H200 Programming Assignment № 3

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

Indiana University, Bloomington, IN, USA

September 19, 2019

Contents

Introduction

In this homework, you'll start mastering your skill in writing functions and be introduced to your first data structure: dicitionaries. **All the Deliverables for Homework**
Add a new folder to your C200 folder named Assignment3. In this new folder you will add the following Python files for Assignment3.

- **funwithfunctions.py**
- **qc1.py**
- **if.py**
- **premetal.py**
- **myclock.py**

Make sure and commit your project and modules by **11pm, Wednesday September 25th 2018.** This later date is due to inclement weather.

As always, all the work should be your own. You will complete this before Wednesday September 25 11:00P. You will submit your work by committing your code to your GitHub repository. You will *not* turn anything in on canvas. If your timestamp is 11:01P or greater, the homework cannot be graded. So do not wait until 10:58P to turn it in.

There are other problems to do that aren't turned-in—these are sessions that you do with your computer and some questions to help improve your understanding of the overall homework.

Problem 1

For each of these real-world problems, you will complete functions. We have provided the *stube* (*header* or *signature*) which means:

```
def functionname (p1, p2, ..., pk):  #This is provided
#TODO: IMPLMEMENT FUNCTION
```

Please study the functions carefully if there's a large number, use the number as it's presented in the description of the problem and not an abridged version.

1. A deposit of d dollars is invested at r interest rate (compounded continuously) for t yields:

$$y(d,r,t) = de^{rt} \quad (1)$$
$$y(1000,.02,10) = \$1221.40 \quad (2)$$

   Use `import math` and `math.exp()` which is a function that raises e to it arguments. `math.exp(1)` produces `2.718281828459045`.

2. According to the Center for Disease Control `https://www.cdc.gov/salmonella/`, the bacteria *Salmonella* causes about 20K hospitalizations and nearly 400 deaths a year. The formula for how fast this bacteria grows is:

$$N(n_0,m,t) = n_0e^{mt} \quad (3)$$
$$N(500,100,4) = 2.610734844882072 \times 10^{176} \quad (4)$$

   where n_0 is the initial number of bacteria, m is the growth rate (*e.g.*, 100 per hr, and t is time. Calculate the size for an initial colony of 500 that grows 100 per hr for four hours.

3. The number of teeth $N_t(t)$ after t days from incubation for Alligator mississippiensis is:

$$N_t(t) = 71.8e^{-8.96e^{-0.0685t}} \quad (5)$$
$$N(1000) = 71 \quad (6)$$

4. Assume t hours after administrating a drug through the bloodstream, the percent concentration is:

$$K(t) = \frac{\frac{9}{2.6}t}{2t^2+3} \quad (7)$$
$$K(1) = 0.69\% = 0.007 \quad (8)$$

5. The average monthly rent for 1000 ft^2 apartment in urban areas from 1998 to 2005 in t years is:

$$r(t) = 1.5207t^4 - 19.166t^3 + 62.91t^2 + 6.0726t + 1026 \quad (9)$$
$$r(4) = 1219.53 \quad (10)$$

6. Suppose the profit in thousands of dollars of an x items sold is:

$$\begin{aligned} p(x) &= 4x^2 - 100x - 1000 && (11) \\ p(10) &= -1600 && (12) \end{aligned}$$

7. If we want to calculate the work done when an ideal expands isothermally (and reversibly) we use for initial and final pressure $P_i = 10\ bar, P_f = 1\ bar$ respectively at $300^o K$. In this problem we are using $\ln$ which is $\log_e$. Because $\log_e$ is used so often, you'll see it just as often abbreivated as $\ln$. This is in the math module as math.log.

$$\begin{aligned} W(P_i, P_f) &= RT \ln(P_i/P_f) && (13) \\ W(10, 1) &= 8.314(300)(\ln 10) = 5740 && (14) \end{aligned}$$

at temperature T (Kelvins) and $R = 8.314\ J/mol$ the universal gas constant.

8. Depreciation spreads the deductible cost of a fixed asset over its *estimate* life. The formula uses original cost c, salvage value s, and estimated life ℓ:

$$\begin{aligned} depreciate(c, s, \ell) &= \frac{c - s}{\ell} && (15) \\ depreciate(20000, 1000, 5) &= \frac{20000 - 1000}{5} = 3800 && (16) \end{aligned}$$

The annual depreciated expense is \$3,800.

9. The Wright Brothers are known for their Flyer and its maiden flight. The formula for lift is:

$$\begin{aligned} L &= kV^2 AC_\ell && (17) \\ L &= 0.0033(33.8)^2(512)0.515 = 994 && (18) \end{aligned}$$

where k is Smeaton's Coefficient ($k = 0.0033$ from their wind tunnel), $V = 33.8\ mph$ is relative velocity over the wing, $A = 512\ ft$ area of wing, and $C_\ell = 0.515$ coefficient of lift. The Flyer weighed $600\,lbs$ and Orville was about $145\,lbs$. We can see that the lift was sufficient since $994 > 745$.

```
import math
#A description with inputs and returns are given fully.   We'd like you to add comments from the readings -- here is an example:

#INPUT dollars, interest rate, yields
#RETURN dollars
#1
def y(d,r,t):
#TO DO: IMPLEMENT FUNCTION

#INPUT
#OUPUT
#2
def N(n_0, m, t):
#TO DO: IMPLEMENT FUNCTION

#INPUT
#OUPUT
#3
def N_t(t):
#TO DO: IMPLEMENT FUNCTION

#INPUT
#OUPUT
#4
def K(t):
#TO DO: IMPLEMENT FUNCTION

#INPUT
#OUPUT
#5
def r(t):
#TO DO: IMPLEMENT FUNCTION

#INPUT
#OUPUT
#6
def p(x):
#TO DO: IMPLEMENT FUNCTION

#INPUT
#OUPUT
#7
def W(P_i,P_f):
#TO DO: IMPLEMENT FUNCTION

```

```
#INPUT
#OUPUT
#8
def depreciate(c,s,life):
#TO DO: IMPLEMENT FUNCTION

#INPUT
#OUPUT
#9
def L(k,V,A,C):
#TO DO: IMPLEMENT FUNCTION

### TESTS

print(y(1000,.02,10))
print(N(500,100,4))
print(math.floor(N_t(1000)))
print(K(1))
print(r(4))
print(p(10))
print(W(10,1))
print(depreciate(20000,1000,5))
print(L(0.0033,33.8,512,0.515))
if __name__=="__main__":
```

```
1221.40275816017
2.610734844882072e+176
71
0.6923076923076923
1219.5256
-1600
5743.107738945749
3800.0
994.0873113599998
```

Deliverables for Problem 1

- Complete the function stubs found in **ursala.py**
- For the last function that models profit, what is the fewest number of items that make a profit?
- Put all of these functions in a new module named **funwithfunctions.py**.

Problem 2

You will **not** turn anything in for this – you are asked to carefully and methodically trace through this code, using environments, to determine its output. We haven't discussed tracing in detail in class, but you will be introduced to it in lab. Essentially tracing is following the flow of the program while keeping track of the variable values as they change and write out what is printed.

tracing.py

```python
def f(x,y):
    def g(x,z):
        if x + z < 2:
            return x+z+10
        else:
            return x+z/10
    if x+y>5:
      return g(y,x-10)
    else:
      return g(2*x,y+1)

def g(x,y,z):
    return f(x,3)

def h(a,b,c,d):
    return f(a,b+d) - g(c-d,a,a+c)

x = 1
y = 2
x = h(x,y,0,3)
y = g(x + h(-10,2,0,30),x+y,h(0,1,2,3))
print(x,y)
```

There are no deliverables for this problem.

Problem 3: 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 are actually simply pairs of real numbers. Python has complex numbers available, albeit slightly different from what people normally use, because (it's said), engineers use j.

$$\text{complex number} \;=\; x \pm y\, i \tag{19}$$

where $x, y \in \mathbb{R}$ (they're just numbers) and there's a lone i. The x is called the real part and the y is called the imaginary part. This is a solution to:

$$x^2 + i \;=\; 0 \tag{20}$$

$$x \;=\; \sqrt{-1} = i \tag{21}$$

For example:

$$i^2 \;=\; \sqrt{-1} \times \sqrt{-1} = -1 \tag{22}$$

In middle school one of the most terrifying first encounter with mathematics is in the form of the dreaded quadratic formula. We have Babylonian tablets that showing students millenia ago having to solve this problem–so take comfort in that. To refresh your memory, given:

$$ax^2 + bx + c \;=\; 0 \tag{23}$$

has two solutions:

$$x \;=\; \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{24}$$

For example, $x^2 - 2x - 4 = 0$

$$x \;=\; \frac{2 \pm \sqrt{4 - (4(1)(-4)}}{2(1)} \tag{25}$$

$$=\; \frac{2 \pm \sqrt{4 + 16}}{2} \tag{26}$$

$$=\; \frac{2 \pm \sqrt{20}}{2} \tag{27}$$

Observe that you'll get zero (on the x-axis or *abscissa*) if you put either of these two x values there. See Fig. 1. below.

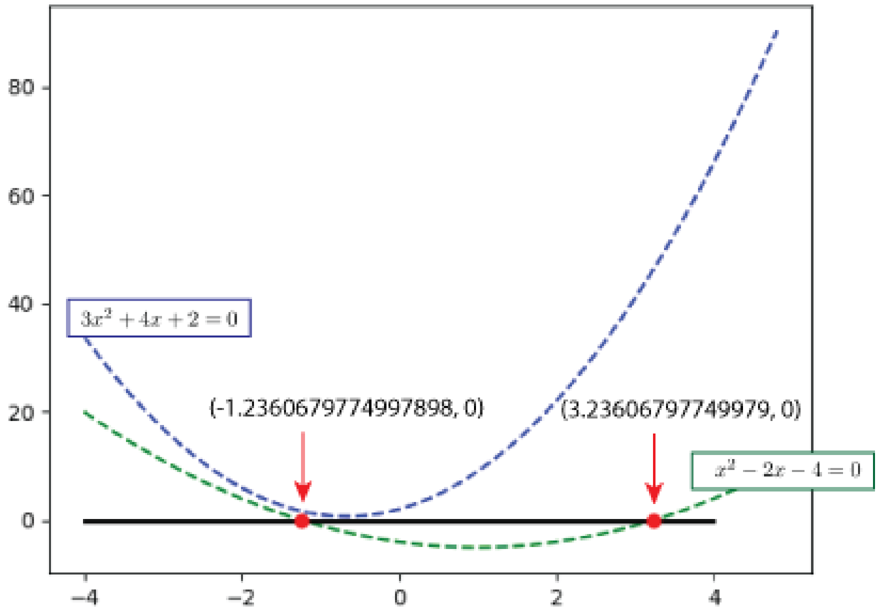


Figure 1: The red dots (with red arrows) are where your solutions are to $x^2 - 2x - 4 = 0$. The dash curve is the function itself. The horizontal line just makes it easier to see the x-axis. The two values are -1.2360679774997898 and 3.23606797749979. For the other function $3x^2 + 4x + 2 = 0$ shown 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!

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.

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

$$\begin{aligned} x &= \frac{-4 \pm \sqrt{-8}}{6} && (28) \\ &= \frac{-4 \pm \sqrt{-2 \times 4}}{6} && (29) \\ &= \frac{-4 \pm \sqrt{-2} \times \sqrt{4}}{6} && (30) \\ &= \frac{-4 \pm 2\sqrt{-2}}{6} && (31) \\ &= -\tfrac{2}{3} \pm \tfrac{\sqrt{2}}{3} i && (32) \end{aligned}$$

We can show you Python is able to use imaginary numbers:

```
import math

def f(x):
    return 3*(x**2) + 4*x + 2

x = -2/3 + math.sqrt(2)/3j
y = -2/3 - math.sqrt(2)/3j

```

```python
print(f(x))
print(f(y))
```

```
0j
0j
```

```python
#imports needed
import numpy as np
import math
import numpy as np
import matplotlib.pyplot as plt

#INPUT ax**2 + bx + c = 0 coefficients to quadratic
#RETURN tuple (x,y) that are solutions
def q(a,b,c):
#TO DO: IMPLEMENT FUNCTION

# Tests
print(q(1,3,-4))
print(q(2,-4,-3))
print(q(9,12,4))
print(q(3,4,2))


# Test & fun stuff!
#assigns the two values into x and y
#because this returns a tuple (firstvalue, secondvalue)
x1,y1 =  q(1,-2,-4) # x**2 - 2*x - 4 = 0
print(x1,y1)
#creates an anonymous function
#this is the function described above
#You should be intrigued by this assignment
#What is it doing?
f = lambda x:x**2 - 2*x - 4
#the output should be zero
print(f(x1),f(y1))

# Plotting Porition
# evenly sampled time at 200ms intervals
x = np.arange(-4.0, 5.0, 0.2)
# Green dashes for line
plt.plot(x, x**2 - 2*x - 4, 'g--')
# Blue dashes for line
plt.plot(x,3*x**2 + 4*x + 2, 'b--')
# Draw horizontal line
plt.plot([-4,4],[0,0], color='k', linestyle='-', linewidth=2)
# Plot red dots as solution
plt.plot([x1,y1],[0,0],'ro')
if __name__=="__main__":
     plt.show()
```

Output for Problem 3

```
Not Complex
(-4.0, 1.0)
Not Complex
(-0.5811388300841898, 2.58113883008419)
Not Complex
(-0.6666666666666666, -0.6666666666666666)
Complex
((-0.67+0.47j), (-0.67-0.47j))
Not Complex
-1.2360679774997898 3.23606797749979
0.0 0.0
```

Python Session

```
>>> (1+j)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'j' is not defined
>>> (1+0j)
(1+0j)
>>> (1+0j)*(1-0j)
(1+0j)
>>> complex(1,0)
(1+0j)
>>> x = complex(1,2)
>>> x
(1+2j)
>>> y = complex(1,-4)
>>> y
(1-4j)
>>> x*y
(9-2j)
>>> x.real
1.0
>>> x.imag
2.0
>>> type(x)
<class 'complex'>
>>> abs(x)
2.23606797749979
>>> (1**2 + (-2)**2))**(1/2)
2.23606797749979
>>> x
(1+2j)
```

Python Session Continued

```
>>> x.conjugate
<built-in method conjugate of complex object at 0x029C6F08>
>>> x.conjugate()
(1-2j)
>>> x*x.conjugate()
(5+0j)
>>> abs(x)
2.23606797749979
>>> (5)**(1/2)
2.23606797749979
>>> y
(1-4j)
>>> yc = complex(1,4)
>>> y*yc
(17+0j)
>>> abs(y*yc)**2
289.0
>>> abs(y*yc)
17.0
>>> z = complex(3,4)
>>> z
(3+4j)
>>> z*z.conjugate()
(25+0j)
>>> abs(z)
5.0
>>>
```

Deliverables for Problem 3

- We assume the quadratic coefficients are $ax^2 + bx + c = 0$ and your task is to produce solutions–some will be normal and some will be complex.
- Do the session for Complex Numbers (so you can understand Quantum Computing
- Here is some Python – explain what's happening:

```
>>> (-2**2)
-4
>>> (-2)**2
4
```

- Complete program
- Remember to use the Python complex(x,y) to change a pair of numbers to complex numbers.
- You should print a message `Not Complex` or `Complex` depending upon the numbers you produce.
- You'll be returning a structure return (x,y) where x and y are the solutions to the quadratic.
- Put the code in a new module named **qc1.py**.

Problem 4

In this problem you will rewrite conditional statements. In the Listing below, there are four different problems.

Listing 1: if.py

```python
x = True
y = False
z = 12
a = 10
b = not (x or not (x or y)) and True

#1#################
if b:
    print("Happy")
if not b and x:
    print("Valentines")
if not b and not x and not y:
    print("day!")
###################


#2#################
if (z > a) or (2*a > z):
    print("C200")
else:
    print("is bliss")
###################


#3#################
if not (not (x and y) or not x):
    print(a)
###################


#4#################
if (2 > z) or x:
    print("1")
elif 2 == 1:
    print("2")
elif y and not x:
    print("3")
else:
    print("4")
###################
```

```python

#5########################
def f(x):
    return (x==4)*100 + (x==3)*10 + (x==2)*1000 + (x!=4)*(x!=3)*(x!=2)*100000

print(f(4))
print(f(3))
print(f(2))
print(f(1))
##########################
if __name__=="__main__":
```

Output of function f(x)

```
Valentines
C200
1
100
10
1000
100000
```

Deliverables for Problem 4

- Each comment has a different task.
  - #1 Rewrite this as an eqiuvalent if-elif-else.
  - #2 Rewrite as an equivalent two if statements.
  - #3 Rewrite the conditional using as few not's as possible.
  - #4 Rewrite this as four single if statements.
  - #5 Rewrite this as an if-elif-else. The output of the prints are shown below.
- The rewrites should produce code that is functionally equivalent – so that if different values are used, the output of this program and your rewrite would be identical. Investigate the output if x = False and y = True, for example.
- Put the code in a new module named **if.py**.

Problem 5

Precious metals are assets that people acquire to hedge against inflation. Here are the prices as of September 19, 2019.

Precious Metal	Spot Price ($/Oz) (dollars per ounce)	$ Change
Gold	1503.35	11.00
Silver	17.91	0.47
Platinum	950.60	(2.50)
Palladium	1468.82	(10.48)

You'll be writing a program to help manage your funds. Some reflection *before* you begin programming–the problem is much simpler than it might seem. We'll give an example. Suppose you have $100. Apples cost $27 each. What's the maximal number of apples you can buy and what amount of money will be left over? Here's a minisession you should do on your own to help better understand.

```
>>> wallet = 100
>>> appleCost = 27
>>> apples = wallet/appleCost
>>> apples
3.7037037037037037
>>> import math
>>> apples = math.floor(apples)
>>> apples
3
>>> wallet - apples*appleCost
19
>>> 19 + 3*27 == 100
True
```

Listing 2: precmetal.py program

```
#imports
import math

########################################
#
# DATA
#
#All values $/ounce abbreviated $/Oz
Gold = 1503.35        #up 11.00
Silver = 17.91          #up 0.47
Platinum = 950.60    #down 2.50
Palladium = 1468.82 #down 10.48

Au, Ag, Pl, Pa = "Au", "Ag", "Pl", "Pa" # to make use of symbols easier

account = 100000  # $100,000 cash assets
Au_amt, Ag_amt, Pl_amt, Pa_amt = 0,0,0,0

#######################################
#INPUT NOTHING
#RETURN amount of money you have
#and amount of precious metals
def holdings():
    print("Your current holdings")
    print("Account   = {0:.2f}".format(account))
    print("Gold      = ",Au_amt)
    print("Silver    = ",Ag_amt)
    print("Platinum  = ",Pl_amt)
    print("Palladium = ",Au_amt)
    print()


#INPUT metal and amount in ounces you want to purchase
#RETURN the cost of the amount of metal you're purchasing
def preciousMetalToDollars(metal, amt):
#TO DO: IMPLEMENT FUNCTION

#INPUT metal and amt you want to purchase
#RETURN IF you have sufficient funds, purchase that amount in
#oz. and add to x_amt and subtract that from account
#If you don't have sufficient funds, output message that you
#can't purchase that amount
def purchase(metal, amt):
    global account
    global Au_amt
    global Ag_amt
```

```
    global Pl_amt
    global Pa_amt
#COMPLETE IMPLEMENTATION OF FUNCTION
#LEAVE GLOBAL variables



######MAIN######

holdings()

print("{:0.2f}".format(preciousMetalToDollars(Au,4)))
print("{:0.2f}".format(preciousMetalToDollars(Ag,100)))
print("{:0.2f}".format(preciousMetalToDollars(Pa,23)))
print("{:0.2f}".format(preciousMetalToDollars(Pl,17)))

print()
purchase(Au,4)
purchase(Ag,100)
purchase(Pa,23)
purchase(Pl,17)
print()

holdings()

purchase(Pl,100000)
if __name__=="__main__":
```

Output precmetal.py

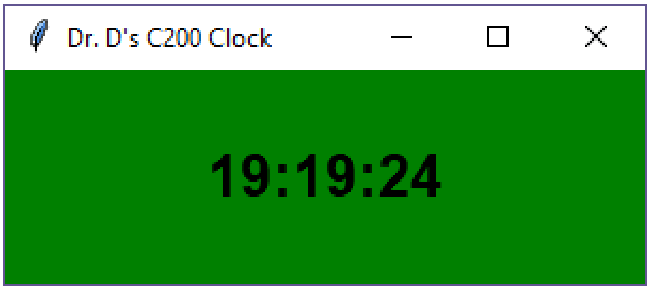
```
Your current holdings
Account   = 100000.00
Gold      =  0
Silver    =  0
Platinum  =  0
Palladium =  0

6013.40
1791.00
33782.86
16160.20

You have puchased 4 oz. of 1503.35 for 6013.40
You have $93986.60 remaining in your account.
You have puchased 100 oz. of 17.91 for 1791.00
You have $92195.60 remaining in your account.
You have puchased 23 oz. of 1468.82 for 33782.86
You have $58412.74 remaining in your account.
You have puchased 17 oz. of 950.6 for 16160.20
You have $42252.54 remaining in your account.

Your current holdings
Account   = 42252.54
Gold      =  4
Silver    =  100
Platinum  =  17
Palladium =  4

You have insufficient funds for this purchase.
```

Deliverables for Programming Problem 5

- Complete the program above. The draft is **metalurgency.py**.
- Put your code for this function in a new Python module named **precmetal.py**.

Problem 6

This problem shows how much fun you can have with Python. In the listing below is the code to display your very own clock. Below is a snapshot.



myclock.py

```python
import tkinter as tk
import time

#time1="" means its defaults to "" initially.
def tick(time1=""):
    time2 = time.strftime('%H:%M:%S')
    if time2 != time1:
        time1 = time2
        clock.config(text = time2)
    clock.after(200,tick)

mywindow = tk.Tk()
mywindow.title("Dr. D's C200 Clock")
mywindow.geometry("300x100")
clock = tk.Label(mywindow, font = ('arial', 20, 'bold'), bg = 'green')
clock.pack(fill = 'both', expand=1)
tick()
mywindow.mainloop()
```

Deliverables for Problem 6

- You might have to install the time module, but you're getting to be an expert now!
- Get the code to run.
- Change the title to your name.
- Change the font from arial to gothic.
- Put your code for this function in a new Python module named **myclock.py**.