

## Part 1

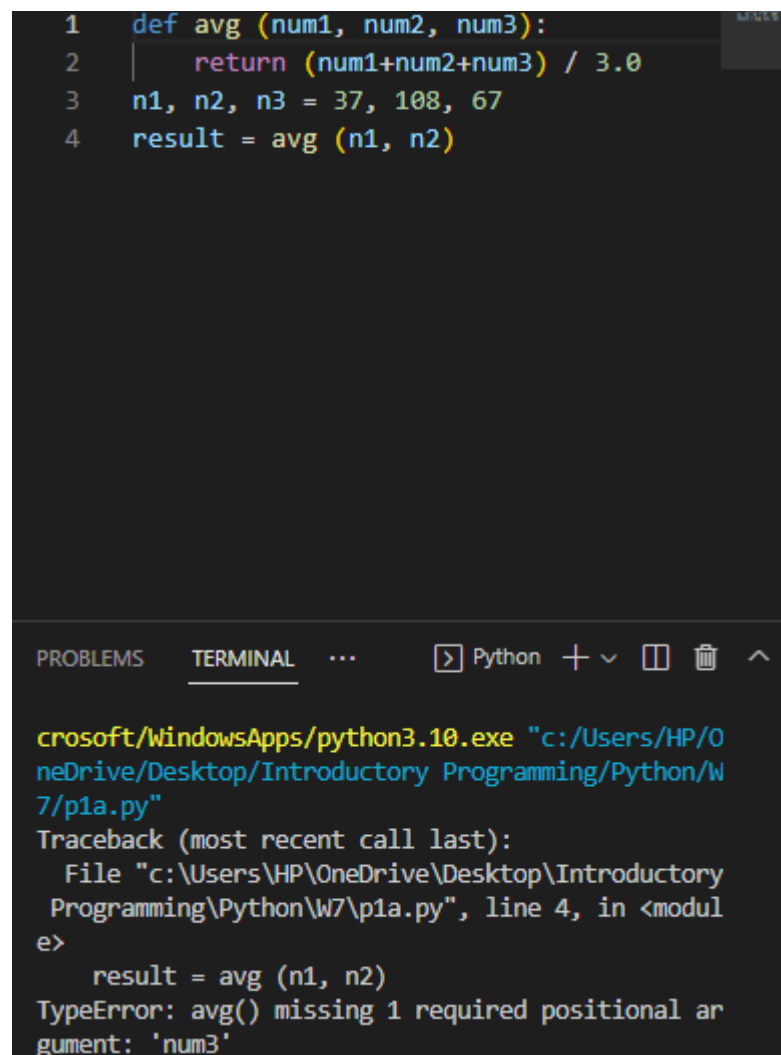
A python function “avg” returns the average of three values, as shown below:

```
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)`



```
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)
```

PROBLEMS TERMINAL ... Python + -

crosoft/WindowsApps/python3.10.exe "c:/Users/HP/OneDrive/Desktop/Introductory Programming/Python/W7/p1a.py"

Traceback (most recent call last):  
File "c:\Users\HP\OneDrive\Desktop\Introductory Programming\Python\W7\p1a.py", line 4, in <module>  
 result = avg (n1, n2)  
TypeError: avg() missing 1 required positional argument: 'num3'

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

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

p1b ×

C:\Users\HP\AppData\Local\Microsoft\Wi  
70.66666666666667

Process finished with exit code 0

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

```
def avg(num1, num2, num3):  
    return (num1 + num2 + num3) / 3.0  
n1, n2, n3 = 6, 108, 67  
result = avg(n1 + n2, n3 + n4, n5 + n6)
```

p1c x  
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.  
Traceback (most recent call last):  
 File "C:\Users\HP\OneDrive\Desktop\Introductory Programmi  
 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))*

```
def avg(num1, num2, num3):  
    return (num1 + num2 + num3) / 3.0  
n1, n2, n3 = 36, 108, 67  
print(avg(n1, n2, n3))
```

p1d x  
C:\Users\HP\AppData\Local\Microsoft\Windows  
70.33333333333333  
Process finished with exit code 0

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

```
def avg(num1, num2, num3):  
    return (num1 + num2 + num3) / 3.0  
n1, n2, n3 = 36, 108, 67  
result = avg(n1, n2, avg(n3, n4, n5))
```

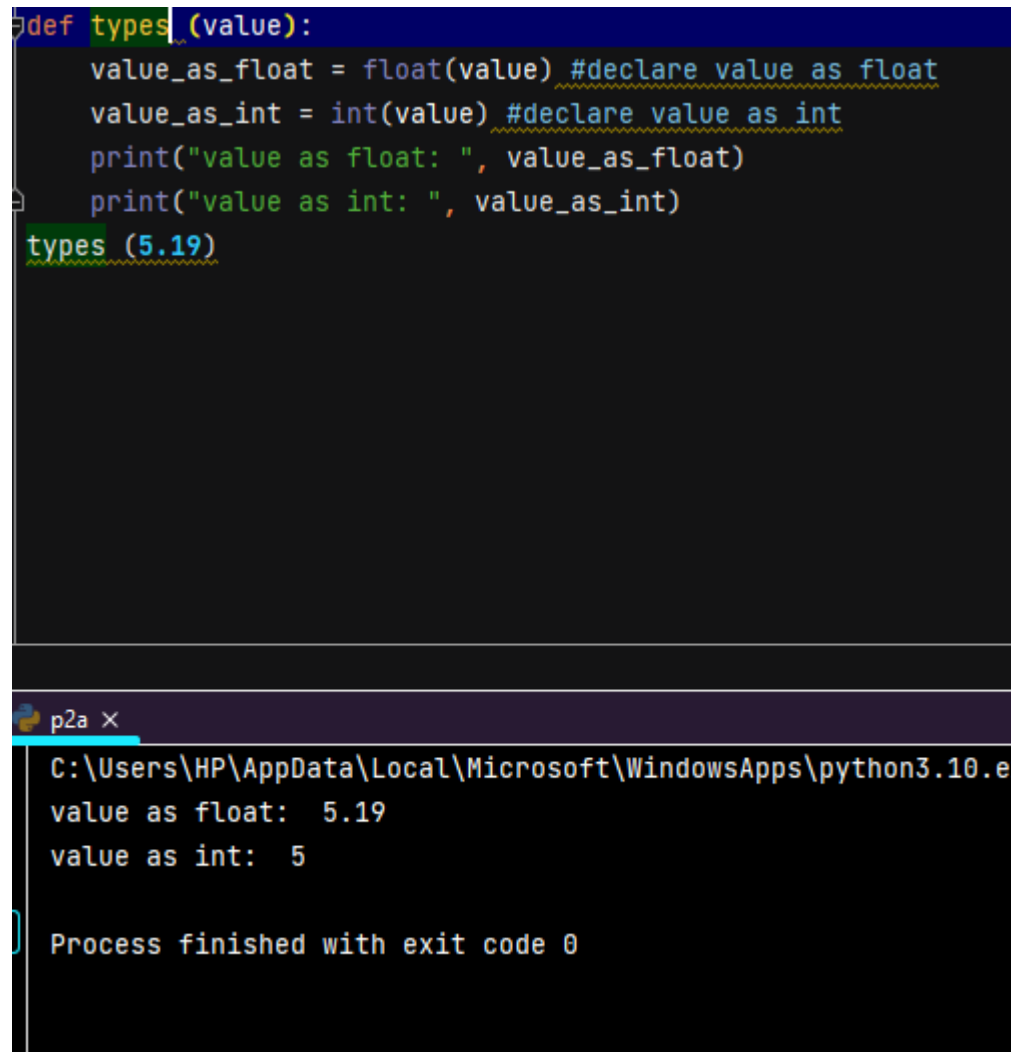
p1e x  
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\HP\OneDrive\Desktop\Introduct  
Traceback (most recent call last):  
 File "C:\Users\HP\OneDrive\Desktop\Introductory Programming\Python\W7\p1e.py", line 4, in <module>  
 result = avg(n1, n2, avg(n3, n4, n5))  
NameError: name 'n4' is not defined. Did you mean: 'n1'?  
Process finished with exit code 1

## Part 2

Define a function:

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

```
def types(value):  
    value_as_float = float(value) #declare value as float  
    value_as_int = int(value) #declare value as int  
    print("value as float: ", value_as_float)  
    print("value as int: ", value_as_int)  
types(5.19)
```

The image shows a code editor window with a dark background. The top part contains a Python function definition for 'types' which takes a 'value' argument, converts it to a float and an integer, and prints both. Below the function definition, the function is called with the argument '5.19'. The bottom part of the image shows a terminal window titled 'p2a x' with the same code executed. The output of the function is 'value as float: 5.19' and 'value as int: 5'. The terminal also shows the file path 'C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe' and 'Process finished with exit code 0'.

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

```
def square(n):  
    """takes as an integer and return the squared value"""  
    return n ** 2  
print(square(4))
```

p2b ×

C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe  
16  
  
Process finished with exit code 0

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

```
def int_to_String(n):  
    """takes an integer value and return as a String"""  
    return str(n)  
print(int_to_String(3))
```

tring0

p2c x

C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.1  
3

Process finished with exit code 0

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

```
def hello_world(name):  
    """takes a parameter and display the following output to the console"""  
    print(f"Hello World! My name is {name}")  
hello_world("Naomi")
```

p2d x

C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\HP\  
Hello World! My name is Naomi

Process finished with exit code 0

(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.



```
def print_ast(n, symbol="*"):
    """takes an integer value n and string value symbol, with a default value of '*'."""
    print(symbol * n)

print_ast(3, " Holiday")
```

p2e x

C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\HP\OneDrive\Desktop\Holiday Holiday Holiday

Process finished with exit code 0

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

```
def improved_average(a, b, c, d, e):  
    """takes five integer parameters."""  
    #create a list of the values  
    values = [a, b, c, d, e]  
    mode = max(set(values), key = values.count)  
    median = sorted(values)[2]  
    mean = sum(values)/len(values)  
    return mode, median, mean  
  
print(improved_average(2, 3, 4, 5, 6))
```

p2f x  
C:\Users\HP\AppData\Local\Microsoft\WindowsApps  
(2, 4, 4.0)  
  
Process finished with exit code 0

(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 either_side(n):  
    print(f"you type {n}, one less than {n} is {n-1}, one more than {n} is {n+1}")  
either_side(12)
```

p2g ×  
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\HP\OneDrive\  
you type 12, one less than 12 is 11, one more than 12 is 13  
Process finished with exit code 0

### 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.

```
def divided_by_second():  
    num1 = int(input("enter the first integer number: "))  
    num2 = int(input("enter the second integer number: "))  
    result = num1 / num2  
    round(result, 2)  
    print(round(result, 2))  
  
divided_by_second()
```

p3.1 x

C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe  
enter the first integer number: 50  
enter the second integer number: 12  
4.17

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

```
def addition(n1, n2):  
    return round(n1+n2, 2)  
def subtraction(n1, n2):  
    return round(n1-n2, 2)  
def multiplication(n1, n2):  
    return round(n1*n2, 2)  
def division(n1, n2):  
    return round(n1/n2, 2)  
def truncated_division(n1, n2):  
    return round(n1//n2, 2)  
def modulus(n1, n2):  
    return round(n1%n2, 2)  
def exponentiation(n1, n2):  
    return round(n1**n2, 2)  
result=addition(56, 23)  
print(result)  
result=subtraction(13, 12)  
print(result)  
result=multiplication(22, 21)  
print(result)  
result=division(55, 5)  
print(result)  
result=truncated_division(13, 2)
```

```
result=trunacated_division(124, 2)
print(result)
result=modulus(15, 3)
print(result)
result=exponentiation(45.55, 55.25)
print(result)
```

```
p3.5 x
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.1
79
1
462
11.0
62
0
4.280387304891007e+91

Process finished with exit code 0
```

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.

```
def addition(n1, n2):  
    return round(n1+n2, 2)  
def subtraction(n1, n2):  
    return round(n1-n2, 2)  
def multiplication(n1, n2):  
    return round(n1*n2, 2)  
def division(n1, n2):  
    if n2 == 0:  
        print("it will show an error")  
        return  
    return round(n1/n2, 2)  
def truncated_division(n1, n2):  
    if n2 == 0:  
        print("it will show an error")  
        return  
    return round(n1//n2, 2)  
def modulus(n1, n2):  
    return round(n1%n2, 2)  
def exponentiation(n1, n2):  
    return round(n1**n2, 2)  
def help():  
    print("addition(n1,n2)")  
    print("subtraction(n1, n2)")
```

```
print("multiplication(n1, n2)")
print("division(n1, n2)")
print("trunacated(n1, n2)")
print("modulus(n1, n2)")
print("exponentiation(n1, n2)")
result=addition(56, 23)
print(result)
result=subtraction(13, 12)
print(result)
result=multiplication(22, 21)
print(result)
result=division(55, 5)
print(result)
result=trunacated_division(124, 2)
print(result)
result=modulus(15, 3)
print(result)
result=exponentiation(45.55, 55.25)
print(result)
help()
```

```
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe
79
1
462
11.0
62
0
4.280387304891007e+91
additon(n1,n2)
subtraction(n1, n2)
multiplication(n1, n2)
division(n1, n2)
trunacated(n1, n2)
modulus(n1, n2)
exponentiation(n1, n2)

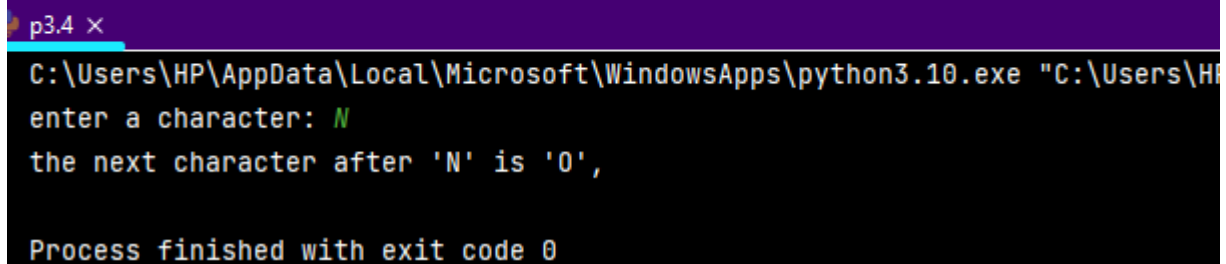
Process finished with exit code 0
```

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]

```
#get a character input from the user
char = input("enter a character: ")
#use the ord function to get the ASCII value of the character
char_value = ord(char)
#increasing the ASCII value by 1 to get the value of the next character
next_char_value = char_value + 1
#use the chr function to convert the ASCII value back into a character
next_char = chr(next_char_value)
#print the result
print(f"the next character after '{char}' is '{next_char}',")
```



```
p3.4 x
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe"
enter a character: N
the next character after 'N' is 'O',

Process finished with exit code 0
```

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

```
while True:
    #get a character input from the user
    char = input("enter a character: ")
    # use the ord function to get the ASCII value of the character
    char_value = ord(char)
    #increasing the ASCII value by 1 tp get the value of the next character
    next_char_value = char_value + 1
    #use the chr function to convert the ASCII value back into a character
    next_char = chr(next_char_value)
    #print the result
    print(f"the next character after '{char}' is '{next_char}'.")
    # asking user if they want to continue or not
    choice = input("do you want to continue, (Y/N)?")
    if choice.upper() != "Y":
        break #break statement is used to stop or terminate the code
```

```
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe
enter a character: N
the next character after 'N' is 'O'.
do you want to continue, (Y/N)?Y
enter a character: A
the next character after 'A' is 'B'.
do you want to continue, (Y/N)?Y
enter a character: O
the next character after 'O' is 'P'.
do you want to continue, (Y/N)?Y
enter a character: M
the next character after 'M' is 'N'.
do you want to continue, (Y/N)?Y
enter a character: I
the next character after 'I' is 'J'.
do you want to continue, (Y/N)?N

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