

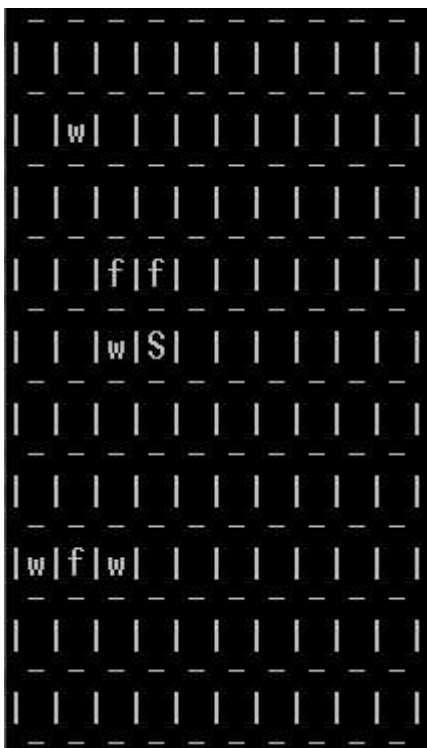
CPSC 231: Assignment 4

Due Wednesday June 21 at 5 PM

New Concepts to be applied for the assignment

- 2D lists
- Solving a more challenging and larger problem
- File output
- Exception handling: during console/keyboard input and as well as during file output

Introduction:



Write a program that simulates the life of a student in CPSC 231 (as shown in Figure 1). The simulated world will be represented by a 2D Python list of single character strings. A student 'S' can either choose to maximize his/her fun times 'f' during the term or the person can choose to expend time working 'w' in order to maximize the term grade point (GPA) and letter graded awarded. In this simulation there is not only a direct but a perfect correlation between time spent working and the grade awarded.¹ Like a semester in "real life" there is only a finite amount of time during a term and the simulation will run for a maximum of 13 turns (weeks). Each time that the user is prompted to move the student (or if the choice is given up) a time unit will be expended.

The initial starting positions will be read in from a text file whose name is specified as the program is started. The student's 'score' (fun and grade points earned along with the term letter grade) is saved to another text file 'scores.txt'. Only one set of scores are retained, each time that the simulation runs previous scores are overwritten.

Figure 1: The simulated world

To add
a level

of uncertainty and challenge to the simulation two random events may occur. "The Pavol" may manifest itself ethereally (no physical appearance) and cause time to appear to stand still for a single week (the time expended by the user's actions won't actually increment time). The Pavol will affect time in the simulation but the Pavol itself doesn't physically appear in the simulation. Also the dread "Taminator" may appear. Unlike The Pavol the Taminator will physically appear. The Taminator can sense the location of a student and it will move towards the person. At maximum speed the Taminator travels at twice the speed of a student. When the Taminator has caught a student, the passage of time will accelerate (3 time units expended instead of 1). Fortunately (and ironically) the time that a Taminator appears is limited (3 'normal' weeks which is not counting the acceleration of time that it causes). Both The Pavol and the Taminator can appear multiple times during the simulation. But only one 'event' or the other will occur: if The Pavol has already appeared then the Taminator won't appear. Or if the Taminator has appeared and hasn't yet vanished then The Pavol won't make a manifestation.

Details:

In order to get credit for some features other features must have been implemented first (e.g. to get credit for moving on top of fun and work points, the ability to move the student has to be working first). Some of the most obvious prerequisites have been listed in the description below.

Initialization

The program starts by reading the starting positions from a text file whose name will be specified by the user at run time. You can either use the functions provided in the starting program or you can write your own. **In order to get any credit for this assignment your program must be able to read the information from file.** This will not only allow your marker to more quickly evaluate the results, but will also make it easier for you to run tests.

Options for using the pre-written file input functions. There is no advantage to using one vs. another. One approach splits the task into two functions to make things easier to interpret by students while the other combines the task into one function. Both functions return a list that has been initialized from the data in the specified file.

1) **Option I**, using a single function: `createListFileRead()`. The function creates a new empty row with each iteration of the outer loop. The inner loop successively appends new data (read in from file) onto the new row.

2) **Option II**, using two functions: `initialize()` and `readFromFile()`. The first function creates the list row by row and then appends a default value for each element '!' to the end of each row. The second function performs all of the file input. Line by line the text from the input file is read in and along that line each character is used to set each element of the list.

The input file will specify the starting positions for most of the main elements in the simulation:

- 'w' represents an opportunity to work on coding a Python program.
- 'f' represent an opportunity for the student to have a fun experience.
- 'S' represents the starting position of the student.
- Spaces represent empty parts of the simulation - since your life is just about school and fun everything else is just a void right? ;)

You can assume that the Taminator 'T' will never be read in from an input file it has to be randomly generated as the game runs (or manually invoked by the user). And as mentioned The Pavol doesn't actually take physical form so it also never appears in the input file.

Another requirement to get any credit for your assignment is that the display of the grid must be done using the `display()` function provided or you must write an equivalent function yourself. This grid is required because it makes the display of world easier to read and interpret for both you and the marker.

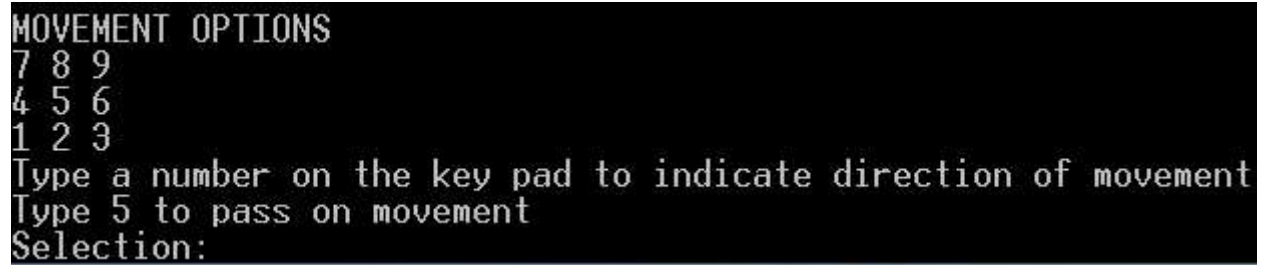
Running the simulation:

The simulation runs for 13 weeks (turns). A turn is broken into parts or 'sub-turns' which occur in the following order:

- Prompt the player for movement and determine the effects of movement
- Determine if The Pavol appears (and account for time appearing to stand still)
- Determine if the Taminator appears

Prompting for movement:

The player can move to any adjacent location that is not currently occupied by the Taminator. The movement options will be represented by a 9 point directional computer keypad grid (Figure 2).



```

MOVEMENT OPTIONS
7 8 9
4 5 6
1 2 3
Type a number on the key pad to indicate direction of movement
Type 5 to pass on movement
Selection:
  
```

Figure 2: Movement menu

Each direction matches a cardinal compass direction (N=8, W=4, E=6, S = 2) or inter-cardinal compass direction (NW=7, SW=1, NE=9, SE=3). Selecting 5 doesn't move the student but still causes a turn to pass. **Unlike previous assignments:** Your program should *not produce a run time error* if an invalid input type is entered (i.e. a string instead of a number). Instead exception handling should be employed which displays a useful error message and continues prompting until a valid value has been received. The program should also continually re-prompt if a value of less than zero or greater than nine has been entered. Entering a value from 1 - 9 (save for 5) will result in the student moving in that direction and a turn being expended. If an attempt is made to move the student outside the bounds of the simulation or onto the Taminator then another useful error message should appear. Time will not pass if an invalid value or if an attempt is made to move the student to an invalid location (outside or Taminator occupied). Entering a zero will result in the secret cheat menu appearing:

```

Cheat menu options
(t)oggle debug mode off
(m)ake the Taminator appear
(p)avol manifests itself
(q)uit cheat menu
  
```

The user will be repeatedly prompted for a selection so long as an option other than one of the ones above have been selected. Toggle debug mode flips the state of the global debug flag between True and False. When the flag is set to True, debugging messages will appear. Their exact content is up to you but generally the message should be visually distinct (e.g. capitals, surrounded with angled brackets, indented etc) and specify the function that where the debugging message appears e.g. <<move() Taminator original location (4,5)>>. Selecting the option to make the Taminator appear will allow the user to select a (row, column) location. A location outside the list or one that one is currently occupied by the student will result in a re-prompting until a valid location is entered. You are not required to type check the Taminator location as you did with the direction of movement (moving the student). Once a valid location has been entered, the Taminator will start moving towards the student (more information on the [Taminator event](#)). Manifesting the Pavol will result in time suspending itself. Trying to get the Taminator or Pavol to appear via the cheat menu when the Taminator has already appeared should result in an error message and the program prompting the user for a cheat menu option again. (Since the Pavol only appears for a turn and [the check for the Pavol's appearance occurs after the player's turn](#) it won't be possible to try to invoke either the Pavol or Taminator when the Pavol has appeared).

Selecting a valid cheat menu option (even the option to quit) will result in a time unit being expended as if the user had selected a valid movement option. **Entering an invalid option will not cause time to advance** (again as was the case with the movement menu).

Moving the student onto the special objects: as mentioned the student cannot be moved outside of the simulation nor can the student being moved onto a location occupied by the Taminator. (No one can 'grapple' the Taminator!) Moving onto the a location with an opportunity for fun 'f' will increase the "fun points" by 1. Moving onto the a location with an opportunity for work 'w' will increase the grade point by 1. In order to get credit for these features, the program must actually allow the user to move the student. The work or fun opportunity is expended by the student and doesn't reappear in the location after the student moves. The current number of fun points and grade points (but not letter grade) awarded is displayed at the start of each turn:

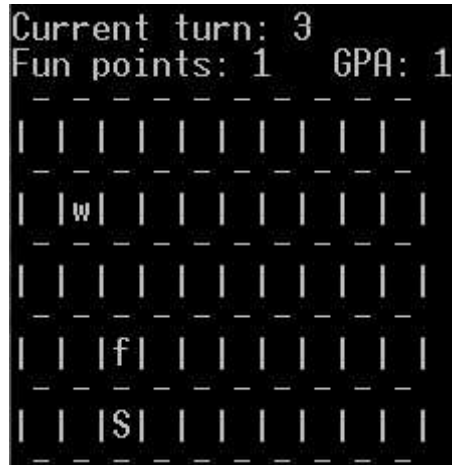


Figure 3: display of fun and grade points

After moving the student, the simulation will check for the possible occurrence of two events in the following order:

Running The Pavol event (you will only receive credit for this feature if the appropriate option in the cheat menu has been fully and correctly implemented and time must be properly tracked in order to get credit for this feature):

There's a 10% chance of this event occurring during a particular turn. This event only last for a single turn and it won't occur if the Taminator event is already in progress. The Pavol doesn't actually physically appear but it only affects the passage of time. The student gets an additional turn (normally the game will advance by one turn but the appearance of The Pavol causes time to roll back by one. So the appearance of The Pavol and time moving forward will cancel each other out...["time appears to stand still when you are lucky enough to be in the positive presence of "The Pavol."](#)

Before The Pavol appears (turn 1):

```

Name of input file: input.txt
Current turn: 1
Fun points: 0   GPA: 0

- - - - -
| | | | | | | | |
- - - - -
| |w| | | | | | |
- - - - -
| | | | | | | | |
- - - - -
| | |f|f| | | | |
- - - - -
| | |w|S| | | | |
- - - - -
| | | | | | | | |
- - - - -
| | | | | | | | |
- - - - -
|w|f|w| | | | | |
- - - - -
| | | | | | | | |
- - - - -
| | | | | | | | |
- - - - -

MOVEMENT OPTIONS
7 8 9
4 5 6
1 2 3
Type a number on the key pad to indicate direction of movement
Type 5 to pass on movement
Selection: 5
The Pavol manefests itself and bathes you in
pure positiveness...Sunshine, Lollipops and
Rainbows :):D:P...

```

Figure 4: The 'turn' when The Pavol makes a manifestation

After the Pavol appears (the student collected a fun point by moving NW and consuming some 'fun' but it's still turn 1)

```

Current turn: 1
Fun points: 1   GPA: 0

- - - - -
| | | | | | | | |
- - - - -
| |w| | | | | | |
- - - - -
| | | | | | | | |
- - - - -
| | |S|f| | | | |
- - - - -
| | | | | | | | |
- - - - -
| |w| | | | | | |
- - - - -

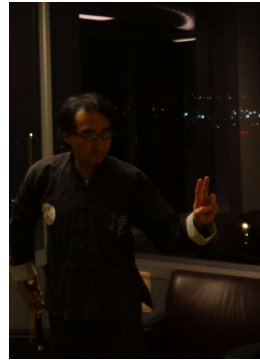
```

Figure 5: The turn after The Pavol has departed

The next turn The Pavol will disappear and time resumes it's normal course (unless the Pavol appears during the next turn as well). The Pavol can appear multiple times during a simulation but as mentioned it won't appear at the same time as the Taminator.

Running the Taminator event (you will only receive credit for this feature if the appropriate option in the cheat menu has been fully and correctly implemented, time must be correctly tracked in the simulation).

If the Pavol hasn't appeared during a turn, the simulation checks against a 25% probability that the Taminator will appear. If that is the case the Taminator will randomly appear at a location that is not currently occupied by the student. If it materializes on top of a work or fun opportunity both are destroyed by the Taminator (the object won't reappear after the Taminator has moved). Immediately after appearing the Taminator will start moving towards the student at a maximum speed of 2 rows and 2 columns. The Taminator will continue moving towards the student until either it's adjacent to the student or it's current time has come to an end.



The Taminator will only move to an adjacent position to the student. It won't move past the student by moving 2 full units (i.e. if it's already close to the student) nor will it move onto the location that is currently occupied by the student. If the Taminator is already in "close range:" the student it will slow down to 1 movement unit. And the Taminator is 'smart' enough so it can move faster along the row or column (only) if the distance is greater along that dimension only (see Figure 8). The Taminator will appear for 3 turns (includes the turn in which the Taminator first appears) and then it will vanish. Whenever the Taminator has caught up to the student (adjacent) the program must advanced time by an additional 2 turns (plus the normal passage of 1 turn = 3 turns passing for each turn that the Taminator is adjacent). It's your choice if time stops at 13 turns or goes beyond that but the simulation must stop at 13 (or more) weeks. While the Taminator is moving it becomes mystically incorporeal like The Pavol. Consequently any work and fun opportunities in it's path the student won't be destroyed. However once it stops moving it becomes solid again and a work or fun opportunity at the destination (where it stops) is destroyed.

The Taminator is by far the most complicated program feature so a more detailed explanation is provided. In this example the Taminator is invoked during Turn 1 via the cheat menu (Figure 6). During Turn 1 the program counts off 1 time unit out of the three units that the Taminator will appear. Also notice that invoking of the cheat menu causes one time unit to pass.

```

Current turn: 1
Fun points: 0   GPA: 0

- - - - -
| | | | | | | | |
- - - - -
| |w| | | | | | |
- - - - -
| | | | | | | | |
- - - - -
| | |f|f| | | | |
- - - - -
| | |w|S| | | | |
- - - - -
| | | | | | | | |
- - - - -
| | | | | | | | |
- - - - -
|w|f|w| | | | | |
- - - - -
| | | | | | | | |
- - - - -
| | | | | | | | |
- - - - -

MOVEMENT OPTIONS
7 8 9
4 5 6
1 2 3
Type a number on the key pad to indicate direction of movement
Type 5 to pass on movement
Selection: 0
Cheat menu options
(t)oggle debug mode on
(m)ake the Taminator appear
(q)uit cheat menu
Cheat menu selection: m
Enter row 0-9: 0
Enter row 0-9: 0
<<< Beware! The TAMINATOR is here! >>>

- - - - -
|T| | | | | | | | <== Initial Taminator location
- - - - -
| |w| | | | | | |
- - - - -
| | | | | | | | |
- - - - -
| | |f|f| | | | |
- - - - -
| | |w|S| | | | |
- - - - -
| | | | | | | | |
- - - - -
| | | | | | | | |
- - - - -
|w|f|w| | | | | |
- - - - -
| | | | | | | | |
- - - - -
| | | | | | | | |
- - - - -

```

Figure 6: Initial appearance of the Taminator

The next turn shows the Taminator has moved 2 rows and 2 columns towards the student during the last turn. (The Taminator reacts quickly and can immediately start moving). The movement was

shown to the user only after the user selected a movement option for the student. Only one time unit has passed in the game because the student has not yet been caught. The program also counts that the Taminator has appeared for 2 turns now. Finally notice that the work opportunity (1,1) that was in direct path (but not the final destination) from the Taminator to the student still remains intact (Location (1,1) in Figure 7).

```

Current turn: 2      <== 1 time unit passes
Fun points: 0   GPA: 0
-----
| | | | | | | | | |
-----
| |w| | | | | | | |
-----
| | |T| | | | | | | <== TAMINATOR immediately moved
-----
| | |f|f| | | | | | |
-----
| | |w|S| | | | | | |
-----

```

Figure 7: The Taminator first appears

The Taminator moves only 1 row the next turn because this is all the distance that is needed in order to enter an adjacent position to the student. (It could move right by 1 column and still be adjacent but the Taminator is efficient in its movement). Now that the student has been caught time advances by 3 time units this turn. Notice that the fun opportunity that was located at the new destination of the Taminator (3,2) is now destroyed (Figure 8).

```

The Taminator has caught up to you!
Taminator speaks: Extensions for everyone!...NOT!!!!
Current turn: 5 <== Student caught. Time advances by +2 more weeks
Fun points: 0   GPA: 0
-----
| | | | | | | | | |
-----
| |w| | | | | | | |
-----
| | | | | | | | | |
-----
| | | | | | | | | |
-----
| | |T|f| | | | | | <= TAMINATOR MOVES 1 row only. Student adjacent
-----
| | |w|S| | | | | | |
-----

```

Figure 8: The Taminator catches the student

The Taminator has made it's appearance for 3 time units and will vanish during the next turn (shown after the student moves). Notice also that fun opportunity at (3,2) has not re-appeared (Figure 9).


```

MOVEMENT OPTIONS
7 8 9
4 5 6
1 2 3
Type a number on the key pad to indicate direction of movement
Type 5 to pass on movement
Selection: 3
The time of the Taminator is over!...for now!!!

Current turn: 6 <== TAMINATOR gone, normal passage of time
Fun points: 0   GPA: 0

- - - - -
| | | | | | | | |
- - - - -
| |w| | | | | | |
- - - - -
| | | | | | | | |
- - - - -
| | | |f| | | | |
- - - - -
| | |w| | | | | |
- - - - -
| | | |S| | | | |
- - - - -
| | | | | | | | |
- - - - -
|w|f|w| | | | | |
- - - - -
| | | | | | | | |
- - - - -
| | | | | | | | |
- - - - -

```

Figure 9: The time of the Taminator has passed

There can only be one Taminator appearing at a time. If the program already randomly generated a Taminator, then the cheat menu won't allow the player to invoke another. As mentioned a suitable error message should be displayed instead. (An error message should also appear if the user manually invoked the Taminator and tries to invoke it again before it has vanished). Time will pass even though the Taminator couldn't be invoked (actually it couldn't be re-invoked) because the cheat menu was summoned. In a similar fashion the program shouldn't check for a new random Taminator event if the Taminator is still making it's appearance whether it was generated randomly or by the user via the cheat menu.

End of the simulation:

When the game is over the game (all weeks have elapsed naturally or in accelerated fashion due to the appearance of the Taminator) the usual statistics will be displayed onscreen (but also the letter grade should be displayed since the final result can be determined). That information should be saved to an output file, in this case the same name 'stats.txt' will be used each time to make it consistent and easier for your marker. Exception handling should be employed to deal with any file output problems during the save. Specific formatting isn't required for the scores file but the information should be presented in a reasonably neat and legible fashion. (Use format specifiers or escape codes as needed). Mapping of grade points to a letter is largely as expected: 0 grade point = F, 1 grade point = D, 2 grade points = C, 3 grade points = B, 4 or more grade points = A.

D2L configuration:

- Multiple submissions are possible for each assignment: You can and should submit many times before the due date. *For this assignment D2L will keep older versions of your submissions.*

- **Important!** Multiple files can be submitted for each assignment. I am allowing you to submit multiple files for each assignment so **don't archive/compress multiple files using a utility such as zip**. However, this means that **TAs will only mark the latest versions** of each file submitted via D2L. Even if the version of a document that you want marked has been uploaded into D2L if it isn't the latest version then you will only get marks for the latest version. (It's unfair to have the TAs check versions or to remark assignments because marking is enough work as-is).

Marking

- Assignments will be marked by your tutorial instructor (the "Teaching Assistant" or "TA"). When you have questions about marking this is the first person that you should be directing your questions towards. If you still have question after you have talked to your TA, then you can talk to your course (lecture) instructor.
- As well as being marked on whether "your program works" you will also be marked on non-functional requirements such as style and documentation. Consequently this assignment will include a separate [\[marking checklist\]](#). Besides seeing your grade point in D2L you can also see the detailed feedback that your TA will enter for each student. You can access the grading sheet in D2L under **Assessments->Dropbox** and then clicking on the appropriate assignment link. If you still cannot find the grading sheet then here is a [\[help link\]](#)

Points to keep in mind:

1. **Due time:** All assignments are due at 5 **PM** on the [due dates](#) listed on the course web page. Late assignments or components of assignments will not be accepted for marking without approval for an extension beforehand. Alternate submission mechanisms (non exhaustive list of examples: email, uploads to cloud-based systems such as Google drive, time-stamps, TA memories) cannot be used as alternatives if you have forgotten to submit work or otherwise have not properly submitted into D2L. **Only files submitted into D2L by the due date is what will be marked**, everything else will be awarded no credit.
2. **Extensions** may be granted for reasonable cases by the course instructor with the receipt of the appropriate documentation (e.g., a doctor's note). Typical examples of reasonable cases for an extension include: illness or a death in the family. Example cases where extensions will not be granted include situations that are typical of student life: having multiple due dates, work commitments etc. Tutorial instructors (TA's) will not be able to provide extension on their own and must receive permission from the course instructor first.
3. **Method of submission:** You are to submit your assignment using D2L [\[help link\]](#). Make sure that you [\[check the contents of your submitted files\]](#) (e.g., is the file okay or was it corrupted, is it the correct version etc.). It's your responsibility to do this! (Make sure that you submit your assignment with enough time before it comes due for you to do a check).
4. **Identifying information:** All assignments should include contact information (full name, student ID number and tutorial section) at the very top of your program in the class where the 'main()' method resides (starting execution point). (Note other documentation is also required for most assignments).
5. **Collaboration:** Assignments must reflect individual work; group work is not allowed in this class nor can you copy the work of others. For more detailed information as to what constitutes academic misconduct (i.e., cheating) for this course please read the following [\[link\]](#).
6. **Execution:** programs must run on the computer science network running Python 3.x. If you write you code in the lab and work remotely using a remote login program such as Putty or SSH. If you choose to install Python on your own computer then it is your responsibility to ensure that your program will run properly here. It's not recommended that you use an IDE for writing your programs but if you use one then make sure that you submit your program in the form of text ".py" file or files.
7. **Use of pre-created Python libraries:** unless otherwise told you are to write the code yourself and not use any pre-created functions. For this assignment the usual acceptable functions include: `print()`, `input()` and the 'conversion' functions such as `int()`, `float()`, `str()`. Look at the particular assignment

description for a list of other classes that you are allowed to use and still get credit in an assignment submission.

Assignment resources:

- You can find sample input files (you are highly encouraged to make your own as well) under:
/home/231/assignments/assignment4/inputs [[Shortcut](#)]
- The starting code for the assignment can be found under:
/home/231/assignments/assignment4/code [[Shortcut](#)]