COSC 101 Homework 4: Spring 2024

The due date for this homework is **Friday, March 8th, 11pm EDT**.

Introduction

This assignment is designed to give you practice with the following new topics:

- Boolean Expressions
- Conditional Statements, including:
  - Nested Conditionals
  - Chained Conditionals

This assignment also aims to give you **practice with the process of designing optimal solutions to real word problems**. Namely, it separates out the different features of a complex problem and requires you to implement and test them first, before worrying about the entire program. This is the process that experts in the field use and you will be required to enact this process in future homework so **it is very important that you follow the steps of this assignment in the order that they are written**.

NOTE: For this homework, you are not allowed to use while loops, break or continue statements, or other constructs/methods/statements that you have not learned in this course.

Taxes

It is tax season and you are looking for a way to hone in your newly acquired programming skills, so you get a job with a local accounting firm. Your first assignment is to implement a program that is able to calculate the tax bracket and amount of tax that must be paid for different incomes of **married couples filling separately**.

Here is the 2024 Tax Bracket:

Bracket	Rate	Taxable Income Separately	Taxable Income Jointly
1	10%	Up to $11,000	Up to $22,000
2	12%	$11,001 to $44,725	$22,001 to $89,450
3	22%	$44,726 to $95,375	$89,451 to $190,750
4	24%	$95,376 to $182,100	$190,751 to $364,200
5	32%	$182,101 to $231,250	$364,201 to $462,500
6	35%	$231,251 to $346,875	$462,501 to $693,750
7	37%	Over $346,876	Over $693,751

Part 1 - tax bracket

In the `hw4_taxes.py` file, write a function called `compute_bracket` that figures out which tax bracket an income falls into. This function **must** use the values in the lists called `brackets` and `rates` provided to you in `main`. This information must be communicated from `main` to this function. **Do not move these lists from `main`**. For example, for an income of $10,000, `compute_bracket` must return 1.

Testing

Inside `main` write testcases to test all the different cases implemented by `compute_bracket` to ensure that it is implemented correctly. Besides exploring the different cases also pay close attention to boundaries. For example, if the income is 346,875.99 the correct tax bracket is 6. Note that bugs love boundaries (bug is a term that computer scientists use to say that the code is incorrect or behaving in an unexpected way). In other words, it is very easy to make a silly mistake when we write boolean expressions, and we employ testing at the boundary between different cases to ensure that our code is correct.

When you are done with this part you may comment out your tests. **Do NOT delete your tests.**

Part 2 - tax owed

In the `hw4_taxes.py` file, write a function called `compute_tax` that calculates how much it's owed in federal income taxes based on the income. This function **must** call `compute_bracket` to figure out the bracket and then with the bracket compute the tax.

For example, an income of $10,000 falls in the first bracket of 10%, so the taxes owed is $1,000.

Note, the tax rate is not applied uniformly for the entire taxable income. For example, someone with a taxable income of $30,000 falls under the 12% bracket. However, this person will not pay 12% * 30000 in taxes. Instead, the first $11,000 will be taxed at 10%, and only the remaining amount (30000-11000) will be taxed at 12%. You can calculate tax owed by a person with a taxable income of $30000 as follows:

```
11000*0.1 + (30000-11000)*0.12
```

Similarly, someone with a taxable income of $70000 will not pay a flat 22% tax. Instead, their first $11,000 will be taxed at 10%, the amounts $11000 to $44725 will be taxed at 12% and only the remaining amount (70000-44725) will be taxed at 22%. You can calculate tax owed by a person with a taxable income of $70000 as follows:

```
11000*0.1 + (44725-11000)*0.12 + (70000-44725)*0.22
```

Lastly, you can calculate tax owed by a person with a taxable income of $100000 as follows:

```
11000*0.1 + (44725-11000)*0.12 + (95,375-44725)*0.22 + (100000-
95,375)*0.24
```

Note: you are not allowed to use the values in the table directly in your computations. Instead, use the values in `brackets` and `rates`.

Testing

Inside `main` write testcases to test all the different cases implemented by `compute_tax` as well. When you are done with this part you may comment out your tests. **Do NOT delete your tests.**

Part 3 - user interface

In the `hw4_taxes.py` file, write a function called `user_interface` that asks a user for their income. If the income is less or equal to 0 then it prints `Invalid income amount` and ends the program. Otherwise, it then asks the user whether they want to know the bracket rate or the tax they owe.

Below are four examples of inputs and corresponding expected outputs:

```
What is your income? 0
Invalid income amount
```

```
What is your income? 30000.0
Enter r for bracket rate and t for tax:  t
You owe $3380.0 in federal taxes
```

```
What is your income? 525600
Enter r for bracket rate and t for tax:  r
Your tax bracket rate is 37%
```

```
What is your income? 525600
Enter r for bracket rate and t for tax:  s
Invalid option
```

Testing

`user_interface` must be called inside `main`. Test your code with different input values to ensure its overall correctness.

Grading

Your assignment will be graded on two criteria:

1. Correctness [90%]: The correctness part of your grade is broken down as follows:

Category	Portion of grade
Part 1	30%
Part 2	30%
Part 3	30%

2. Program design and style [10%]: style and program design become increasingly important the more complex your program becomes. Adhere to the following guidelines:
   - Variable and function names should be meaningful.
   - Type annotations must be present and correct.
   - Code is clean (for example, remove code you commented out) and follows proper style (for example, limit excessive code, ask yourself, can I do this a simpler way)

Challenge Problem (OPTIONAL)

Challenge problems are entirely optional extensions to the homework. If you complete them successfully, you are rewarded with a sense of accomplishment and a small number of extra points on the homework. Note that any extra points apply only to the homework on which the challenge problem is attempted, not to other homeworks or other grading categories. They are intended for students who want to explore a little further; only pursue the challenge problem after you have successfully completed the homework.

Expand your code to also work for married couples filling jointly. Modify the `user_interface` method to also ask the user if filling jointly or separately and pass that information to the `compute_bracket` and `compute_tax` functions. Hint: see if there is a relation between the brackets for filling jointly vs. separately. You won't receive full credit if you just simply implement a separate `compute_bracket` or `compute_tax` with the rates from the Taxable Income Jointly column. Reflect on how the filling jointly or separately option impacts the brackets ranges.

DO NOT modify the `hw4_taxes.py` file. Instead you should copy the initial code in a new file called `hw4_challenge.py` and modify it according to the challenge.