Part 1

A python function "avg" returns the average of three values, as shown below:

def avg (num1, num2, num3):

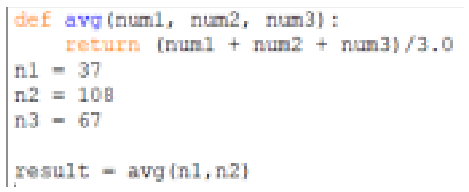
return (num1 + num2 + num3) / 3.0

n1 = 37, n2 = 108, n3 = 67

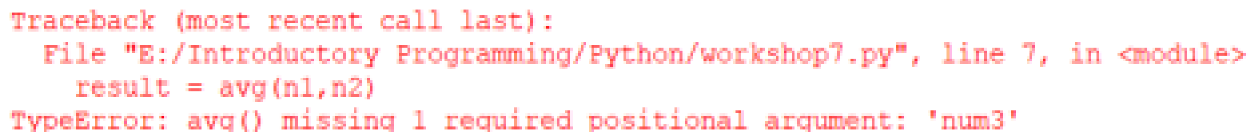
Define the function and variable declarations given above in IDLE shell and execute the following expressions. Which of the statements are valid?

Note down the response to each. Do they differ from what you would expect?

(A) *result = avg(n1, n2)*

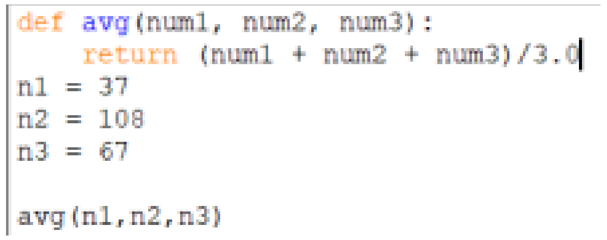


```
def avg(num1, num2, num3):
    return (num1 + num2 + num3)/3.0
n1 = 37
n2 = 108
n3 = 67

result = avg(n1,n2)
```



```
Traceback (most recent call last):
  File "E:/Introductory Programming/Python/workshop7.py", line 7, in <module>
    result = avg(n1,n2)
TypeError: avg() missing 1 required positional argument: 'num3'
```

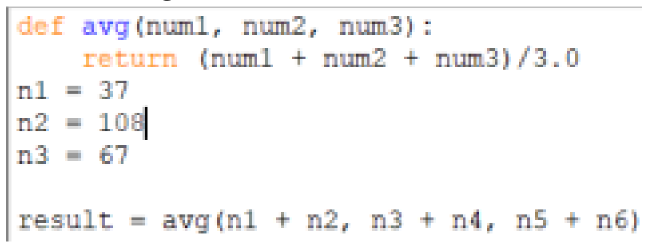
The result is as I expected because n3 is not called which gives error. (B) *avg(n1, n2, n3)*



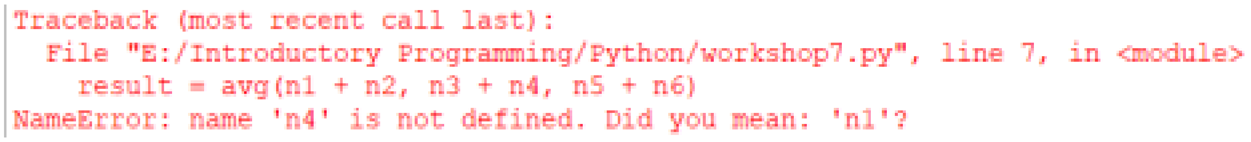
```
def avg(num1, num2, num3):
    return (num1 + num2 + num3)/3.0
n1 = 37
n2 = 108
n3 = 67

avg(n1,n2,n3)
```

The result does not give error as all parameters are called.

(C) *result = avg(n1 + n2, n3 + n4, n5 + n6)*



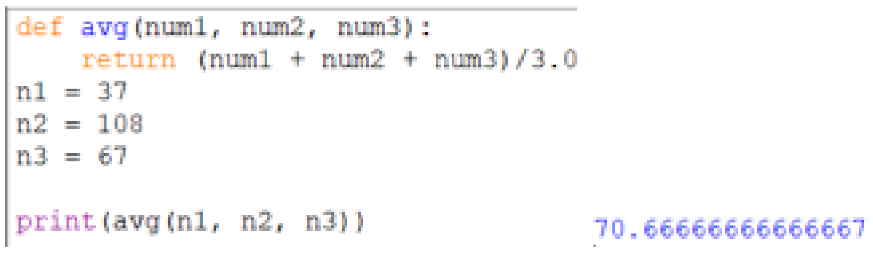
```
def avg(num1, num2, num3):
    return (num1 + num2 + num3)/3.0
n1 = 37
n2 = 108
n3 = 67

result = avg(n1 + n2, n3 + n4, n5 + n6)
```



```
Traceback (most recent call last):
  File "E:/Introductory Programming/Python/workshop7.py", line 7, in <module>
    result = avg(n1 + n2, n3 + n4, n5 + n6)
NameError: name 'n4' is not defined. Did you mean: 'n1'?
```

Gives error as n4, n5 and n6 are not defined.

(D) *print(avg(n1, n2, n3))*

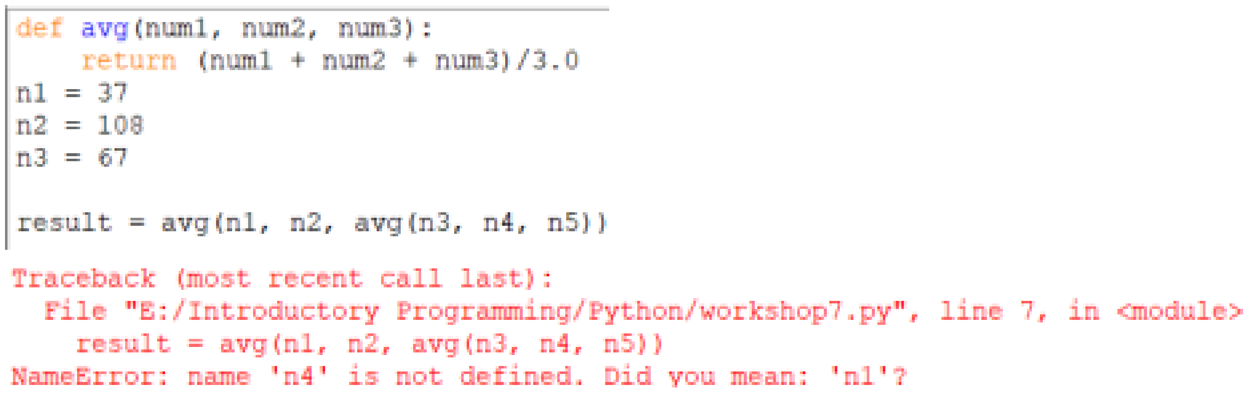


```
def avg(num1, num2, num3):
    return (num1 + num2 + num3)/3.0
n1 = 37
n2 = 108
n3 = 67

print(avg(n1, n2, n3))
```

```
70.66666666666667
```

Prints the output as all parameters are called.

(E) *result = avg(n1, n2, avg(n3, n4, n5))*



```
def avg(num1, num2, num3):
    return (num1 + num2 + num3)/3.0
n1 = 37
n2 = 108
n3 = 67

result = avg(n1, n2, avg(n3, n4, n5))
```

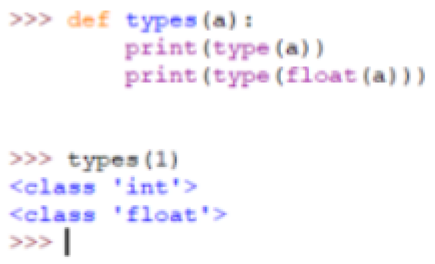
```
Traceback (most recent call last):
  File "E:/Introductory Programming/Python/workshop7.py", line 7, in <module>
    result = avg(n1, n2, avg(n3, n4, n5))
NameError: name 'n4' is not defined. Did you mean: 'n1'?
```

Dosenot give output as you cannot call fuction inside a function and n4 and n5 are not defined.

Part 2

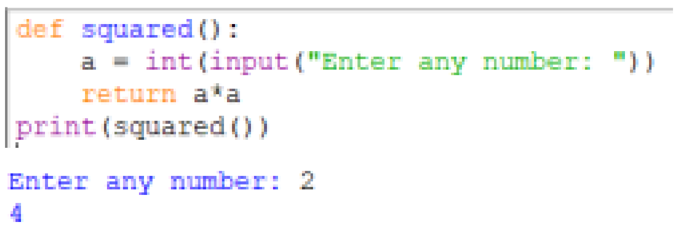
Define a function:

(A) *types( )* that prints a given value both as a float and an integer



```
>>> def types(a):
        print(type(a))
        print(type(float(a)))


>>> types(1)
<class 'int'>
<class 'float'>
>>> |
```

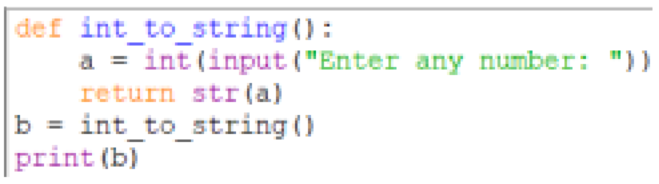
(B) *squared( )* that take an integer and returns the value squared.



```
def squared():
    a = int(input("Enter any number: "))
    return a*a
print(squared())
```

```
Enter any number: 2
4
```

(C) *int_to_string( )* that takes an integer value and returns it as a string.



```
def int_to_string():
    a = int(input("Enter any number: "))
    return str(a)
b = int_to_string()
print(b)
```

```
Enter any number: 2
2
```

(D) *hello_world( )* that takes a parameter name and displays the following output to the console: "Hello World, my name is name".

Code:

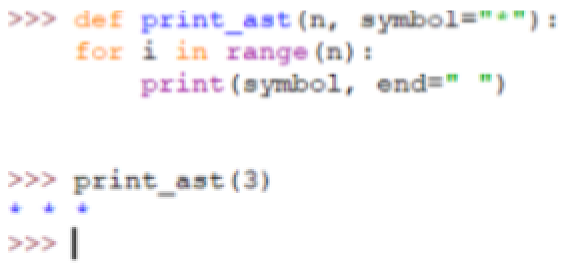
```
def hello_world(name):
    print ("Hello World, my name is "+name)

hello_world("Nabodip")
```

Output:

```
Hello World, my name is Nabodip
```

(E) *print_ast( )* that takes an integer value *n* and a string value symbol, with a default value of "*". This character should be printed *n* times to the console.

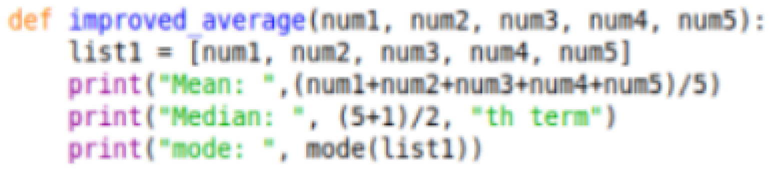
Code:



```
>>> def print_ast(n, symbol="*"):
    for i in range(n):
        print(symbol, end=" ")


>>> print_ast(3)
* * *
>>> |
```

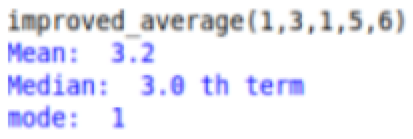
(F) *improved_average( )* that takes five integer parameters. It should return the mode, median and mean values of the numbers passed to the function.

Code:



```
def improved_average(num1, num2, num3, num4, num5):
    list1 = [num1, num2, num3, num4, num5]
    print("Mean: ",(num1+num2+num3+num4+num5)/5)
    print("Median: ", (5+1)/2, "th term")
    print("mode: ", mode(list1))
```

Output:

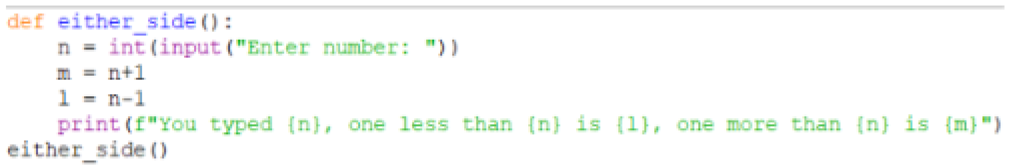


```
improved_average(1,3,1,5,6)
Mean:  3.2
Median:  3.0 th term
mode:  1
```

(G) *either_side( )* which when passed an integer value also prints the values which are one less and one more than that value e.g.

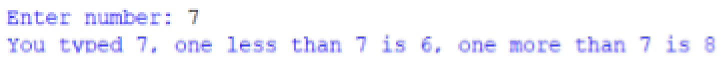
"You typed 4, one less than 4 is 3, one more than 4 is 5"

Code:



```python
def either_side():
    n = int(input("Enter number: "))
    m = n+1
    l = n-1
    print(f"You typed {n}, one less than {n} is {l}, one more than {n} is {m}")
either_side()
```

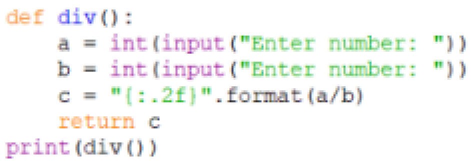
Output:



```
Enter number: 7
You typed 7, one less than 7 is 6, one more than 7 is 8
```

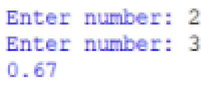
Part 3

1. Create a function that prompts the user for two integer values and displays the results of the first number divided by the second to two decimal places.
   Code:



```python
def div():
    a = int(input("Enter number: "))
    b = int(input("Enter number: "))
    c = "{:.2f}".format(a/b)
    return c
print(div())
```

Output:



```
Enter number: 2
Enter number: 3
0.67
```

2. Create a Python program called calculator with functions to perform the following arithmetic calculations, each should take two decimal parameters and return the result of the arithmetic calculation in question.

A. Addition

B. Subtraction
C. Multiplication

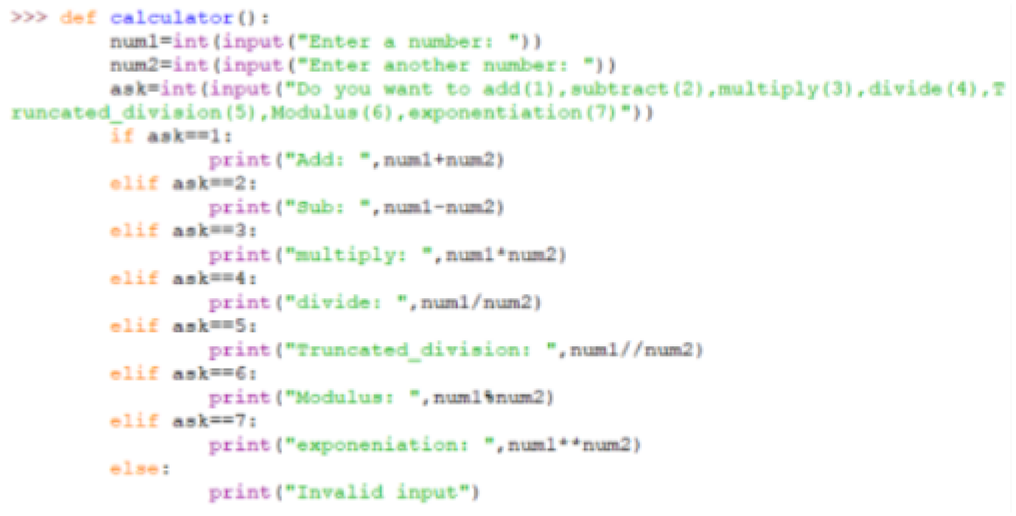
D. Division

E. Truncated division

F. Modulus

G. Exponentiation

Code:



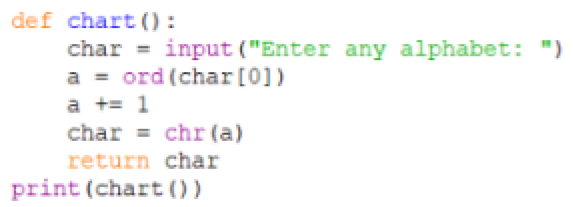
```
>>> def calculator():
	num1=int(input("Enter a number: "))
	num2=int(input("Enter another number: "))
	ask=int(input("Do you want to add(1),subtract(2),multiply(3),divide(4),T
runcated_division(5),Modulus(6),exponentiation(7)"))
	if ask==1:
		print("Add: ",num1+num2)
	elif ask==2:
		print("Sub: ",num1-num2)
	elif ask==3:
		print("multiply: ",num1*num2)
	elif ask==4:
		print("divide: ",num1/num2)
	elif ask==5:
		print("Truncated_division: ",num1//num2)
	elif ask==6:
		print("Modulus: ",num1%num2)
	elif ask==7:
		print("exponeniation: ",num1**num2)
	else:
		print("Invalid input")
```

3. Go back and add multi-line Docstrings to each of the functions you defined in the previous question. Use the help function to check them afterwards.

4. Take a character input from the user and convert the character into next character in the alphabetical order. Use *ord( )* and *chr( )* ASCII functions.

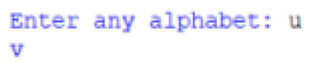
[Hint: for input of ‘a’, print ‘b’ and so on]

Code:



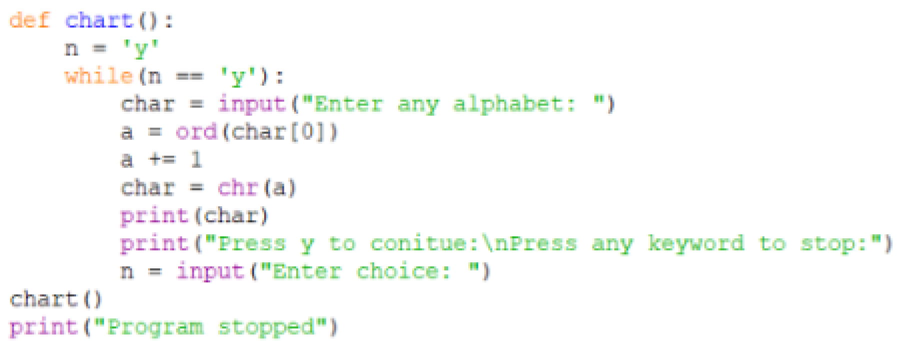
```
def chart():
    char = input("Enter any alphabet: ")
    a = ord(char[0])
    a += 1
    char = chr(a)
    return char
print(chart())
```

Output:



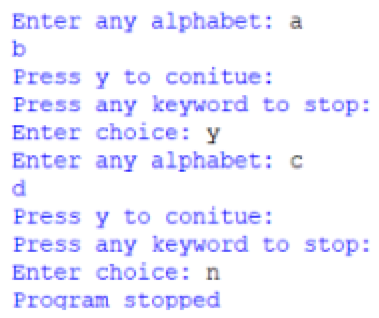
```
Enter any alphabet: u
v
```

5. Use a looping statement to take user’s choice to continue for the above program. Code:



```
def chart():
    n = 'y'
    while(n == 'y'):
        char = input("Enter any alphabet: ")
        a = ord(char[0])
        a += 1
        char = chr(a)
        print(char)
        print("Press y to conitue:\nPress any keyword to stop:")
        n = input("Enter choice: ")
chart()
print("Program stopped")
```

Output:



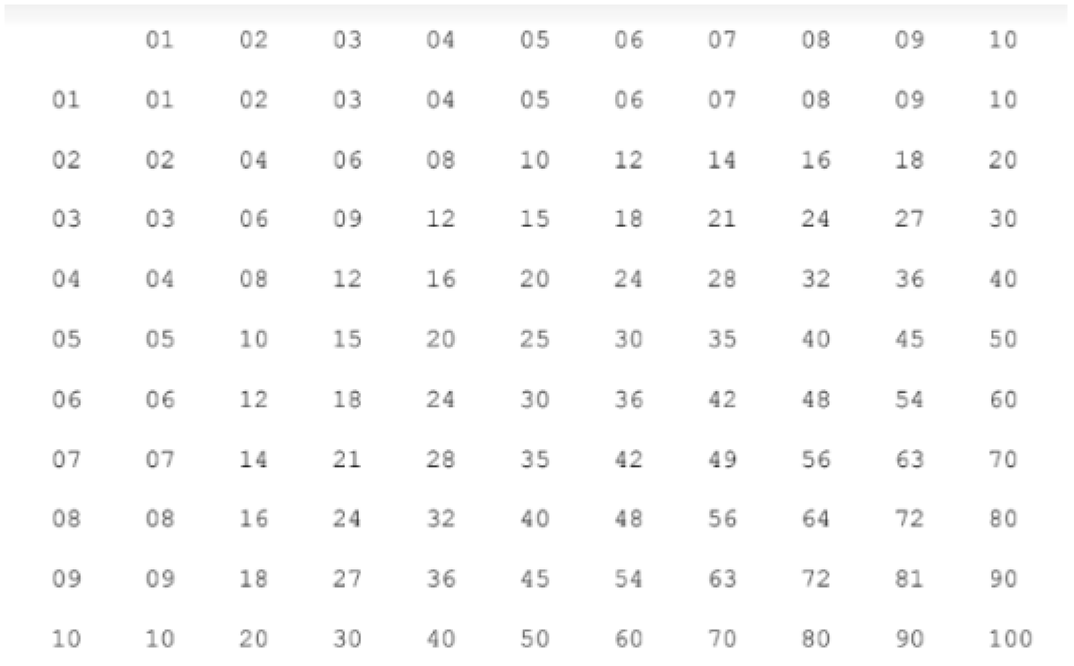
```
Enter any alphabet: a
b
Press y to conitue:
Press any keyword to stop:
Enter choice: y
Enter any alphabet: c
d
Press y to conitue:
Press any keyword to stop:
Enter choice: n
Program stopped
```

Part 4 (Optional)

You will need to understand control structures to complete the following questions. Therefore, you should carry out some independent research before attempting this. However, it will also be covered next week in class.

1. Create a function *multiplication_table( )*. It should take a single parameter *n*, which determines the size of the grid to be output e.g.

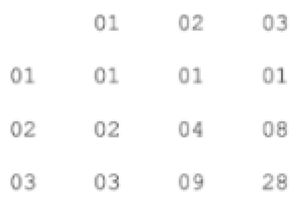
multiplication_table(10)



```
      01   02   03   04   05   06   07   08   09   10

01    01   02   03   04   05   06   07   08   09   10

02    02   04   06   08   10   12   14   16   18   20

03    03   06   09   12   15   18   21   24   27   30

04    04   08   12   16   20   24   28   32   36   40

05    05   10   15   20   25   30   35   40   45   50

06    06   12   18   24   30   36   42   48   54   60

07    07   14   21   28   35   42   49   56   63   70

08    08   16   24   32   40   48   56   64   72   80

09    09   18   27   36   45   54   63   72   81   90

10    10   20   30   40   50   60   70   80   90   100
```

2. Modify your existing function to take an additional parameter: *power*, with a default value of *False*. If a value of *True* is provided, your multiplication table should instead apply the top row as *powers* instead of multiplying the numbers.

multiplication_table(3, True)



```
      01   02   03

01    01   01   01

02    02   04   08

03    03   09   28
```