✖

Homework 1

10 questions

1
point

1.
“And now for something completely different.” - Monty Python

Homework philosophy

In "An Introduction to Interactive Programming in Python" (IIPP), the focus of the class was to learn the basics of programming in Python. In IIPP, there was only a minimal emphasis on building code that was well-structured and efficient. In "Principles of Computing" (PoC), we will follow a more disciplined approach to programming that emphasizes understanding the structure and efficiency of your programs.

While the mini-projects in PoC will focus on coding style and structure, all of the homework assignments in PoC after this one are designed to hone your mathematical skills. These skills are a critical part of learning to think like a computer scientist. To become proficient at solving computational problems, one need not only be skilled at programming, but also skilled at thinking computationally.

As a final note, we remind you to read and think about each homework problem carefully. Several of the problems are "tricky". In particular, your immediate answer may not necessarily be the correct one. Learning to pay close attention to detail is an important skill for successful programmers.

Assessing your knowledge of Python

Our first homework will focus solely on assessing and reviewing your knowledge of basic programming in Python. The standard prerequisite for this class is IIPP. This homework (and the first mini-project) are designed such that students who have finished IIPP should be able to complete these assignments.

However, some students may wish to take this class without having taken IIPP.

If you have not taken or completed IIPP, please use your results on these assignments to help self-assess whether your knowledge of Python is sufficient to be successful in this class. For the assessment provided by this homework to be most accurate, we suggest that you attempt problems 1-8 below **without** testing solutions in Python.

If you are in doubt, you are welcome to continue in the class. However, you may find the class to be very difficult since PoC assumes a broad knowledge of basic Python.

Basic Python knowledge

For the first question in this assignment, enter the type of the Python expression `3.14159` below. Remember that capitalization is important.

float

1 point

2.
An `if` statement can have at most how many `else` parts?

- ○ Unlimited, i.e., 0 or more
- ○ 0
- ○ 1

1 point

3.
Consider the following Python code snippet:

```
def clock_helper(total_seconds):
    """
    Helper function for a clock
    """
    seconds_in_minute = total_seconds % 60
```

Enter the value returned by Python after evaluating the expression `clock_helper(90)` below.

30

---

1 point

4.

In Python, what character always appears at the end of the line before the start of an indented block of code?

- =
- Nothing
- {
- [✓ :]
- ;

---

1 point

5.

Which of the following expressions returns the last character in the non-empty string `my_string`?

- `my_string[len(my_string)]`
- [✓ `my_string[-1]`]
- `my_string.pop()`
- `my_string.last()`

---

1 point

6.
What is the primary difference between a list and a tuple?

- ○ For tuples, all entries must be of the same type. Entries of lists may be of different type.
- ○ There is no diference. The words "list" and "tuple" are interchangeable in Python.
- ○ List are immutable. Tuples are mutable.
- ○ Lists are mutable. Tuples are immutable.

1
point

7.
Consider the following snippet of Python code. What is the value of `val2[1]` after executing this code?

```
val1 = [1, 2, 3]
val2 = val1[1:]
val1[2] = 4
```

3

1
point

8.
Which of the following Python expressions is a valid key in a Python dictionary?

- ☐ set([])
- ☐ []
- ☐ ""
- ☐ False
- ☐ {}
- ☐

None

1 point

9.

Write a function in Python that takes a list as input and repeatedly appends the sum of the last three elements of the list to the end of the list. Your function should loop for 25 times.

```
def appendsums(lst):
    """
    Repeatedly append the sum of the current last three elements
    of lst to lst.
    """
```

If your function works correctly, the following code should print 230:

```
sum_three = [0, 1, 2]
appendsums(sum_three)
print sum_three[10]
```

Enter the value of `sum_three` [20] below.

101902

1 point

10.

First, complete the following class definition:

```
class BankAccount:
    """ Class definition modeling the behavior of a simple bank account """

    def __init__(self, initial_balance):
        """Creates an account with the given balance."""
        …
    def deposit(self, amount):
        """Deposits the amount into the account."""
        …
    def withdraw(self, amount):
        """
        Withdraws the amount from the account. Each withdrawal resulting
        in a negative balance also deducts a penalty fee of 5 dollars
        from the balance.
        """
        …
        …
    def get_balance(self):
        """Returns the current balance in the account."""
        …
    def get_fees(self):
        """Returns the total fees ever deducted from the account."""
        …
```

The `deposit` and `withdraw` methods each change the account balance. The `withdraw` method also deducts a fee of 5 dollars from the balance if the withdrawal (before any fees) results in a negative balance. Since we also have the method `get_fees`, you will need to have a variable to keep track of the fees paid.

Here's one possible test of the class. It should print the values 10 and 5, respectively, since the withdrawal incurs a fee of 5 dollars.

```
my_account = BankAccount(10)
my_account.withdraw(15)
my_account.deposit(20)
print my_account.get_balance(), my_account.get_fees()
```

Copy-and-paste the following much longer test. What two numbers are printed at the end? Enter the two numbers, separated only by spaces.

```
my_account = BankAccount(10)
my_account.withdraw(5)
my_account.deposit(10)
my_account.withdraw(5)
my_account.withdraw(15)
my_account.deposit(20)
my_account.withdraw(5)
my_account.deposit(10)
my_account.deposit(20)
my_account.withdraw(15)
my_account.deposit(30)
my_account.withdraw(10)
my_account.withdraw(15)
my_account.deposit(10)
my_account.withdraw(50)
my_account.deposit(30)
my_account.withdraw(15)
my_account.deposit(10)
my_account.withdraw(5)
my_account.deposit(20)
my_account.withdraw(15)
my_account.deposit(10)
my_account.deposit(30)
my_account.withdraw(25)
my_account.withdraw(5)
my_account.deposit(10)
my_account.withdraw(15)
my_account.deposit(10)
my_account.withdraw(10)
my_account.withdraw(15)
my_account.deposit(10)
my_account.deposit(30)
my_account.withdraw(25)
my_account.withdraw(10)
my_account.deposit(20)
my_account.deposit(10)
my_account.withdraw(5)
my_account.withdraw(15)
my_account.deposit(10)
my_account.withdraw(5)
my_account.withdraw(15)
my_account.deposit(10)
my_account.withdraw(5)
print my_account.get_balance(), my_account.get_fees()
```

-60, 75

Upgrade to submit