MISE Summer Programming Camp

Class 2 Worksheet - Functions and Questions

This worksheet has 3 sections, which should take you around 1 hour to complete. Section 1 contains a series of short answer questions about what we did in class. You'll have to run some code in your IDE to see the result and write it down. Section 2 will introduce a new topic we haven't seen in class and so it is optional. You'll be guided through it since it doesn't assume knowledge of the topic. Finally on Section 3 you'll have to write some programs and submit them to CodeForces (**this section is the most important and you should try to solve it!**)

1. Review Questions

Question 1. In class we learned about Boolean operators and how to use them in if conditions. The following are short pieces of code that use different Boolean operators. Look at them first and think about what you think they should do, then type them into your IDE (Visual Studio or the online one) and run them to see if it matches what you thought. If not, try to think about why it does what it does.

a. `9 == 3*3`
b. `8 < 3*3 or 6 >= 3+3`
c. `8 > 3*3 and 6 == 3+3`
d. `'6' == 6`
e. `int('6') == 6`
f. `not(True)`

Question 2. The reason why we care about Boolean operators is because they are useful in conditionals. The following are short pieces of code that perform some task to some user input. Look at them first and think about what you think they should do, then type them into your IDE and run them to see if it matches what you thought. If not, try to think about why it does what it does.

a.
```
val = int(input())
if val > 0:
    print("Positive")
elif val < 0:
    print("Negative")
else:
    print("Zero")
```

b.
```
val = int(input())
if val % 2 == 0:
    print("Even")
else
    print("Odd")
```

c.
```
unit = input()
val = int(input())
if unit == 'Celsius':
    print(val, 'Celsius is', int(val * 1.8 + 32), 'Fahrenheit')
else:
    print(val, 'Fahrenheit is', int((val - 32) / 1.8), 'Celsius')
```

d.
```
val_1 = int(input())
val_2 = int(input())
val_3 = int(input())
if val_1 <= val_2 and val_2 <= val_3:
    print("The input is sorted")
else:
    print("The input isnt sorted")
```

Question 3. In class we also learned about functions, which allow us to organize code into blocks that perform a certain task. In each of the following you'll have to implement a function to solve a short task. After the description of each task there is a small code that achieves that as well as an example function call. Read the task and think about how you'd solve it, then read the solution and see if it matches what you thought. Type up the solution and run it to verify that it does what it's supposed to do.

a. Write a function that given a string prints "Hello" followed by the input.
```
def sayHello(x):
    print("Hello", x)
sayHello("Ghana")
```

b. Write a function that given a number returns its absolute value.
```
def abs(x):
    if x < 0:
        return -x
    else:
        return x
print(abs(-4))
```

c. Write a function that given two integers *a* and *b* returns true if *a* is a multiple of *b*.

```
def multiple(a, b):
    return a % b == 0
print(multiple(10, 2))
```

d. Write a function that given three integers *a, b, c*, returns true if you can form a triangle with sides of lengths *a, b, c*.

```
def triangle(a, b, c):
    if a >= b + c or b >= a + c or c >= a + b:
        return False
    else:
        return True
print(triangle(4, 2, 6))
```

2. Code Debugging (Optional)

(This section is optional. It teaches some new interesting topics, but you don't need to solve it in order to follow the rest of the class)

When we write code we are going to make mistakes eventually. We call these mistakes "bugs" and we call finding and fixing them "debugging". Debugging is part art, part science, so we are going to practice that in this section.

One thing we'll have to learn to be able to debug programs is reading error messages. If you tried writing enough programs you have probably seen some error saying something like "File file.py, line 3 SyntaxError: …". It is important to read these because they tell us where there is something wrong. In the next question we will look at various programs that have some bug that triggers a different error.

Question 1. The following pieces of code have some error. If you run them you'll see an error message. Your task is to read the error message to try to understand where the error is and then think about how you'd fix the error. Note that there are many more error types, this is just a small sample.

a.
```
print("Hello"
```
The error message: *SyntaxError: unexpected EOF while parsing*
(A Syntax Error is a violation of the rules of what makes a valid Python program. It usually happens when there is a mismatched parentheses or quotes, when there is a colon missing after an if statement or others).

b.

```python
def fun(x):
    return x + 1
print(fun())
```

The error message: *TypeError: fun() missing 1 required positional argument: 'x'*.
(A Type Error happens when a function or operation is applied to an object of an incorrect type, or when there is a wrong number of parameters in a function call).

c.

```python
if 4 < 5:
print("Hello")
```

The error message: *IndentationError: expected an indented block.*
(An Indentation Error happens when there is an incorrect indentation).

d.

```python
a = int(input())
print(b + 3)
```

The error message: *NameError: name 'b' is not defined*
(A Name Error happens when a variable is not found).

Question 2. Another important skill related to debugging is finding why a program gives a result different from what is expected. There are different tricks you'll learn, like to print the value of variables in different places to ensure that they are what you expected at different points of the program. To practice that, in the following we have some programs that solve the problems of Question 3 of Section 1 but they have a bug. Find the bug!

a.

```python
def sayHello(x):
    print("Hello")
    print(x)
```

b.

```python
def abs(x):
    if x > 0:
        return -x
    else:
        return x
```

c.

```python
def multiple(b, a):
    return a % b == 0
```

d.

```python
def triangle(a, b, c):
    return a >= b + c or b >= c + a or c >= a + b
```

3. Coding Questions

You should see a new problem set in codeforces called “Class 2 Problems” (https://codeforces.com/group/K1Fxw6skwV/contest/442336). There are 4 problems available, but only the first 2 (“Budget” and “All in order”) are mandatory. The other 2 (“Back to school” and “All in order - Hard version”) are harder and optional, so if you feel up to it you can give it a go, but you don’t have to.