H2 Computing Practical Worksheet 1

Set 1 – Please attempt the following questions WITHOUT the use of a computer.

1 What is the output of the following code?

```
a = 5
b = a + 7
a = 10
print(str(b))
```

2 The Python interpreter has strict rules for variable names. Which of the following are legal Python variable names? If the name is not legal, state the reason.

i. `and`
ii. `_and`
iii. `Var`
iv. `var1`
v. `1var`
vi. `my-name`
vii. `your_name`
viii. `COLOUR`

3 It is important to know the type of the value that a variable refers to – e.g., this would allow us to use the correct operators. Python automatically infers the type from the value you assign to a variable. Write down the type of the values stored in each of the variables below. Pay special attention to punctuation: values are not always the type they seem!

i. `a = False`
ii. `b = 3.7`
iii. `c = 'Alex'`
iv. `d = 7`
v. `e = 'True'`
vi. `f = 12 ** 3`
vii. `g = '17'`
viii. `h = True`
ix. `i = '3.14159'`
x. `j = 12 / 27`
xi. `k = 2.0 / 1`
xii. `l = (5 == "5")`
xiii. `m = str((-4 + abs(-5) / 2 ** 3) + 321 - ((64 / 16) % 4) ** 2)`

To verify your answers, you can use the interactive Python shell (as shown below). However, first try to do the exercise without the help of the shell.

```
>>> x = 100
>>> type(x)
<type 'int'>
>>>
```

4 Boolean operators can seem tricky at first, and it takes practice to evaluate them correctly. Write the value (True or False) produced by each expression below, using the assigned values of the variables a, b, and c. Try to do this without using your interpreter, but you should check yourself when you think you've got it.

Hint: Work from the inside out, starting with the innermost expressions, similar to arithmetic.

Let:

```
a = False
b = True
c = False
```

Would the expressions below evaluate to `True` or `False`?

i. `b and c`
ii. `b or c`
iii. `not a and b`
iv. `(a and b) or not c`
v. `not b and not (a or c)`

5 What is the output for each of the following lines of code?

i.
```
print(5 == 5.0)
```
ii.
```
print(float(1/2))
```
iii.
```
print(float(1)/2)
```
iv.
```
print(5 == "5")
```
v.
```
print("sdf" != "sdf")
```
vi.
```
print(True and (False or not True))
```
vii.
```
print(str(53) + str(True))
```
viii.
```
a = 20
print(str(15 - (a - 15)) + ", ", end = " ")
a = 10
print(15 - (a - 15))
```

6 Consider the following code.

```
x = input("Enter a positive integer value for x: ")
if x > 10 and x % 6 = 3:
    print("a", end = "")
elif x > 10 and x < 20:
    print("b", end = "")
else:
    print("c", end = "")
print("")
```

Given the following outputs, determine the corresponding value(s) for `x` that would have generated that output. If no values of a can produce the listed output, write none.

i. `ab`
ii. `a`
iii. `b`
iv. `c`

7 What is the output for each of the following pieces of code?

Note: if the code does not terminate, write infinite loop.

i.
```
a = 5
while a < 8:
    print("X", end = "")
print("")
```

ii.
```
a = -1
while a < 3:
    print("X" , end = "")
    a = a + 1
print("")
```

iii.
```
a = 1
while a % 7 != 0:
    if a % 2 == 0:
        print("O" , end = "")
    if a == 2:
        print("X" , end = "")
    a = a + 1
print("")
```

8 What is the output for each of the following pieces of code?

i.
```
num = 10
while num > 3:
    print(num)
    num = num - 1
```

ii.
```
divisor = 2
for i in range(0, 10, 2):
    print(i // divisor)
```

iii.
```
num = 10
while True:
    if num < 7:
        break
    print(num)
    num -= 1
```

iv.
```
count = 0
for letter in 'Snow!':
    print('Letter #' + str(count), ' is ', letter)
    count += 1
```

9 What is the output for each of the following pieces of code?

i.
```
keep_going = True
a = 0
b = 0
while keep_going:
    print("O", end = "")
    a = a + 5
    b = b + 7
    if a + b >= 24:
        keep_going = False
print("")
```

ii.
```
keep_going = True
a = 0
b = 0
while keep_going:
    print("O" , end = "")
    if a + b >= 24:
        keep_going = False
    a = a + 5
    b = b + 7
print("")
```

iii.
```
keep_going = True
a = 0
b = 0
while keep_going:
    print("O" , end = "")
    a = a + 5
    b = b + 7
    if a + b > 24:  # note that ">" is used here … vs ">=" in (i)
        keep_going = False
print("")
```

iv.
```
keep_going = True
a = 0
b = 0
while keep_going:
    print("O" , end = "")
    if a + b > 24:
        keep_going = False  # note that ">" is used here … vs ">=" in (ii)
    a = a + 5
    b = b + 7
print("")
```

10 What is the output for each of the following pieces of code? Note: if the code does not terminate, write infinite loop.

i.
```
a = 0
while a < 3:
    while True:
        print("X", end = "")
        break
    print("O", end = "")
    a = a + 1
print("")
```

ii.
```
a = 1
while a < 3:
    while a < 3:
        print("O", end  = "")
    a = a + 1
print("")
```

iii.
```
a = 1
while a < 3:
    if a % 2 == 0:
        b = 1
        while b < 3:
            print("X", end = "")
            b = b + 1
    print("O", end = "")
    a = a + 1
print("")
```

11 The following code loops infinitely when run, Fix the code so that its output is "OOOXOXOO".

```
a = 1
while a < 3:
    b = 1
    while b < 3:
        if a == 2:
            print("X", end = "")
        print "O",
        b = b + 1
    print("O", end = "")
print("")
```

Set 2 – Please attempt the following questions by implementing the solutions using Python 3.

12 Entering and printing your name.

The goal of this programming exercise is simply to get you more comfortable with using IDLE, and to begin using simple elements of Python. Standard elements of a program include the ability to print out results (using the **`print`** function), the ability to read input from a user at the console (e.g., using the **`input`** function), and the ability to store values in a variable, so that the program can access that value as needed.

The problem:

- Ask the user to enter his/her last name.
- Ask the user to enter his/her first name.
- Print out the user's last and first names in that order.

Example interaction between the program and the user (note: words printed in blue are from the computer, based on the program, while the words in black are a user's input - the colours are simply here to help you distinguish the two components):

```
>>>
Enter your last name:
Daren
Enter your first name:
Ler
Ler
Daren
>>>
```

13 Printing out multiplication tables.

Your program should request in integer input, between 1 and 12 (inclusive), from the user, and then output the multiplication table for that output (calculating the 1st to 12th multiple of that input number).

The problem:

- Request an integer input from the user (between 1 and 12 inclusive).
  Let us denote this input as `i`.
- Print out the products of `i * k`, such that `k` = 1, …, 12.

Example interaction:

```
>>>
Please input an integer between 1 and 12 (inclusive):
7
Multiples of 7:
7
14
21
28
35
42
49
56
63
70
77
84
>>>
```

14 Unit converter.

Write a program that helps the user to convert a given quantity from one unit of (distance, area or volume) measurement to another.

The problem:

- Request an quantity from the user.
- Have the user specify if the quantity is a length, area or volume.
- Have the user select the base unit of measurement from the following list:
  - **mm**,
  - **cm**,
  - **m**, or
  - **km**.
- Then have the user select the desired base unit of measurement that the given quantity should be converted to.
- Print out converted quantity (including the new unit of measurement).

Example interaction:

```
>>>
Input a quantity: 10
Select either length, area or volume: area
Select the base unit of measurement (mm, cm, m, km): m
Select which base unit of measurement to convert to (mm, cm, m, km): cm
10 m^2 = 100000 cm^2
>>>
```

15 Rock, Paper, Scissors, Lizard, Spock.

In this exercise, you are going to practice using conditionals (`if`, `elif`, `else`). You will write a small program that will determine the result of a Rock, Paper, Scissors, Lizard, Spock game.

For the rules of the game, please which the following clip from "The Big Bang Theory":
https://www.youtube.com/watch?v=_PUEoDYpUyQ

The problem:

- Write a program that will first request the input for Player 1 and Player 2's choices.
- Next, the program will print out the result (i.e., who the winner is).

Example interaction:

```
>>>
Player1? rock
Player2? scissors
Player 1 wins.
>>>
```

16 Vending machine change.

Your program should simulate the calculation of the change that a vending machine must return. The vending machine only sells drinks that cost $1.20 and $0.80. The machine only accepts 10-cent, 20-cent, 50-cent, and 1-dollar coins. It must return the least possible number of coins.

The problem:

- The vending machine only sells drinks that either cost $0.80 or $1.20
- The vending machine may only accept 10-cent, 20-cent, 50-cent, and 1-dollar coins
- The vending machine has an unlimited supply of the coins it accepts
- There is no limit on the amount of money a person can insert before making a purchase (and then receiving change). However, we assume that the person has already inserted all the money that he or she wants to, and has already chosen the drink to purchase.
- Your program must ask for 5 inputs:
    - The amount of 10-cent coins that the user has inserted.
    - The amount of 20-cent coins that the user has inserted.
    - The amount of 50-cent coins that the user has inserted.
    - The amount of 1-dollar coins that the user has inserted.
    - The choice of a $0.80 or $1.20 drink (note, you should expect that the value of all the coins inserted must be greater than the cost of the drink selected).
- Calculate and print the number each type of coin to return as change such that the total number of coins returned by the machine is the least possible.

Example interaction:

```
>>>
Enter the number of 10-cent coins inserted:
5
Enter the number of 20-cent coins inserted:
5
Enter the number of 50-cent coins inserted:
1
Enter the number of 1-dollar coins inserted:
3
Please enter either the price of the drink, i.e., 0.8 or 1.2:
1.2
Total inserted: $5.0
The machine returns a total of $3.8, in the form of:
3 x 1-dollar coin(s)
1 x 50-cent coin(s)
1 x 20-cent coin(s)
1 x 10-cent coin(s)
>>>
```

17 Birthdays and Zeller's algorithm.

Zeller's algorithm computes the day of the week on which a given date will fall (or fell). Use the description of Zeller's algorithm, given below, to write a Python script that computes the day of the week on which the user's birthday fell in the year they were born and prints the result to the screen.

Zeller's algorithm:

- We first define the following variables (which each stores a non-negative integer):

`month`	This variable corresponds to the month of the year (ranging from 3 to 14), with March having the value 3, April the value 4, …, December the value 12, and January and February (of the following year) being counted as months 13 and 14.
`year`	This variable corresponds to the year of the century (e.g., `year = 89` for the year 1989; `year = 5` for the year 2005). Note that if `A == 13` or `A == 14`, then you should adjust `C` such that `C = C - 1`.
`yy`	This variable corresponds to the last 2 digits of `year`. For examples, `yy == 89` for the year 1989 and `yy == 5` for the year 2005.
`century`	This variable corresponds to the first 2 digits of `year`. For examples, `century == 19` for the year 1989 and `century == 20` for the year 2005.
`day`	This variable corresponds to the day of the month, i.e., an integer values between 1 and 31.

- The day of the week is then calculated using the following formula:

$$\left[\text{day} + \left\lfloor\frac{13(\text{month}+1)}{5}\right\rfloor + \text{yy} + \left\lfloor\frac{\text{yy}}{4}\right\rfloor + \left\lfloor\frac{\text{century}}{4}\right\rfloor - 2 \times \text{century}\right] \bmod 7$$

- To determine the actual day of the week, the resultant value is then determined by referring to the following table:

Resultant Value	Day of the Week
0	Saturday
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday

The problem:

- Have the user input their birth date.
- Use Zeller's algorithm to compute the day of the week on which they were born.
- Print out that day.

Example interaction:

```
>>>
Day Born? 20
Month Born? 5
Year Born? 2005
Your birthday fell on: Friday
>>>
```

18 Caesar cipher.

The goal of this exercise is to write a cyclic cipher to encrypt messages. This type of cipher was used by Julius Caesar to communicate with his generals. It is very simple to generate but it can actually be easily broken and does not provide the security one would hope for.

The key idea behind the Caesar cipher is to replace each letter by a letter some fixed number of positions down the alphabet. For example, if we want to create a cipher shifting by 3, you will get the following mapping:

- Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ
- Cipher: DEFGHIJKLMNOPQRSTUVWXYZABC

To be able to generate the cipher above, we need to understand a little bit about how text is represented inside the computer. Each character has a numerical value and one of the standard encodings is ASCII (American Standard Code for Information Interchange). It is a mapping between the numerical value and the character graphic. For example, the ASCII value of 'A' is 65 and the ASCII value of 'a' is 97. To convert between the ASCII code and the character value in Python, you can use the following code:

```
letter = 'a'
# converts a letter to ascii code
ascii_code = ord(letter)
# converts ascii code to a letter
letter_res = chr(ascii_code)
print ascii_code, letter_res
```

The problem:

- Create a file called cipher.py. Start your program by asking the user for a phrase to encode and the shift value. Then begin the structure of your program by entering in this loop (we'll build on it more in a bit):

```
encoded_phrase = ''
for c in phrase:
   encoded_phrase = encoded_phrase + c
```

  What does this loop do? Make sure you understand what the code does before moving on!

- Now modify the program above to replace all the alphabetic characters with 'x'. For example:

```
Enter sentence to encrypt: Mayday! Mayday!
Enter shift value: 4
The encoded phrase is: Xxxxxx! Xxxxxx!
```

  We are going to apply the cipher only to the alphabetic characters and we will ignore the others.

- Now modify your code, so that it produces the encoded string using the cyclic cipher with the shift value entered by the user. Let's see how one might do a cyclic shift. Let's say we have the sequence:

```
012345
```

  If we use a shift value of 4 and just shift all the numbers, the result will be:

```
456789
```

- We want the values of the numbers to remain between 0 and 5. To do this we will use the modulus operator. The expression x % y will return a number in the range 0 to y - 1 inclusive, e.g., 4 % 6 = 4, 6 % 6 = 0, 7 % 6 = 1. Thus the result of the operation will be:

  ```
  450123
  ```

  Hint: Note that the ASCII value of 'A' is 65 and 'a' is 97, not 0. So you will have to think how to use the modulus operator to achieve the desired result. Apply the cipher separately to the upper and lower case letters.

Example interaction:

```
>>>
Enter sentence to encrypt: Mayday! Mayday!
Enter shift value: 4
The encoded phrase is: Qechec! Qechec!
>>>
```

19 Significant figure calculator.

Your program should request in number input (i.e., either int or float) from the user, as well as the number of significant figures desired, and then output the first value input rounded to the number of significant figures specified.

The problem:

- Request a number from the user.
- Request the number of significant figures desired.
- Print out the first input corrected to the number of significant figures specified.

Example interaction:

```
>>>
Input a number: 3.14159265359
Number of significant figures: 10
The value 3.14159265359 rounded to 10 significant figures is: 3.141592654
>>>
```

20 Prime numbers.

The problem:

- Request an integer input from the user.
- Compute the `i`-th prime number, such that `i` is the integer input by the user above.
- Print the `i`-th prime number.
-

You may use the calculator on the following website to check your answer:
http://www.bigprimes.net/archive/prime/

Example interaction:

```
>>>
Select the prime number you want (i.e., an integer): 101
Prime number 101: 547
>>>
```

Additional Practice Questions

- Types, Values, Expressions; Variables and Binding
  https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-01sc-introduction-to-electrical-engineering-and-computer-science-i-spring-2011/python-tutorial/part-i/

- Functions and Scope
  https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-01sc-introduction-to-electrical-engineering-and-computer-science-i-spring-2011/python-tutorial/part-2/

- Using if, else, and while
  https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-01sc-introduction-to-electrical-engineering-and-computer-science-i-spring-2011/python-tutorial/part-3/

- Loops and List Comprehensions
  https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-01sc-introduction-to-electrical-engineering-and-computer-science-i-spring-2011/python-tutorial/part-5/

- Arrays as Lists of Lists
  https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-01sc-introduction-to-electrical-engineering-and-computer-science-i-spring-2011/python-tutorial/part-6/

- Association Lists
  https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-01sc-introduction-to-electrical-engineering-and-computer-science-i-spring-2011/python-tutorial/part-7/

- (Optional Exercise) Quadratic Roots
  https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-01sc-introduction-to-electrical-engineering-and-computer-science-i-spring-2011/python-tutorial/part-4/

Even More Additional Practice Questions

If you would like additional practice, you may also attempt the following:

1	https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/assignments/pset1a.pdf
2	https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/assignments/pset1a.pdf
3	https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/assignments/pset3.pdf
4	https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/assignments/pset4.pdf
5	https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/assignments/pset5.pdf
6	https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/assignments/pset6.pdf
7	https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/assignments/pset7.pdf

Additional References

- MIT Introduction to Computing Video Lecture 1 (start watching from 16:10):
  https://www.youtube.com/watch?v=k6U-i4gXkLM
- "How to Think Like a Computer Scientist " Chapter 1:
  http://www.greenteapress.com/thinkpython/thinkCSpy/html/chap01.html
- MIT Introduction to Computing Video Lecture 2:
  https://www.youtube.com/watch?v=Pij6J0HsYFA
- "How to Think Like a Computer Scientist" Chapter 2:
  http://www.greenteapress.com/thinkpython/thinkCSpy/html/chap01.html
- "Python Programming" Section on Variables and Strings:
  https://en.wikibooks.org/wiki/Python_Programming/Variables_and_Strings
- "Python Programming" Section on Input and Output (excluding Sections 1.3 and 2.3):
  https://en.wikibooks.org/wiki/Python_Programming/Input_and_Output
- "How to Think Like a Computer Scientist" Sections 4.1 – 4.7:
  http://www.greenteapress.com/thinkpython/thinkCSpy/html/chap04.html
- "How to Think Like a Computer Scientist" Sections 6.1 – 6.2:
  http://www.greenteapress.com/thinkpython/thinkCSpy/html/chap06.html
- "Python Programming" Section on Loops:
  https://en.wikibooks.org/wiki/Python_Programming/Loops
- "How to Think Like a Computer Scientist" Chapter 7:
  http://www.greenteapress.com/thinkpython/thinkCSpy/html/chap07.html

There are many other exceptional resources available on the internet. If you find any, please share with the class via group chat.