Contents

1 Basic Test Results

```
Starting tests...
Thu Nov  7 15:01:28 IST 2013
fdd2b0559c15655f35a509d8ab8962ca4e766b0e  -


ex3.py
README

Testing README...
Done testing README...

Testing ex3.py...
result_code    badinputs    50    1
result_code    growth    10    1
result_code    inflation    10    1
result_code    variable    10    1
result_code    choose    5    1
result_code    constant    10    1
result_code    post    10    1
Done testing ex3.py

Grading summary
---------------
****** constant:
Number of failed tests: 0
Total number of tests : 10
Penalty: 0.0
****** variable:
Number of failed tests: 0
Total number of tests : 10
Penalty: 0.0
****** choose:
Number of failed tests: 0
Total number of tests : 5
Penalty: 0.0
****** growth:
Number of failed tests: 0
Total number of tests : 10
Penalty: 0.0
****** inflation:
Number of failed tests: 0
Total number of tests : 10
Penalty: 0.0
****** post:
Number of failed tests: 0
Total number of tests : 10
Penalty: 0.0
****** badinputs:
Number of failed tests: 0
Total number of tests : 50
Penalty: 0.0
******
Expected automatic grade: 100.0
******
Submission passed!
Tests completed
```

2 aaa expected autograde

```
Grading summary
---------------
****** constant:
Number of failed tests: 0
Total number of tests : 10
Penalty: 0.0
****** variable:
Number of failed tests: 0
Total number of tests : 10
Penalty: 0.0
****** choose:
Number of failed tests: 0
Total number of tests : 5
Penalty: 0.0
****** growth:
Number of failed tests: 0
Total number of tests : 10
Penalty: 0.0
****** inflation:
Number of failed tests: 0
Total number of tests : 10
Penalty: 0.0
****** post:
Number of failed tests: 0
Total number of tests : 10
Penalty: 0.0
****** badinputs:
Number of failed tests: 0
Total number of tests : 50
Penalty: 0.0
******
Expected automatic grade: 100.0
******
Submission passed!
```

3 aaa hint result.png

4 README

```
roigreenberg
305571234
roi greenberg

================================
=  README for ex3: Retirement  =
================================



==================
=  Description:  =
==================

The function calculate the pension in given amount of year with constant rate (task 1)
or variable rates (task 2).
Also it can find the best fund for your pension.(task 3)
In task 4 you can check the growth rate in a spesific year(4.1) or update the list of
the growth rates after the market inflation(4.2).
In the last task(5) you can calculate your retirement saving after your yearly withdraw.



============================
=  List of submitted files: =
============================

README        This file
ex3.py        pension, growth rates and retirement caculations

======================
=  Special Comments  =
======================
```

5 ex3.py

```
1  ##############################################################
2  # FILE: ex3.py
3  # WRITER: Roi Greenberg + roigreenberg + 305571234
4  # EXERCISE : intro2cs ex3 2013-2014
5  # Description: pension, growth rates and retirement
6  # caculations.
7  ##############################################################
8
9  # calculate pension with constant growth rate
10 def constant_pension(salary, save, growth_rate, years):
11     # verifies the input
12     if salary < 0 or growth_rate < -100 or years < 0\
13        or save < 0 or save > 100:
14         return
15
16     if years == 0:  # return empty list if years is 0
17         return []
18     # calculate the pention
19     pension = [salary * save * 0.01]
20     # run for the length of the years given
21     for i in range(1, years):
22         pension.append(pension[i - 1]*(1 + growth_rate*0.01)\
23                        + salary*save*0.01)
24     return pension  # return list of the pension value for each year
25
26 # calculate pension with variable growth rates
27 def variable_pension(salary, save, growth_rates):
28     # verifies the input
29     for i in growth_rates:
30         if float(i) < -100:
31             return
32     if salary < 0 or save < 0 or save > 100:
33         return
34
35     if len(growth_rates) == 0: # return empty list if no growth rates given
36         return []
37
38     # calculate the pention
39     pension = [salary * save * 0.01]
40
41     # run for the length of the growth rates list
42     for i in range(1, len(growth_rates)):
43         pension.append(pension[i - 1]*(1 + float(growth_rates[i])*0.01)\
44                        + salary*save*0.01)
45
46     return pension  # return list of the pension value for each year
47
48 # finding the fund with the best pention
49 def choose_best_fund(salary,save,funds_file):
50     if salary < 0 or save < 0 or save > 100: # verifies the input
51         return
52
53     file = open(funds_file)  # open the given file
54     funds = file.readlines()  # create list of the file lines
55     for i in range(len(funds)):  # divide the Fund name from the growth rates
56         funds[i] = funds[i].split(",", 1)
57     if len(funds[0]) == 1:  # return empty list in no growth rates given
58         return []
59
```

try to avoid the using of magic numbers. in the next ex's point will be reduced.

```
60      # Create list from the growth rates and strip the fund name and the last
61      # growth rate from unnessary chars ('#' "\n")
62      for i in range(len(funds)):
63          funds[i][1] = funds[i][1].strip("\n").split(",")
64          funds[i][0] = funds[i][0].strip("#")
65
66      # create new list of every fund and it value in the last year
67      funds_compare = []
68
69      # Calculate the pension values the takes only the name and the last value
70      # and put the into the new list
71      for i in range(len(funds)):
72          pension_values = variable_pension(salary,save,funds[i][1])
73          fund_last_value = [funds[i][0], pension_values[len(funds[i][1]) - 1]]
74          funds_compare.append(fund_last_value)
75
76      # reorder the funds from the highest value to the lowest
77      for i in range(len(funds)-1):
78          for  j in range (i,len(funds)):
79              if funds_compare[i][1] < funds_compare[j][1]:
80                  funds_compare[i], funds_compare[j] =\
81                  funds_compare[j], funds_compare[i]
82      file.close()
83
84      return tuple(funds_compare[0]) # return a tuple with the best fund
85
86  # find the growth rate in a given yaer
87  def growth_in_year(growth_rates,year):
88      # verifies the input
89      if len(growth_rates) == 0 or year < 0:
90          return
91      for i in growth_rates:
92          if float(i) < -100:
93              return
94      if len(growth_rates) <= year:
95          return
96
97
98      return growth_rates[year]  # return the growth rate in the given yaer
99
100
101 # update the growth rates list with inflation value
102 def inflation_growth_rates(growth_rates,inflation_factors):
103     # verifies the input
104     if len(inflation_factors) == 0:
105         return growth_rates
106     for i in growth_rates:
107         if float(i) < -100:
108             return
109     for i in inflation_factors:
110         if float(i) <= -100:
111             return
112
113     # return empty list in no growth rates given
114     if len(growth_rates) == 0:
115         return []
116
117     update_rates = []  # create new list of the update rates
118
119     # run for the shortest list and calculate the update rates
120     for i in range(min(len(growth_rates),len(inflation_factors))):
121             update_rates.append(100*((100+float(growth_rates[i]))/\
122                                      (100+float(inflation_factors[i]))-1))
123
124     # added the rates from the original rates that didn't update
125     if len(growth_rates)>len(inflation_factors):
126         for i in range(len(growth_rates)-len(inflation_factors)+1,\
127                        len(growth_rates)+1):
```

there are more efficient ways to short an array. and its not really needed hear...

you weren't ask for checking the legality of the given list. its make this method much less efficient -O(N) instead of O(1).

```
128             update_rates.append(float(growth_rates[i-1]))
129
130     return update_rates  # return list of the updated rates
131
132 # calculate the retirement savings
133 def post_retirement(savings, growth_rates, expenses):
134     # verifies the input
135     if len(growth_rates) == 0:
136         return []
137     for i in growth_rates:
138         if float(i) < -100:
139             return
140     if savings <= 0 or expenses < 0:
141         return
142
143     # Calculate the retirement saving
144     # and put the into the new list
145     retirement = [savings*(1+float(growth_rates[0])*0.01)-expenses]
146     for i in range(1, len(growth_rates)):
147         retirement.append(retirement[i-1]*(1+float(growth_rates[i])\
148                                            *0.01)-expenses)
149
150     return retirement  # return list of the retirement saving in each year
```

Good job !