Intro2cs Ex4 – Pensioners

Submission deadline

Thursday,14 November, 20:55

Objectives

In this exercise you will practice the use of basic algorithms as binary search and bubble sort. You will also have a deeper understanding of functions and containers.

Ex3
Ex3 is also online. you must complete Ex3 before start working on this exercise. Some of the skills needed for Ex4 will be discussed in the "tirgulim" of next week. But, after completing ex3, you may choose to start working on this exercise before them.

Being a Pensioner
Ex3 dealt with the issue of "planning for the future". On Ex4, you should find your time machine in the garage since you are becoming a pensioner.

You may use your own code from Ex3 (and only yours) in this exercise, but it must be copy pasted into your code (i.e. you are not allowed to import it)

We will suppose you want to "live like a king" until your death. You don't have any children and you want to make the most out of your retirement – i.e. you want to maximize your expenses without entering into debts. We will assume constant expenses on each year. How may we calculate it? One option would be to try feeding different values of expenses to post_retirement() – and find the best value (i.e. maximum expenses such that the last value in the returned list will still be a positive)
Remember that in class we saw the idea of binary search. We can use this idea also here to find the correct value. In this assignment you may choose to use also pure arithmetics for this calculation.

Assignment 1
Implement `live_like_a_king`, which takes five arguments: a `salary`, a percentage of your salary to save (`save`), a list of annual growth rate before retirement (`pre_retire_growth_rates`), a list for after the retirement (`post_retire_growth_rates`), and a value for stop searching (`epsilon`). As before the length of the `pre_retire_growth_rates` will be the amount of working years and `post_retire_growth_rates` will account for the number of years you live after retirement. Find the maximal expenses you can withdraw each year from your retirement fund, such that by your

death, you will have a positive amount in your bank. You may use binary search in order to find the maximal expenses. In order not to continue forever you should stop when your expenses result in a positive value which is smaller than epsilon. **Special 5 points bonus** will be given to those of you who instead of using binary search will use arithmetics and will return a value that will leave your account with **0** in the end of your life. Specify in your README which method did you use.
Your function should return the maximal expenses value you found.
Complete:

```
live_like_a_king(salary, save, pre_retire_growth_rates,
                post_retire_growth_rates, epsilon)
```

Write your code in the appropriate place in the ex4.py template. To test your function, compare it with the school solution.

Assignment 2

Implement `bubble_sort_2nd_value`. In this assignment, you will implement a function that receives a list of tuples, where each tuple is composed of a string value and a float value. The function will return a **NEW** list that is sorted by the 2nd value of the tuple, the numerical one. The sorting direction should be from the lowest to the largest. sort should be stable (if values are equal, use original order).
We've learned in class "bubble sort", use this algorithm to sort the list. you cannot use any given function by python (as sort()) or any different algorithm.
Complete: `bubble_sort_2nd_value(tuple_list)`
Write your code in the appropriate place in the ex4.py

Bingo! Bingo is an important activity.
To enjoy the rest of your life, one should choose the best retirement house he can afford. In the next assignment you'll search for the most expensive retirement house, such that you won't finish your savings. In this assignment you must use binary search.

Assignment 3

Implement *choosing_retirement_home,* a function with 3 arguments: an initial amount on your account, `savings`; a list of annual growth percentages on investments while you are retired (`growth_rates`), and a list of tuples of `retirement_houses`, where the first value is a string - the name of the house and the second is the annual rent of it. The function should return the name of most expensive house one can afford, without getting into debts.
Implement the function using binary search. Note that the list of tuples won't be sorted according to their prices. First, sort the list of houses by their prices and then run a binary search, to find your house.
complete *choosing_retirement_home(savings,*`growth_rates,retirement_houses)`.
The function should return a string - the name of the chosen retirement house.
Write your code in the appropriate place in the ex4.py template.
Theoretical and practical important CS comment: Linear search is always faster than first sorting and then performing a binary search. (Even if the search is efficient, which bubble sort isn't). We ask you for this implementation only as an exercise.

While retired, you will need some leisure activity.
You must write some functions for a Battleship game.
See rules here: http://boardgames.about.com/od/battleship/a/Rules-of-Battleship.htm
The following assignment will be written in a file named battleship.py

---

Assignment 4a

Implement `new_board`, a function with 2 arguments: The `width` of the board and the `height` of the board. In case of no height argument, the default board is a square. The default value for width is 10 (new_board() is 10x10, new_board(8) is 8x8, new_board(8,12) is rectangular). Each row of the board is represented by a list, (left to right,) and the board is a list of lists of rows, (top to bottom). The top left cell is indexed as [0][0], and the bottom right cell is indexed as [height-1][width-1]. As the board starts empty, each of the inner arrays is a list of 'None's.
Return an empty grid of the right size.

Assignment 4b:
Implement `place_ship`, a function with 4 arguments: The `board` in which to place the ships, the length of the ship, `ship_length`, a tuple (x,y) representing the `bow` of the ship, and a tuple (dx,dy) representing the direction the ship is facing - `ship_direction`.
Remember that the cell (x,y) can be accessed as `board`[y][x]. Remember that if a ship is facing north, the rest of the ship is to the south of the bow. A ship may not go over the edges of the grid, and may not occupy an already occupied space. How a ship is represented in the grid will be described later. each ship will have a unique index - described below.
Returns the index of the placed ship, if the placement was successful, and 'None' otherwise. If ship placement was not successful, the board should not have been modified.

Assignment 4c:
Implement `fire`, a function with 2 arguments: The `board` to fire on, and a tuple (x,y) representing the `target` to fire on. If the target is an illegal target, return None.
Otherwise returns a tuple (hit,ship), where hit is True/False depending if the the shot hit, and ship is the index of the ship which was completely destroyed, or 0 if no ship was completely destroyed.

Each cell in the grid contains one of the following:
1. None, if the cell has no ship.
2. A tuple (index,part,[size]), where index is the index of the ship, part is a number from 0 at the bow towards length-1 aft (backwards towards the stern) and [size] is a length 1 list, containing the number of the remaining parts of the ship.

When successfully placing a ship, do the following: 1. Find the maximum ship index in the grid 'm'. The new ship's index will be 'm+1'. (The first ship placed will have index of 1.) 2. The part number is 0 for the bow (front of the ship), length-1 for the stern (back). 3. The starting size of the ship is its length. 4. In each cell occupied by the ship, place a tuple
containing the above information. Remember that size is stored in a list. Why is that necessary?

When successfully hitting a ship, do the following: 1. Decrease the size of the ship. If the size becomes 0, the ship is destroyed, and its index will have to be returned. You may not search the grid in order to update the ship size.
2. Remove the ship from the relevant cell, i.e., set the cell to None. Note that this means that shooting a cell a second time, after it is hit, will be considered a miss. It also means that if a ship added in middle of a game can cross a partially destroyed ship.

Back in ex4.py, we will try to choose our retirement home by different measurement. We will care for two things - we must make sure the house is affordable, but the value of the house will be given by a formula depends both on its rent and the number of battleship opponents in it.

Assignment 5a:
Implement `get_value_key`, a function with 1 argument: the `value` added per opponent (default value = 0),
Returns a key function which accepts a triple containing (name of the house ,annual rent, number of opponents) and returns keys such that 'sorted(home_list, key=get_value_key(n))' will return a new sorted list from low to high according to this ranking. you are asked to **return a function** that given a container of size 3 will return the new value of the house - the rent plus the value from battleship opponents.

Assignment 5b:
Implement `choose_retirement_home_opponent`s, a function with 3 arguments: a `budget`, a function of the type returned by get_value_key - `key`, and a list of tuples containing information on `retirement_houses`, where the first value is a string - the name of the house, the second is the annual rent on it, and the third is the number of battleship opponents the home hosts.
Returns the name of the retirement home which provides the best value and which is affordable. Note that the key may not have come from get_value_key, and might not be linear.
Note that the school solution demo receives a value instead of a function in the 2ns parameter. Your code is supposed to work like this:
f2=get_value_key(3)
choose_retirement_home_opponents(budget,f2,retirement_houses)

Editing the definitions of functions.
In few assignments you have arguments that receive a default value - You should edit the function definitions in the template file such that they will specify the requested default values. Do not change anything else in the definition. example
template: def get_value_key(value):
Submitted ex: def get_value_key(value=0)

Dealing with errors and bad inputs
In this exercise, as in Ex3 you cannot assume that the input you've received from the user is correct. You may assume that the type of each argument is correct but not its value. In the ex4.py and in battleship.py you will find specific details how to deal with errors, in each function, but generally speaking in case of bad input (wrong values) you should print nothing and return **None**.

Template files

Template -ex4.py: http://moodle.cs.huji.ac.il/cs13/file.php/67101/ex4/ex4.py
Battleship template: http://moodle.cs.huji.ac.il/cs13/file.php/67101/ex4/battleship.py

School Solution

can be found in ~intro2cs/bin/ex4/ex4 or in ~intro2cs/bin/ex4/battleship. It uploads a menu that is not demanded by your code. Your code only need to implement the functions. Note that also the print board function is not part of the exercise. To enter a list write its length followed by values(L = [(h1,1),(h2,3),(h3,4)] -> 3 h1 1 h2 3 h3 4).
To test it on your own code, you can use the import function as we have learned in class.

Submission and Further Guidelines:

Submission:

1. Implement assignments 1,2,3,5 in the ex4.py file and assignment 4 in battleship.py
2. Create a tar file containing ex4.py, battleship.py and the README
3. Submit the file ex4.tar via the "Upload File" link on the course home page, under the ex4 link.
4. Make sure you pass the automatic tests

Running the tests yourself

- There are two ways you can run these testers and see the results without submitting your exercise and getting the pdf:
  - The first – place your tar file in an empty directory, go to that directory using the shell and type: `~intro2cs/bin/testers/ex4 ex4.tar`
  - The second – download a file -ex4testing.tar.bz2. Extract the contents (using tar –xjv ex4testing.tar.bz2) of the file and follow the instructions on the file named TESTING