DATA STRUCTURES AND ALGORITHMS IN PYTHON

Assignment #3

TOPICS: FUNCTIONS AND LISTS

Indian Creek School

DUE : *15 September 2023*

For problems that ask you to write a program, place your code in a .py file and write a comment at the beginning of the file that includes your name, date, and description of the contents of the file. Suggested file names appear in bold at the beginning of problems that ask you to write a program.

EXAMPLE:

```python
# Name: Doug McNally
# Date: 2023-06-10
# Description: Lucky number checker

num = int( input("Pick a number: ") )
if num == 42:
     print("You entered the right number!")
else:
     print("You lose!")
```

PROBLEMS THAT REQUIRE YOU TO WRITE A PYTHON FILE: 6 – 8
PROBLEMS THAT REQUIRE ANSWERS IN A WORD/PDF DOCUMENT: 2 – 4

1. Read section 1.5 in *Data Structures and Algorithms in Python*.

2. Explain the primary reasons for using functions in your programs.

3. There are two errors in the following code. Identify both issues.

```python
def multiply(x, y):
    return x*y

result = multiply(5)
print(result)
```

4. There are two errors in this code that deals with lists. Identify both of them.

```python
fruits = ["apple", "banana", "cherry"]
print(fruit[3])
```

5. Complete the following on CodingBat.com. Instructions on how to get an account set up on the website will be given in class.
Remember to always log in before starting so that I can see your work!

   - Warmup-1
     - Complete any 3 problems. Warmup problems have solutions available - you may look at them if you get stuck, but don't just copy them.
   - Warmup-2
     - Complete any 3 problems. Warmup problems have solutions available - you may look at them if you get stuck, but don't just copy them.
   - List-1
     - Complete any 3 of the problems.
   - List-2
     - Complete any 1 of the problems.
   - Logic-1
     - Complete any 3 of the problems.
   - Logic-2
     - Complete any 1 of the problems.

   So altogether, this week you should complete at least 14 problems on CodingBat, 3 from Warmup-1, 3 from Warmup-2, 3 from List-1, 1 from List-2, 3 from Logic-1, and 1 from Logic-2. A few of the problems on the website may require using concepts we haven't covered yet. You can choose which ones you complete, so if you are stuck on one just try a different one instead.

6. **computepay.py** Write a program to prompt the user for hours and rate per hour and compute their gross pay. Award time-and-a-half for the hourly rate for all hours worked above 40 hours. Put the logic to do the computation of time-and-a-half in a function called `computepay()` and use the function to do the computation. The function should return a value. Use 45 hours and a rate of 10.50 per hour to test the program (the pay should be 498.75). You should use `input()` to read a string and `float()` to convert the string to a number. Do not worry about error checking the user input unless you want to - you can assume the user types numbers properly. Do not name your variable sum or use the sum() function. Here is a skeleton of the code to get you started (*this code will not work, it is just here to show you the structure of the program*):

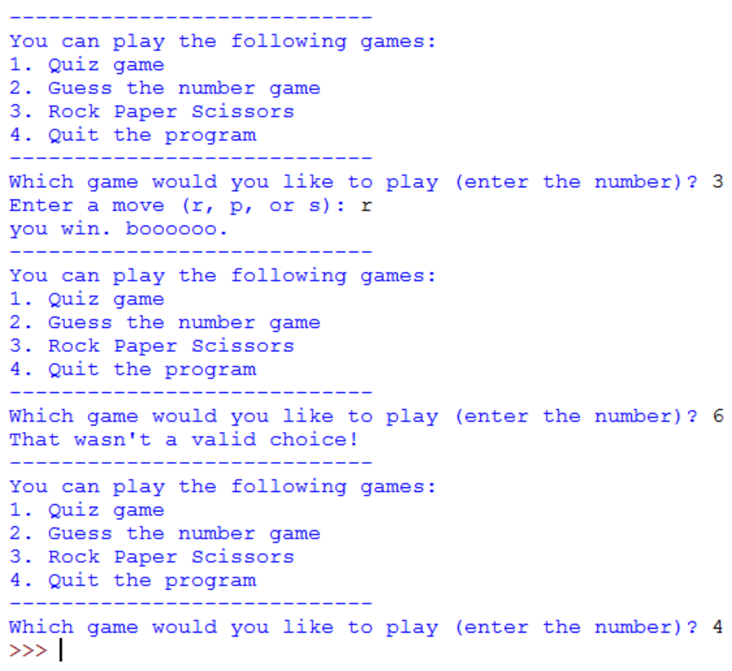
```
def computepay(h, r):
    return 42.37

hrs = input("Enter Hours:")
p = computepay(10,20)
print("Pay", p)
```

7. **mygame.py** For this problem you will choose one of the games from Invent Your Own Computer Games with Python to recreate. You can access the source code for several games at this link: https://is.gd/JnHAnm. Choose one of the following games to recreate and customize:

   - Guess the Number
   - Hangman
   - Tic-Tac-Toe
   - Bagels

   You should read through the corresponding chapter and re-write any code you want to use yourself (meaning don't just copy/paste it).

8. **menu.py** Create a program that presents the user with a choice to play any the games you've made so far. It might look something like this when you run it:



```
----------------------------
You can play the following games:
1. Quiz game
2. Guess the number game
3. Rock Paper Scissors
4. Quit the program
----------------------------
Which game would you like to play (enter the number)? 3
Enter a move (r, p, or s): r
you win. boooooo.
----------------------------
You can play the following games:
1. Quiz game
2. Guess the number game
3. Rock Paper Scissors
4. Quit the program
----------------------------
Which game would you like to play (enter the number)? 6
That wasn't a valid choice!
----------------------------
You can play the following games:
1. Quiz game
2. Guess the number game
3. Rock Paper Scissors
4. Quit the program
----------------------------
Which game would you like to play (enter the number)? 4
>>> |
```

As always your program doesn't have to look exactly like mine, but it should have the same basic functionality. The idea for the code is to place the code for the games you've written previously into functions, and use those functions inside your program loop. Here is an incomplete sketch of how the program should look:

```
# import statements needed go here

def quiz():
    # code from your quiz game goes here
def custom_game():
    # code for your final game goes here
def rock_paper_scissors():
    # code for your RPS game goes here

while True:
    choice = int( input("What would you like to play? ") )
    if choice == 1:
        quiz()
    elif choice == 2:
        guess_the_number()
    elif choice == 3:
        rock_paper_scissors()
    elif choice == 4:
        break
    else:
        print("invalid choice!")
```