# C200 Programming Assignment № 4

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

Indiana University, Bloomington, IN, USA

February 17, 2022

## Introduction

**Due Date: 10:59 PM, Thursday, February, 24, 2022** Submit your work before the deadline and remember to add, commit and push.

In this homework, you'll start mastering your skill in writing functions. Make sure to start working early on this assignment–it is abridged (shortened).

## Problem 1: Day of the Week

The modulus operator is denoted by the symbol %. For $x\%y$ it returns returns the remainder after $y$ is divided into $x$ a whole number of times. While you've seen this before, the format is likely different. To refresh your memory you can visit `https://realpython.com/python-modulo-operator/` or do your own search. For this problem, you will also need to use the floor function in python - think of floor as a way to round down a floating point number to the nearest integer. For example, floor(18.90) is 18 and floor(14.25) is 14.

An approximate formula for computing the day of the week given dlst = [d,m,y] where d is day, m is month, and y is year is:

$$
\begin{aligned}
dlst &= [d, m, y] && (1)\\
a(dlst) &= y - \frac{(14-m)}{12} && (2)\\
x &= a(dlst) + \frac{a(dlst)}{4} - \frac{a(dlst)}{100} + \frac{a(dlst)}{400} && (3)\\
b(dlst) &= floor(x) && (4)\\
c(dlst) &= m + 12(\frac{14-m}{12}) - 2 && (5)\\
day((d,m,y)) &= (d + b(dlst) + (31 \cdot \frac{c(dlst)}{12}))\%7 && (6)
\end{aligned}
$$

The function $day$ returns a number $1, 2, \ldots, 7$ where $1 = Monday, 2 = Tuesday, \ldots$. If we use the number as a key and day of the week as a value, we can use the value to produce meaningful text. Here is the dictionary used:

```
week = {1:"Mon", 2:"Tue", 3:"Wed", 4:"Thu", 5:"Fri", 6:"Sat", 7:"Sun"}
```

For example,

$$
\begin{aligned}
day([14,2,2000]) &= \text{Mon} && (7)\\
print(day([14,2,1963])) &= \text{Thu} && (8)\\
print(day([14,2,1972])) &= \text{Mon} && (9)
\end{aligned}
$$

2/14/2000 falls on a Monday; 2/14/1963 falls on a Thursday; 2/14/1972 falls on a Monday.

**Deliverables for Problem 1**

- Complete the functions described above.
- For calculating b(dlst), you can use the math.floor() function from the math library.

## Problem 2: Starting quantum computing

Quantum Computing is a different model of computation imagined by the physicist Feynman. Instead of whole numbers, like we use in the Turing model, it uses complex numbers. Complex numbers, if you remember, are actually pairs of real numbers written as:

$$\text{complex number} = x \pm y\, i \tag{10}$$

where $x, y \in \mathbb{R}$ (they're just numbers) and there's a lone $i = \sqrt{-1}$. The x is called the real part and the y is called the imaginary part. For example,

$$x^2 + 1 = 0 \tag{11}$$
$$x = \sqrt{-1} = i \tag{12}$$

Then

$$x^2 + 1 = (\sqrt{-1})^2 + 1 = -1 + 1 = 0 \tag{13}$$

In Python we use the function complex(x,y) to form a complex number $(x \pm yj)$ where $j$ represents $i$ and there's no space between the $\pm$ sign and numbers. Let's write the function on line (10), build the complex number, and show it's a solution.

```
>>> def f(x):
...     return x**2 + 1
...
>>> root = complex(0,1)
>>> f(root)
0j
>>> f(root) == 0
True
>>> f(root).real
0.0
>>> f(root).imag
0.0
```

For a quadratic from last homework, you know that the answer is either real or complex. To remind you, the answer is complex when the discriminant is negative. Fig. 1. visualizes what's happening–the curve for complex roots does **intersect**

the abscissa. Observe that you'll get zero (on the $x$-axis or *abscissa*) if you put either of these two x values there. Sometimes the discriminant (the value in the squareroot) is negative. This is where $i$ comes in. We simply multiply the value by -1, thereby allowing us to take the squareroot, but then we append an $i$ to signal it's imaginary.

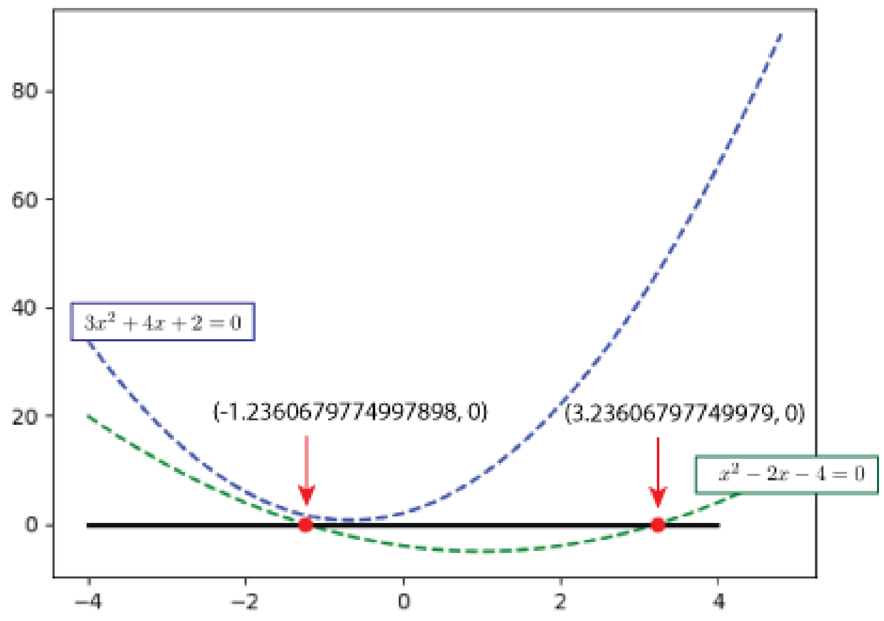

Figure 1: The red dots (with red arrows) are where your solutions are to $x^2 - 2x - 4 = 0$. The dash curve is the function itself. The horizontal line just makes it easier to see the x-axis. The two values are -1.2360679774997898 and 3.23606797749979. For the other function $3x^2+4x+2=0$ shown in blue, because it has an imaginary part, it cannot cross the axis. You'll use the matplotlib library you were introduced last homework to actually plot this!

For example, suppose we have $3x^2+4x+2=0$. Then we find, after some algebra:

$$\begin{aligned}
x &= \frac{-4 \pm \sqrt{-8}}{6} && (14)\\
&= \frac{-4 \pm \sqrt{-2 \times 4}}{6} && (15)\\
&= \frac{-4 \pm \sqrt{-2} \times \sqrt{4}}{6} && (16)\\
&= \frac{-4 \pm 2\sqrt{-2}}{6} && (17)\\
&= -\tfrac{2}{3} \pm \tfrac{\sqrt{2}}{3} i && (18)\\
&= (-\tfrac{2}{3} + \tfrac{\sqrt{2}}{3} i, -\tfrac{2}{3} - \tfrac{\sqrt{2}}{3} i) && (19)
\end{aligned}$$

In Python we would write:

```
>>> import math
>>> complex(-2/3,math.sqrt(2)/3),complex(-2/3,-math.sqrt(2)/3)
((-0.6666666666666666+0.47140452079103173j), ↩
    (-0.6666666666666666-0.47140452079103173j))
>>> x = round(-2/3,2)
>>> y = round(math.sqrt(2)/3)
>>> complex(x,y), complex(x,-y)
((-0.67+0j), (-0.67+0j))
>>> def f(x):
...     return 3*(x**2) + 4*x + 2
```

```
...
>>> f(complex(x,y))
(0.6667000000000001+0j)
>>> f(complex(x,-y))
(0.6667000000000001+0j)
>>> f(complex(-2/3,math.sqrt(2)/3))
0j
```

Observe the difference between using the full root and only to two decimal places. You'll write a function q(t) where:

$$q((a,b,c)) = \begin{cases} (r_0, r_1) & \text{real roots if } b^2 - 4ac \geq 0 \\ (c_0, c_1) & \text{complex roots if } b^2 - 4ac < 0 \end{cases} \tag{20}$$

When returning the roots, round to two decimal places, *i.e.*, round($x$,2). Here are a few examples:

$$q((3,4,2)) = ((-0.67 + 0.47j), (-0.67 - 0.47j)) \tag{21}$$

$$q((1,3,-4)) = (-4.0, 1.0) \tag{22}$$

$$q((1,-2,-4)) = (-1.24, 3.24) \tag{23}$$

**Deliverables for Problem 2**

- Complete the function.

## Problem 3: Computing Relationships

To match people, companies use trigonometry. Assume you have a list of people $[p_0, p_1, \ldots, p_m]$, we find the two different people who have the smallest angle. The way we calculate this is to assume each person is a list (mathematically vector) of 0s and 1s. We need three basic functions: inner product, magnitude, and $\cos^{-1}$. We assume two lists $x = [x_0, x_1, \ldots, x_n], y = [y_0, y_1, \ldots, y_n]$ of 0s and 1s of the same length:

$$inner\_prod(x, y) = x_0 y_0 + x_1 y_1 + \cdots + x_n y_n \tag{24}$$

$$mag(x) = \sqrt{inner\_prod(x, x)} \tag{25}$$

For the last function (where we calculate the angle), we know that:

$$\cos(\theta) = \frac{inner\_prod(x, y)}{mag(x) mag(y)} \tag{26}$$

In class we learned that we can invert a function and the inverted function is denoted as $f^{-1}$. So we can:

$$\cos^{-1}(\cos(\theta)) = \cos^{-1}(\frac{inner\_prod(x, y)}{mag(x) mag(y)}) \tag{27}$$

$$\theta = \cos^{-1}(\frac{inner\_prod(x, y)}{mag(x) mag(y)}) \tag{28}$$

The math module has math.acos() for $\cos^{-1}()$. Further, Python returns $\theta$ in radians. We have:

$$\pi \text{ radians} = 180 \text{ degrees} \tag{29}$$

Thus, to convert from radians to degree, you **must** multiply your answer by $\frac{180}{\pi}$.

The task is to write two functions. The match function takes a list of people $p_i$ where each person is a list of 0s 1s, and returns all unique pairs with the angle in degrees. The function best_match takes the result from match and returns the pair with the "best" match–the smallest degree. We can assume there's only one best match. Here is a run (with some extra output) for your perusal.

```python
people0 = [[0,1,1],[1,0,0],[1,1,1]]
print(match(people0))
print(best_match(match(people0)))
```

gives an output

```
[[[0, 1, 1], [1, 0, 0], 90.0],
 [[0, 1, 1], [1, 1, 1], 35.26],
 [[1, 0, 0], [1, 1, 1], 54.74]]
([0, 1, 1], [1, 1, 1], 35.26)
```

We're displaying the result of match so you can see the structure. Each unique pair as an angle. For example:

$$\begin{aligned} inner\_prod([1,0,0],[1,1,1]) &= 1(1)+0(1)+0(1)=1 && (30)\\ mag([1,0,0]) &= \sqrt{inner\_prod([1,0,0],[1,0,0])} = \sqrt{1} = 1 && (31)\\ mag([1,1,1]) &= \sqrt{inner\_prod([1,1,1],[1,1,1])} = \sqrt{3} && (32)\\ \theta &= (\frac{180}{\pi})\text{math.acos}(\frac{1}{1\sqrt{3}}) \approx 54.74 && (33) \end{aligned}$$

**Deliverables for Problem 3**

- Complete the functions.

## Problem 4: Intersecting Lines

Given two intersecting lines $y = m_0x + b_0, y = m_1x + b_1$ at a single point $(x', y')$ means that $y' = m_0x' + b_0$ and $y' = m_1x' + b_1$. The function $intersect$ takes two lines described by the slope and intercept, and returns their point of intersection.

For example, $y = 2x + 3$ and $y = -\frac{1}{2}x + 2$ we have

$$\begin{aligned} \ell_0 &= (2, 3) && (34) \\ \ell_1 &= (-\frac{1}{2}, 2) && (35) \\ intersect(\ell_0, \ell_1) &= (-0.4, 2.2) && (36) \\ intersect((1, 4), (-1/2, 1/2)) &= (-2.33, 1.67) && (37) \end{aligned}$$

**Deliverables for Problem 4**

- Complete the function.
- Remember to round your result to 2 decimal places.

## Problem 5: Toward Statistical Analysis

In this problem, you'll write fundamental statistical functions. We assume a list of numbers $lst = [x_0, x_1, \ldots, x_n]$.

$$\begin{aligned}
mean(lst) &= (x_0 + x_1 + \cdots + x_n)/\text{len}(lst) && (38)\\
\mu &= mean(lst) && (39)\\
var(lst) &= \frac{1}{\text{len}(lst)}((x_0-\mu)^2 + (x_1-\mu)^2 + \cdots + (x_n-\mu)^2) && (40)\\
std(lst) &= \sqrt{var(lst)} && (41)
\end{aligned}$$

For example, $lst = [1, 3, 3, 2, 9, 10]$, rounding to two places

$$\begin{aligned}
mean(lst) &= 4.67 && (42)\\
var(lst) &= 12.22 && (43)\\
std(lst) &= 3.5 && (44)
\end{aligned}$$

The last function mean_centered takes a list of numbers $lst = [x_0, x_1, \ldots, x_n]$ and returns a new list $[x_0 - \mu, x_1 - \mu, \ldots, x_n - \mu]$. An interesting feature of the mean-centered list is that if you try to calculate its mean, it's zero:

$$\begin{aligned}
\mu &= (x_0 + x_1 + \cdots + x_n)/n && (45)\\
lst &= [x_0 - \mu, x_1 - \mu, \ldots x_n - \mu] && (46)\\
mean(lst) &= ((x_0 - \mu) + \cdots (x_n - \mu))/n && (47)\\
&= ((x_0 + \ldots + x_n) + n\mu)/n && (48)\\
&= \mu - \mu = 0 && (49)
\end{aligned}$$

Using the same list we have

$$mean(mean\_centered(lst)) = -0.0 = 0 \qquad (50)$$

**Deliverables for Problem 5**

- Complete the functions.

## Problem 6: Market Equilibrium

When modeing market behavior we use two curves to indicate supply and demand. You can also think of supply as quantity and demans as want. When the two curves intersect (the common point where they cross), see Fig. 2, there is a point that satisfies both equations. Review the problem of intersecting two lines. Here we have specifically two quadratic functions. Assume

$$s(x) = -.025x^2 - .05x + 60 \qquad (51)$$

$$d(x) = 0.02x^2 + .6 + 20 \qquad (52)$$

for supply $s$ and demand $d$. We have a function $equi$ that takes the coefficients of $s, d$ and returns the solution:

$$equi((-.025, -.5, 60), (0.02, .6, 20)) = (20.0, -44.44) \qquad (53)$$

Our solution returns the roots to a quadratic equation. Since $s, d$ models the presence of items, only 20 makes sense. HINT: use your $q$ function in Problem 2.

**Deliverables for Problem 6**

- Complete the function.
- Use $q$ in Problem 2 to make the solution very quick.

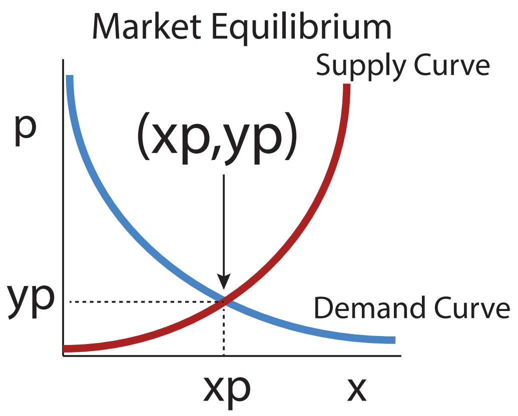

Figure 2: Market equilibrium with supply and demand

## Partners for Programming

mabdayem@iu.edu, bolabanj@iu.edu
ahnabrah@iu.edu, ldownin@iu.edu
adamsjf@iu.edu, dazamora@iu.edu
dadeyeye@iu.edu, cl101@iu.edu
cmaguila@iu.edu, daparent@iu.edu, evewalsh@iu.edu
ahmedrr@iu.edu, powelchr@iu.edu
malshama@iu.edu, tclady@iu.edu
olalbert@iu.edu, nps1@iu.edu
nalemanm@iu.edu, lancswar@iu.edu
faysalza@iu.edu, sydfoste@iu.edu
rnameen@iu.edu, cbalbuen@iu.edu
svamin@iu.edu, egmorley@iu.edu
jaygul@iu.edu, ssalama@iu.edu
bbacso@iu.edu, vvictori@iu.edu
rbajaj@iu.edu, timbogun@iu.edu
ikbanist@iu.edu, jleverty@iu.edu
zsbanks@iu.edu, logbrads@iu.edu
mbarrant@iu.edu, vramkum@iu.edu
tymbarre@iu.edu, jaespin@iu.edu
dcblakle@iu.edu, sahmir@iu.edu
mlboukal@iu.edu, apoellab@iu.edu
gabradle@iu.edu, sj110@iu.edu
kbburnet@iu.edu, remarche@iu.edu
lburrola@iu.edu, luilmill@iu.edu
cbylciw@iu.edu, evtomak@iu.edu
aidcarli@iu.edu, mjerrell@iu.edu
dcaspers@iu.edu, kamdelmo@iu.edu
mathchen@iu.edu, jwrohn@iu.edu
joecool@iu.edu, owenaj@iu.edu
ccoriag@iu.edu, msronan@iu.edu
blcrane@iu.edu, jugallow@iu.edu
gcruzcor@iu.edu, linweix@iu.edu
jacuau@iu.edu, dsenisai@iu.edu
acuazitl@iu.edu, owinston@iu.edu
ddahodu@iu.edu, edepke@iu.edu
rpdeady@iu.edu, kevko@iu.edu
cadelaga@iu.edu, cy30@iu.edu
emdelph@iu.edu, sprabhak@iu.edu
shrdesai@iu.edu, cadwilco@iu.edu

jpdiskin@iu.edu, gillenj@iu.edu
shadoshi@iu.edu, notsolo@iu.edu
eeconomo@iu.edu, ethfrago@iu.edu
ereilar@iu.edu, nogoch@iu.edu
jpenrigh@iu.edu, aj110@iu.edu
mfanous@iu.edu, jjwelp@iu.edu
nfarhat@iu.edu, jtkrug@iu.edu
jayfish@iu.edu, mattroac@iu.edu
fraustom@iu.edu, njindra@iu.edu
nfrische@iu.edu, yuljiao@iu.edu
gaoxinl@iu.edu, ejharms@iu.edu
jmgebhar@iu.edu, astrouf@iu.edu
ggivan@iu.edu, mguleria@iu.edu
bgloor@iu.edu, silmudee@iu.edu
noahgrah@iu.edu, gkarnuta@iu.edu
halejd@iu.edu, kviele@iu.edu
ehallor@iu.edu, mlumbant@iu.edu
hamedi@iu.edu, asaokho@iu.edu
pheile@iu.edu, sothor@iu.edu
jchobbs@iu.edu, binyhu@iu.edu
jthurd@iu.edu, srpothir@iu.edu
srimmadi@iu.edu, kpalus@iu.edu
yjan@iu.edu, grtalley@iu.edu
gjarrold@iu.edu, johnslia@iu.edu
cjohanns@iu.edu, ndvanbur@iu.edu
kjwalapu@iu.edu, shevphil@iu.edu
fkanmogn@iu.edu, camitong@iu.edu
akaushal@iu.edu, divpatel@iu.edu
sskauvei@iu.edu, mdtanner@iu.edu
jk130@iu.edu, rosavy@iu.edu
phklein@iu.edu, reedrobe@iu.edu
jhlazar@iu.edu, dylomall@iu.edu
wlegear@iu.edu, ibnash@iu.edu
yl181@iu.edu, chlzhang@iu.edu
lopezis@iu.edu, gkyoung@iu.edu
eluthra@iu.edu, ryou@iu.edu
vimadhav@iu.edu, actoney@iu.edu
gmanisca@iu.edu, jsm13@iu.edu
pmanolis@iu.edu, raia@iu.edu
mmansoo@iu.edu, anniye@iu.edu
tymath@iu.edu, yudsingh@iu.edu

rlmcdani@iu.edu, amystaff@iu.edu
mooralec@iu.edu, jamoya@iu.edu
hnasar@iu.edu, mschauss@iu.edu
boconno@iu.edu, gtutton@iu.edu
mppan@iu.edu, apapaioa@iu.edu
perkcaan@iu.edu, sowvemul@iu.edu
aramo@iu.edu, mzakman@iu.edu
sasaluja@iu.edu, ansiva@iu.edu
chsand@iu.edu, samuwagn@iu.edu
eliserr@iu.edu, hdwatter@iu.edu
ashankwi@iu.edu, weidzhen@iu.edu
mvincen@iu.edu, rwan@iu.edu
yangyuc@iu.edu, lzinn@iu.edu