separated by spaces.
- (2) Convert the input string into a list of characters.
- (3) Print the list.
- (4) Print the list as a string with no spaces between the characters.
- (5) Call function `swapAdjacentElements()` to modify the list.  
See instructions above. This function does not create a new list; instead, it should modify the list passed as an argument.
- (6) Print the list again to show that the list has been modified.
- (7) Print the list as a string again with no spaces between the characters.

*See next page for example output.*

**Example of task 2 running (user input in red) – see video link for more examples.**

----- Task2: Swap Adjacent List Elements -----

Input two or more chars separated by spaces: **a b a b a b a b**

Initial list

['a', 'b', 'a', 'b', 'a', 'b', 'a', 'b']

String version: 'abababab'

Modified list

['b', 'a', 'b', 'a', 'b', 'a', 'b', 'a']

String version: 'babababa'

Example of case 1 (even # of items)

----- Task2: Swap Adjacent List Elements -----

Input two or more chars separated by spaces: **t h i s i s a t e s t**

Initial list

['t', 'h', 'i', 's', 'i', 's', 'a', 't', 'e', 's', 't']

String version: 'thisisatest'

Modified list

['h', 't', 's', 'i', 's', 'i', 't', 'a', 's', 'e', 't']

String version: 'htsisitaset'

Example of case 2 (odd # of items)

----- Task2: Swap Adjacent List Elements -----

Input two or more chars separated by spaces: **1 2 3 4 5**

Initial list

['1', '2', '3', '4', '5']

String version: '12345'

Modified list

['2', '1', '4', '3', '5']

String version: '21435'

----- Task2: Swap Adjacent List Elements -----

Input two or more chars separated by spaces: **1**

Initial list

['1']

String version: '1'

Modified list

Traceback (most recent call last):

File "final.py", line 119, in <module>

main()

File "final.py", line 110, in main

task2()

File "Cfinal.py", line 71, in task2

swapAdjacentElements(theList)

File "final.py", line 55, in swapAdjacentElements

assert len(theList) >= 2, "Error"

AssertionError: Error

Example of that causes an  
AssertionError

### Task 3 – Student Information Unpacking (Topics: Nested collections, flow-control, multi-file, strings) [25pnts]

Requires: `utilities.py` -> variables `students` and `studentsInfo`

Task 3 is to print out a list of students and their information. This information is stored in two variables (a list and a dictionary of lists). The data is organized as shown in the diagram below. Specifically, the variable `students` is a list of 47 strings of student names; the variable `studentsInfo` is a dictionary where the keys indicate information about the students. Each key is associated with a list of integers that indexes into the `students` list variable.

The keys in the dictionary indicate the following information: 'Year 1' to 'Year 4' is what year a student is studying. For example, 'Sasha' is a 'Year 2' student (see below). Keys 'CS', 'DM', 'BZ', 'SE' indicate the students' program of study (e.g., 'CS' is comp sci, 'DM' is digital media, 'BZ' is business, and 'SE' is software engineering). For example, 'Solar' is a 'CS' student. Keys 'On Campus' and 'Off Campus' indicate the students' living arrangements. For example, 'Mahmoud' lives 'On Campus'.

```
index      0          1          2          3          4          5          ...          45          46
students = ['Juhong', 'Sasha', 'Solar', 'Ali', 'Amy', 'Gopi', ..., 'Mahmoud', 'Rutha']

studentsInfo = {
    'Year 1': [42, 24, 12, 37, 35, 23, 32, 5, 34, 19],
    'Year 2': [1, 28, 36, 46, 30, 15, 6, 33, 13, 7, 18, 44],
    'Year 3': [26, 11, 21, 3, 2, 14, 31, 27],
    'Year 4': [25, 4, 20, 0, 16, 17, 40, 39, 29, 22, 41, 9, 8, 10, 38, 43, 45],
    'CS': [24, 1, 33, 12, 2, 30, 23, 34, 3, 45, 6, 22, 39, 11, 14, 28, 15, 26, 21, 7],
    'DM': [44, 18, 16, 40, 29, 13, 46],
    'BZ': [10, 5, 9, 31, 41, 20, 8, 25, 17, 42, 35, 43, 38, 0],
    'SE': [4, 19, 27, 32, 36, 37],
    'On Campus': [24, 13, 6, 3, 22, 4, 23, 12, 15, ..., 33, 40, 43, 35, 8, 45],
    'Off Campus': [30, 1, 37, 36, 41, 16, 9, 25, 29, 17, 0, 18, 31, 44, 5, 32, 2, 20, 28]
}
```

Task 3 should behave as follows:

- 1) Use the two variables above to print out each student's information sorted by the student's name (see below).
- 2) Format the output for each student as follows:

```
Vlادن (Year 3) Program: CS Housing: On Campus
^^^^^^^^^^
```

There should be ten characters for each name. If a name has less than ten characters, you'll need to pad it with leading spaces. Using ten characters for the name ensures the final output of each student aligns, as shown below.

Example of task 3 running – – see video link to see an example of the full list of names being printed.

```
Abisai (Year 4) Program: BZ Housing: On Campus
Ajay (Year 2) Program: SE Housing: Off Campus
Alexi (Year 4) Program: DM Housing: On Campus
Ali (Year 3) Program: CS Housing: On Campus
Amy (Year 4) Program: SE Housing: On Campus
...
Ursula (Year 2) Program: CS Housing: Off Campus
Vlادن (Year 3) Program: CS Housing: On Campus
Wen (Year 1) Program: SE Housing: Off Campus
Weng-Fai (Year 3) Program: CS Housing: On Campus
Yasu (Year 4) Program: CS Housing: On Campus
Yuki (Year 4) Program: BZ Housing: On Campus
Yunjo (Year 2) Program: DM Housing: Off Campus
```

## Task 4 – Aquarium (Topics: classes/objects, flow-control, string formatting, multi-file, collections) [35pnts]

This task simulates a text-based aquarium with five random "sea creatures" that move back and forth on the screen. This task uses the class defined in the `utilities.py` file named `SeaLife`. The class has already defined several class variables that are useful to perform this task. You need to complete the class definition as follows:

### SeaLife class

**Class data** (already defined for you).

```
category = ["fish1", "fish2", "jelly", "crab"]           # categories of sea life
draw = { "fish1":["><>", "<><"], "fish2":[">()", "(<"],    # characters used for each category
        "jelly":["~>"], "<~"], "crab":["^_^", "^_^"] }    # and for the two directions 0 or 1
speed_range = {"fish1":(5,10), "fish2":(1,10), "jelly":(1,5), "crab":(10,20)} # speed ranges
```

### Instance data (4)

**cat** a *string* that records what category an object is: fish1, fish2, jelly, or crab  
**direction** an *integer* that is 0 or 1 to indicate the direction the object is moving -- see `move()` and `getStr()`  
**pos** a *float* that represents the sea life's current position  
**speed** a *float* between [0.1 – 2.0] depending on the category that is the object's speed -- see `__init__()`

### Methods (3)

`__init__(params: direction, pos)` two parameters for the starting direction and position.

- assert if `direction` isn't 1 or 0 with error message "direction must be 0 or 1".
- assert if `pos` isn't greater than 0 and less than 39 with error message "pos must be between 1 and 38".
- set instance variables `direction` and `pos` to the parameters `direction` and `pos`.
- randomly set instance variables `cat` to be "fish1", "fish2", "jelly", or "crab".
- Depending on the category, look up the speed range in the class variable `speed_range` and set instance variable `speed` to a random `int` between the speed range for the object's category and divide by 10. For example, if the category for this object is "crab", look up the speed range in the `speed_range` class dictionary. The speed range for crab is (10, 20), so set `speed` to a random `int` between 10 and 20 and then divide by 10 (so a crab's speed is between 1.0 and 2.0). If the category is "jelly", set `speed` to random `int` between 1 and 5 and divide by 10 (so a jelly's speed is between 0.1 and 0.5).

`move()` no parameters; no return

- if the `direction` is 0, set `pos` equal to `pos + speed` (i.e., moving from left to right)
- if the `direction` is 1, set `pos` equal to `pos - speed` (i.e., moving from right to left)
- if the `pos` is greater than 40, set `direction` to 1 (i.e., reverse direction)
- if the `pos` is less than 1, set `direction` to 0 (i.e., reverse direction)

`getStr()` no parameters; returns a *string* -- see details below

- this method returns a string that depicts the sea life moving in the aquarium. The string depends on the object's category and direction. The characters that you should draw are in the class dictionary variable `draw`.
- Example #1, if the object's category is "fish1" and direction is 0, then draw "><>" for the sea life.
- Example #2, if the object's category is "jelly" and the direction is 1, then draw "<~" for the sea life.
- See the diagram below on how to format the string.

For this example, assume the object's instance variables are as follows:

`int(pos)` is 18 (recall, `pos` is a float, so `int()` is converting it to an integer)

`cat` is "jelly", `direction` is 1

Return a string as follows:

`' ' + "(<~" + ' ' + 'jelly'`

pos (18) spaces      category      (42-pos) spaces      cat

characters for direction 1

#### Task 4 should behave as follows:

1. First, ask the user to "Press enter to start aquarium."
2. Create a list of five `SeaLife` objects
  - randomly set the starting position to an integer between 5 and 30
  - randomly set the starting direction to either 0 or 1
  - note that the `__init__()` method will randomly select the category of sea life for each of your creatures
3. Set the current time step to 1
4. Print out the current time step as shown in the example below (40 "-" + "Time: " + current time step)
5. For each `SeaLife` object in your list
  - get the object's "move string" via the `getStr()` method and print it out
  - call the object's `move()` method
6. Slow down the output by calling `sleep()` for 0.3 seconds (see *Lab 8* if you forgot how to use `sleep()` )
7. Repeat from step (4) for 50 times-steps as shown below

#### Example of task 4 running – see video link for more examples.

----- Task4: Aquarium -----

Press enter to start aquarium...

```
-----Time: 1
      ^ ^
      _
      ><>
    <><
    >()
  (<~
-----Time: 2
      ^ ^
      _
      ><>
    <><
    >()
  (<~
-----Time: 3
      ^ ^
      _
      ><>
    <><
    >()
  (<~
...
-----Time: 48
      ^ ^
      _
      <><
    ><>
  ()<
  (<~
-----Time: 49
      ^ ^
      _
      <><
    ><>
  ()<
  (<~
-----Time: 50
      ^ ^
      _
      <><
    ><>
  ()<
  (<~
```

-- END OF EXAM --



#### 4. Grading Scheme

Task 0 [ 0 points if correct, -10 deducted if you don't do it]

Task 1 [20 points]

Task 2 [20 points]

Task 3 [25 points]

Task 4 [35 points]

100 points total

#### FINAL COMMENTS

**For tasks 1-4, you may introduce additional functions and variables or, in the case of task 4, additional methods. However, it is not required and will not affect your grade either way.**

See the pages below on how to submit your final exam code.

**MAKE SURE TO SELECT *final* with websubmit.**

## 5. SUBMISSIONS (EECS web-submit)

You will submit your lab using the EECS web submit.

Click on the following URL: <https://webapp.eecs.yorku.ca/submit>

### Web Submit Login


To access Web Submit:

- Use your **Passport York** account by [clicking here](#), or,
- Use your EECS account by logging in below:

EECS Username:

EECS Password:

Login



York University  
Department of Electrical Engineering and Computer Science  
Lassonde School of Engineering

**STEP 1** -- If you don't have an EECS account, click here to use Passport York (everyone has a passport York account).

If you do have an EECS account, enter here and go to **STEP 3**.

**Passport YORK**

Passport York authenticates you as a member of the York community and gives you access to a wide range of computing resources and services.

Username:

Password:

Login

☐ Click this box before logging in to change your Passport York password.

**STEP 2** – Enter your passport York username/password.

Academic Year: 2021-2022 ▼

Term: F ▼

Course: 1015A ▼

Assignment: final ▼

Submit Status: Submission  
Enabled

Feedback: None

Please specify files to submit:  
(You can submit multiple files at once!)

Choose Files	final.py
Choose Files	utilities.py
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen

Submit Files Logout

**STEP 3** – Select the correct menu option as follows. Term "F", Course "1015", Assignment "Lab9".  
Also, remember to select the correct section.

**STEP 3 cont'** – Select your file. Make sure you attached both files Lab9.py and MinMaxList.py

**STEP 3 cont'** – once you have entered everything above, click "Submit Files".

webapp.eecs.yorku.ca says

\*\*\*\*\* ATTENTION \*\*\*\*\*

You are submitting files to:

Course:\*\*\*1015  
Assignment:\*\*\*Lab1  
Academic Year:\*\*\*2020-21  
Term:\*\*\*F

Failure to submit your assignment to the proper course

OK Cancel

**STEP 4** – Confirm that you have entered everything in correctly. If you make a mistake here and submit to the wrong course, or wrong lab, we won't be able to tell and will mark your lab as not submitted. Please double check before clicking OK.

**Feedback:** None

Please specify files to submit:  
(You can submit multiple files at once!)

Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen
Choose Files	No file chosen

**Messages:**

- **final.py** submitted
- **utilities.py**

~~You have submitted these files:~~

- [final.py](#) (6 B) 09/13/2020 21:58:41
- [utilities.py](#)

**STEP 5** – After you submit, your webpage will refresh and show that you have submitted the files and the time.

I recommend you logout.

You can resubmit the file if you make changes. However, if the TA has already graded your lab, they will not grade it again, so I recommend you only upload once you have it work.

For more details on websubmit, see EECS department instructions:

<https://wiki.eecs.yorku.ca/dept/tdb/services:submit:websubmit>