C200 Programming Assignment №6

Professor M.M. Dalkilic
Computer Science
School of Informatics, Computing, and Engineering

Indiana University, Bloomington, IN, USA

March 27, 2023

Instructions

The submission deadline is **Sunday, April 2, 2023 at 12:59 AM**.

We are comparing the code submitted by different groups. If the submissions made by different groups are similar, we will give a zero in the first instance. In the second instance, we will likely take more drastic actions.

You must submit to the Autograder (`c200.luddy.indiana.edu`) and also push to GitHub before the deadline.

1. Make sure that you are **following the instructions** in the PDF, especially the format of output returned by the functions. For example, if a function is expected to return a numerical value, then make sure that a numerical value is returned (not a list or a dictionary). Similarly, if a list is expected to be returned then return a list (not a tuple, set or dictionary).
2. Test **debug** the code well (syntax, logical and implementation errors), before submitting to the Autograder. These errors can be easily fixed by running the code in VSC and watching for unexpected behavior such as, program failing with syntax error or not returning correct output.
3. Make sure that the **code does not have infinite loop (that never exits) or an endless recursion (that never completes)** before submitting to the Autograder. You can easily check for this by running in VSC and watching for program output, if it terminates timely or not.
4. Given that you already tried points 1-3, if you see that Autograder does not do anything (after you press 'submit') and waited for a while (30 seconds to 50 seconds), try refreshing the page or using a different browser.
5. Once you are done testing your code, comment out the tests i.e. the code under the __name__ == "__main__" section.

Problem 1: Recursion on Numbers

In this problem, you'll implement the following recursive functions. Look back at Lab 6 to review similar recursion problems.

$$
\begin{aligned}
p(0) &= 10000 && (1)\\
p(n) &= p(n-1) + 0.02p(n-1) && (2)
\end{aligned}
$$

$$
\begin{aligned}
c(1) &= 9 && (3)\\
c(n) &= 9c(n-1) + 10^{n-1} - c(n-1) && (4)
\end{aligned}
$$

$$
\begin{aligned}
d(0) &= 1 && (5)\\
d(n) &= 3d(n-1) + 1 && (6)
\end{aligned}
$$

$$
\begin{aligned}
f(0) &= 1 && (7)\\
f(1) &= 1 && (8)\\
f(n) &= f(n-1) + f(n-2) && (9)
\end{aligned}
$$

$$
\begin{aligned}
e(1) &= 12 && (10)\\
e(n) &= -5e(n-1) && (11)
\end{aligned}
$$

$$
\begin{aligned}
M(0,i) &= 1 \; when \; c = 0 && (12)\\
M(c,i) &= \begin{cases} 0 & c < 0 \;\vee\; f(i) > 10 \\ M(c - f(c), i+1) + M(c, i+1) & otherwise \end{cases}\\
\binom{n}{k} &= \begin{cases} 1 & k = 0 \;\vee\; n = k \\ \binom{n-1}{k-1} + \binom{n-1}{k} & otherwise \end{cases} && (13)
\end{aligned}
$$

Programming Problem 1: Recursion on Numbers

- Complete each of the functions.
- Observe that M requires f
- Note that $\binom{n}{k}$ indicates choosing k from n and in the starter code it is denoted by the function c(n, k).

Problem 2: Maximal Value

In this problem, you're given a list of numbers. The function should return a list of three values (in this same order) namely, the interval start, interval stop and the value of the greatest sum. For example if the list is: x = [7, -9, 5, 10, -9, 6, 9, 3, 3, 9] then the greatest interval is from x[2:10] giving 36:

$$sum(x[2,10]) \quad = \quad 5 + 10 - 9 + 6 + 9 + 3 + 3 + 9 = 36 \tag{14}$$

The function should return a list of values i.e., [2, 10, 36]

For this problem you are allowed to use sum(lst) which returns the sum of a non-empty list of numbers.

Programming Problem 2: Maximal Value

- Complete the function msi(x)
- You are allowed to use the sum function
- You might find the following from the lecture useful:

```
>>> x = [1,2,3,-2]
>>> sum(x[1:3])
5
```

Problem 3: Move the Cheese

In this problem, you must read the constraints carefully. You are given a list of two types of cheese, one called zero (denoted by 0) and the other one (denoted by 1). The actual names are so long and complicated, it'd make the problem even more difficult. You must move all the zero cheeses next to each other in the front of the list (because zero is more popular than one) leaving all the one cheeses next to each other. I will supply the loop (you need only one) and you'll be given two variables. You

- cannot build any extra lists
- cannot use any list operations other than single subscripting
- cannot use any more loops

Here is the starting code:

```
data = [[1,0],[0,1,0,1,0,1,0],[1,1,1,1,0,0,0,0]]
def move(x):
    lo,hi = 0,len(x)-1
```

```
   while lo < hi:
    pass  #you can only add code here -- see the constraints above
    return x

for d in data:
    print(f"{d} => {move(d)}")
```

has output:

```
[1, 0] => [0, 1]
[0, 1, 0, 1, 0, 1, 0] => [0, 0, 0, 0, 1, 1, 1]
[1, 1, 1, 1, 0, 0, 0, 0] => [0, 0, 0, 0, 1, 1, 1, 1]
```

Programming Problem 3: Move the Cheese

- Complete the function.
- Follow the constraints.
- Think about small problems first–don't over think it :).

Problem 4: Polynomial Regression using numpy

In this problem, you'll be modeling rock bass otolith data: the size as a function of years. The data is in a csv fish_data.txt. When plotted, the data appears polynomial–size three (see Fig. 1).

For this we'll use numpy's polyfit function found here: `https://numpy.org/doc/stable/reference/generated/numpy.polyfit.html`.

To be specific, we have data D and a model

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \tag{15}$$

From D, we compute the "best" values of a_0, a_1, a_2, a_3. We saw in the previous homework we can use bounded loops to find these values, but there are a myriad of packages and modules available for us to use. Numpy has a polyfit function that takes the data $D = [(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)]$ as a list of x values and y values and returns an array of a_0, a_1, a_2, a_3. We can use these with numpy to build a function called poly1d( ). In this problem you'll need to:

1. Read the data from the file
2. Read about the package from the url provided and build the graph

```
def get_fish_data(path,name):
   pass

#INPUT two lists
```

Age (year)	Length (inches)
1	4.8
2	8.8
2	8.0
3	7.9
4	11.9
5	14.4
6	,14.1
6	15.8
7	15.6
8	17.8
9	18.2
9	17.1
10	18.8
10	19.5
11	18.9
12	21.7
12	21.9
13	23.8
14	26.9
14	25.1

```
#RETURN a polynomial function degree 3
def make_function(X,Y):
    pass


path,name = "","fish_data.txt"
X,Y = get_fish_data(path,name)
data4 = [[i,j] for i,j in zip(X,Y)]
print(data4)

plt.plot(X,Y,'ro')                #plots data D in red

xp = np.linspace(1,14,10) #makes input for model
p3 = make_function(X,Y)  #builds model from data
plt.plot(xp,p3(xp),'b')  #plots  model in bule

plt.xlabel("Age (years)")  #visualization elements
plt.ylabel("Length (inches)")
plt.title("Rock Bass Otolith")
plt.show()
```

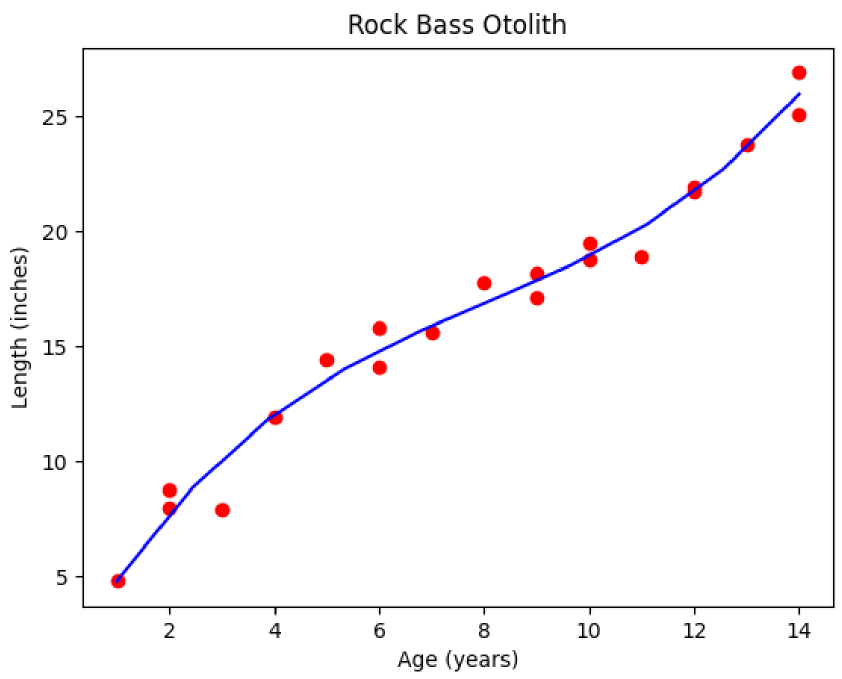

Figure 1: Plots of rock bass data (red) and model (blue). We use numpy's polyfit to do a regression on a polynomial of degree three.

gives output:

```
[[1, 4.8], [2, 8.8], [2, 8.0], [3, 7.9], [4, 11.9], [5, 14.4], [6, 14.1], ↩
    [6, 15.8], [7, 15.6], [8, 17.8], [9,
18.2], [9, 17.1], [10, 18.8], [10, 19.5], [11, 18.9], [12, 21.7], [12, ↩
    21.9], [13, 23.8], [14, 26.9], [14, 25.1]]
```

and the plot.

Programming Problem 4: Polynomial Regression with Numpy

- Please read the documentation as you see necessary for solution–please do the small examples shown.
- Complete the functions.
- You should plot the data first (red dots) without plotting the function (blue line) to see if the data is correct.
- Keep in mind that after you have tested your code, you must comment out the 'import matplotlib' and the plotting code given under the __main__ before submitting to the Autograder. Autograder can not draw graphical plots on the web browser so it will return an error if you do not comment out the plotting code/import matplotlib.

Paired Programming Partners

ghyatt@iu.edu, sabola@iu.edu, zhaozhe@iu.edu
anthoang@iu.edu, rair@iu.edu
aketcha@iu.edu, sowmo@iu.edu
bgabbert@iu.edu, yasmpate@iu.edu
jaybaity@iu.edu, yz145@iu.edu
jonhick@iu.edu, jwescott@iu.edu
nbakken@iu.edu, leplata@iu.edu
hollidaa@iu.edu, rmatejcz@iu.edu
sbehman@iu.edu, ss126@iu.edu
hc51@iu.edu, pvinod@iu.edu
ljianghe@iu.edu, dy11@iu.edu
jasoluca@iu.edu, jmissey@iu.edu
mgambett@iu.edu, alexschu@iu.edu
siqidong@iu.edu, hmerrit@iu.edu
eddykim@iu.edu, vthakka@iu.edu
aadidogr@iu.edu, jonvance@iu.edu
ecastano@iu.edu, dannwint@iu.edu
ogift@iu.edu, elyperry@iu.edu
chnbalta@iu.edu, isarmoss@iu.edu
anderblm@iu.edu, schwani@iu.edu
jacbooth@iu.edu, mszczas@iu.edu
moalnass@iu.edu, shethsu@iu.edu
aladkhan@iu.edu, tshore@iu.edu
jl263@iu.edu, lenonti@iu.edu
rythudso@iu.edu, sjxavier@iu.edu
sjkallub@iu.edu, mostrodt@iu.edu
cheng47@iu.edu, esmmcder@iu.edu
jacklian@iu.edu, arnpate@iu.edu
tifhuang@iu.edu, peschulz@iu.edu
agoldswo@iu.edu, nireilly@iu.edu
sijatto@iu.edu, criekki@iu.edu
jegillar@iu.edu, webejack@iu.edu
qhodgman@iu.edu, danwils@iu.edu
wdoub@iu.edu, omilden@iu.edu
coldjone@iu.edu, jorzhang@iu.edu
abolad@iu.edu, svaidhy@iu.edu
arykota@iu.edu, macsvobo@iu.edu
aibitner@iu.edu, snuthala@iu.edu
aalesh@iu.edu, cmulgrew@iu.edu

brakin@iu.edu, psaggar@iu.edu
stfashir@iu.edu, patel88@iu.edu
actonm@iu.edu, jarymeln@iu.edu
mdgamble@iu.edu, roelreye@iu.edu
sfawaz@iu.edu, parisbel@iu.edu
lsherbst@iu.edu, chnico@iu.edu
megofolz@iu.edu, ajneel@iu.edu
akundur@iu.edu, davthorn@iu.edu
dombish@iu.edu, crfmarti@iu.edu
abdufall@iu.edu, lmamidip@iu.edu
arjbhar@iu.edu, dpepping@iu.edu
sakinolu@iu.edu, sjvaleo@iu.edu
dilkang@iu.edu, avpeda@iu.edu
joecool@iu.edu, shahneh@iu.edu
neybrito@iu.edu, zisun@iu.edu
sl92@iu.edu, bashih@iu.edu
jacuau@iu.edu, grschenc@iu.edu
nbulgare@iu.edu, yijwei@iu.edu
marbelli@iu.edu, cartmull@iu.edu
jl335@iu.edu, rpmahesh@iu.edu
jolawre@iu.edu, cjromine@iu.edu
joshbrin@iu.edu, bensokol@iu.edu
adagar@iu.edu, aw149@iu.edu
pkasarla@iu.edu, dwinger@iu.edu
tyfeldm@iu.edu, nkmeade@iu.edu
dud@iu.edu, ornash@iu.edu
klongfie@iu.edu, almoelle@iu.edu
ivycai@iu.edu, imalhan@iu.edu
joracobb@iu.edu, trewoo@iu.edu
nwbarret@iu.edu, johwarre@iu.edu
qugriff@iu.edu, chrinayl@iu.edu
swcolson@iu.edu, edshipp@iu.edu
aaberma@iu.edu, hvelidi@iu.edu
achimeba@iu.edu, greymonr@iu.edu
jackssar@iu.edu, jvalleci@iu.edu
bcarl@iu.edu, btpfeil@iu.edu
sihamza@iu.edu, jmom@iu.edu
bradhutc@iu.edu, bpoddut@iu.edu
moesan@iu.edu, hnichin@iu.edu
jadbenav@iu.edu, kvanever@iu.edu
ekkumar@iu.edu, imaychru@iu.edu

lgflynn@iu.edu, caitreye@iu.edu
mgorals@iu.edu, jyamarti@iu.edu
vilokale@iu.edu, wuyul@iu.edu
egillig@iu.edu, mw154@iu.edu
katzjor@iu.edu, cartwolf@iu.edu
sgcolett@iu.edu, jcpilche@iu.edu
davgourl@iu.edu, saseiber@iu.edu
kbchiu@iu.edu, kalomart@iu.edu
nokebark@iu.edu, sousingh@iu.edu
dk80@iu.edu, alescarb@iu.edu
georliu@iu.edu, nmonberg@iu.edu