Part 1

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

def avg (num1, num2, num3):

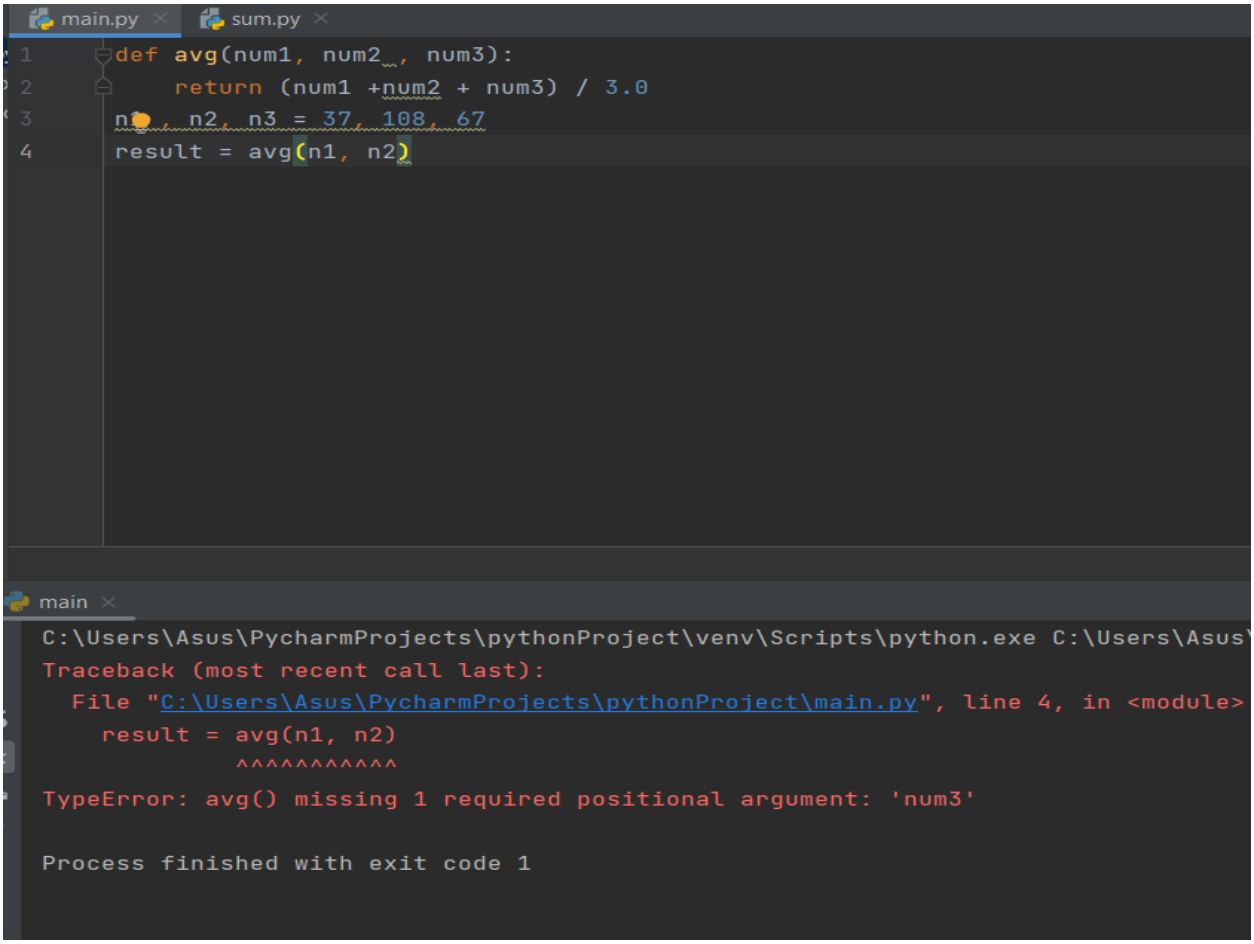
return (num1 + num2 + num3) / 3.0

n1 = 37, n2 = 108, n3 = 67

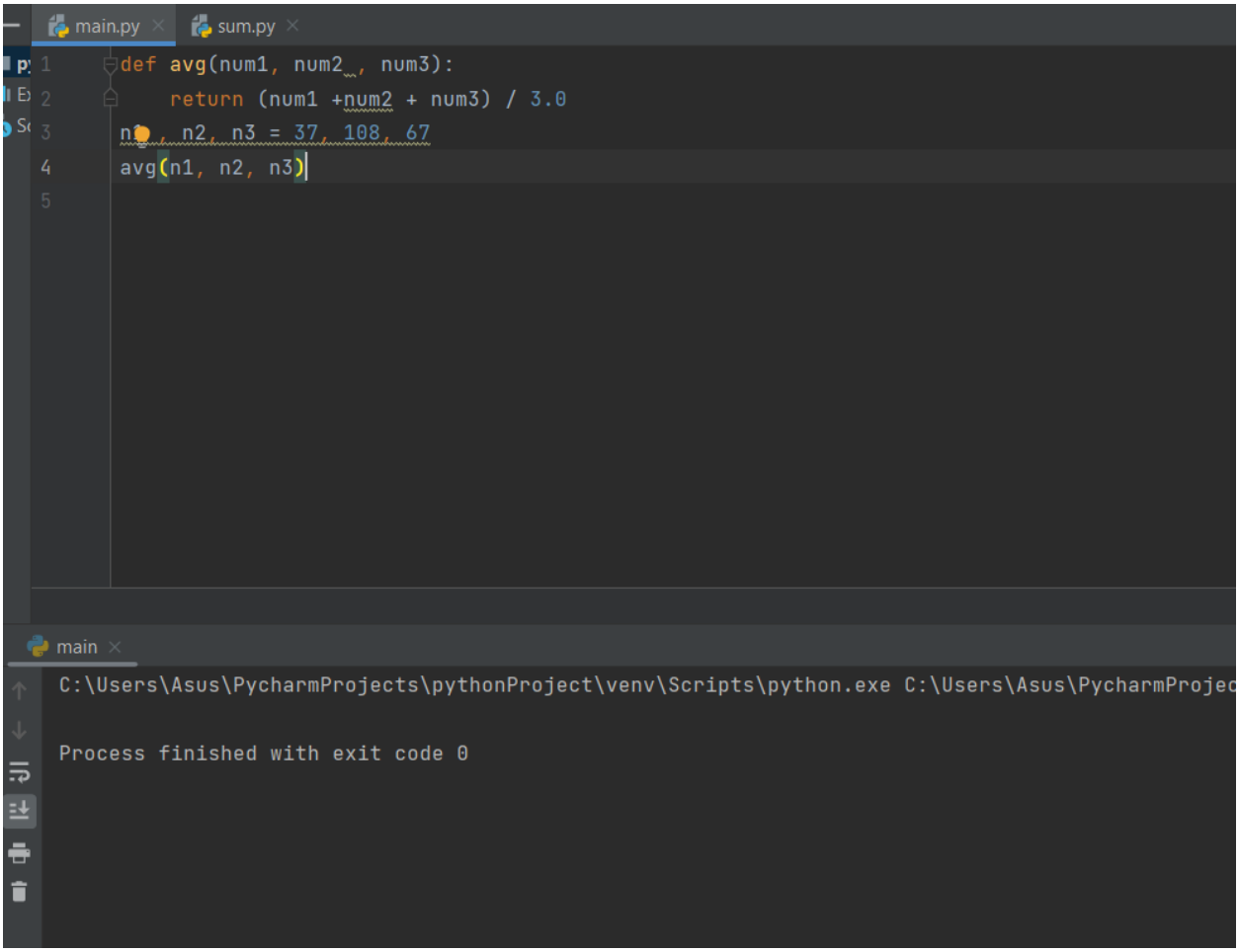
Define the function and variable declarations given above in IDLE shell and execute the following expressions. Which of the statements are valid?

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

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



(B) *avg(n1, n2, n3)*



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

```
main.py   sum.py
1  def avg(num1, num2, num3):
2      return (num1 +num2 + num3) / 3.0
3  n1 , n2, n3 = 37, 108, 67
4  result = avg(n1 + n2, n3 + n4, n5 + n6)
```

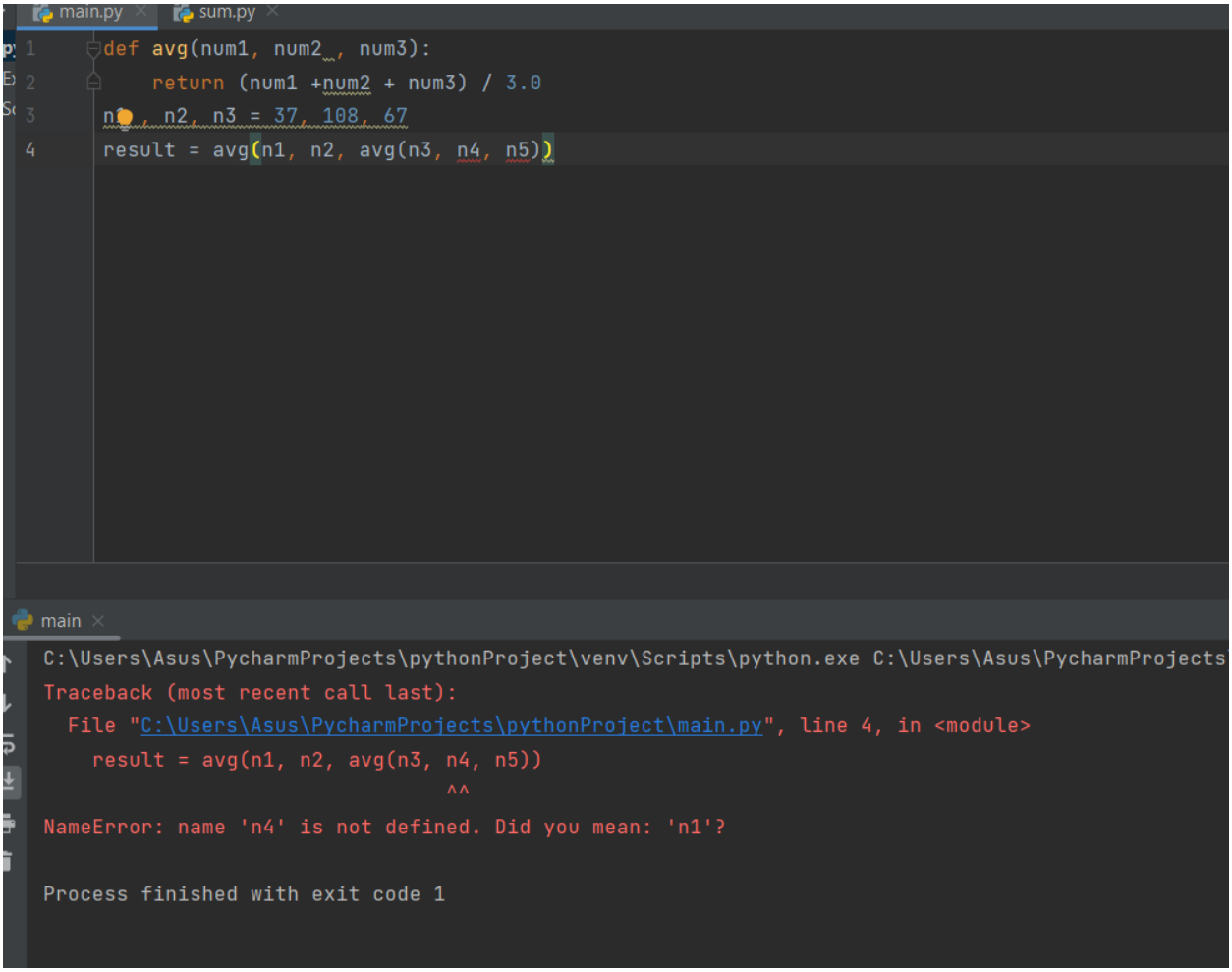
```
main
C:\Users\Asus\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Asus\PycharmProjects\pythonProject\main.py
Traceback (most recent call last):
  File "C:\Users\Asus\PycharmProjects\pythonProject\main.py", line 4, in <module>
    result = avg(n1 + n2, n3 + n4, n5 + n6)
                          ^^
NameError: name 'n4' is not defined. Did you mean: 'n1'?

Process finished with exit code 1
```

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

```
main.py   sum.py
1  def avg(num1, num2, num3):
2      return (num1 +num2 + num3) / 3.0
3  n1 , n2, n3 = 37, 108, 67
4  print(avg(n1, n2, n3))
```

```
main
C:\Users\Asus\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Asus\PycharmProjects\pythonPr
70.66666666666667

Process finished with exit code 0
```

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



Part 2

Define a function:

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

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

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

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

(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.

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

(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"

```
def types(value):
    """Prints a given value both as a float and an integer."""
    # Print the value as a float with one decimal place
    print(f"As a float: {value:.1f}")
    # Print the value as an integer
    print(f"As an integer: {int(value)}")

types(15) # calling the function and passing the argument value to the parameter

def squared(n):
    """Takes an integer and returns the value squared."""
    return n ** 2

print(squared(2)) # calling the function and passing the argument value to the parameter

def int_to_string(n):
    """Takes an integer value and returns it as a string."""
    return str(n)

print(int_to_string(22)) # calling the function and passing the argument value to the parameter
print(type(int_to_string(22)))

def hello_world(name):
    """Takes a parameter name and displays the following output to the console:
    "For Example: Hello World, my name is name"."""
    print(f"Hello World, my name is {name}")

hello_world("praa") # calling the function and passing the argument value to the parameter
```

```python
def print_ast(n, symbol="*"):
    """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."""
    print(symbol * n)

print_ast(2, "abc") # calling the function and passing the argument value to the parameter

def improved_average(a, b, c, d, e):
    """Takes five integer parameters.
    It should return the mode, median and mean values of the numbers passed to the function."""
    # Create a list of the values
    values = [a, b, c, d, e]
    # Calculate the mode
    mode = max(set(values), key=values.count)
    # Calculate the median
    median = sorted(values)[2]
    # Calculate the mean
    mean = sum(values) / len(values)
    # Return all three values
    return mode, median, mean

print(improved_average(2,3,4,5,6)) # calling the function and passing the argument value to the parameter

def either_side(n):
    """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"."""
```

```python
def either_side(n):
    """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"."""
    print(f"You typed {n}, one less than {n} is {n-1}, one more than {n} is {n+1}")


either_side(10) # calling the function and passing the argument value to the parameter
```

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.

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

```python
def calculate_difference():
    global x, y
    result = x - y
    return result
def calculate_sum():
    result = x + y
    return result
def calculate_product():
    result = x * y
    return result
def calculate_division():
    result = x / y
    return result
def calculate_modulus():
    result = x % y
    return result
def calculate_exponential():
    result = x ** y
    return result
x = int(input("enter a number: "))
y = int(input("enter a number: "))
sum_result = calculate_sum()
difference_result = calculate_difference()
product_result = calculate_product()
division_result = calculate_division()
modulus_result = calculate_modulus()
exponential_result = calculate_exponential()
print('sum is {} and {} is {}'.format(x, y, sum_result))
print('difference is {} and {} is {}'.format(x, y, difference_result))
```

```
 = int(input("enter a number: "))
 = int(input("enter a number: "))
um_result = calculate_sum()
ifference_result = calculate_difference()
roduct_result = calculate_product()
ivision_result = calculate_division()
odulus_result = calculate_modulus()
xponential_result = calculate_exponential()
rint('sum is {} and {} is {}'.format(x, y,sum_result))
rint('difference is {} and {} is {}'.format(x, y,difference_result))
rint('product is {} and {} is {}'.format(x, y,product_result))
rint('division is {} and {} is {}'.format(x, y,division_result))
rint('modulus is {} and {} is {}'.format(x, y,modulus_result))
rint('exponential is {} and {} is {}'.format(x, y,exponential_result))
```

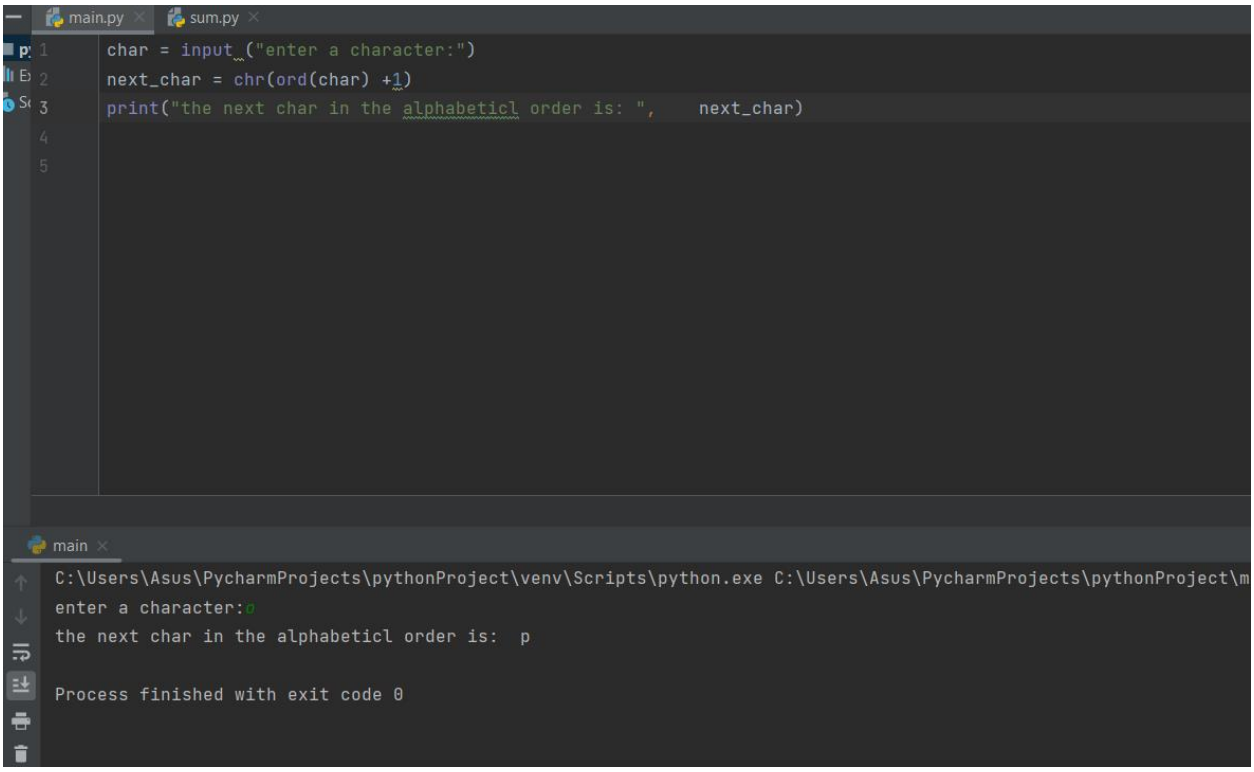
3. Go back and add multi-line Docstrings to each of the functions you defined in the previous question. Use the help function to check them afterwards.

```
enter a number: 4
enter a number: 2
sum is 4 and 2 is 6
HGHGHGHG
difference is 4 and 2 is 2
product is 4 and 2 is 8
division is 4 and 2 is 2.0
modulus is 4 and 2 is 0
exponential is 4 and 2 is 16

Process finished with exit code 0
```

```
Welcome to Python 3.11's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the internet at https://docs.python.org/3.11/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help>
```

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]



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

```python
while True:
    char = input("enter a character: ")
    next_char = chr(ord(char)+1)
    print("the next character in the alphabetic order is :", next_char)
    choice = input("do u want to continue?(yes/no)")
    if choice == "no":
        break
```

while True › if choice == "no"

main

```
C:\Users\Asus\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Asus\PycharmProjects\pythonProject\main.py
enter a character: r
the next character in the alphabetic order is : s
do u want to continue?(yes/no)yes
enter a character: d
the next character in the alphabetic order is : e
do u want to continue?(yes/no)no

Process finished with exit code 0
```

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