Python Programming Language

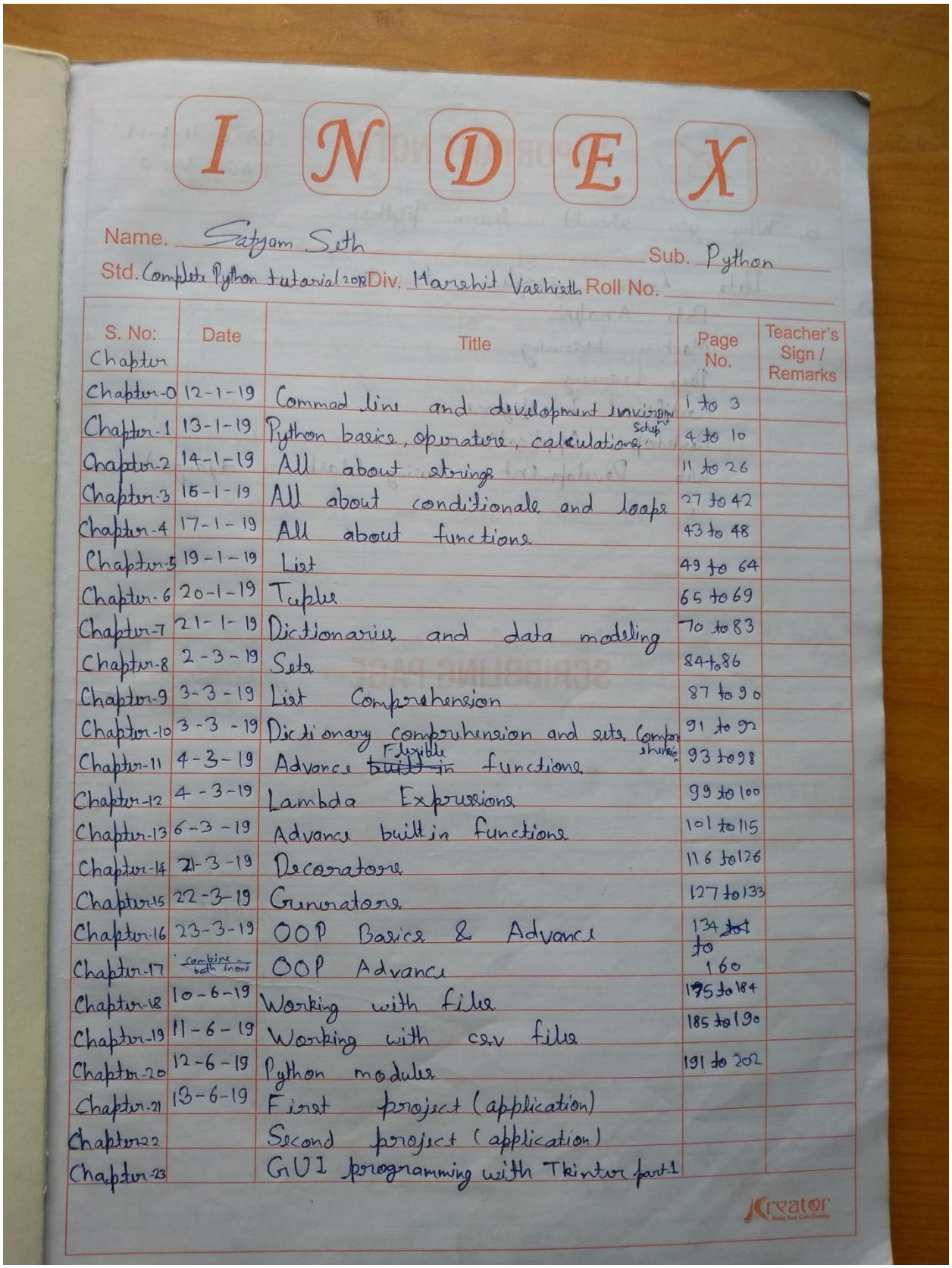
HARSHIT VASHISTH YOUTUBE CHANNEL LEARING NOTES

Source Code- https://github.com/satyam-seth-learnings/python_learning/tree/main/Harshit%20Vashisth

Playlist Link- https://youtube.com/playlist?list=PLwgFb6VsUj_lQTpQKDtLXKXElQychT_2j

Satyam Seth

19-09-2021

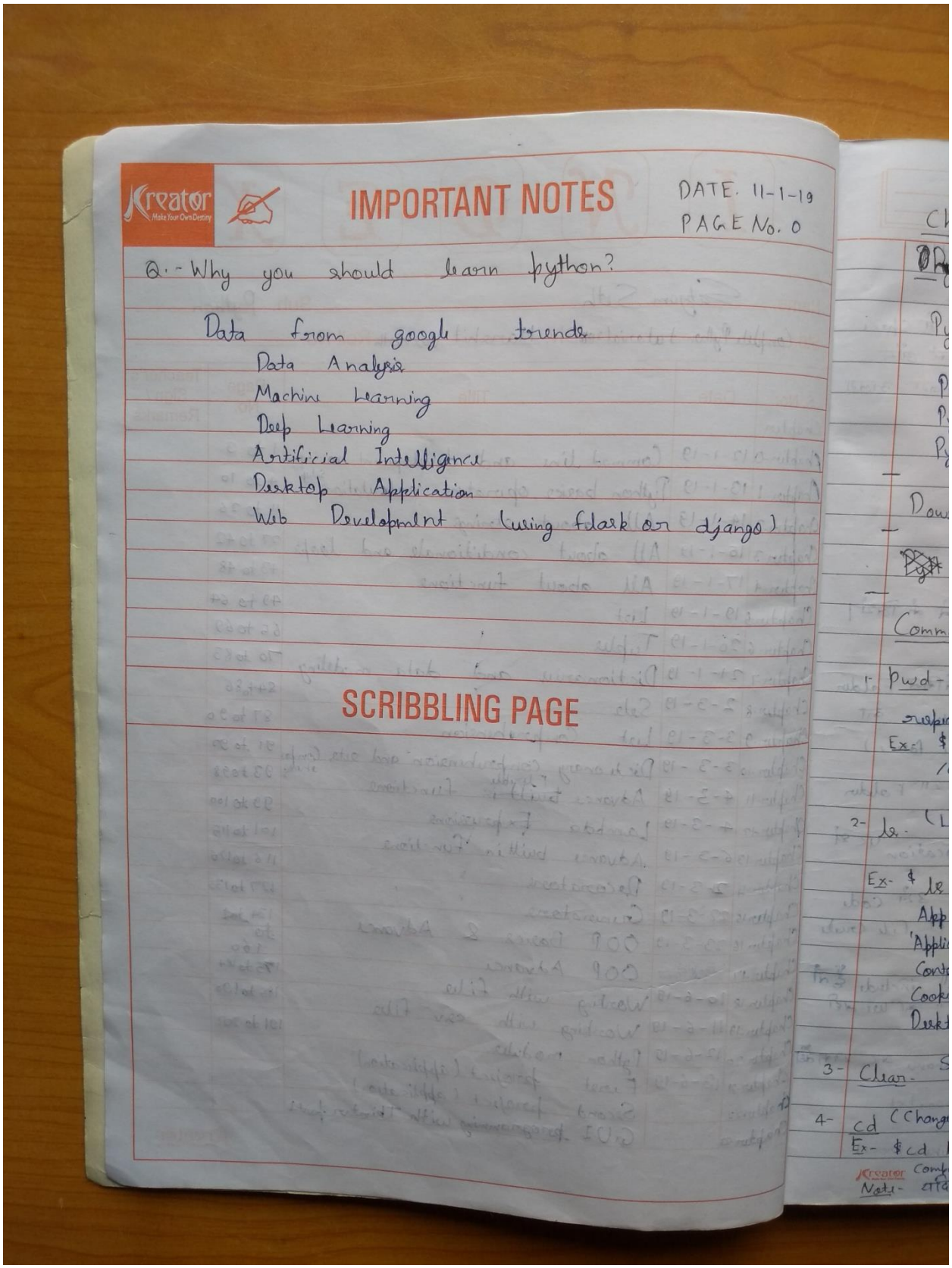


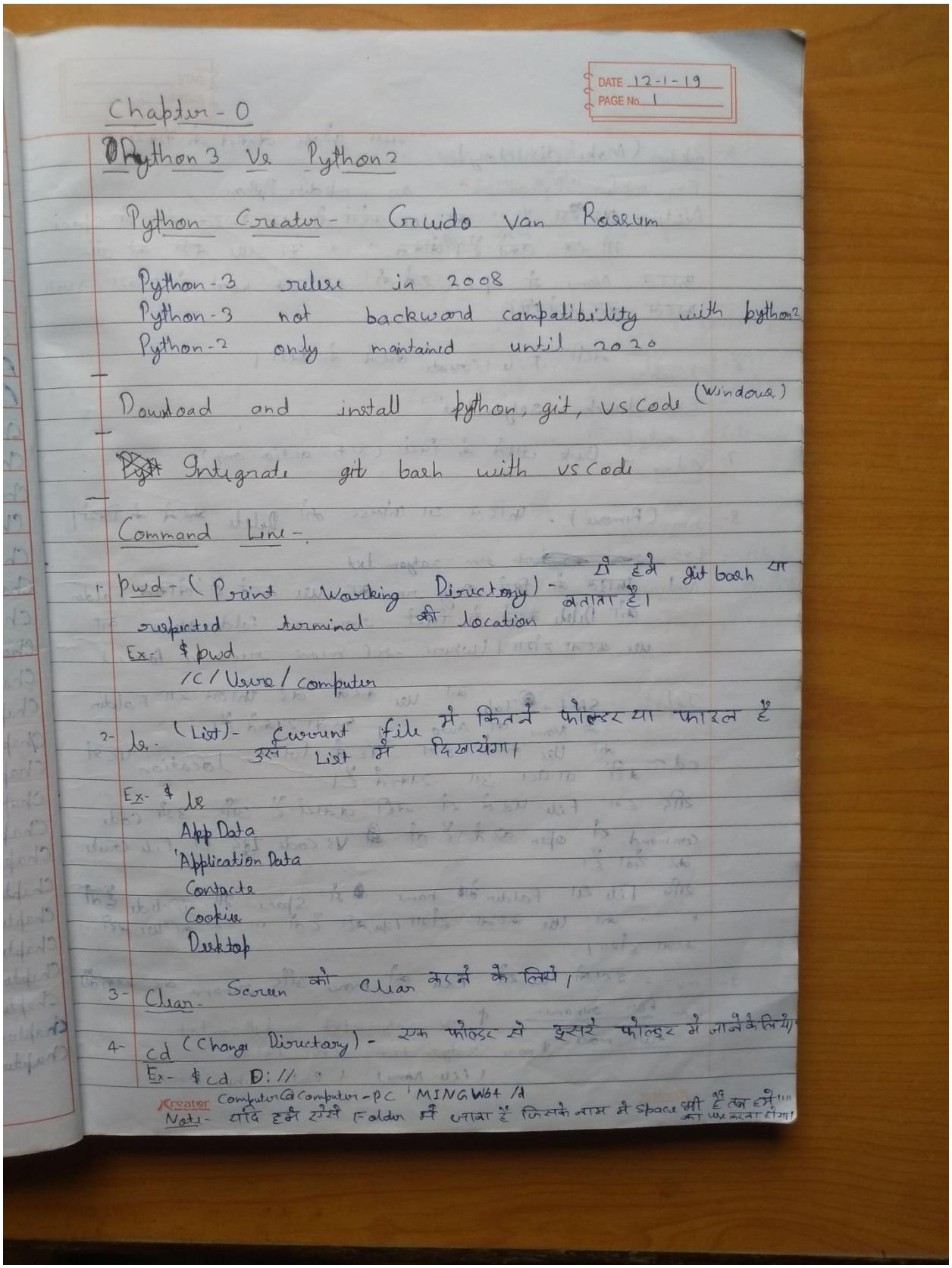
INDEX

Name. Satyam Seth Sub. Python

Std. Complete Python tutorial 2018 Div. Harshit Vashisth Roll No.

Kreator Make Your Own Destiny


IMPORTANT NOTES
DATE. 11-1-19
PAGE No. 0
Q. - Why you should learn python?
Data from google trends
Data Analysis
Machine Learning
Deep Learning
Artificial Intelligence
Desktop Application
Web Development (using flask or django)
SCRIBBLING PAGE



DATE 12-1-19
PAGE No. 1

Chapter - 0

Python 3 Vs Python 2

Python Creator - Guido Van Rossum

Python-3 relase in 2008
Python-3 not backward compatibility with python2
Python-2 only maintained until 2020

Download and install python, git, vscode (Windows)

Integrate git bash with vscode

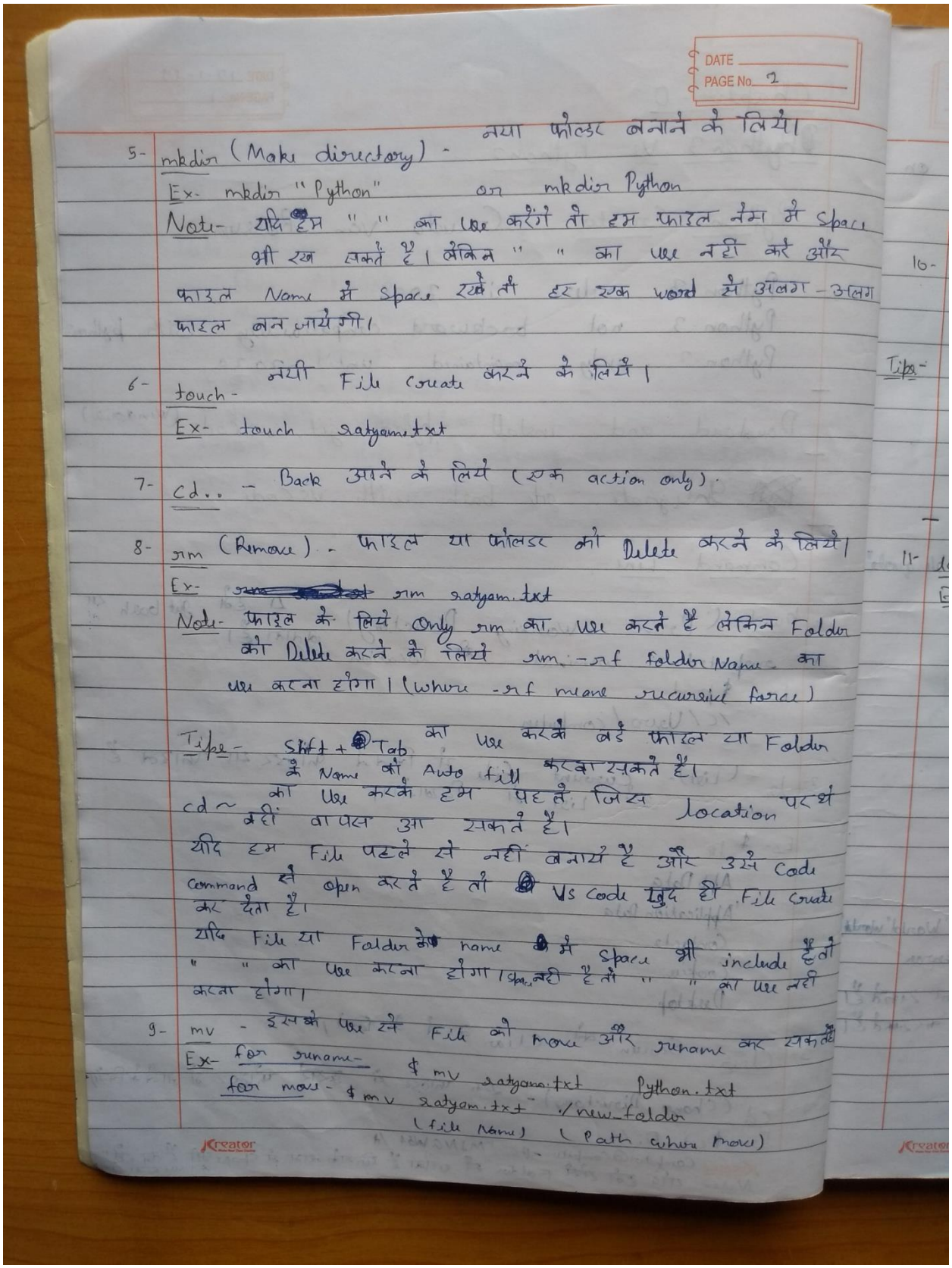
Command Line -

1- pwd - (Print Working Directory) - ये हमें git bash या reported terminal की location बताता है।
Ex- $ pwd
/c/Users/computer

2- ls - (List) - Current file में कितने फोल्डर या फाइल हैं उस List में दिखायेगा।
Ex- $ ls
AppData
'Application Data'
Contacts
Cookies
Desktop

3- Clear - Screen को Clear करने के लिये।

4- cd (Change Directory) - एक फोल्डर से दूसरे फोल्डर में जाने के लिये।
Ex- $ cd D://
Creator Computer@Computer-PC 'MINGW64 /d
Note - यदि हमें ऐसे Folder में जाना है जिसके नाम में space भी है तब हमें "" का use करना होगा।



DATE
PAGE No. 1

5- mkdir (Make directory) - नया फोल्डर बनाने के लिये।
Ex- mkdir "Python" or mkdir Python
Note- यदि हम " " का use करेंगे तो हम फाइल नेम में Space भी रख सकते हैं। लेकिन " " का use नहीं करें और फाइल Name में Space रखें तो हर एक word से अलग-अलग फाइल बन जायेगी।

6- touch - नयी File create करने के लिये।
Ex- touch satyam.txt

7- cd.. - Back आने के लिये (एक action only).

8- rm (Remove) - फाइल या फोल्डर को Delete करने के लिये।
Ex- rm satyam.txt
Note- फाइल के लिये only rm का use करते हैं लेकिन Folder को Delete करने के लिये rm -rf Folder Name का use करना होगा। (where -rf mean recursive force)

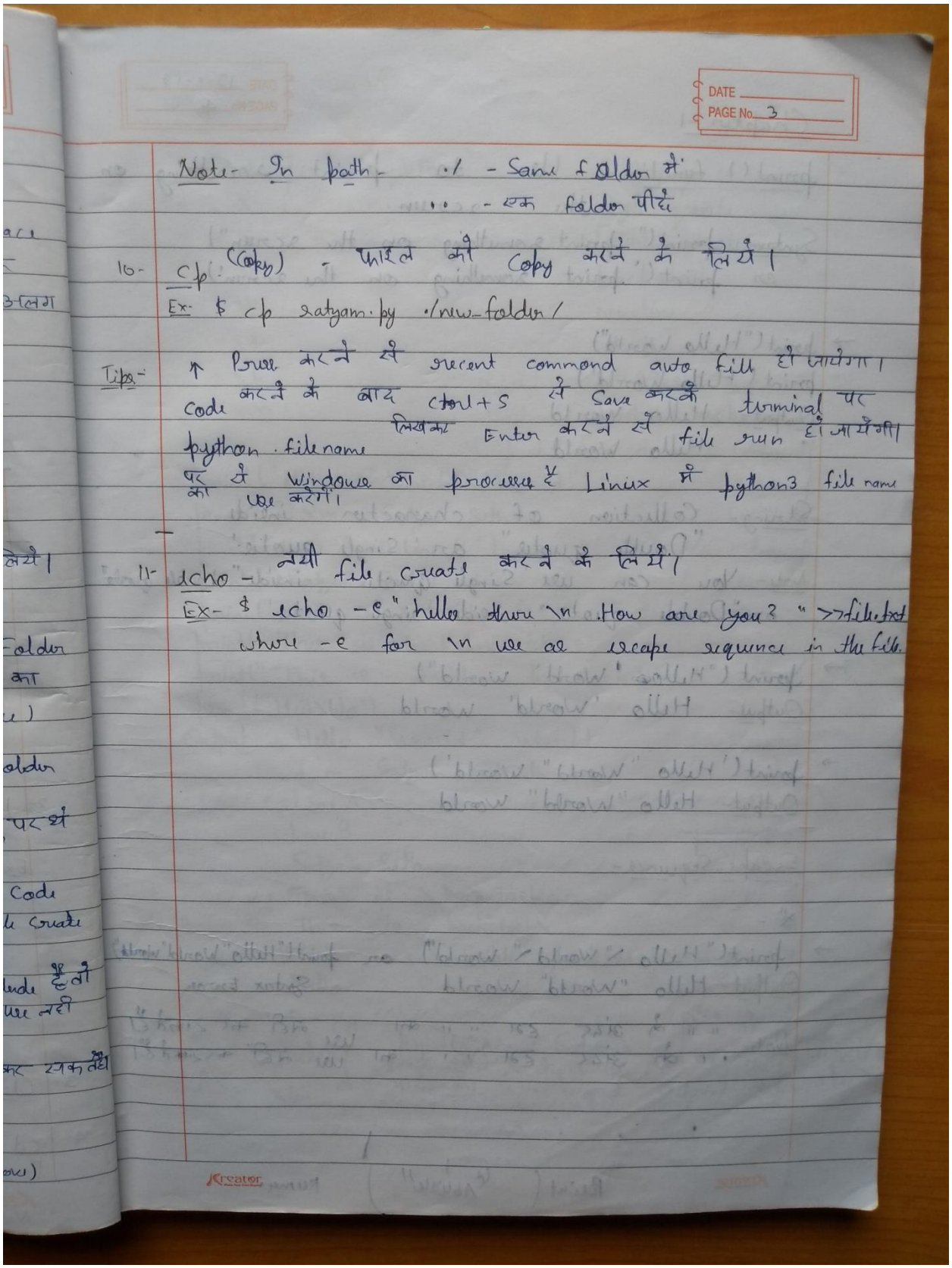
Tips- Shift + Tab का use करके बड़ी फाइल या Folder के Name को Auto fill करवा सकते हैं।
cd ~ का use करके हम पहले जिस location पर थे वहीं वापस आ सकते हैं।
यदि हम File पहले से नहीं बनाये हैं और उसे Code command से open करते हैं तो Vs code खुद ही File create कर देता है।
यदि File या Folder के name में space भी include है तो " " का use करना होगा। अगर नहीं है तो " " का use नहीं करना होगा।

9- mv - इसके use से File को move और rename कर सकते हैं
Ex- for rename- $ mv satyam.txt Python.txt
for move - $ mv satyam.txt ./new_folder
(file name) (Path where move)

Tips-

10-

11-

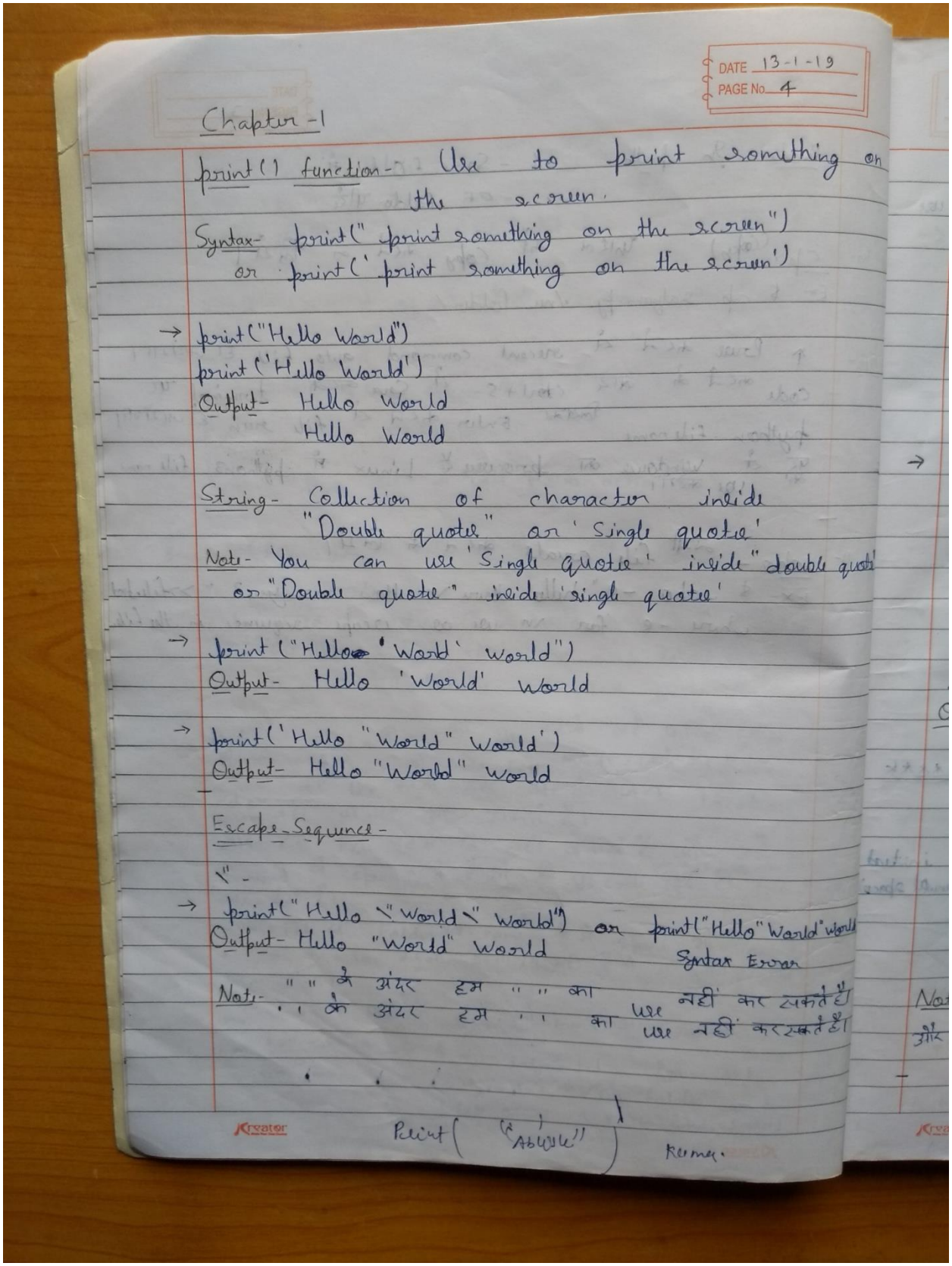


Note- In path - ./ - Same folder में
.. - एक folder पीछे

10- cp (copy) - फाईल को copy करने के लिये।
Ex- $ cp satyam.py ./new-folder/

Tips- ↑ Press करने से recent command auto fill हो जायेगा।
Code करने के बाद ctrl+S से Save करके terminal पर python filename लिखकर Enter करने से file run हो जायेगी।
पर से Windows का process है Linux में python3 file name का use करेंगे।

11- echo - नयी file create करने के लिये।
Ex- $ echo -e "hello there \n .How are you? " >>file.txt
where -e for \n we use as escape sequence in the file.



DATE 13-1-19
PAGE No. 4

Chapter -1

print() function- Use to print something on the screen.

Syntax- print("print something on the screen")
or print('print something on the screen')

→ print("Hello World")
print('Hello World')
Output- Hello World
Hello World

String- Collection of character inside "Double quote" or 'Single quote'

Note- You can use 'Single quote' inside "double quote" or "Double quote" inside 'single quote'

→ print("Hello 'World' World")
Output- Hello 'World' World

→ print('Hello "World" World')
Output- Hello "World" World

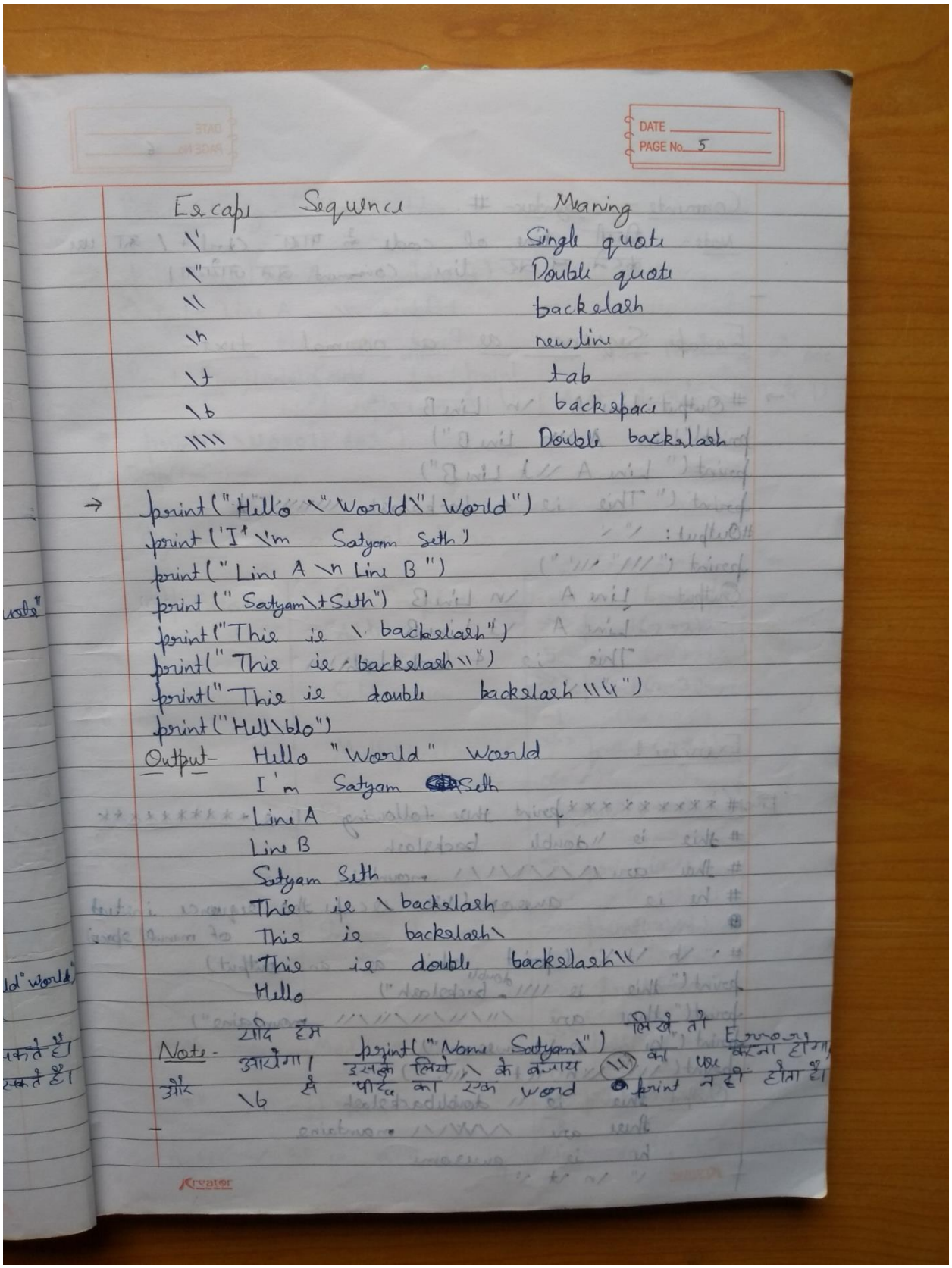
Escape Sequence-

\" -

→ print("Hello \"World\" World") or print("Hello"World"World
Output- Hello "World" World Syntax Error

Note- " " के अंदर हम " " का use नहीं कर सकते हैं
' ' के अंदर हम ' ' का use नहीं कर सकते हैं

Kreater Print("Abuse") Kreater



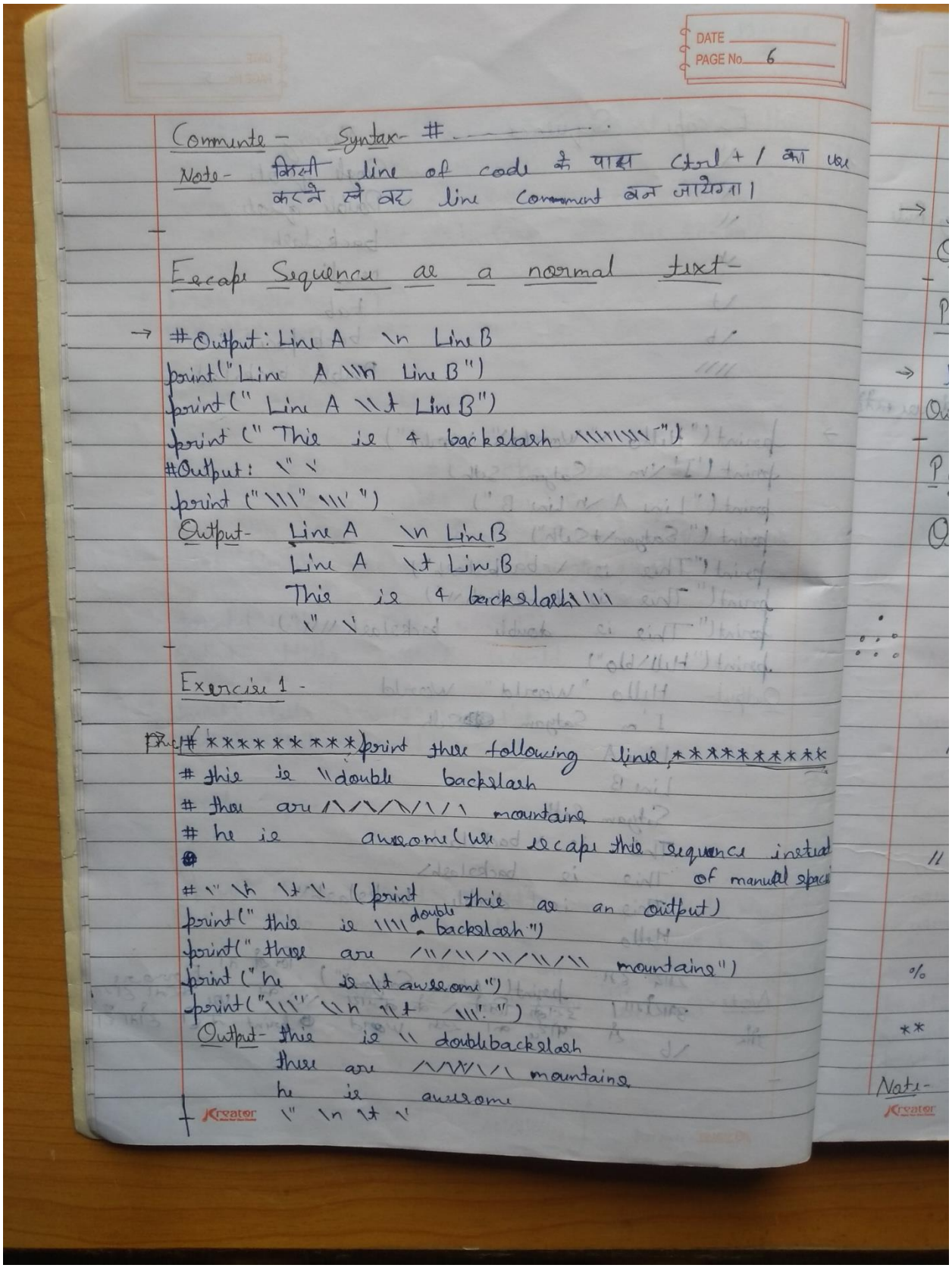
DATE
PAGE No. 5

Escape Sequence	Meaning
\'	Single quote
\"	Double quote
\\	backslash
\n	newline
\t	tab
\b	backspace
\\\\	Double backslash

→
```
print("Hello \"World\" World")
print('I\'m Satyam Seth')
print("Line A\n Line B")
print("Satyam\tSeth")
print("This is \ backslash")
print("This is backslash\\")
print("This is double backslash\\\\")
print("Hell\bblo")
```

Output-
```
Hello "World" World
I'm Satyam Seth
Line A
Line B
Satyam Seth
This is \ backslash
This is backslash\
This is double backslash\\
Hello
```

Note- यदि हम print("Name Satyam\") लिखें तो Error आएगा। इसके लिए \ के बजाय (\\) का use करना होगा, और \b से पहले का एक word print नहीं होता है।



DATE
PAGE No. 6

Comments - Syntax - #

Note - किसी line of code के पहले Ctrl + / का use करने से वह line comment बन जायेगा।

Escape Sequence as a normal text -

→ #Output: Line A \n Line B
print("Line A \\n Line B")
print("Line A \\t Line B")
print("This is 4 backslash \\\\")
#Output: \" \'
print("\\\" \\\' ")

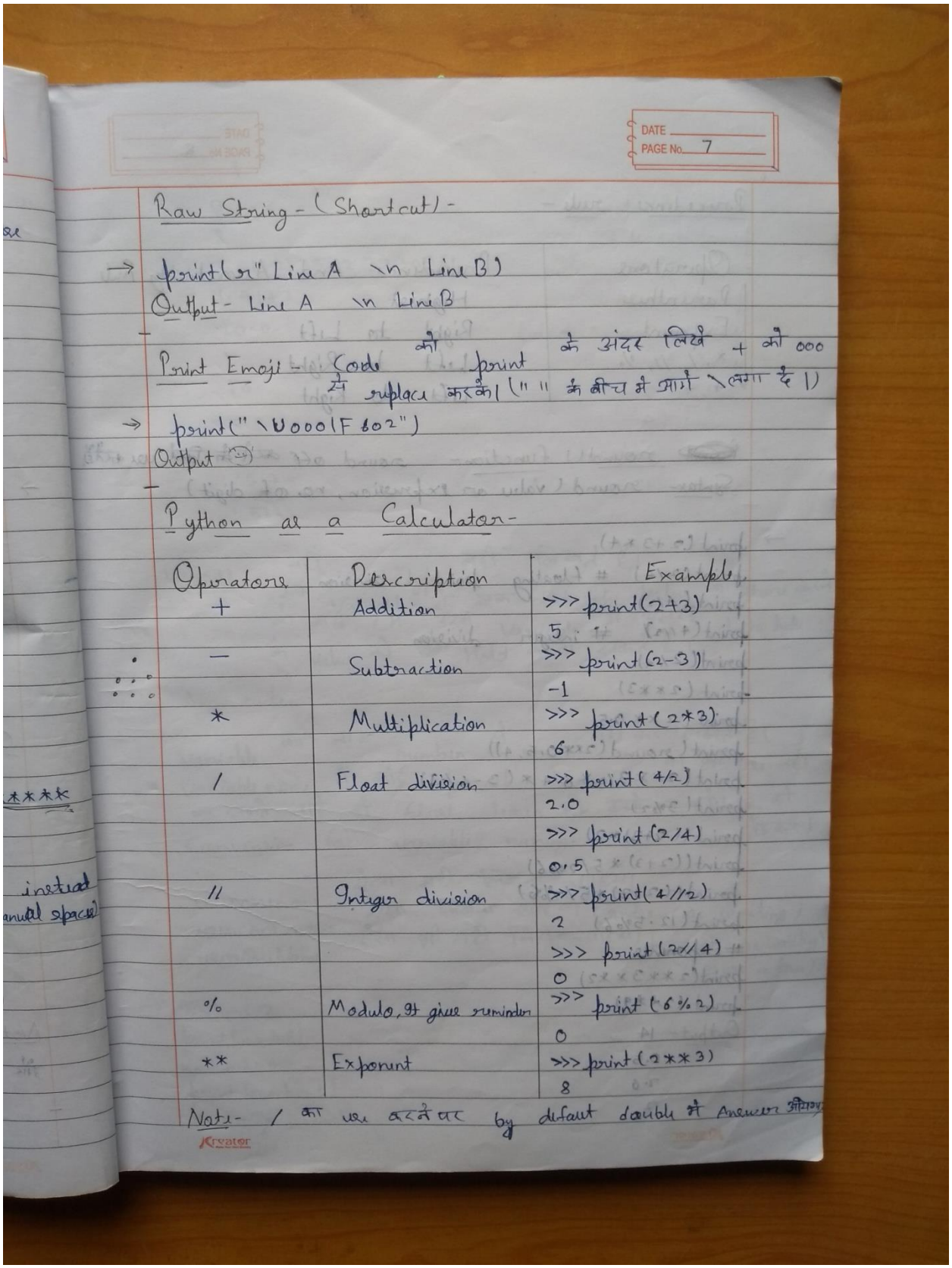
Output- Line A \n Line B
Line A \t Line B
This is 4 backslash \\\\
\" \'

Exercise 1 -

Q1 # ********** print these following line **********
# this is \\ double backslash
# these are /\/\/\/\/\ mountains
# he is awesome (use escape this sequence instead of manual space)
# \" \n \t \' (print this as an output)
print("this is \\\\ double backslash")
print("these are /\\/\\/\\/\\/\\ mountains")
print("he is \t awesome")
print("\\\" \\n \\t \\\'.")

Output- this is \\ double backslash
these are /\/\/\/\/\ mountains
he is awesome
\" \n \t \'

Note -



DATE
PAGE No. 7

Raw String - (Shortcut) -

→ print(r"Line A \n Line B)
Output - Line A \n Line B

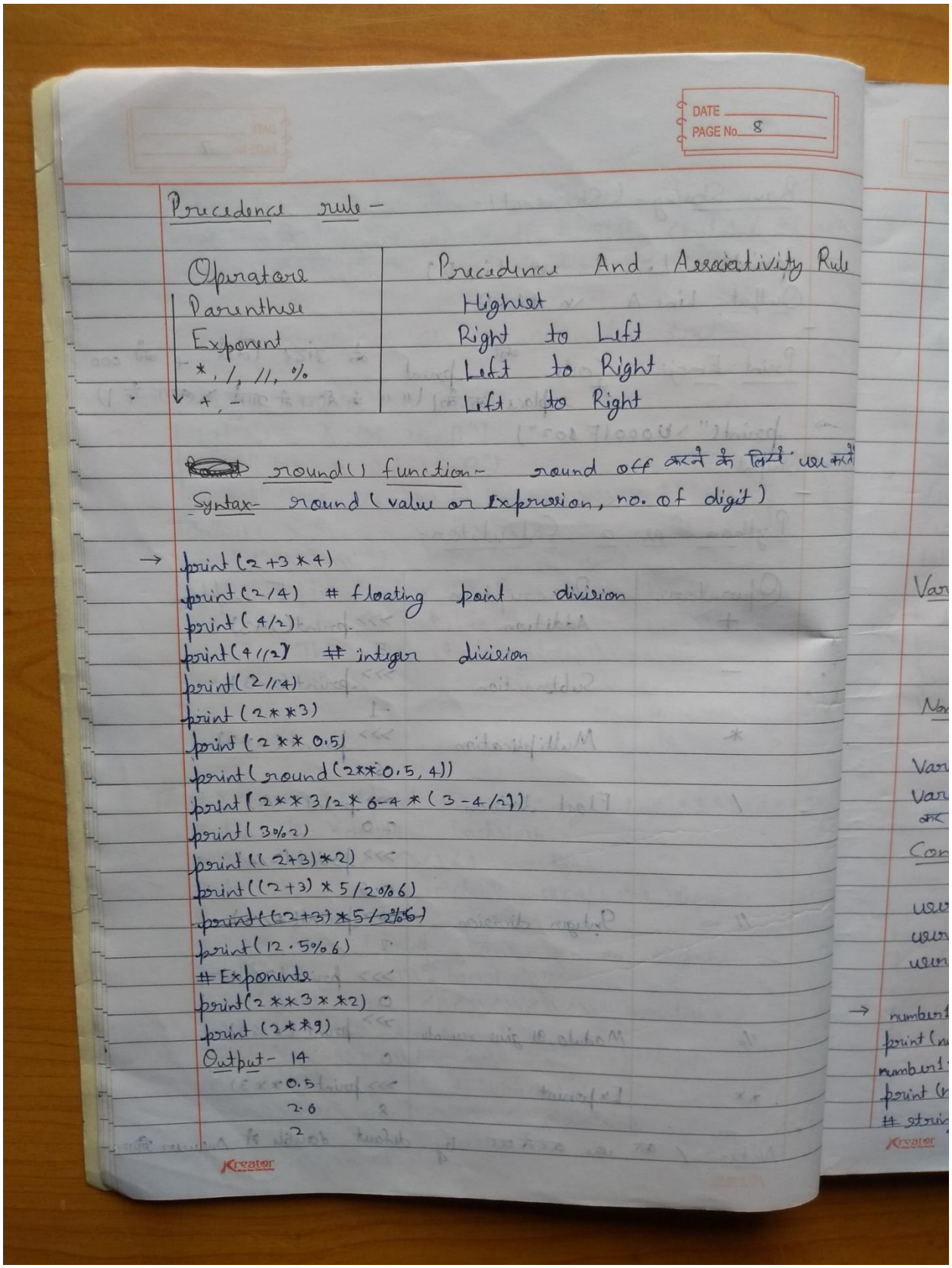
Print Emoji - Code को print के अंदर लिखें + को 000 से replace करके। (" " के बीच में आगे \ लगा दें।)

→ print("\U000|F602")
Output 😂

Python as a Calculator -

Operators	Description	Example
+	Addition	>>> print(2+3) 5
−	Subtraction	>>> print(2-3) -1
*	Multiplication	>>> print(2*3) 6
/	Float division	>>> print(4/2) 2.0
		>>> print(2/4) 0.5
//	Intger division	>>> print(4//2) 2
		>>> print(2//4) 0
%	Modulo, It give reminder	>>> print(6%2) 0
**	Exponent	>>> print(2**3) 8

Note - / का use करने पर by difaut double में Answer आएगा



DATE
PAGE No. 8

Precedence rule -

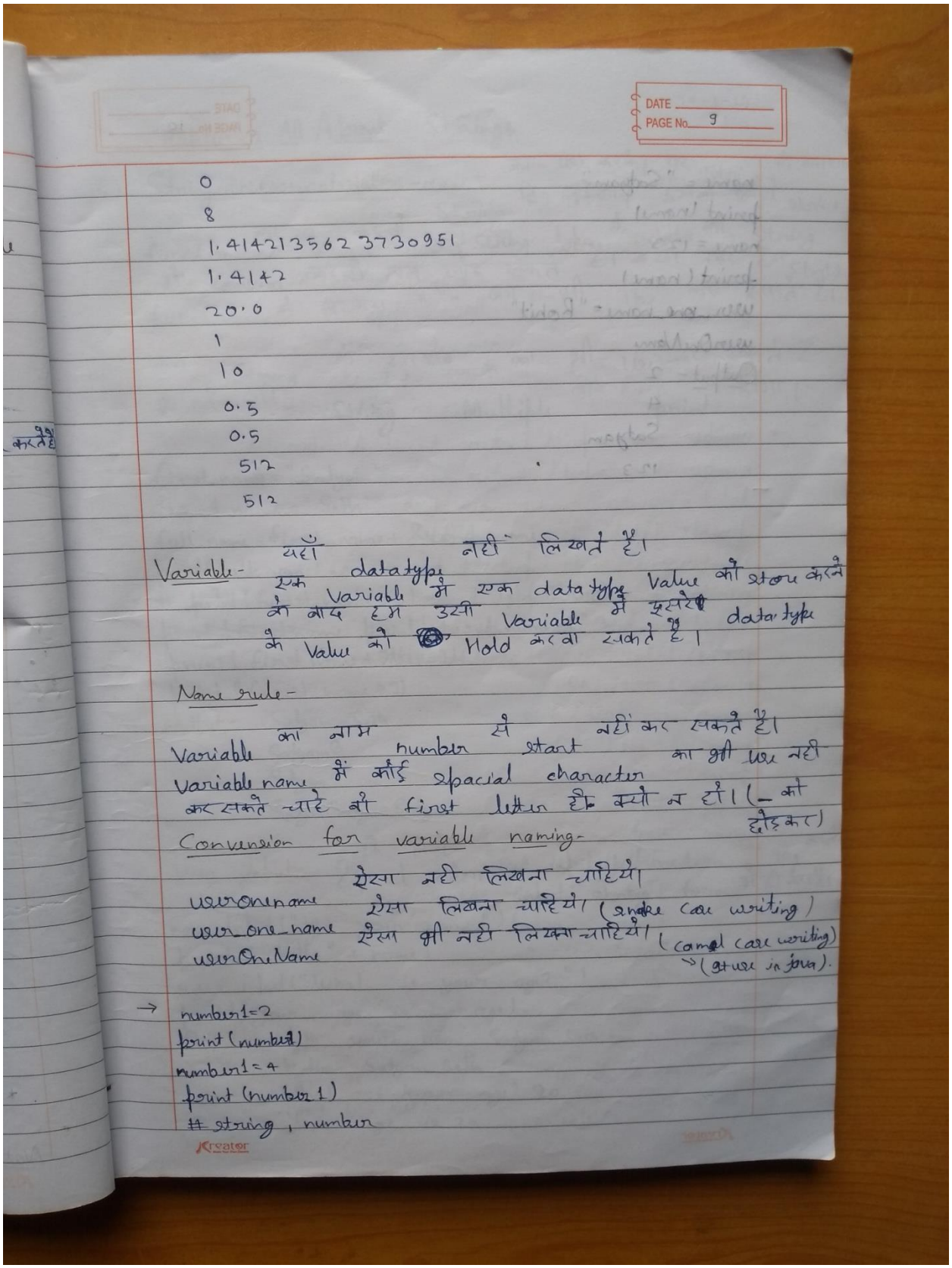
Operators	Precedence And Associativity Rule
Parenthesis	Highest
Exponent	Right to Left
*, /, //, %	Left to Right
+, -	Left to Right

round() function - round off करने के लिये use करते

Syntax- round (value or expression, no. of digit)

```
print(2+3*4)
print(2/4) # floating point division
print(4/2)
print(4//2) # integer division
print(2//4)
print(2**3)
print(2**0.5)
print(round(2**0.5,4))
print(2**3/2*6-4*(3-4/2))
print(3%2)
print((2+3)*2)
print((2+3)*5/2%6)
print((2+3)*5/2%6)
print(12.5%6)
# Exponents
print(2**3**2)
print(2**9)
```

Output- 14
0.5
2.0
2



DATE
PAGE No. 9

0
8
1.4142135623730951
1.4142
20.0
1
10
0.5
0.5
512
512

Variable- यहाँ नहीं लिखते हैं।
एक datatype Variable में एक data type Value को store करने के बाद हम उसी Variable में दूसरे data type के Value को Hold करवा सकते हैं।

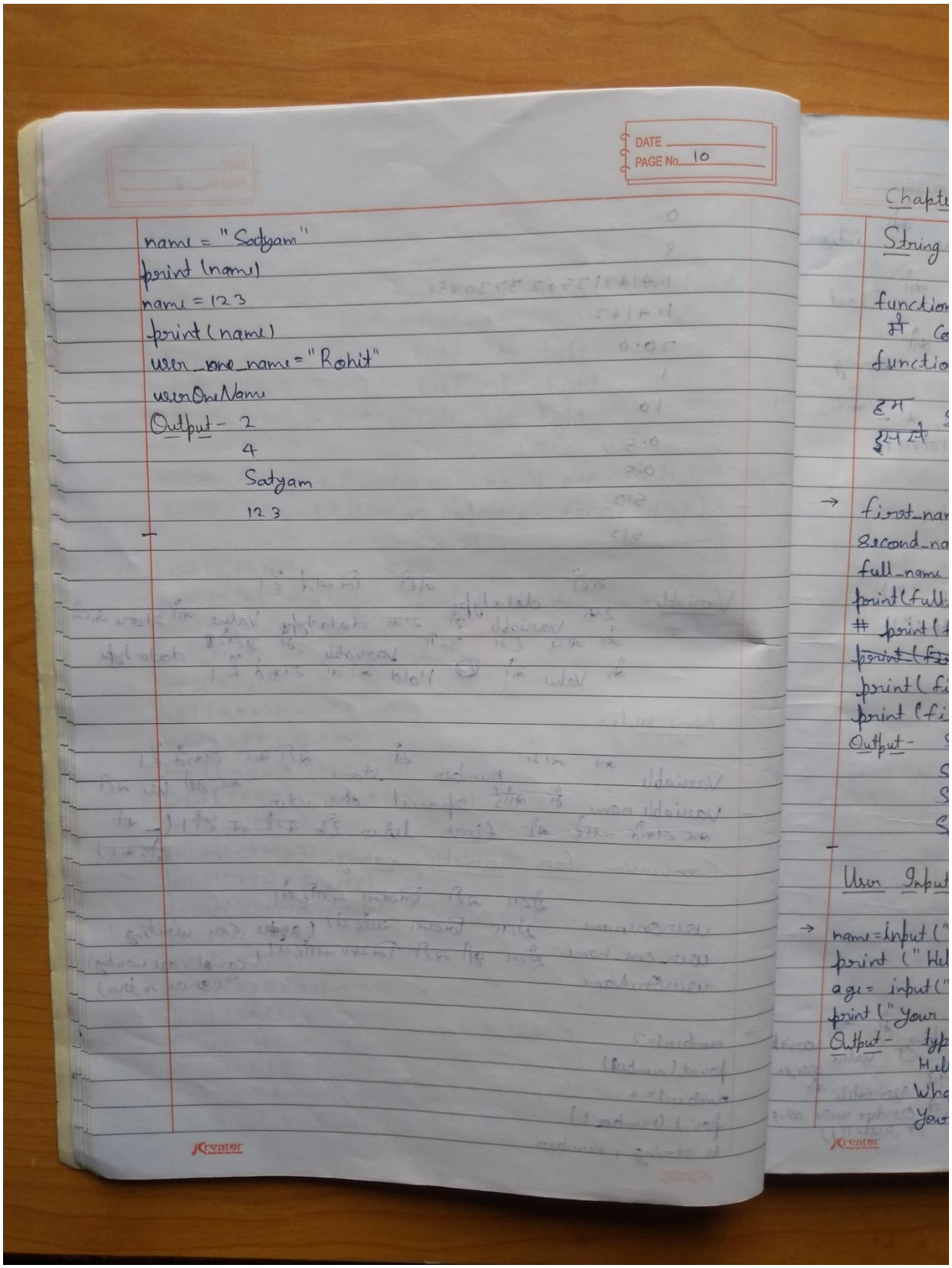
Name rule-

Variable का नाम number से start नहीं कर सकते हैं।
Variable name में कोई spacial character का भी use नहीं कर सकते चाहे वो first letter ही क्यों न हो। (_ को छोड़कर)

Convension for variable naming-

ऐसा नहीं लिखना चाहिये।
userOnename
user_one_name ऐसा लिखना चाहिये। (snake case writing)
userOneName ऐसा भी नहीं लिखना चाहिये। (camel case writing)
↳ (at use in java).

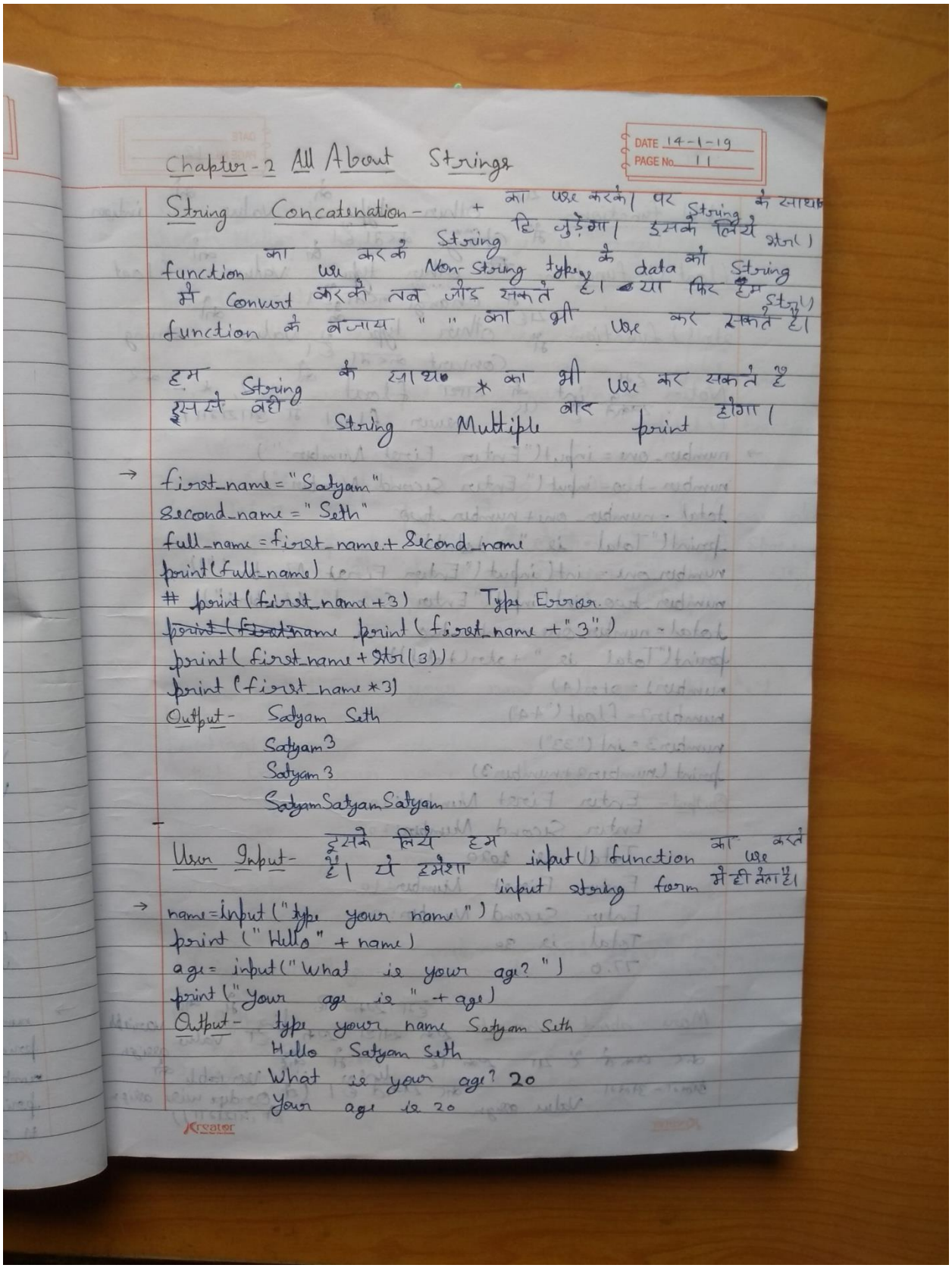
→
```
number1=2
print(number1)
number1=4
print(number1)
# string , number
```



DATE
PAGE No. 10

```
name = "Satyam"
print (name)
name = 123
print (name)
user_one_name = "Rohit"
userOneName
```

Output - 2
4
Satyam
123



DATE 14-1-19
PAGE No. 11

Chapter-2 All About Strings

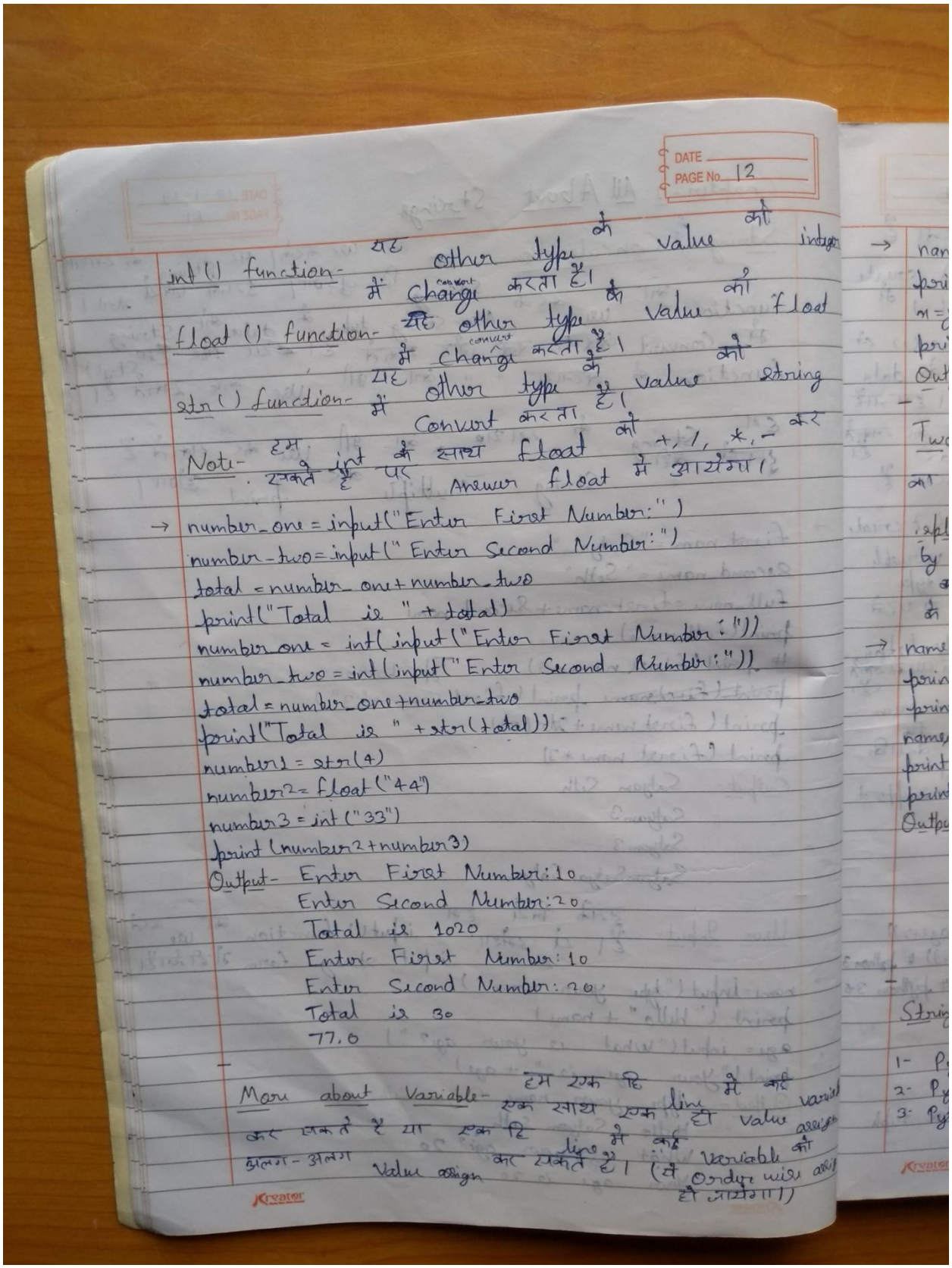
String Concatenation - + का use करके हम String के साथ String को जोड़ेंगे। पर String के साथ किसी और data type को जोड़ने के लिये str() function का use करके Non-String type के data को String में convert करके तब जोड़ सकते हैं। या फिर हम str() function के बजाय " " का भी use कर सकते हैं।

हम String के साथ * का भी use कर सकते हैं इससे वही String Multiple बार print होगा।

→
```
first_name = "Satyam"
second_name = "Seth"
full_name = first_name + second_name
print(full_name)
# print(first_name + 3)  Type Error.
print(first_name + "3")
print(first_name + str(3))
print(first_name * 3)
```
Output - Satyam Seth
Satyam3
Satyam3
SatyamSatyamSatyam

User Input - इसके लिये हम input() function का use करते हैं। ये हमेशा input string form में ही लेता है।

→
```
name = input("type your name")
print("Hello" + name)
age = input("What is your age?")
print("Your age is" + age)
```
Output - type your name Satyam Seth
Hello Satyam Seth
What is your age? 20
Your age is 20



int() function- यह other type के value को integer में change (convert) करता है।

float() function- यह other type के value को float में change (convert) करता है।

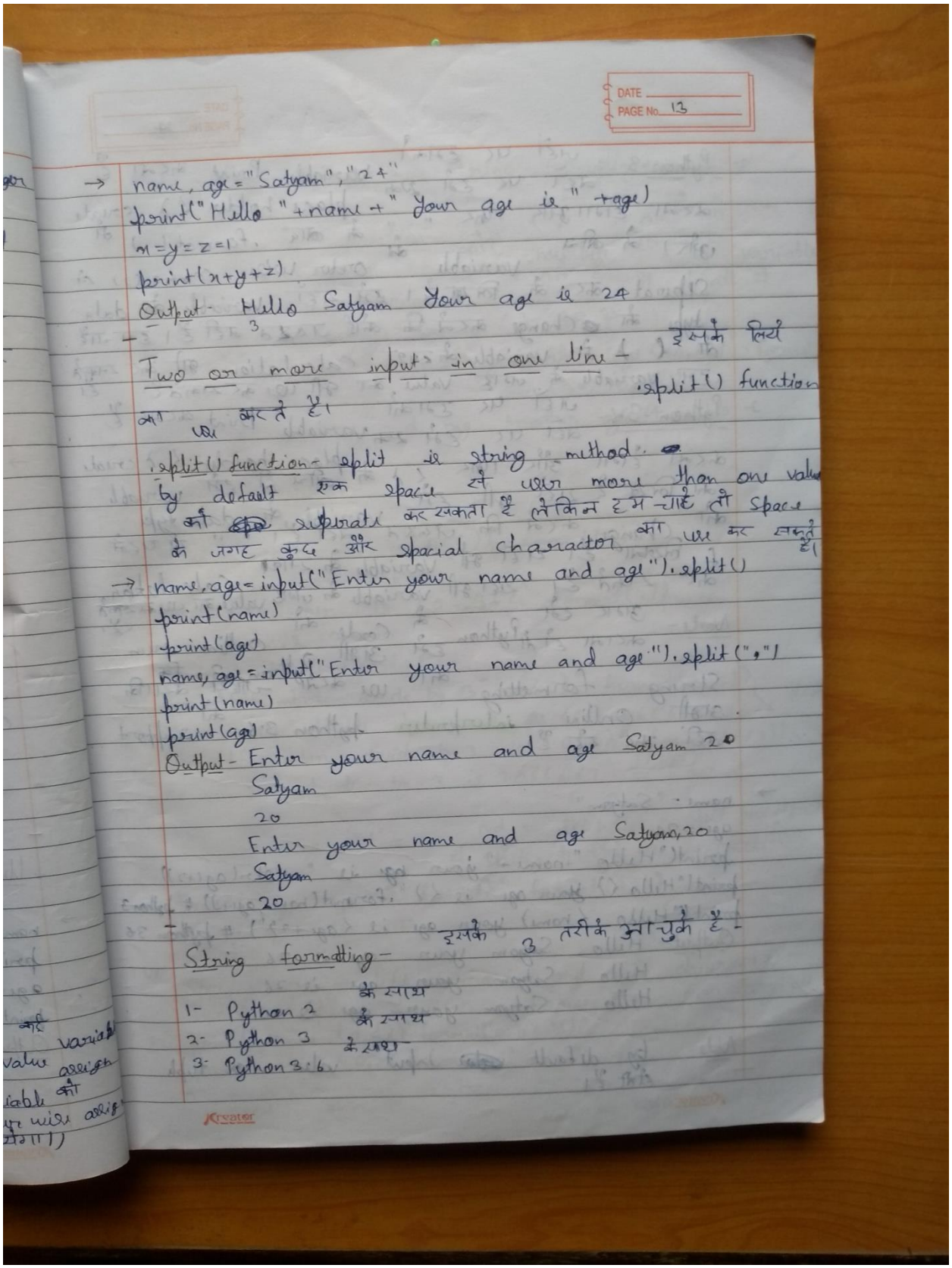
str() function- यह other type के value को string में convert करता है।

Note- हम int के साथ float को +, /, *, - कर सकते हैं पर Answer float में आयेगा।

```
number_one = input("Enter First Number:")
number_two = input("Enter Second Number:")
total = number_one + number_two
print("Total is " + total)
number_one = int(input("Enter First Number:"))
number_two = int(input("Enter Second Number:"))
total = number_one + number_two
print("Total is " + str(total))
number1 = str(4)
number2 = float("4.4")
number3 = int("33")
print(number2 + number3)
```

Output- Enter First Number: 10
Enter Second Number: 20
Total is 1020
Enter First Number: 10
Enter Second Number: 20
Total is 30
77.0

More about Variable- हम एक ही line में कई variable एक साथ एक ही value assign कर सकते हैं या एक ही line में कई variable अलग-अलग value assign कर सकते हैं। (ये Ordre wise assign होते जायेंगे।)



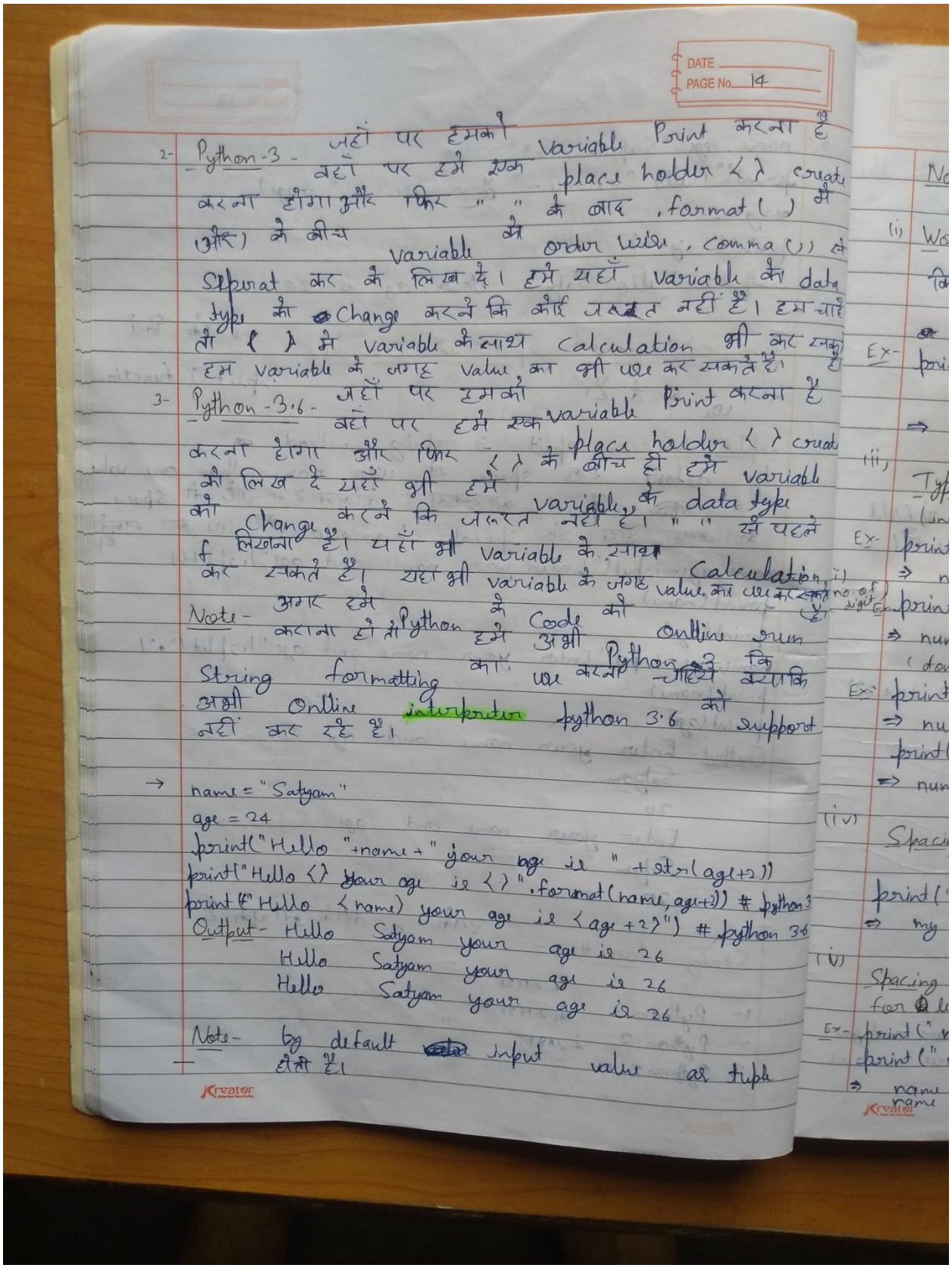
→ name, age = "Satyam", "24"
print("Hello "+name+" Your age is "+age)
x=y=z=1
print(x+y+z)
Output- Hello Satyam Your age is 24
3

Two or more input in one line - इसके लिये
split() function का use करते हैं।

split() function - split is string method.
by default एक space से user more than one value को separate कर सकता है लेकिन हम चाहें तो space के जगह कुछ और special character का use कर सकते हैं।

→ name, age = input("Enter your name and age").split()
print(name)
print(age)
name, age = input("Enter your name and age").split(",")
print(name)
print(age)
Output- Enter your name and age Satyam 20
Satyam
20
Enter your name and age Satyam,20
Satyam
20

String formatting - इसके 3 तरीके आ चुके हैं -
1- Python 2 के साथ
2- Python 3 के साथ
3- Python 3.6 के साथ



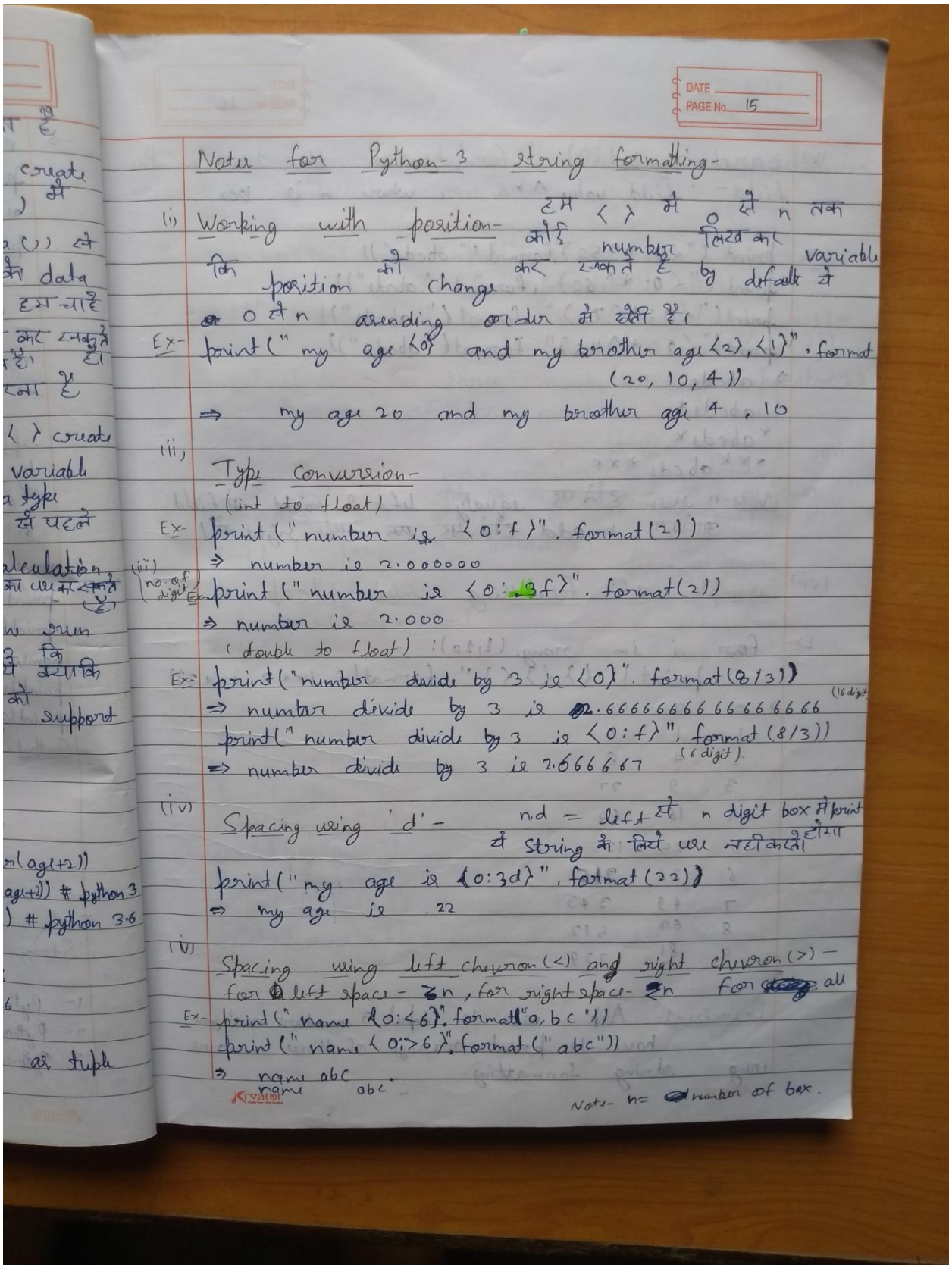
2- Python-3 - जहाँ पर हमको Variable Print करना है वहाँ पर हमें एक place holder { } create करना होगा और फिर " " के बाद .format( ) में (और) के बीच Variable को order wise , comma (,) से Sprrat कर के लिख दे। हमें यहाँ variable का data type को Change करने कि कोई जरूरत नहीं है। हम चाहें तो { } में variable के साथ calculation भी कर सकते हैं। हम variable के जगह value का भी use कर सकते हैं।

3- Python-3.6 - जहाँ पर हमको Print करना है वहाँ पर हमें एक variable create करना होगा और फिर { } के बीच ही हमें variable को लिख दे यहाँ भी हमें variable के data type को Change करने कि जरूरत नहीं है। " " से पहले f लिखना है। यहाँ भी variable के साथ Calculation कर सकते हैं। यहाँ भी variable के जगह value का use कर सकते हैं।

Note - अगर हमें Python के Code को Online run कराना हो तो Python हमें अभी Python 3 का use करने नहीं देता क्योंकि अभी Online interpreter python 3.6 को support नहीं कर रहे हैं।

→
```
name = "Satyam"
age = 24
print("Hello "+name+" your age is "+str(age+2))
print("Hello {} Your age is {}".format(name,age+2)) # python 3
print(f"Hello {name} your age is {age+2}") # python 3.6
```
Output - Hello Satyam your age is 26
Hello Satyam your age is 26
Hello Satyam your age is 26

Note - by default input value as tuple होते हैं।



DATE
PAGE No. 15

Notes for Python-3 string formatting-

(i) Working with position- हम < > में 0 से n तक कोई number लिख कर variable कि position को change कर सकते है by default ये 0 से n asending order में होती है।

Ex- print("my age {0} and my brother age {2}, {1}".format(20, 10, 4))

⇒ my age 20 and my brother age 4, 10

(ii) Type Conversion-
(int to float)

Ex- print("number is {0:f}".format(2))
⇒ number is 2.000000

(iii) (no. of digit)
print("number is {0:.3f}".format(2))
⇒ number is 2.000
(double to float)

Ex- print("number divide by 3 is {0}".format(8/3))
⇒ number divide by 3 is 2.6666666666666666 (16 digit)
print("number divide by 3 is {0:f}", format(8/3))
⇒ number divide by 3 is 2.666667 (6 digit)

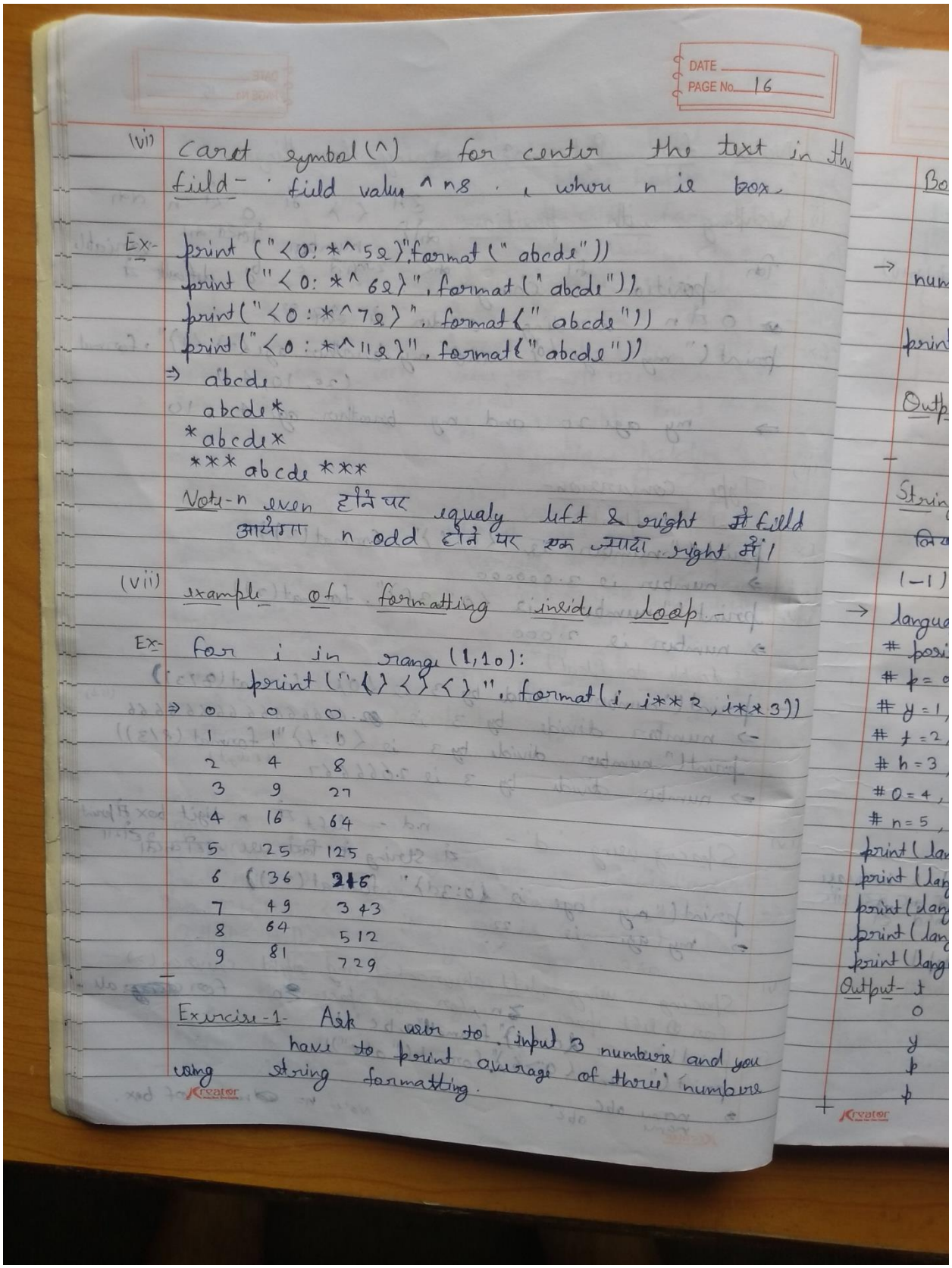
(iv) Spacing using 'd'- n.d = left से n digit box में print में string के लिये use नहीं करते होगा

print("my age is {0:3d}".format(22))
⇒ my age is 22

(v) Spacing using left chevron (<) and right chevron (>) -
for left space - <n, for right space- >n for all

Ex- print("name {0:<6}".format("a, b c"))
print("name {0:>6}".format("abc"))
⇒ name abc
name abc

Note- n= number of box.



(vi) Caret symbol (^) for center the text in the field - field value ^ ns , where n is box.

Ex- print ("<0:*^5s>".format("abcde"))
print ("<0:*^6s>".format("abcde"))
print ("<0:*^7s>".format("abcde"))
print ("<0:*^11s>".format("abcde"))
⇒ abcde
abcde*
abcde
abcde

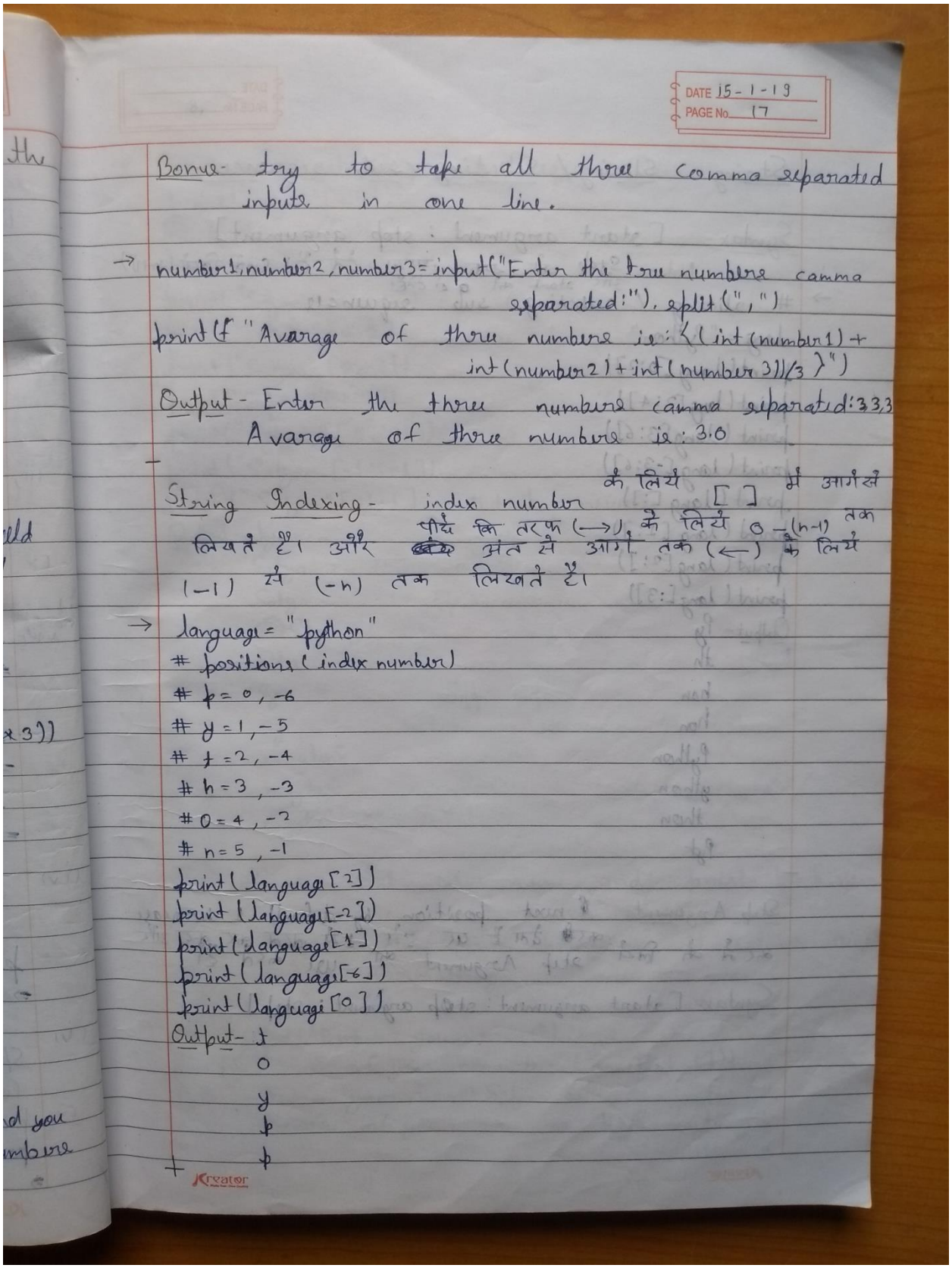
Note- n even होने पर equaly left & right में field आयेगा। n odd होने पर एक ज्यादा right में।

(vii) Example of formatting inside loop -

Ex- for i in range(1,10):
print ("{} {} {}".format(i, i**2, i**3))

⇒
0	0	0
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729

Exercise-1- Ask user to input 3 numbers and you have to print average of three numbers using string formatting.



DATE 15-1-19
PAGE No. 17

Bonus- try to take all three comma separated inputs in one line.

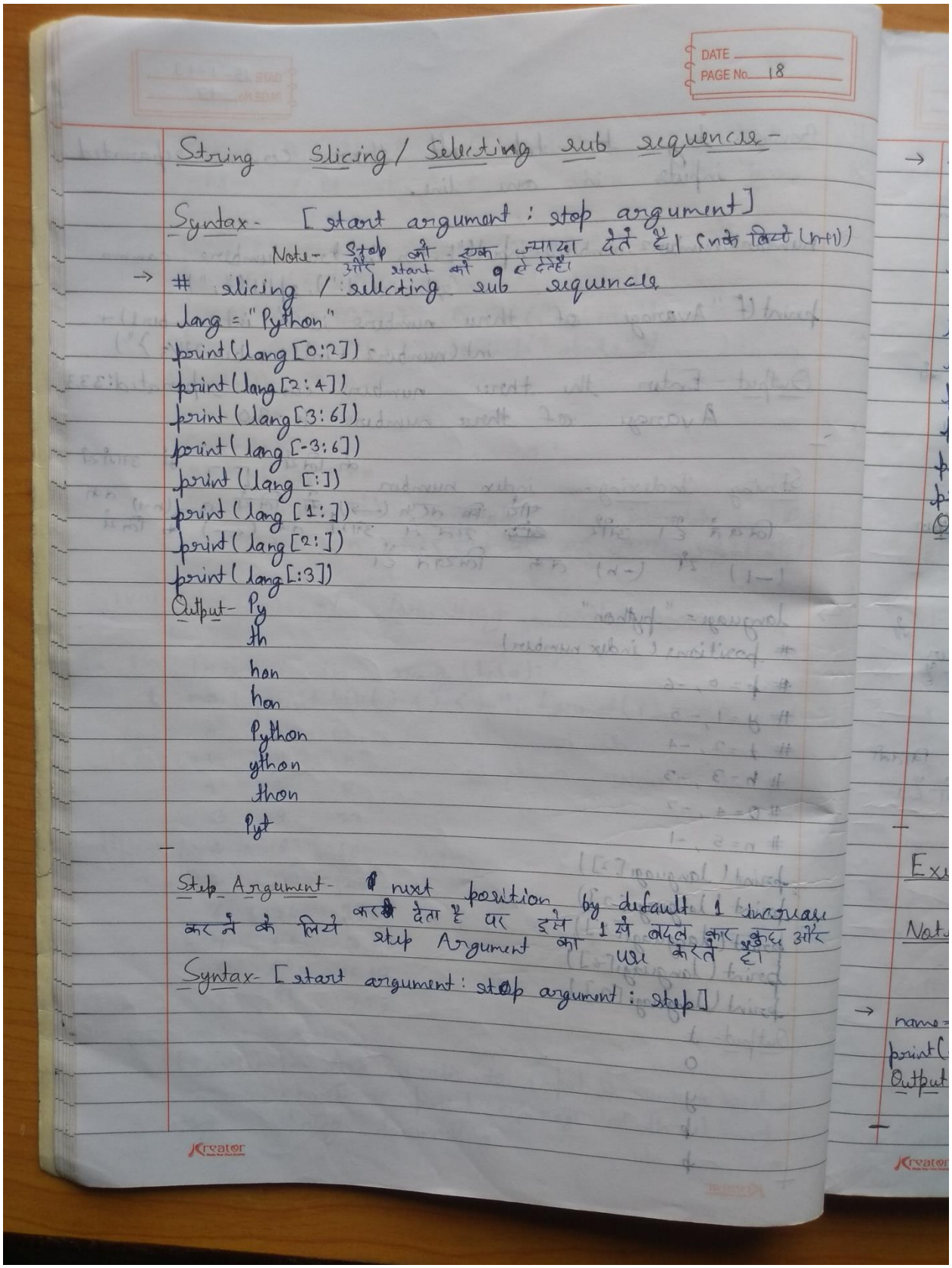
→
```
number1,number2,number3=input("Enter the three numbers comma separated:").split(",")
print("Avarage of three numbers is:{(int(number1)+int(number2)+int(number3))/3}")
```

Output- Enter the three numbers comma separated:3,3,3
Avarage of three numbers is:3.0

String Indexing- index number [ ] के लिये में आगे से पीछे कि तरफ (→), के लिये 0 – (n-1) तक लिखते है। और अंत से आगे तक (←) के लिये (-1) से (-n) तक लिखते है।

→
```
language="python"
# positions (index number)
# p=0,-6
# y=1,-5
# t=2,-4
# h=3,-3
# o=4,-2
# n=5,-1
print(language[2])
print(language[-2])
print(language[1])
print(language[-6])
print(language[0])
```
Output- t
o
y
p
p

Kreator



String Slicing / Selecting sub sequence-

Syntax- [start argument : stop argument]

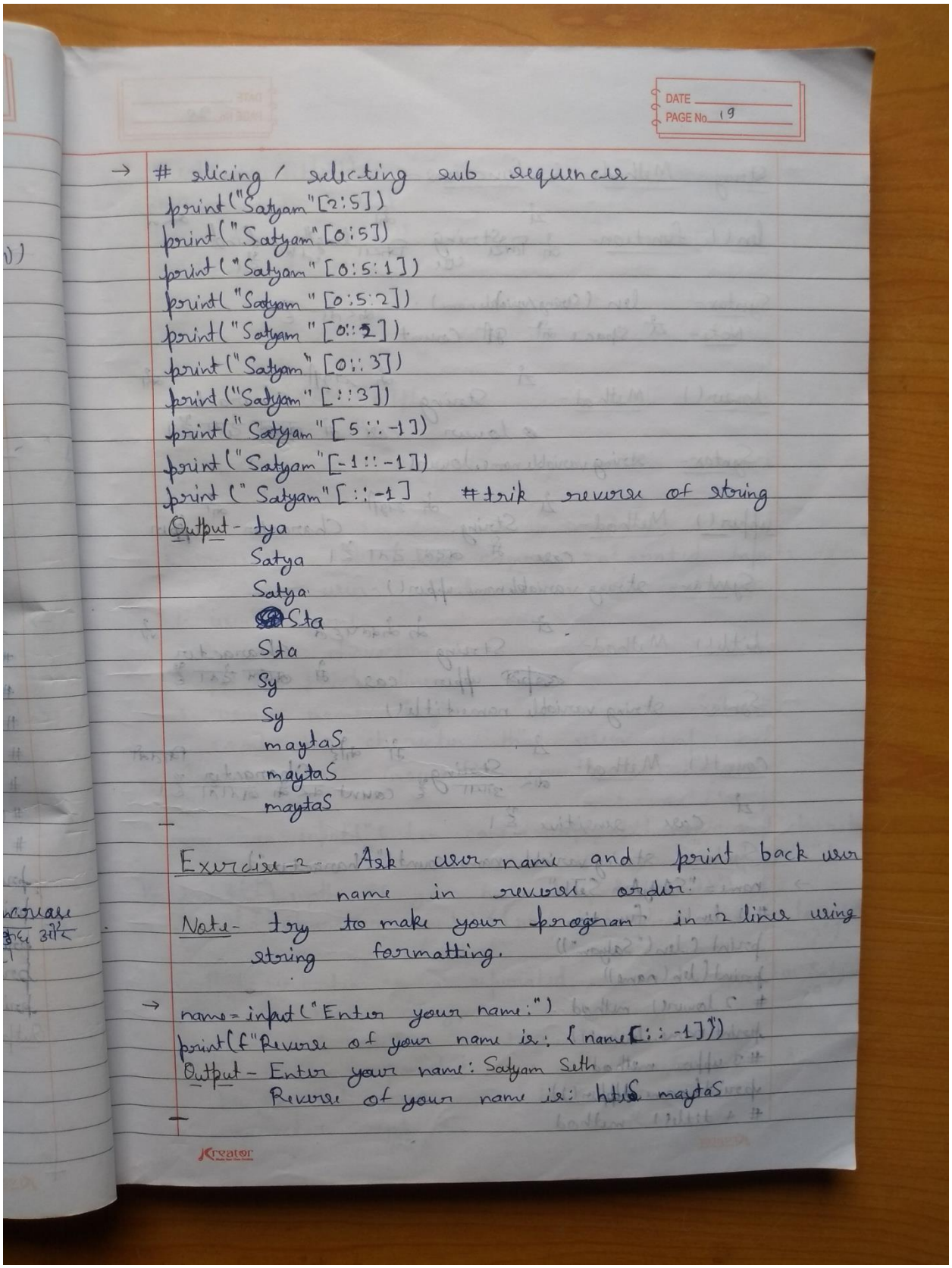
Note- Stop को एक ज्यादा देते है। (n को लिखते (n+1))
और start को 0 दे देते है।

```
# slicing / selecting sub sequence
lang = "Python"
print(lang[0:2])
print(lang[2:4])
print(lang[3:6])
print(lang[-3:6])
print(lang[:])
print(lang[1:])
print(lang[2:])
print(lang[:3])
```

Output- Py
th
hon
hon
Python
ython
thon
Pyt

Step Argument- next position by default 1 increase करके देता है पर इसे 1 से बदल कर कुछ और कर ने के लिये step Argument का use करते है।

Syntax- [start argument : stop argument : step]



→ # slicing / selecting sub sequences
print("Satyam"[2:5])
print("Satyam"[0:5])
print("Satyam"[0:5:1])
print("Satyam"[0:5:2])
print("Satyam"[0::2])
print("Satyam"[0::3])
print("Satyam"[::3])
print("Satyam"[5::-1])
print("Satyam"[-1::-1])
print("Satyam"[::-1] #trik reverse of string

Output - tya
Satya
Satya
Sta
Sta
Sy
Sy
maytaS
maytaS
maytaS

—

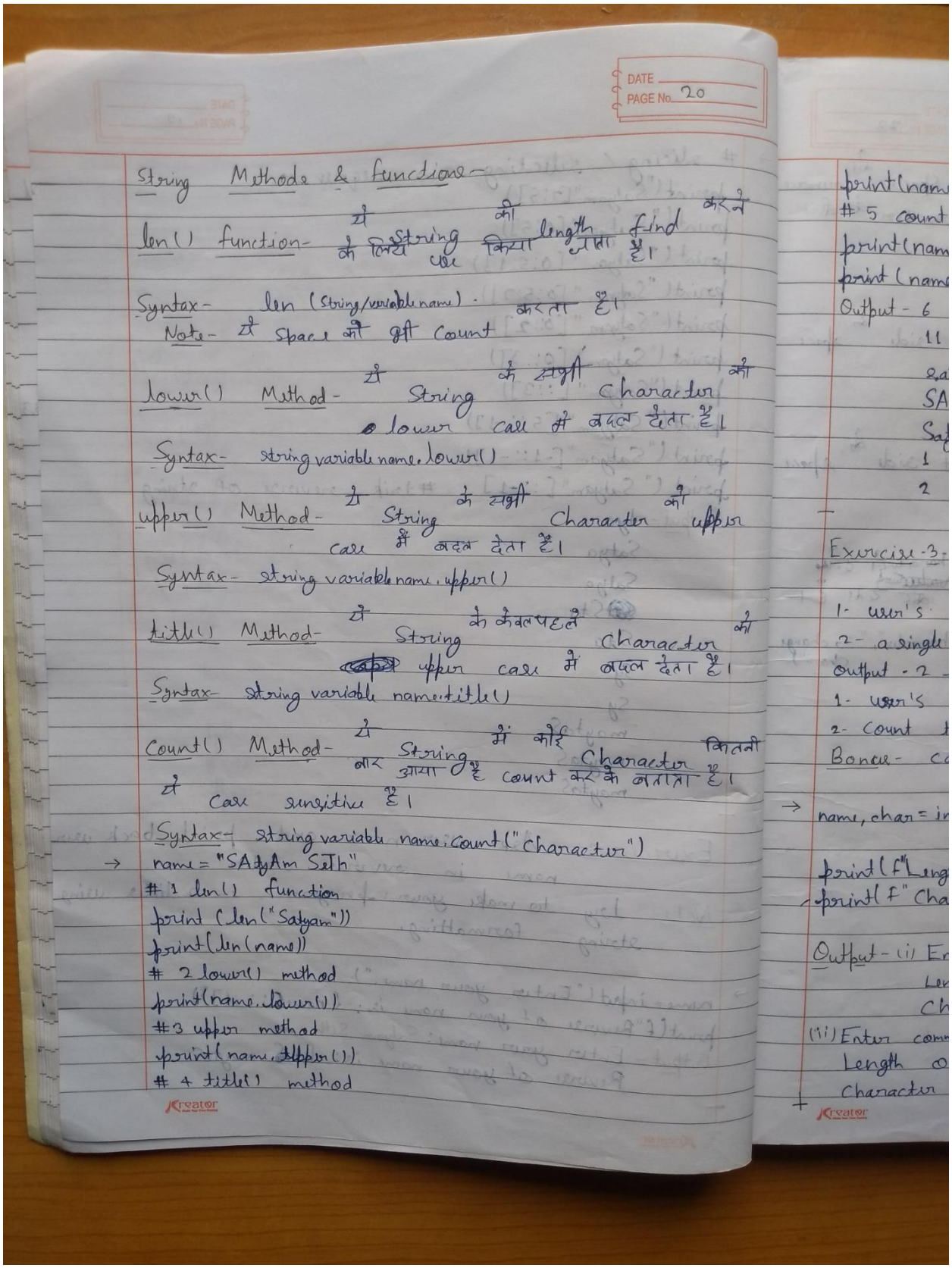
Exercise-2 Ask user name and print back user name in reverse order.

Note- try to make your program in 2 lines using string formatting.

→ name = input("Enter your name:")
print(f"Reverse of your name is: {name[::-1]}")

Output - Enter your name: Satyam Seth
Reverse of your name is: htS maytaS

—



String Methods & functions-

len() function- से String के लिए use किया length find करने का तरीका है।

Syntax- len (String/variable name) करता है।

Note- से Space को get Count

lower() Method- से String के सभी Character को lower case में बदल देता है।

Syntax- string variable name.lower()

upper() Method- से String के सभी Character को upper case में बदल देता है।

Syntax- string variable name.upper()

title() Method- से String के केवल पहले Character को upper case में बदल देता है।

Syntax- string variable name.title()

Count() Method- से String में कोई Character कितनी बार आया है count कर के बताता है।

Case sensitive है।

Syntax- string variable name.count("Character")

```
→ name = "SAtyAm SeTh"
# 1 len() function
print (len("Satyam"))
print(len(name))
# 2 lower() method
print(name.lower())
#3 upper method
print(name.upper())
# 4 title() method
```

print(nam
5 count
print(nam
print (nam
Output - 6
11
SA
Sa
1
2

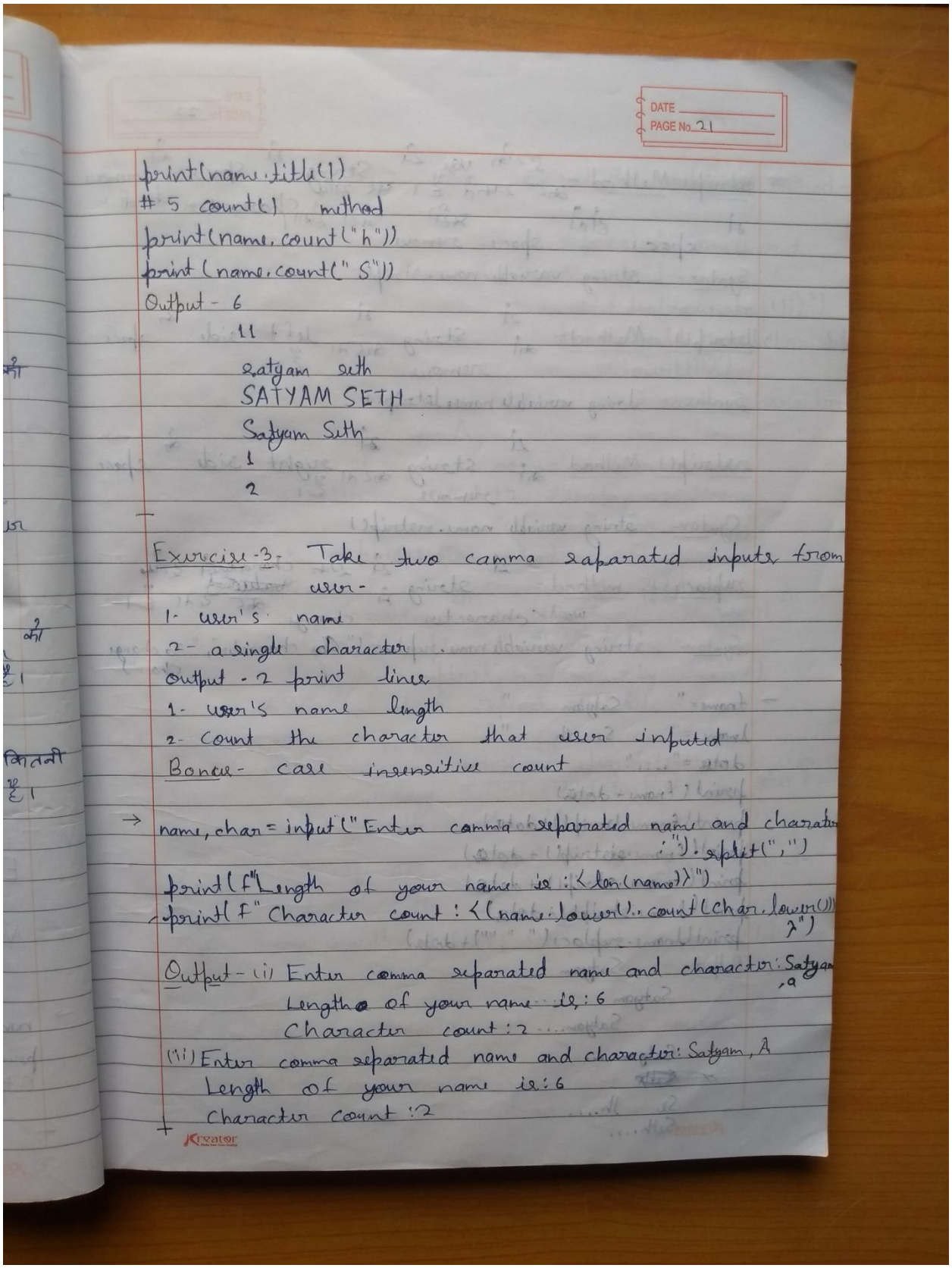
Exercise-3-

1- user's
2- a single
Output - 2
1- user's
2- Count
Bonus- c

→ name, char = in

print(f"Leng
print(f"Cha

Output - (i) En
Le
Ch
(ii) Enter com
Length o
Character



DATE
PAGE No. 21

```
print(name.title())
# 5 count() method
print(name.count("h"))
print(name.count("S"))
Output - 6
         11
```

satyam seth
SATYAM SETH
Satyam Seth
1
2

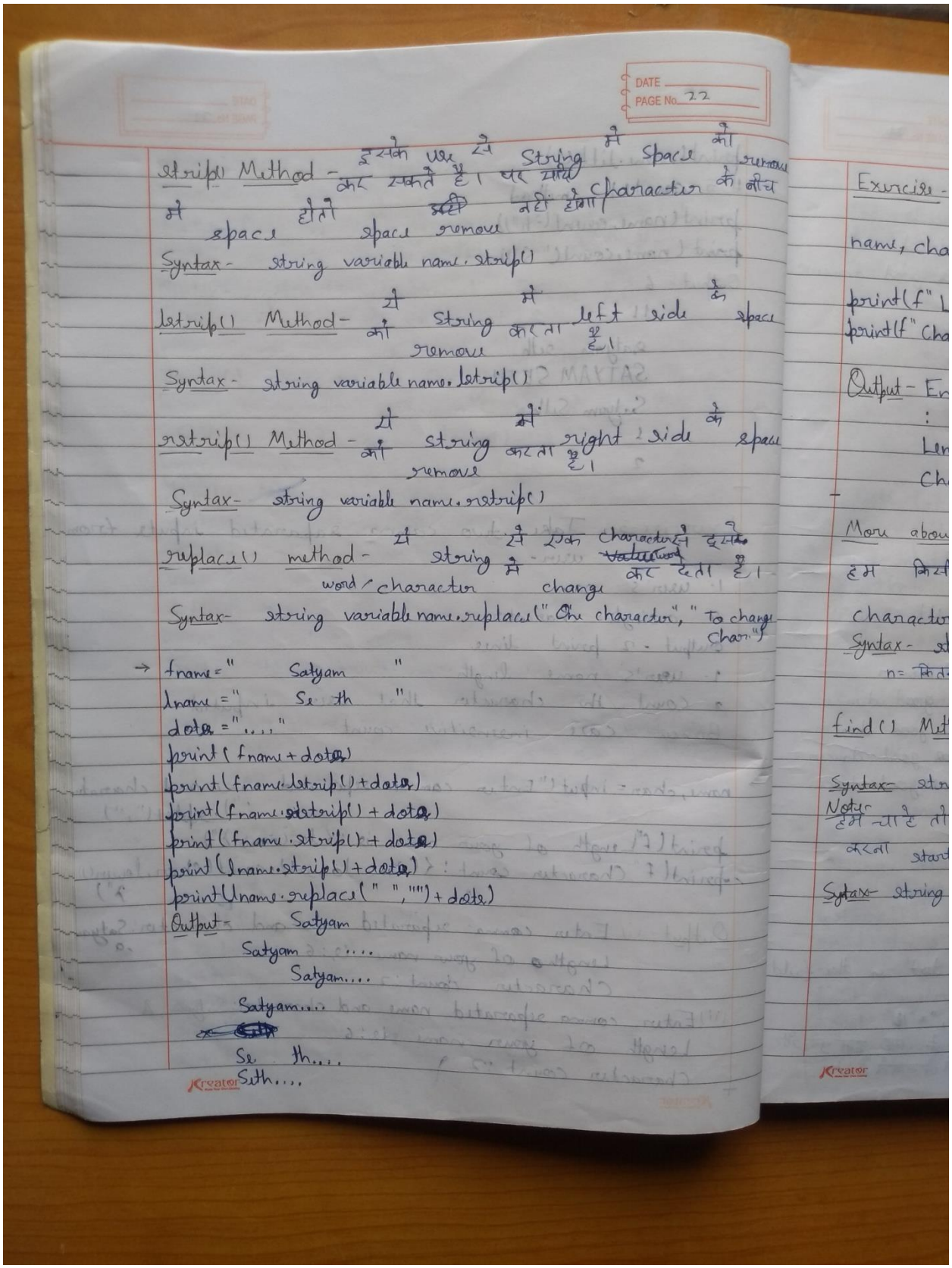
Exercise-3- Take two camma saparated inputs from user-
1- user's name
2- a single character
Output - 2 print lines
1- user's name length
2- count the character that user inputed
Bonus- case insensitive count

→ `name, char = input("Enter camma separated name and charator :").split(",")`

`print(f"Length of your name is :{len(name)}")`
`print(f" Character count : {(name.lower().count(char.lower()))}")`

Output - (i) Enter camma separated name and charactor: Satyam, a
Length of your name is: 6
Character count: 2

(ii) Enter camma separated name and charactor: Satyam, A
Length of your name is: 6
Character count: 2



strip() Method - इसके use से String में space को remove कर सकते हैं। यह यदि character के बीच में space होती है तो space remove नहीं होगा।

Syntax - string variable name.strip()

lstrip() Method - से String में left side के space को remove करता है।

Syntax - string variable name.lstrip()

rstrip() Method - से String में right side के space को remove करता है।

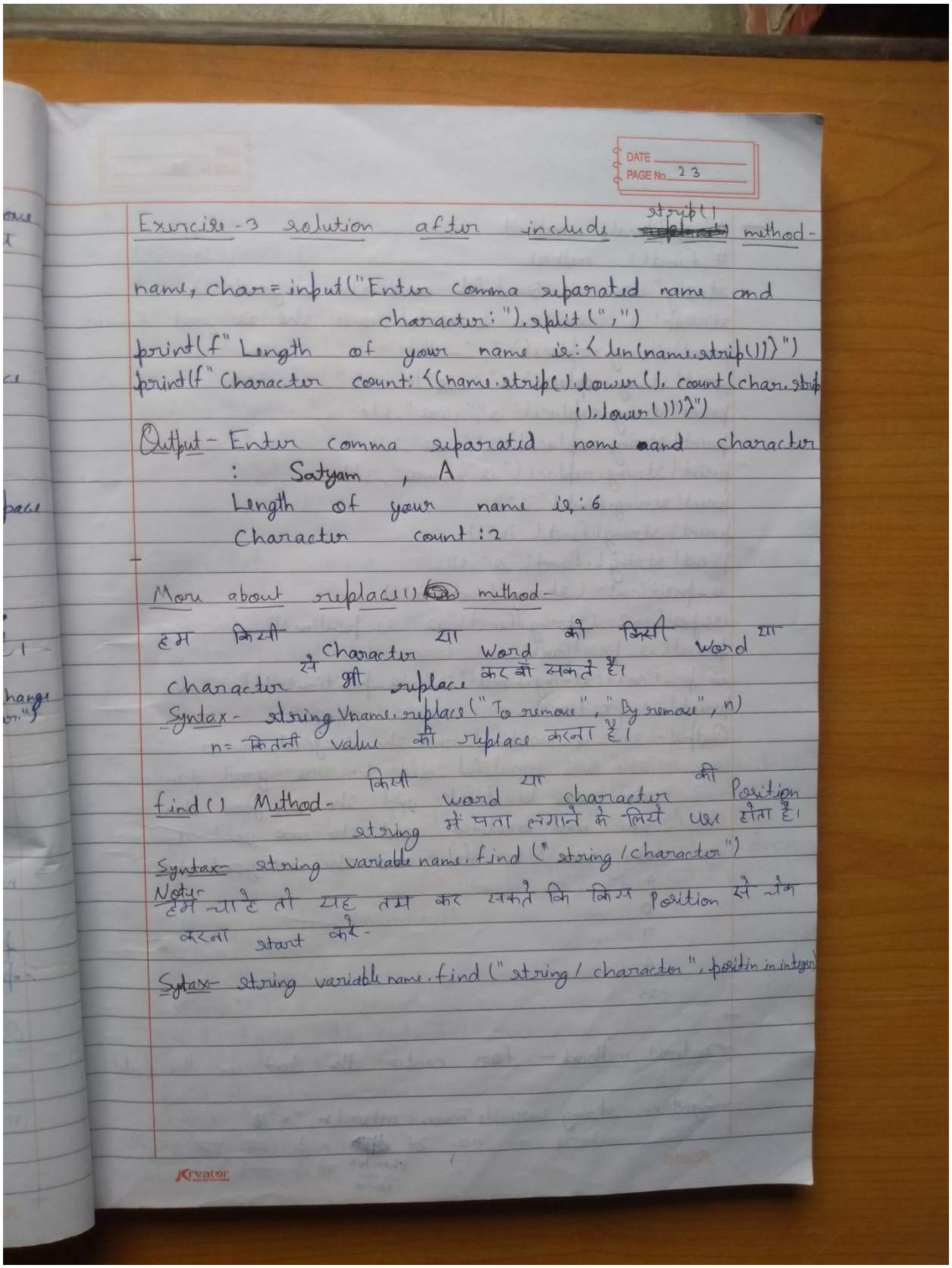
Syntax - string variable name.rstrip()

replace() method - से String में एक character दूसरे word/character से change कर देता है।

Syntax - string variable name.replace("Old character", "To change char.")

→ fname = " Satyam "
lname = " Se th "
data = "...."
print(fname + data)
print(fname.lstrip() + data)
print(fname.rstrip() + data)
print(fname.strip() + data)
print(lname.strip() + data)
print(lname.replace(" ", "") + data)

Output - Satyam....
Satyam....
Satyam....
Satyam....
Se th....
Seth....



DATE
PAGE No. 23

Exercise-3 solution after include strip() method-

```
name, char = input("Enter comma separated name and
                    character: ").split(",")
print(f"Length of your name is: {len(name.strip())}")
printlf"Character count: {(name.strip().lower(). count(char.strip
                    ().lower()))}")
```

Output- Enter comma separated name and character
: Satyam , A
Length of your name is: 6
Character count: 2

More about replace() method-

हम किसी Character या Word को किसी Word या Character से भी replace कर सकते है।

Syntax- string Vname.replace("To remove", "By remove", n)
n= कितनी value को replace करना है।

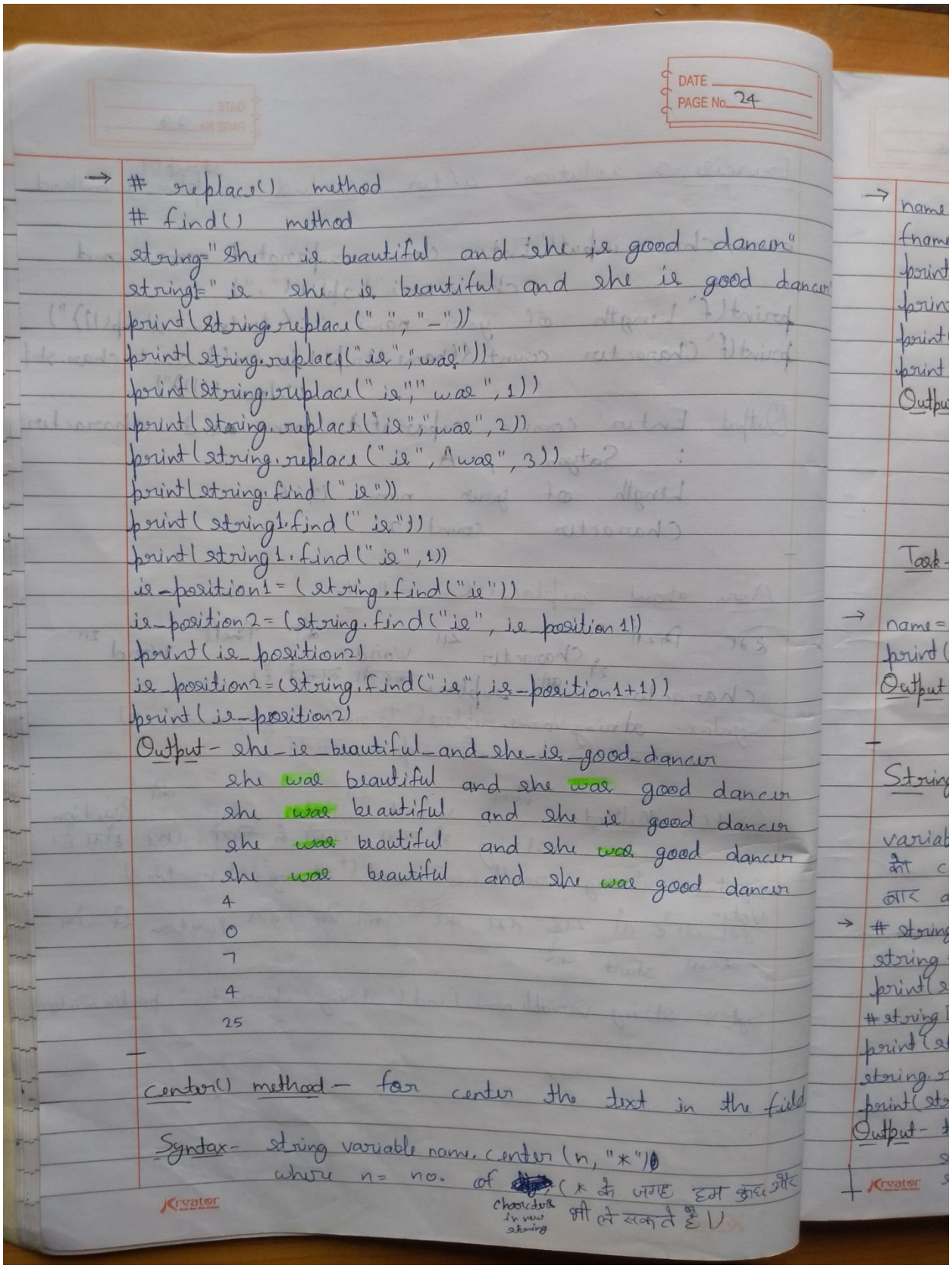
find() Method- किसी word या character की Position string में पता लगाने के लिये use होता है।

Syntax- string variable name.find("string/character")

Note- हम चाहे तो यह तय कर सकते कि किस Position से चेक करना start करें-

Sytax- string variable name.find("string/character", positin in intger)

Harshit Vashisth Page | 25

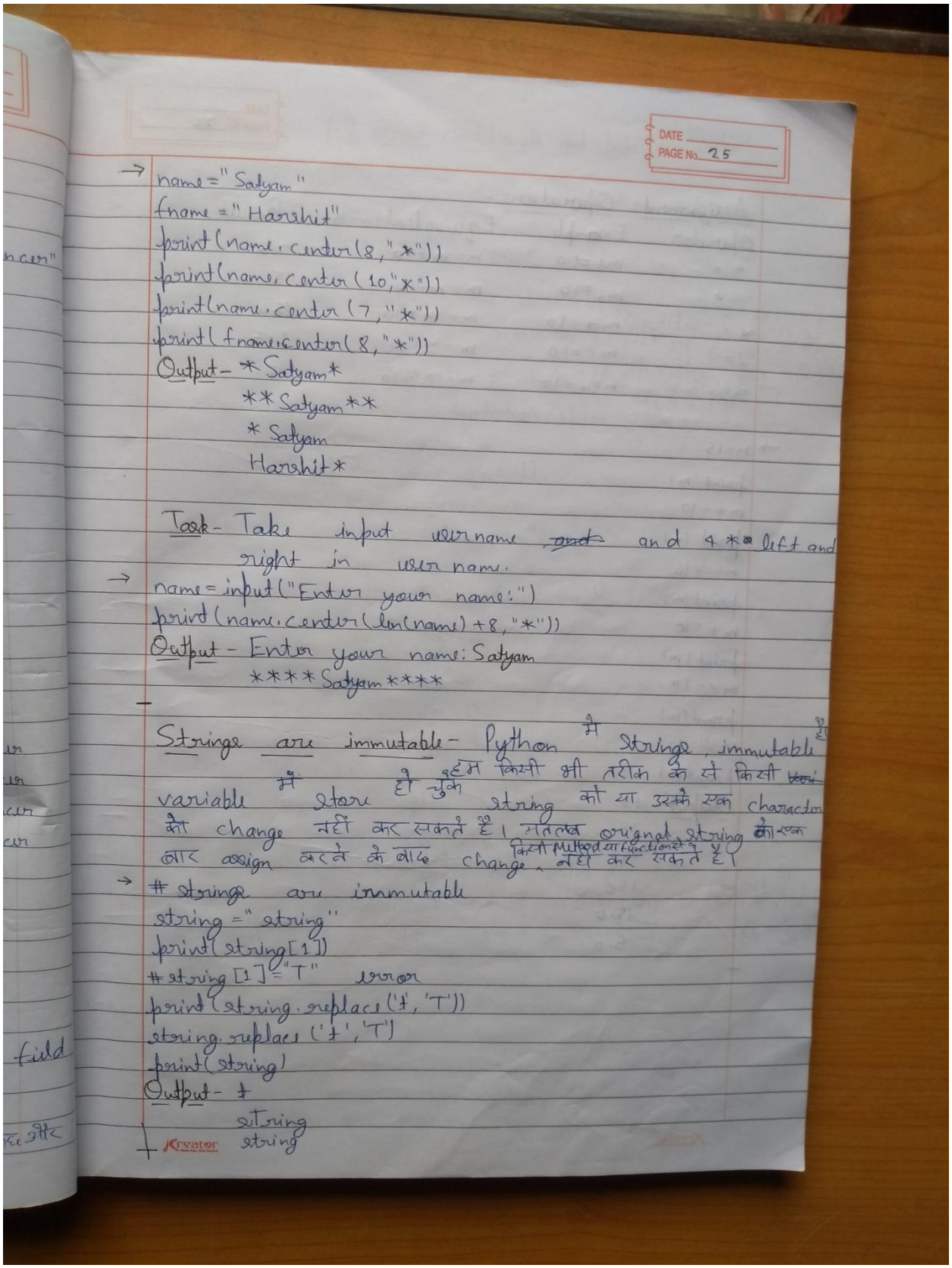


```
# replace() method
# find() method
string="she is beautiful and she is good dancer"
string1="is she is beautiful and she is good dancer"
print(string.replace(" ","-"))
print(string.replace("is","was"))
print(string.replace("is","was",1))
print(string.replace("is","was",2))
print(string.replace("is","was",3))
print(string.find("is"))
print(string1.find("is"))
print(string1.find("is",1))
is_position1=(string.find("is"))
is_position2=(string.find("is",is_position1))
print(is_position2)
is_position2=(string.find("is",is_position1+1))
print(is_position2)
```

Output- she-is-beautiful-and-she-is-good-dancer
she was beautiful and she was good dancer
she was beautiful and she is good dancer
she was beautiful and she was good dancer
4
0
7
4
25

center() method- for center the text in the field

Syntax- string variable name.center(n,"*")
where n= no. of characters in new string (* के जगह हम कुछ और भी ले सकते है।)



DATE
PAGE No. 25

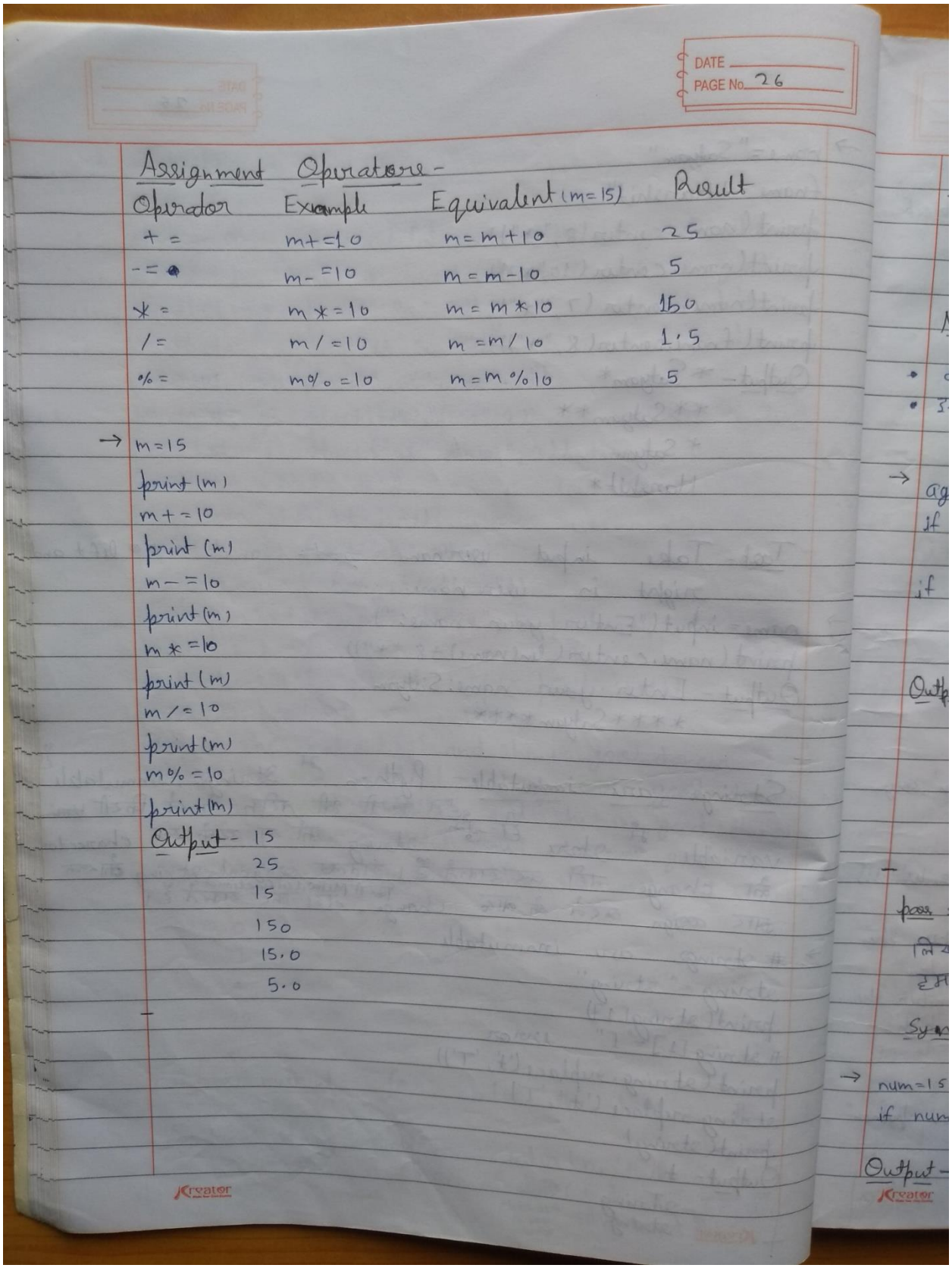
→
```
name = "Satyam"
fname = "Harshit"
print(name.center(8,"*"))
print(name.center(10,"*"))
print(name.center(7,"*"))
print(fname.center(8,"*"))
```
Output - *Satyam*
Satyam
*Satyam
Harshit*

Task - Take input username and 4 ** left and right in username.

→
```
name = input("Enter your name:")
print(name.center(len(name)+8,"*"))
```
Output - Enter your name: Satyam
****Satyam****

Strings are immutable - Python में String, immutable है। हम किसी भी तरीके से किसी variable में store हो चुके string को या उसके एक character को change नहीं कर सकते हैं। अतएव original string को एक बार assign करने के बाद किसी method या function से change नहीं कर सकते हैं।

→
```
# strings are immutable
string = "string"
print(string[1])
# string[1] = "T"  error
print(string.replace('t','T'))
string.replace('t','T')
print(string)
```
Output - t
string
string



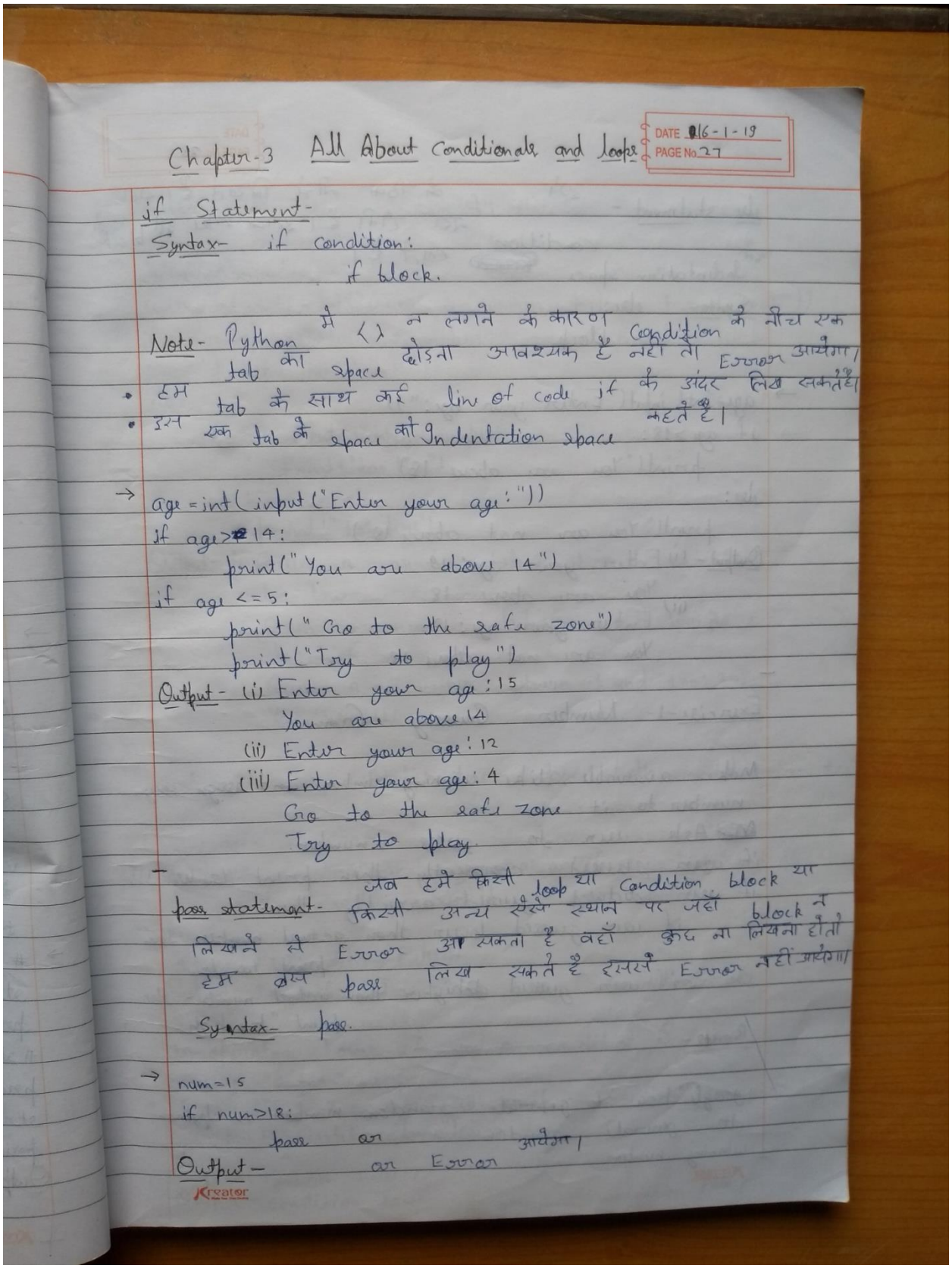
DATE
PAGE No. 26

Assignment Operators -

Operator	Example	Equivalent (m=15)	Result
+=	m+=10	m=m+10	25
-=	m-=10	m=m-10	5
*=	m*=10	m=m*10	150
/=	m/=10	m=m/10	1.5
%=	m%=10	m=m%10	5

→
```
m=15
print(m)
m+=10
print(m)
m-=10
print(m)
m*=10
print(m)
m/=10
print(m)
m%=10
print(m)
```

Output -
```
15
25
15
150
15.0
5.0
```



Chapter-3 All About Conditionals and loops

DATE 16-1-19
PAGE No. 27

if Statement-

Syntax- if condition:
    if block.

Note- Python में <> न लगने के कारण condition के नीचे एक tab का space छोड़ना आवश्यक है नहीं तो Error आयेगा।

- हम tab के साथ कई line of code if के अंदर लिख सकते हैं।
- इस एक tab के space को Indentation space कहते हैं।

→
```
age = int(input("Enter your age:"))
if age>14:
    print("You are above 14")
if age<=5:
    print("Go to the safe zone")
    print("Try to play")
```

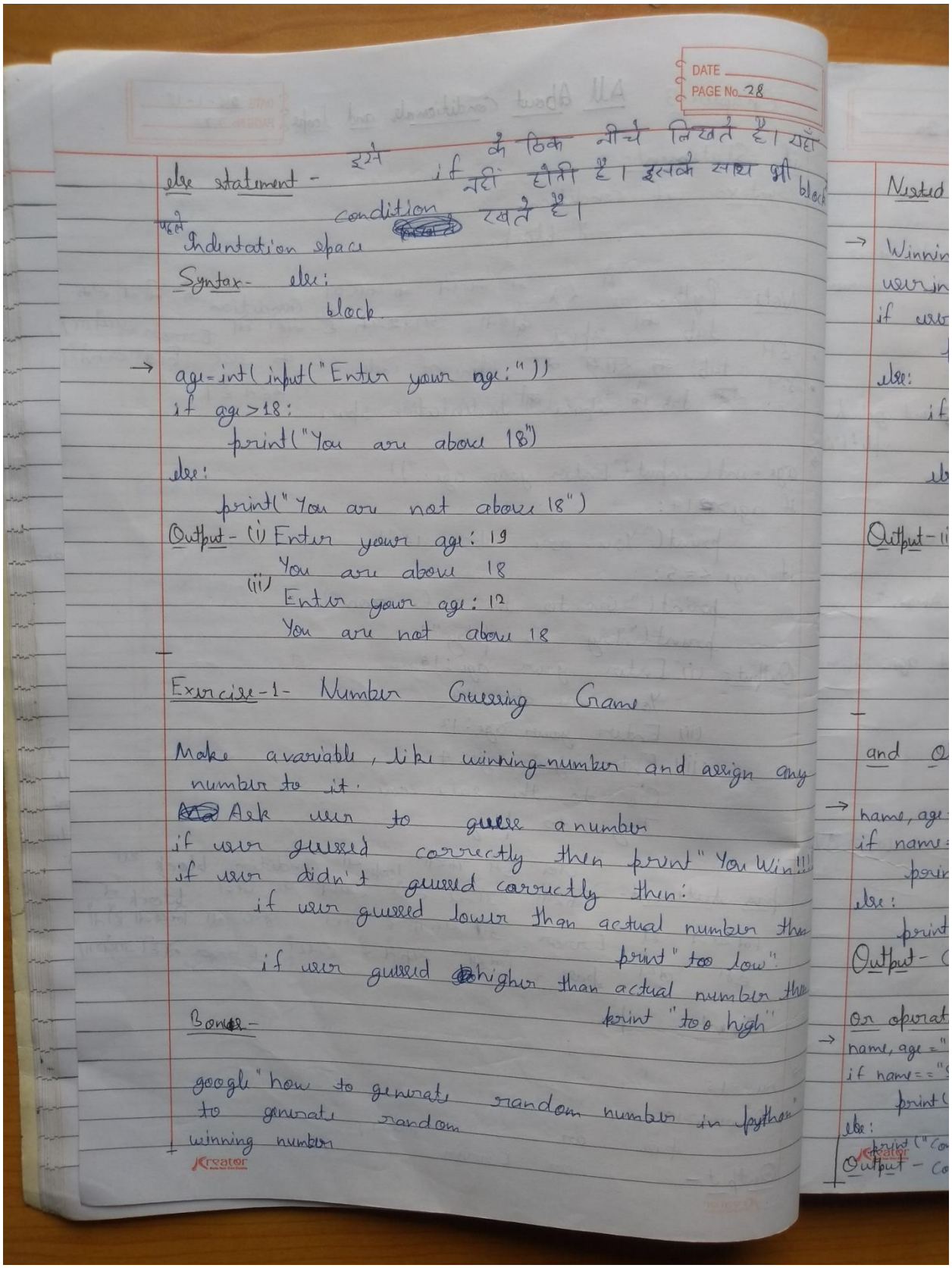
Output- (i) Enter your age: 15
You are above 14
(ii) Enter your age: 12
(iii) Enter your age: 4
Go to the safe zone
Try to play

pass statement- जब हमें किसी loop या condition block या किसी अन्य ऐसे स्थान पर जहाँ block न लिखने से Error आ सकता है वहाँ कुछ ना लिखना होतो हम वहाँ pass लिख सकते हैं इससे Error नहीं आयेगा।

Syntax- pass

→
```
num=15
if num>18:
    pass
```

Output- कोई Error आयेगा।



DATE
PAGE No. 28

else statement - इसे if के ठीक नीचे लिखते हैं। यहाँ condition नहीं होती है। इसके साथ भी block रखते हैं।
पहले Indentation space

Syntax- else:
    block

→ age=int(input("Enter your age:"))
if age>18:
    print("You are above 18")
else:
    print("You are not above 18")

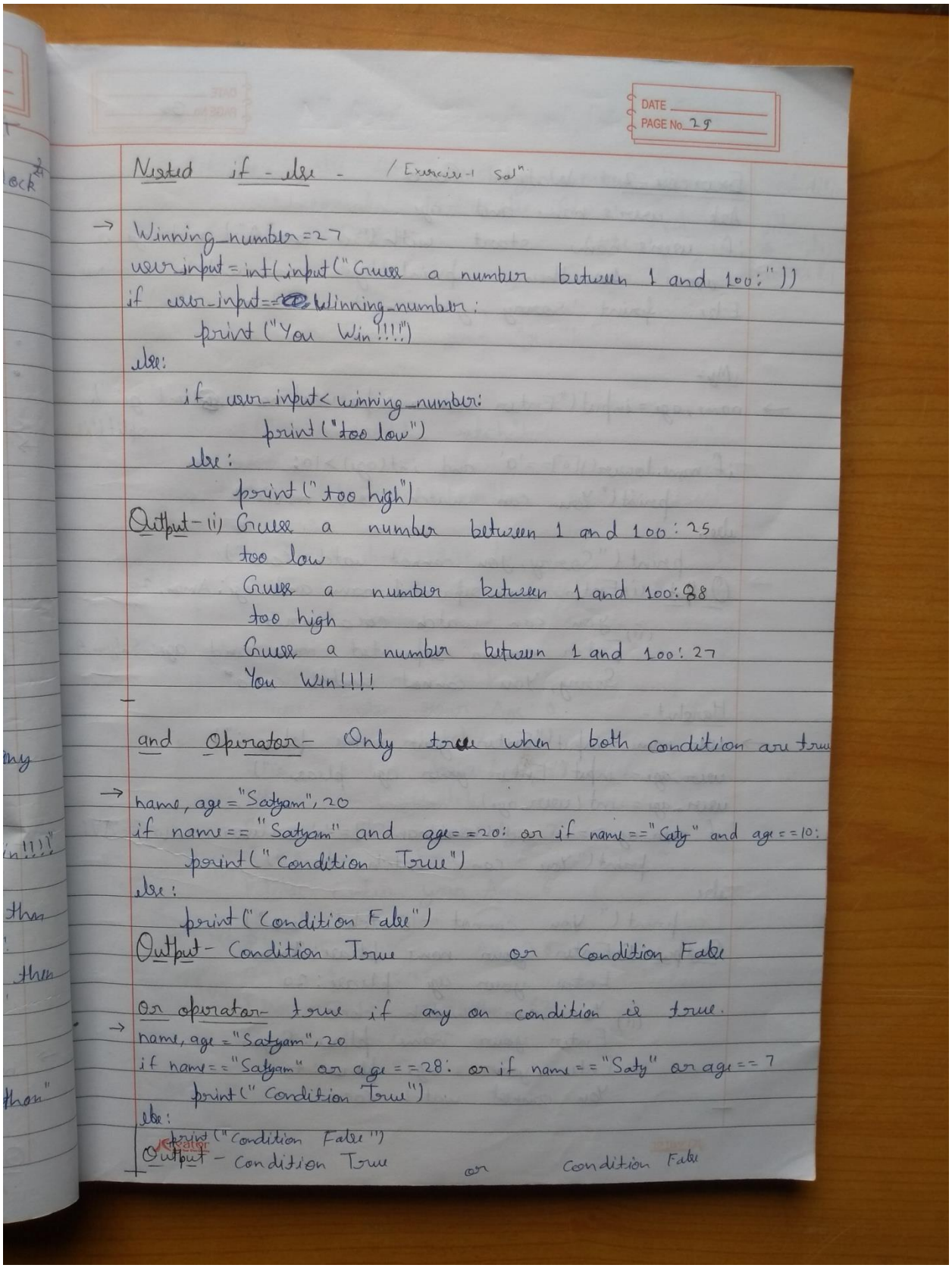
Output- (i) Enter your age: 19
    You are above 18
(ii) Enter your age: 12
    You are not above 18

Exercise-1- Number Guessing Game

Make a variable, like winning_number and assign any number to it.
Ask user to guess a number
if user guessed correctly then print "You Win!!!"
if user didn't guessed correctly then:
    if user guessed lower than actual number then print "too low"
    if user guessed higher than actual number then print "too high"

Bonus-

google "how to generate random number in python"
to generate random winning number



DATE
PAGE No. 29

Nested if - else - / Exercise-1 Soln

→
```
Winning_number=27
user_input=int(input("Guess a number between 1 and 100:"))
if user_input==Winning_number:
    print("You Win!!!!")
else:
    if user_input<winning_number:
        print("too low")
    else:
        print("too high")
```

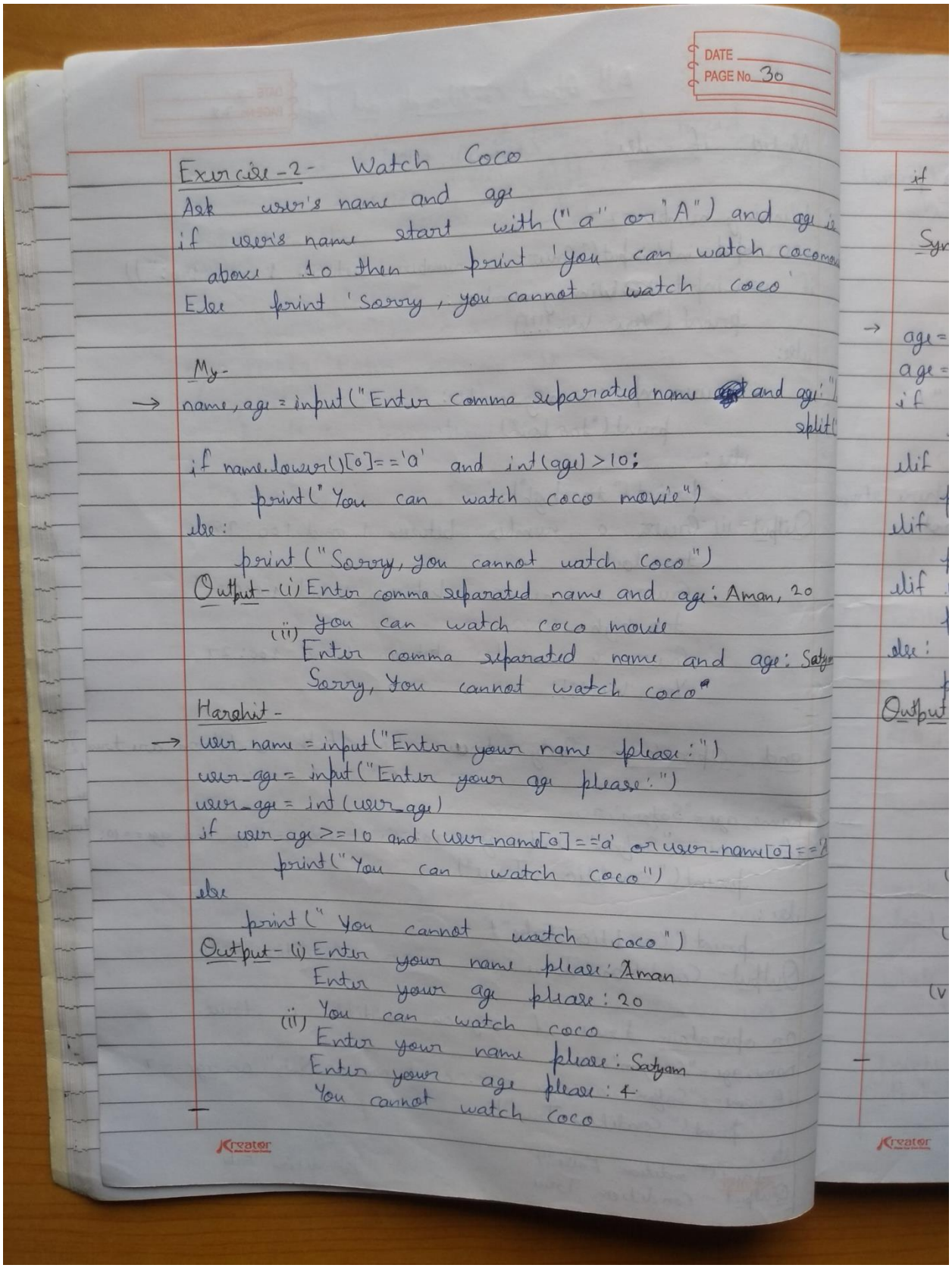
Output- ii) Guess a number between 1 and 100: 25
too low
Guess a number between 1 and 100: 88
too high
Guess a number between 1 and 100: 27
You Win!!!

and Operator- Only true when both condition are true

→
```
name, age="Satyam", 20
if name=="Satyam" and age==20: or if name=="Saty" and age==10:
    print("Condition True")
else:
    print("Condition False")
```
Output- Condition True or Condition False

Or operator- true if any on condition is true.

→
```
name, age="Satyam", 20
if name=="Satyam" or age==28: or if name=="Saty" or age==7
    print("Condition True")
else:
    print("Condition False")
```
Output - Condition True or Condition False



DATE
PAGE No. 30

Exercise-2- Watch Coco

Ask user's name and age
if user's name start with ("a" or "A") and age is above 10 then print 'you can watch cocomovie'
Else print 'Sorry, you cannot watch coco'

My-

→ name, age = input("Enter comma separated name and age:").split()

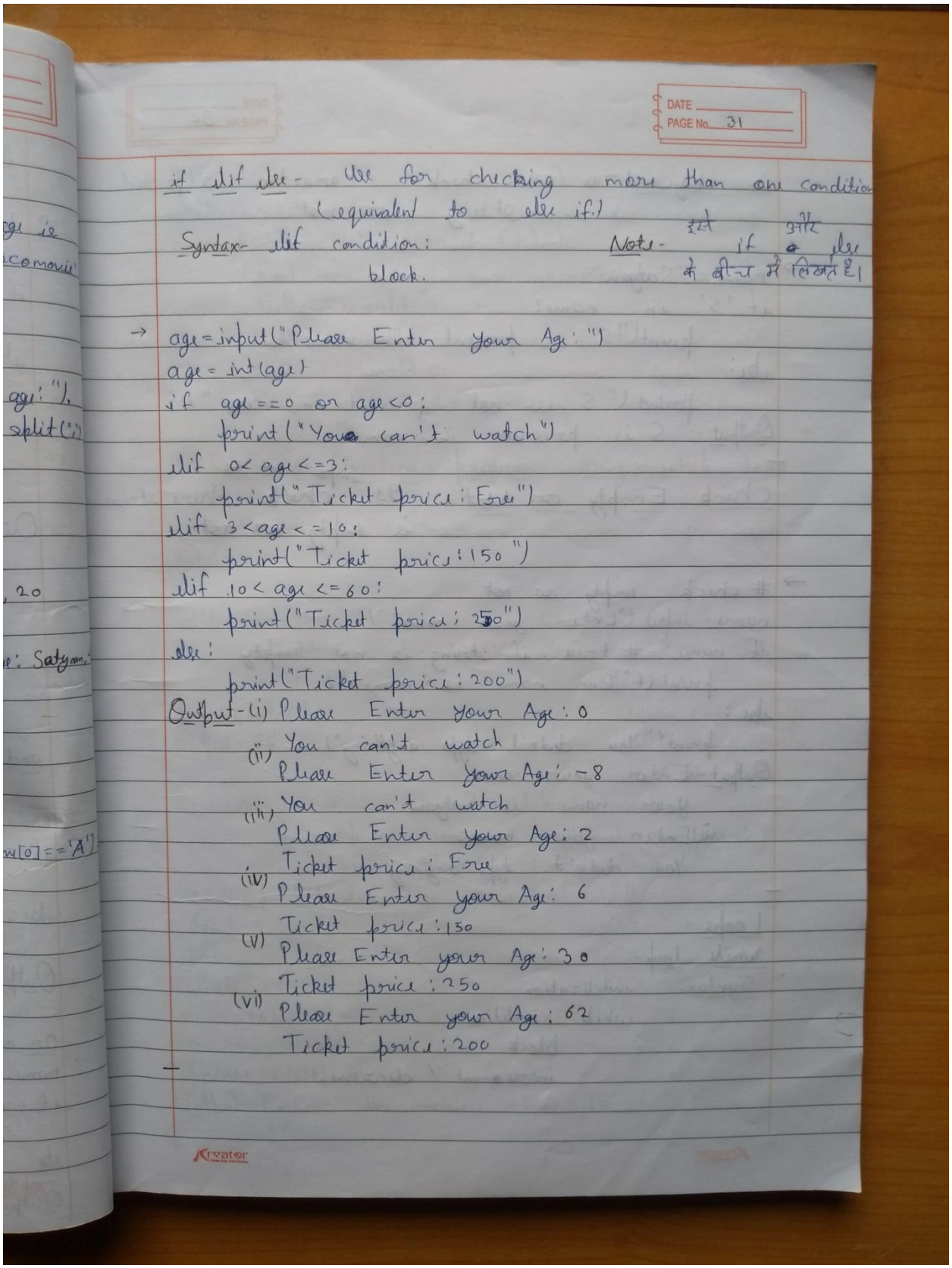
if name.lower()[0]=='a' and int(age)>10:
    print("You can watch coco movie")
else:
    print("Sorry, you cannot watch coco")

Output- (i) Enter comma separated name and age: Aman, 20
You can watch coco movie
(ii) Enter comma separated name and age: Satyam, 4
Sorry, You cannot watch coco

Harshit-

→ user_name = input("Enter your name please:")
user_age = input("Enter your age please:")
user_age = int(user_age)
if user_age >= 10 and (user_name[0]=='a' or user_name[0]=='A'):
    print("You can watch coco")
else
    print("You cannot watch coco")

Output- (i) Enter your name please: Aman
Enter your age please: 20
You can watch coco
(ii) Enter your name please: Satyam
Enter your age please: 4
You cannot watch coco



DATE
PAGE No. 31

if elif else - Use for checking more than one condition
(equivalent to else if.)

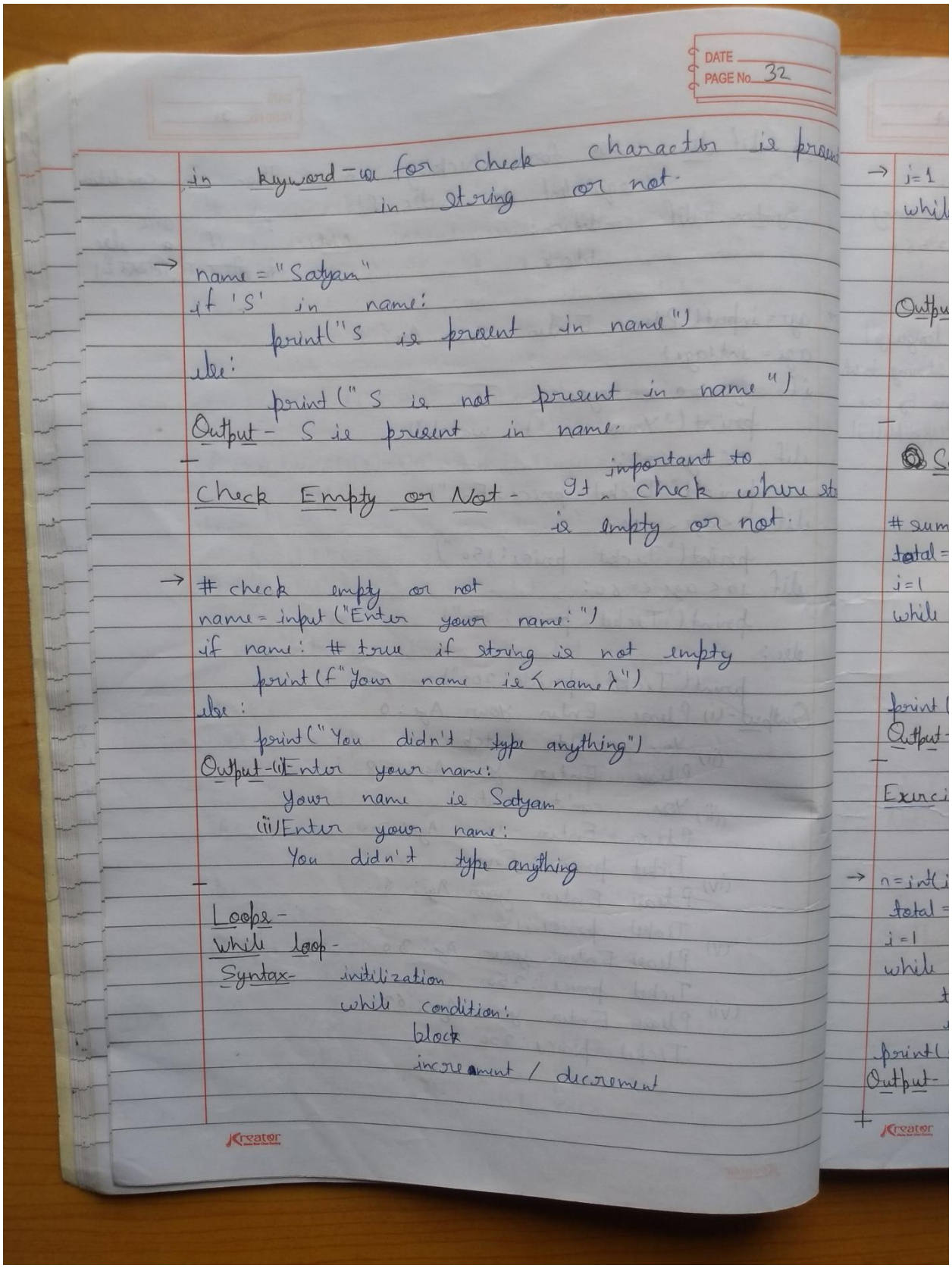
Syntax- elif condition:
block.

Note- if और else के बीच में लिखते हैं।

→
```
age=input("Please Enter your Age: ")
age = int(age)
if age==0 or age<0:
    print("You can't watch")
elif 0< age <=3:
    print("Ticket price: Free")
elif 3<age<=10:
    print("Ticket price:150")
elif 10< age <=60:
    print("Ticket price: 250")
else:
    print("Ticket price:200")
```

Output- (i) Please Enter your Age: 0
(ii) You can't watch
Please Enter your Age: -8
(iii) You can't watch
Please Enter your Age: 2
(iv) Ticket price: Free
Please Enter your Age: 6
(v) Ticket price: 150
Please Enter your Age: 30
(vi) Ticket price: 250
Please Enter your Age: 62
Ticket price: 200

Kreator



DATE
PAGE No. 32

in keyword - use for check character is present in string or not.

```
name = "Satyam"
if 'S' in name:
    print("S is present in name")
else:
    print("S is not present in name")
```

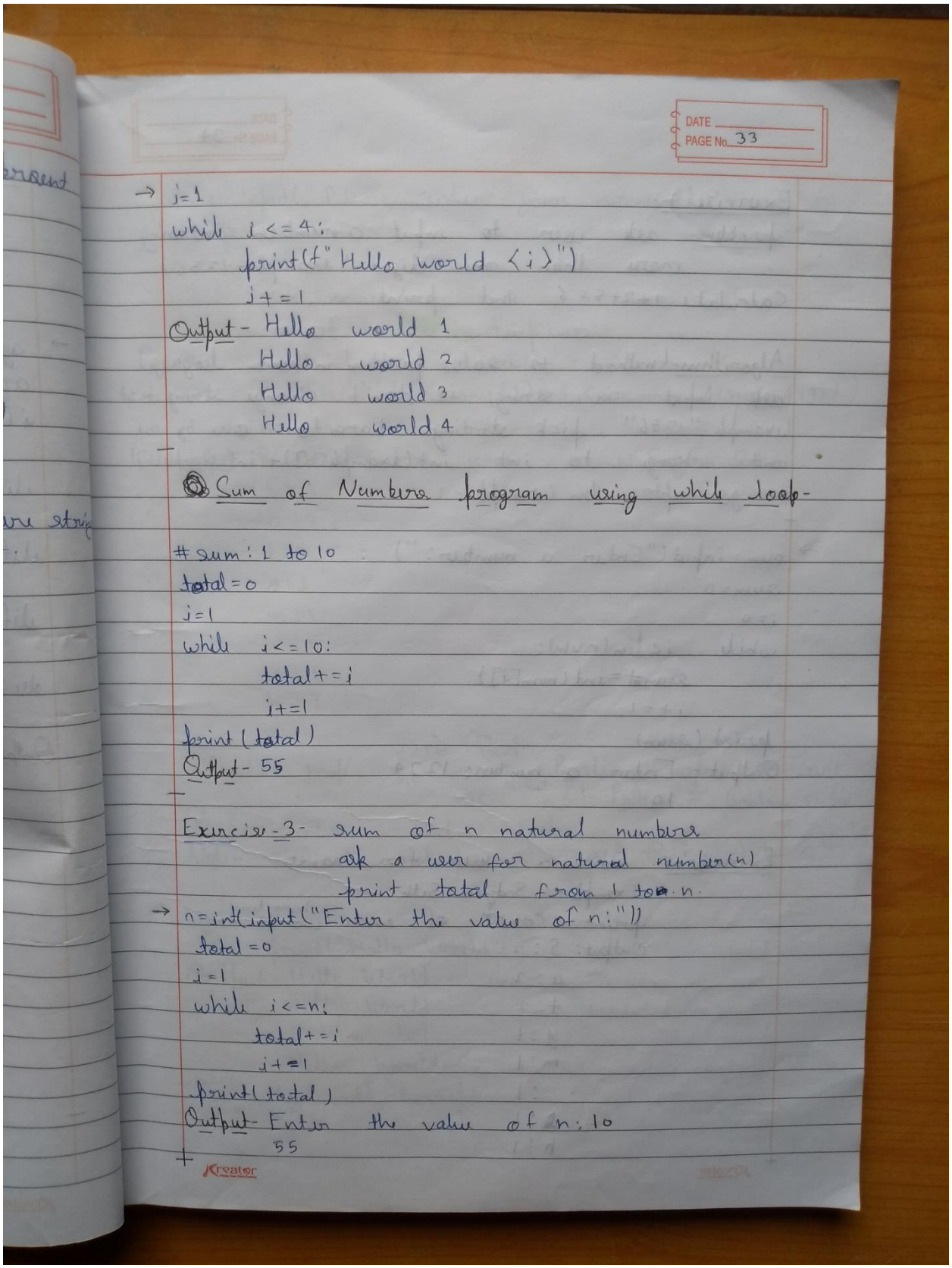
Output - S is present in name.

Check Empty or Not - It is important to check where string is empty or not.

```
# check empty or not
name = input("Enter your name: ")
if name: # true if string is not empty
    print(f"Your name is {name}")
else:
    print("You didn't type anything")
```

Output - (i) Enter your name:
Your name is Satyam
(ii) Enter your name:
You didn't type anything

Loops -
while loop -
Syntax - initilization
while condition:
block
increament / decrement



```
i=1
while i<=4:
    print(f"Hello world {i}")
    i+=1
```

Output - Hello world 1
Hello world 2
Hello world 3
Hello world 4

Sum of Numbers program using while loop -

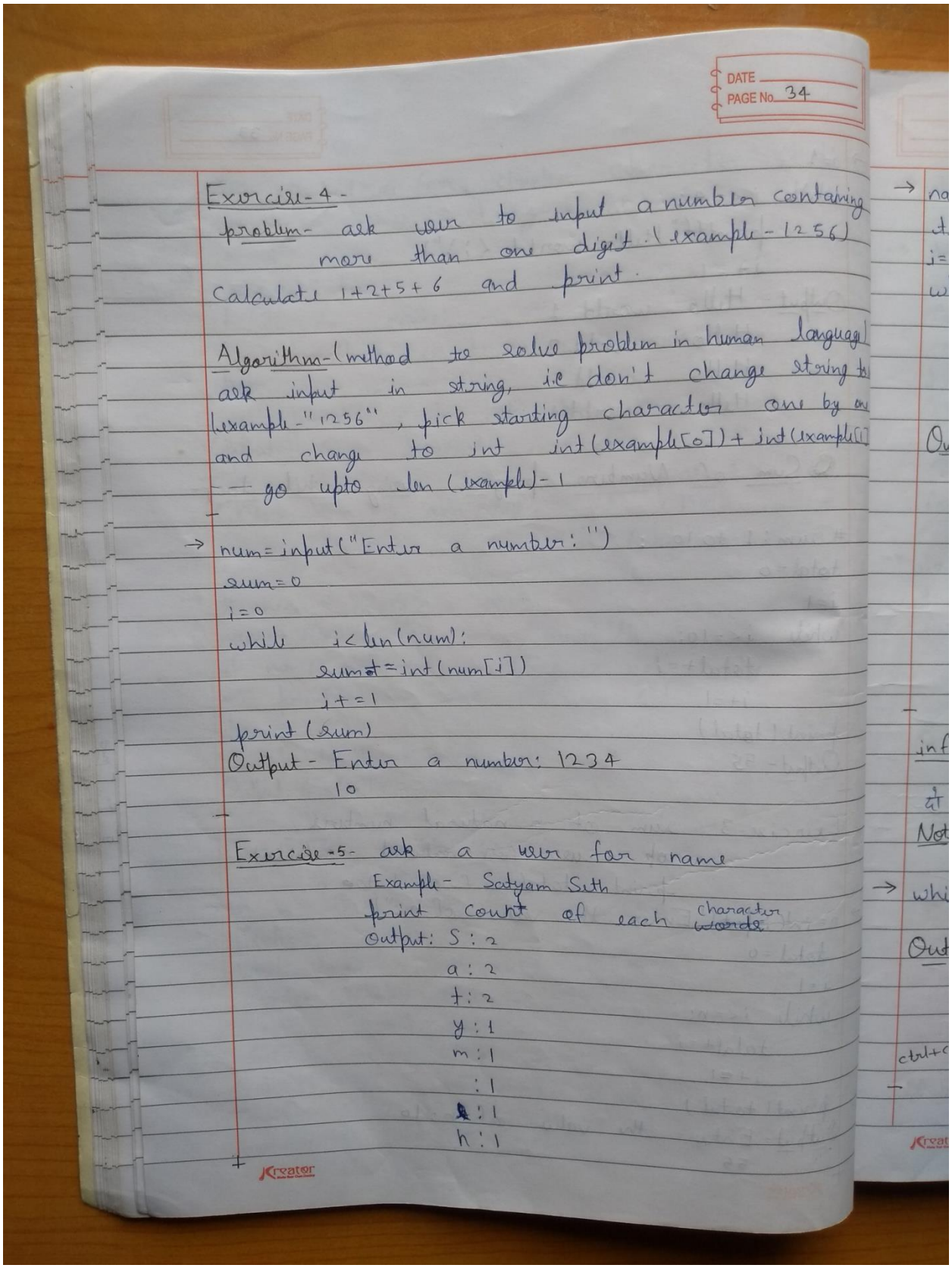
```
# sum: 1 to 10
total=0
i=1
while i<=10:
    total+=i
    i+=1
print(total)
```

Output - 55

Exercise-3- sum of n natural numbers
ask a user for natural number(n)
print total from 1 to n.

```
n=int(input("Enter the value of n:"))
total=0
i=1
while i<=n:
    total+=i
    i+=1
print(total)
```

Output - Enter the value of n: 10
55



DATE
PAGE No. 34

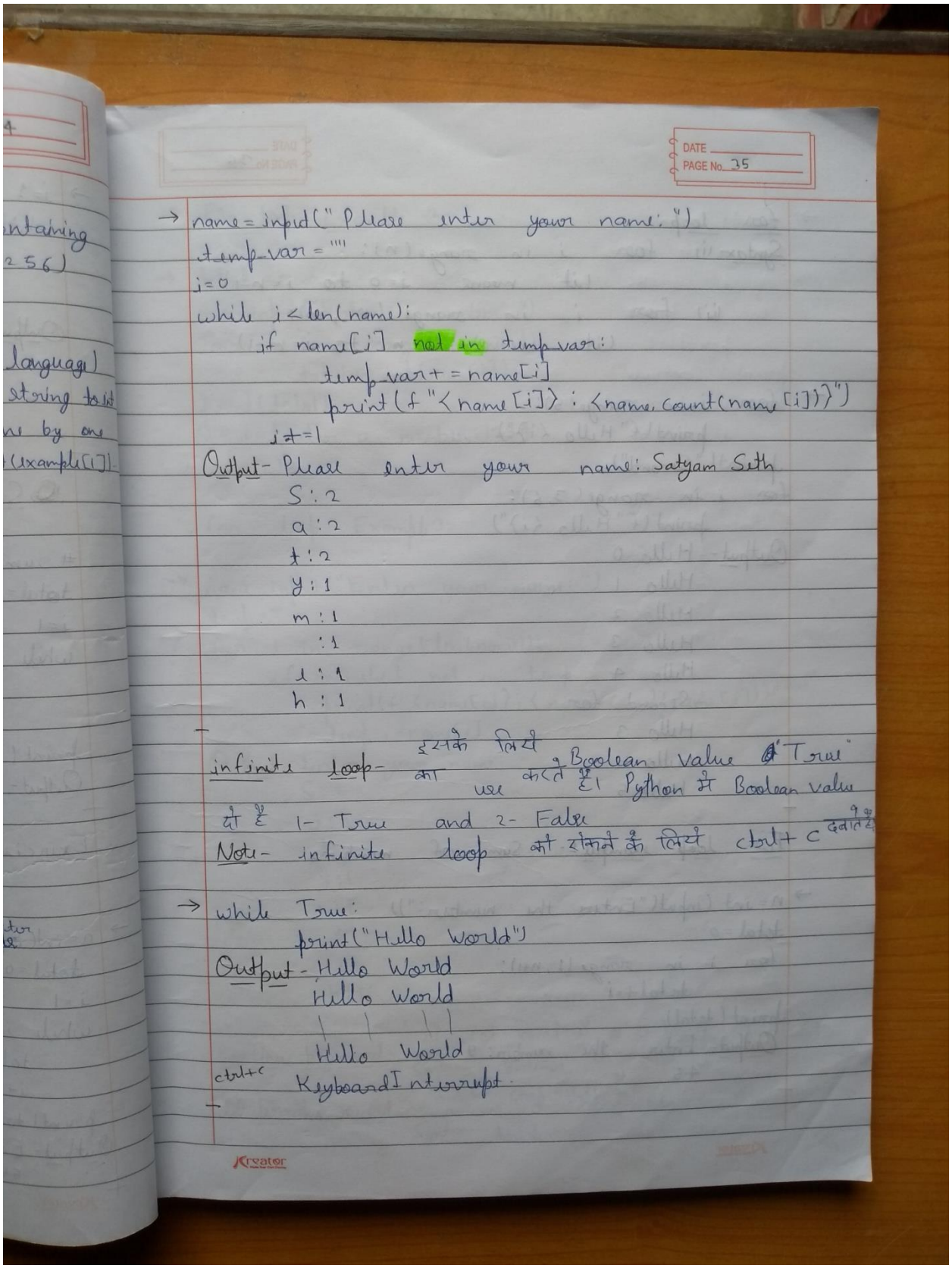
Exercise-4 -
problem- ask user to input a number containing more than one digit. (example - 1256)
Calculate 1+2+5+6 and print.

Algorithm- (method to solve problem in human language)
ask input in string, i.e don't change string to
(example - "1256", pick starting character one by one
and change to int int(example[0]) + int(example[1]
— go upto len(example) - 1

```
num = input("Enter a number: ")
sum = 0
i = 0
while i < len(num):
    sum += int(num[i])
    i += 1
print(sum)
```

Output - Enter a number: 1234
10

Exercise-5- ask a user for name
Example - Satyam Seth
print count of each character
Output: S : 2
a : 2
t : 2
y : 1
m : 1
: 1
[illegible] : 1
h : 1



```
name = input(" Pluase inter your name: ")
temp_var = ""
i=0
while i<len(name):
    if name[i] not in temp_var:
        temp_var += name[i]
        print(f"{name[i]} : {name.count(name[i])}")
    i+=1
```

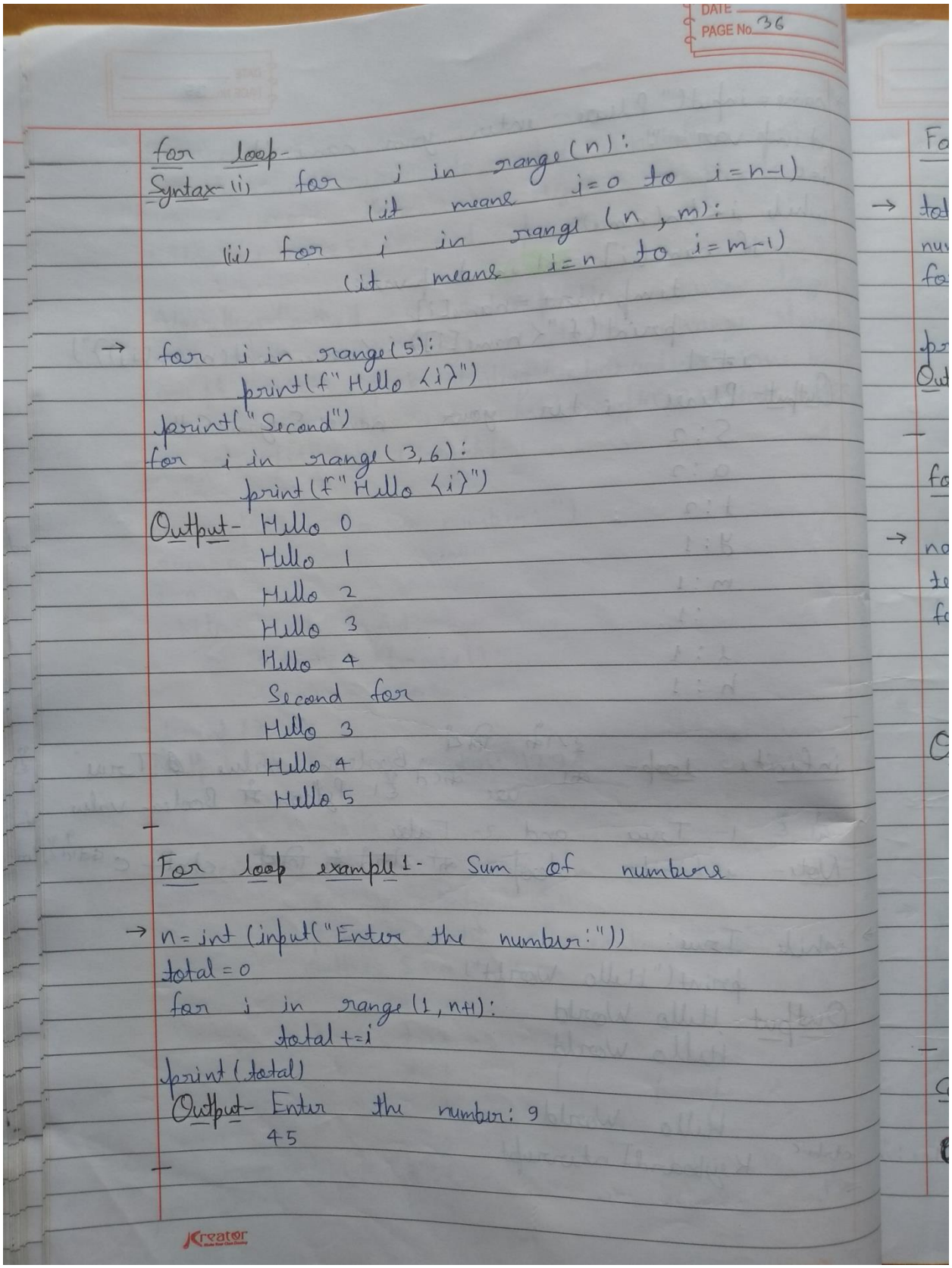
Output- Pluase enter your name: Satyam Seth
S : 2
a : 2
t : 2
y : 1
m : 1
: 1
e : 1
h : 1

infinite loop- इसके लिये Boolean value "True" का use करते हैं। Python में Boolean value दो हैं 1- True and 2- False

Note- infinite loop को रोकने के लिये ctrl + c दबाते हैं

```
while True:
    print("Hello World")
```

Output- Hello World
Hello World
| | | |
Hello World
ctrl+c KeyboardInterrupt



for loop-

Syntax- (i) for i in range(n):
(it means i=0 to i=n-1)

(ii) for i in rangi (n, m):
(it means i=n to i=m-1)

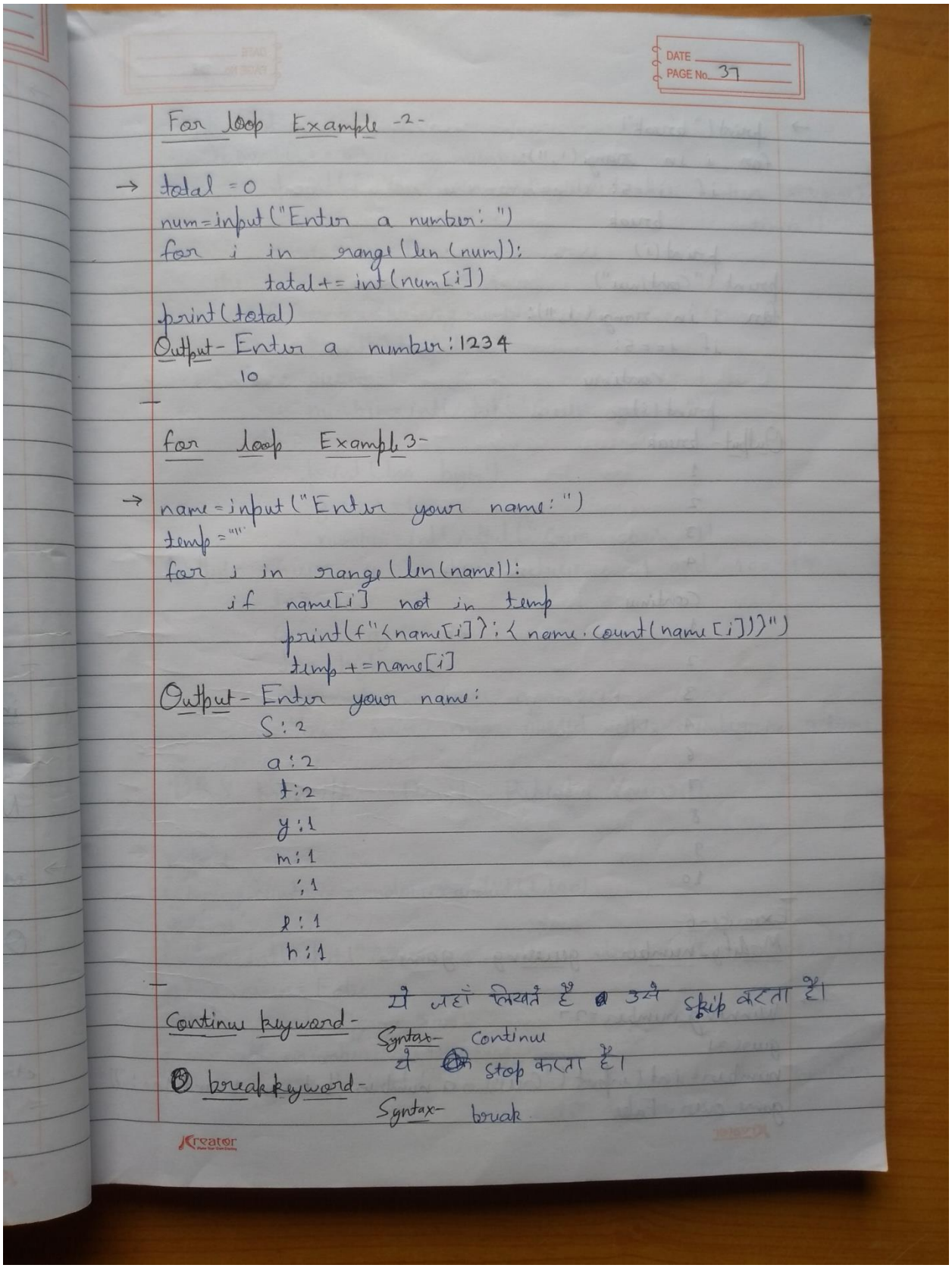
→ for i in range(5):
    print(f"Hello {i}")
print("Second")
for i in range(3,6):
    print(f"Hello {i}")

Output- Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
Second for
Hello 3
Hello 4
Hello 5

For loop example 1- Sum of numbers

→ n = int(input("Enter the number:"))
total = 0
for i in range(1, n+1):
    total += i
print(total)

Output- Enter the number: 9
45



DATE
PAGE No. 37

For loop Example -2-

→
```
total = 0
num = input("Enter a number: ")
for i in range(len(num)):
        total += int(num[i])
print(total)
```
Output- Enter a number: 1234
10

—

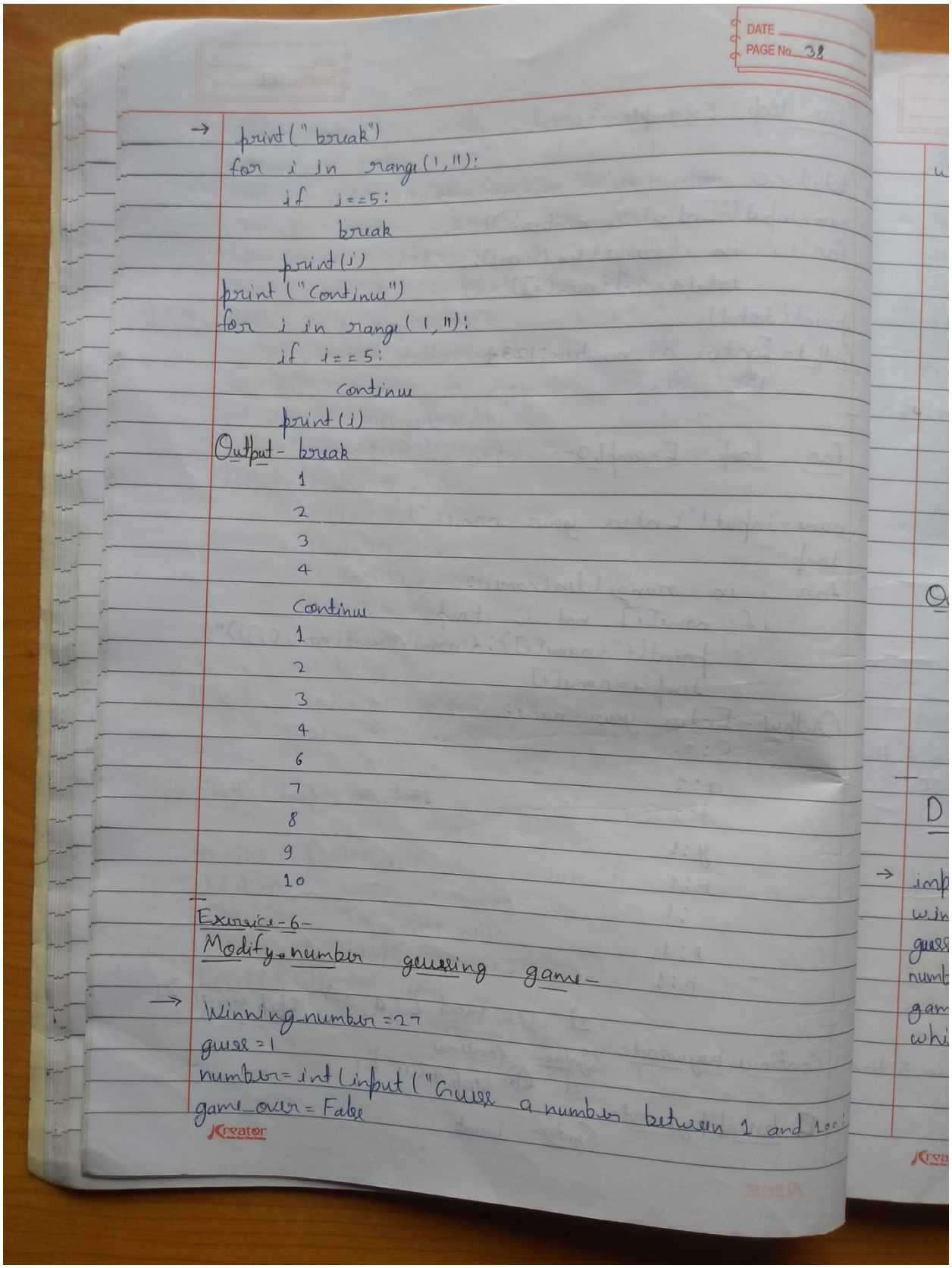
for loop Example 3-

→
```
name = input("Enter your name: ")
temp = ""
for i in range(len(name)):
    if name[i] not in temp
        print(f"{name[i]}: {name.count(name[i])}")
        temp += name[i]
```
Output- Enter your name:
S: 2
a: 2
t: 2
y: 1
m: 1
: 1
k: 1
h: 1

—

Continue keyword- यह जहाँ लिखते हैं वहाँ से skip करता है।
Syntax- continue

② break keyword- यह stop करता है।
Syntax- break



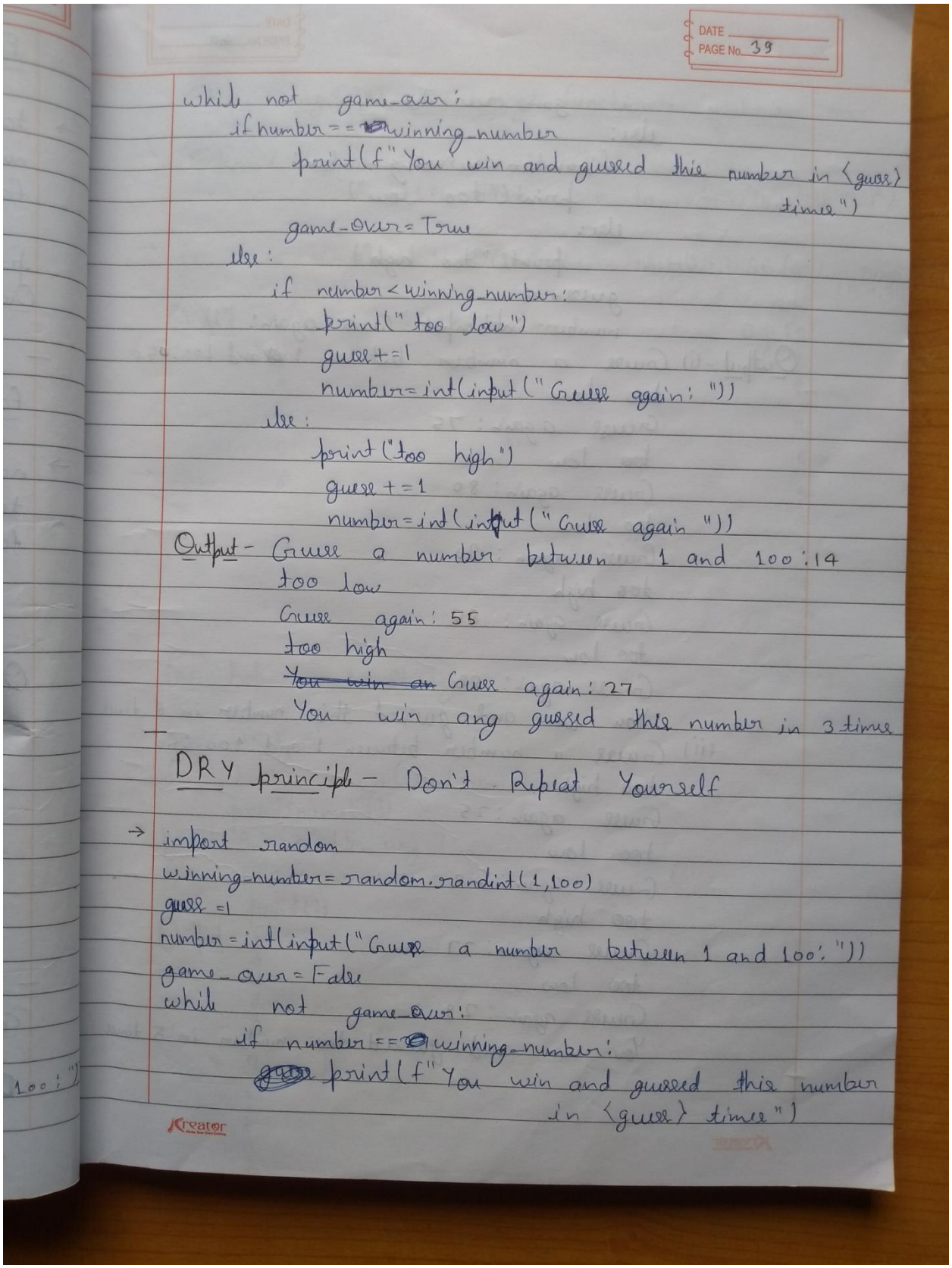
```
print("break")
for i in range(1,11):
    if i==5:
        break
    print(i)
print("continue")
for i in range(1,11):
    if i==5:
        continue
    print(i)
```

Output- break
1
2
3
4

Continue
1
2
3
4
6
7
8
9
10

Exercise-6-
Modify number guessing game-

```
winning_number=27
guess=1
number=int(input("Guess a number between 1 and 100:
game_over=False
```



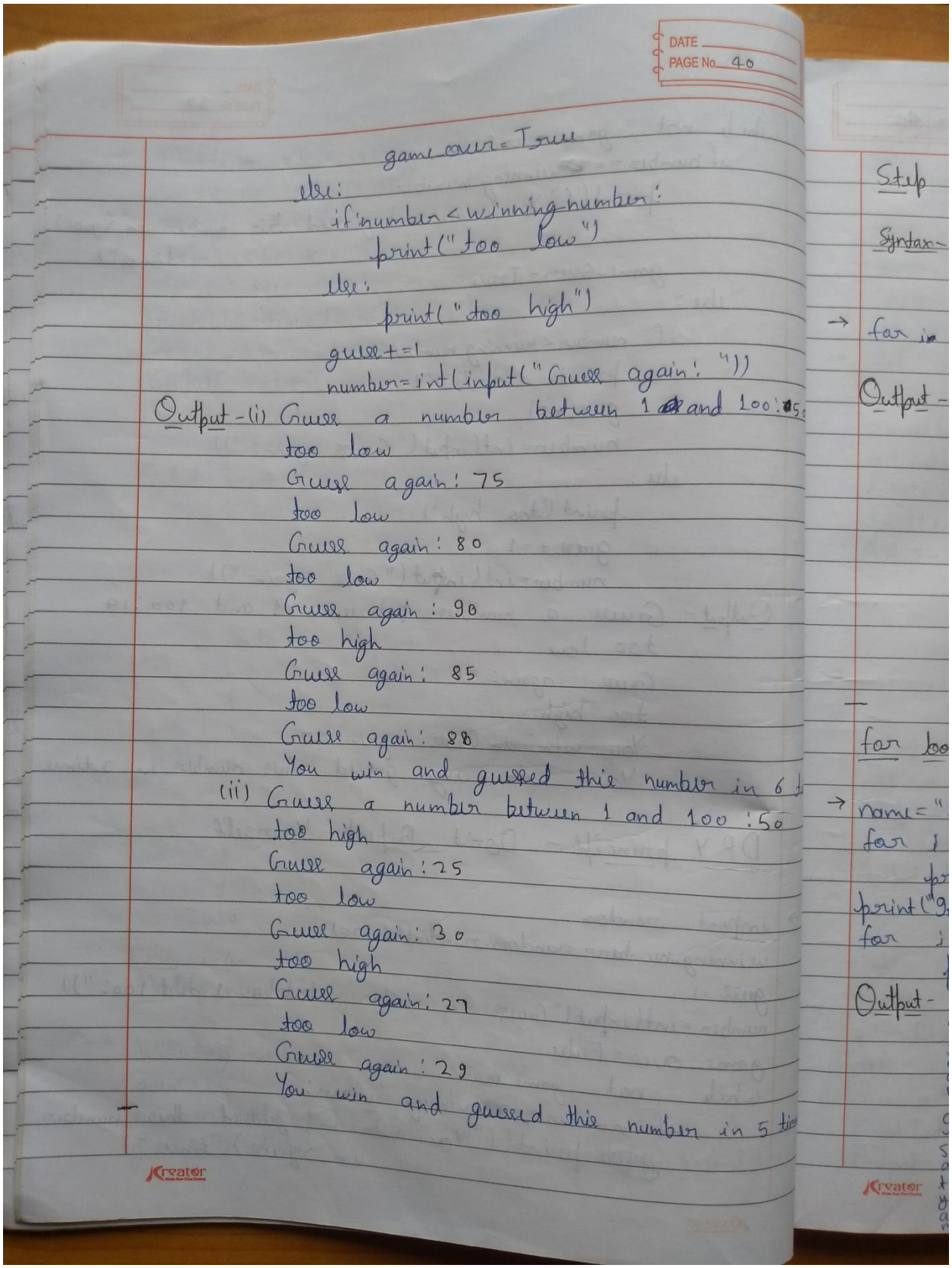
```
while not game_over:
    if number == winning_number
        print(f"You win and guessed this number in {guess} times")
        game_over = True
    else:
        if number < winning_number:
            print("too low")
            guess += 1
            number = int(input("Guess again: "))
        else:
            print("too high")
            guess += 1
            number = int(input("Guess again "))
```

Output - Guess a number between 1 and 100: 14
too low
Guess again: 55
too high
~~You win an~~ Guess again: 27
You win ang guessed this number in 3 times

DRY principle - Don't Repeat Yourself

→

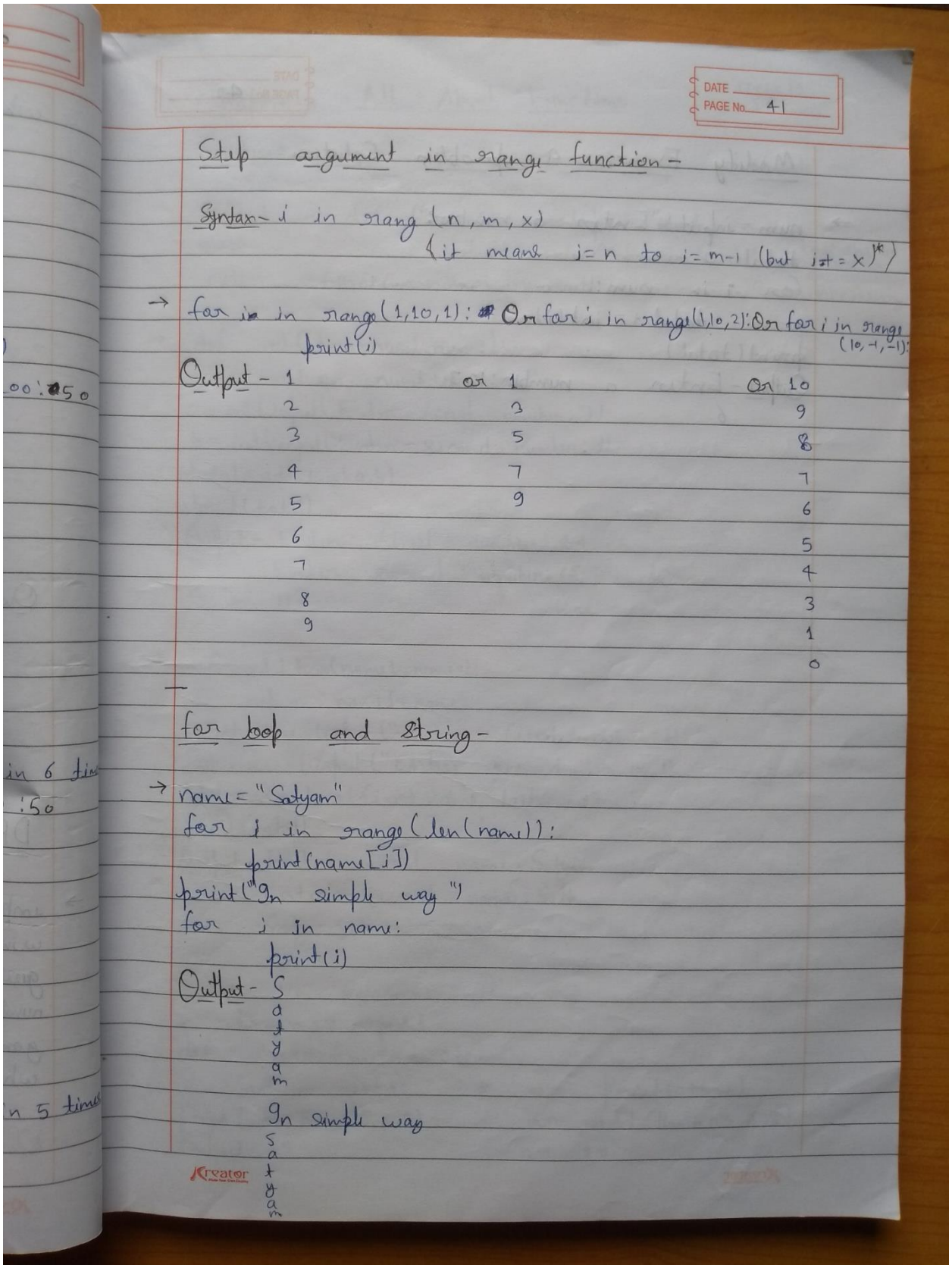
```
import random
winning_number = random.randint(1,100)
guess = 1
number = int(input("Guess a number between 1 and 100: "))
game_over = False
while not game_over:
    if number == winning_number:
        print(f"You win and guessed this number in {guess} times")
```



```
        game_over = True
else:
    if number < winning_number:
        print("too low")
    else:
        print("too high")
    guess += 1
    number = int(input("Guess again: "))
```

Output - (i) Guess a number between 1 and 100: 50
too low
Guess again: 75
too low
Guess again: 80
too low
Guess again: 90
too high
Guess again: 85
too low
Guess again: 88
You win and guessed this number in 6 t

(ii) Guess a number between 1 and 100: 50
too high
Guess again: 25
too low
Guess again: 30
too high
Guess again: 27
too low
Guess again: 29
You win and guessed this number in 5 tim



DATE
PAGE No. 41

Step argument in rangi function -

Syntax - i in rang (n, m, x)
(it means i = n to i = m-1 (but i+ = x)*)

→ for in in range(1,10,1): Or for i in range(1,10,2): Or for i in range (10,-1,-1):
print(i)

Output - 1 2 3 4 5 6 7 8 9

or 1 3 5 7 9

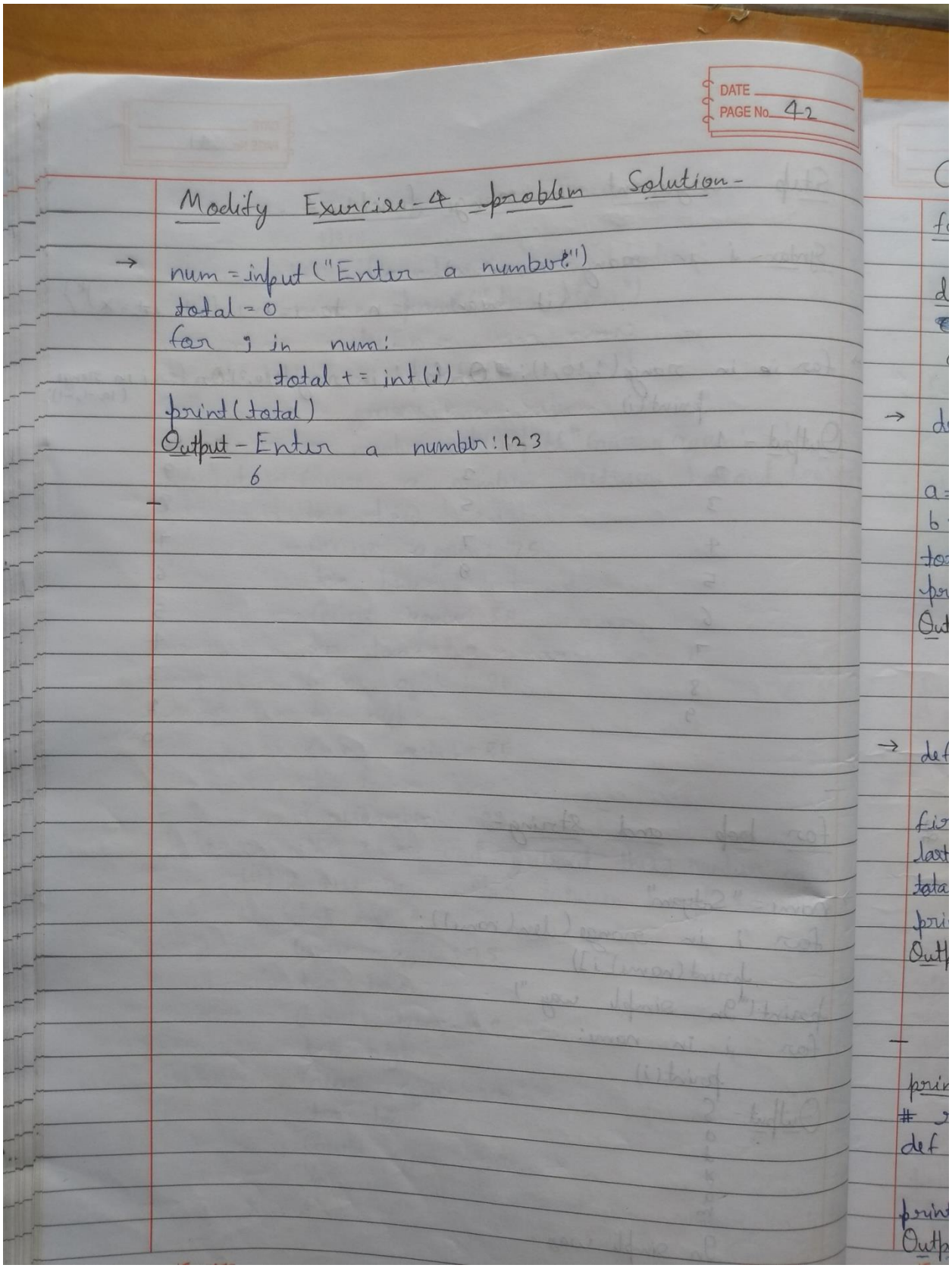
Or 10 9 8 7 6 5 4 3 1 0

—

for loop and String -

→
```
name = "Satyam"
for i in range(len(name)):
    print(name[i])
print("In simple way")
for i in name:
    print(i)
```

Output -
```
S
a
t
y
a
m

In simple way
S
a
t
y
a
m
```

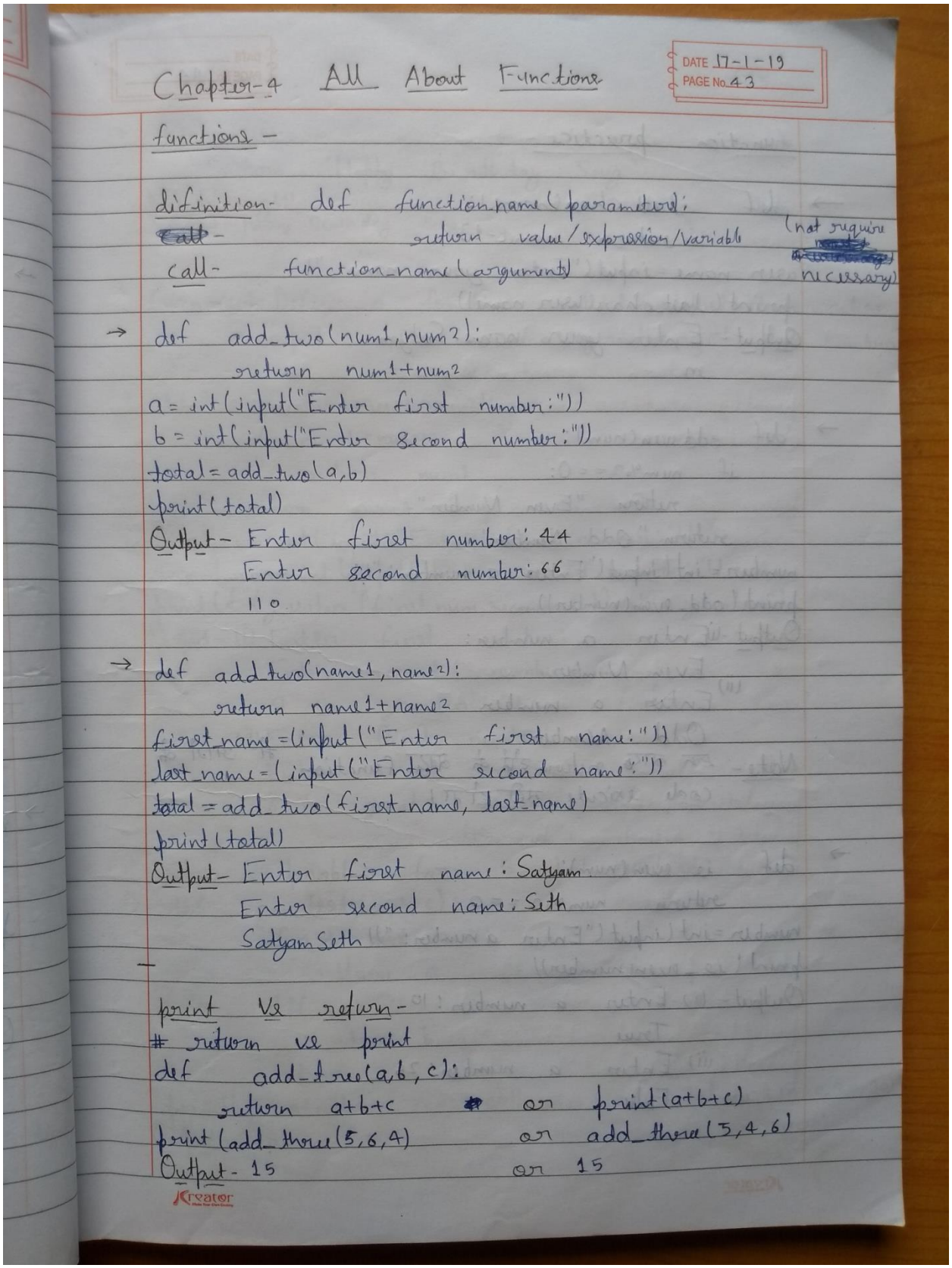


DATE
PAGE No. 42

Modify Exercise-4 problem Solution-

```
num = input("Enter a number:")
total = 0
for i in num:
    total += int(i)
print(total)
```

Output - Enter a number: 123
6



Chapter-4 All About Functions

DATE 17-1-19
PAGE No. 43

functions –

difinition- def function_name (parameters):
~~Call~~ - return value / expression / variable (not require necessary)

call- function_name (arguments)

→
```
def add_two(num1, num2):
    return num1+num2
a = int(input("Enter first number:"))
b = int(input("Enter second number:"))
total = add_two(a,b)
print(total)
```

Output-
```
Enter first number: 44
Enter second number: 66
110
```

→
```
def add_two(name1, name2):
    return name1+name2
first_name = (input("Enter first name:"))
last_name = (input("Enter second name:"))
total = add_two(first_name, last_name)
print(total)
```

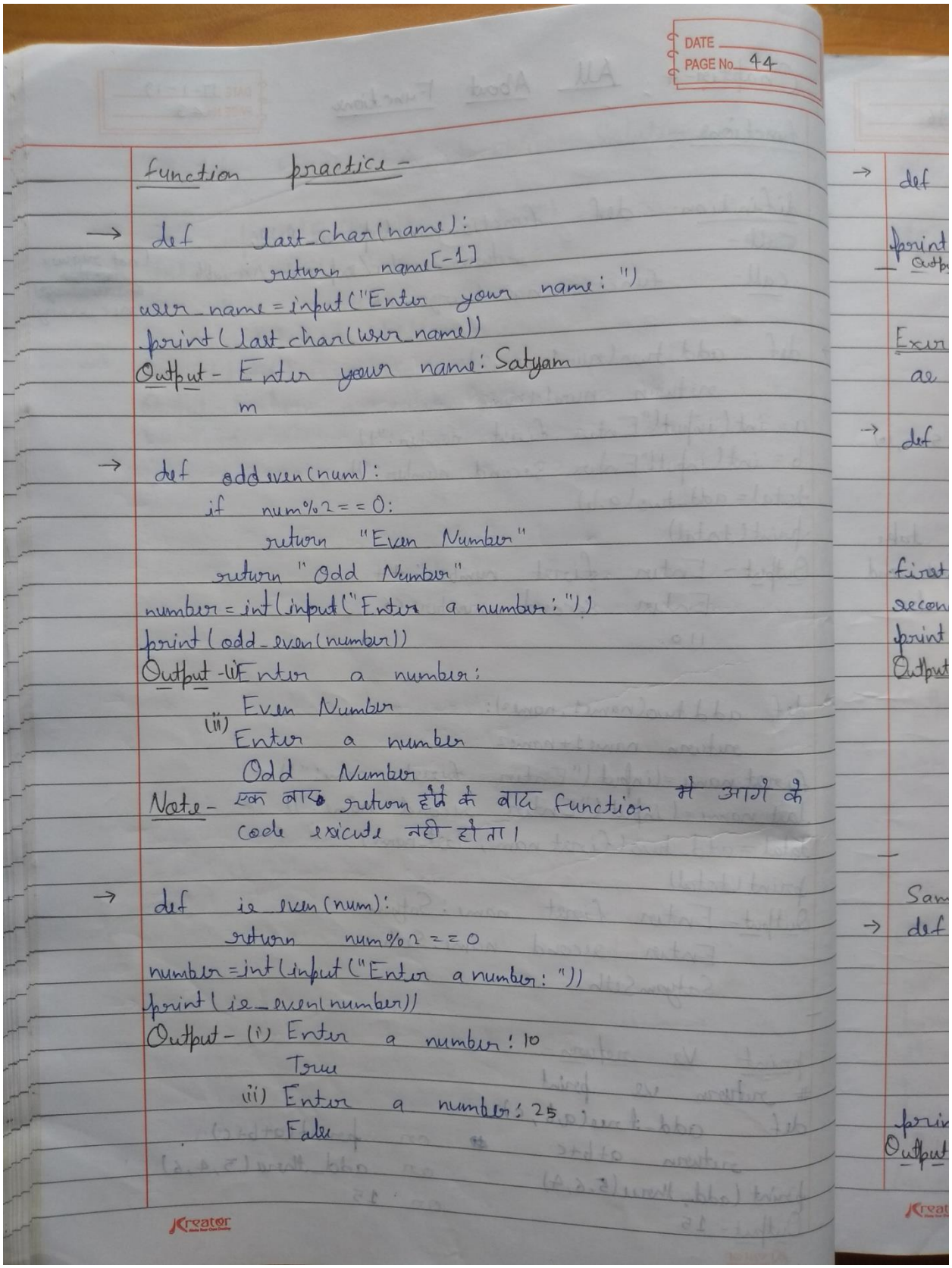
Output-
```
Enter first name: Satyam
Enter second name: Seth
SatyamSeth
```

—

print Vs return-

```
# return vs print
def add_three(a,b,c):
    return a+b+c          * or print(a+b+c)
print(add_three(5,6,4))   or add_three(5,4,6)
```

Output- 15 or 15



DATE
PAGE No. 44

Function practice -

```
def last_char(name):
    return name[-1]
user_name = input("Enter your name: ")
print(last_char(user_name))
```

Output - Enter your name: Satyam
m

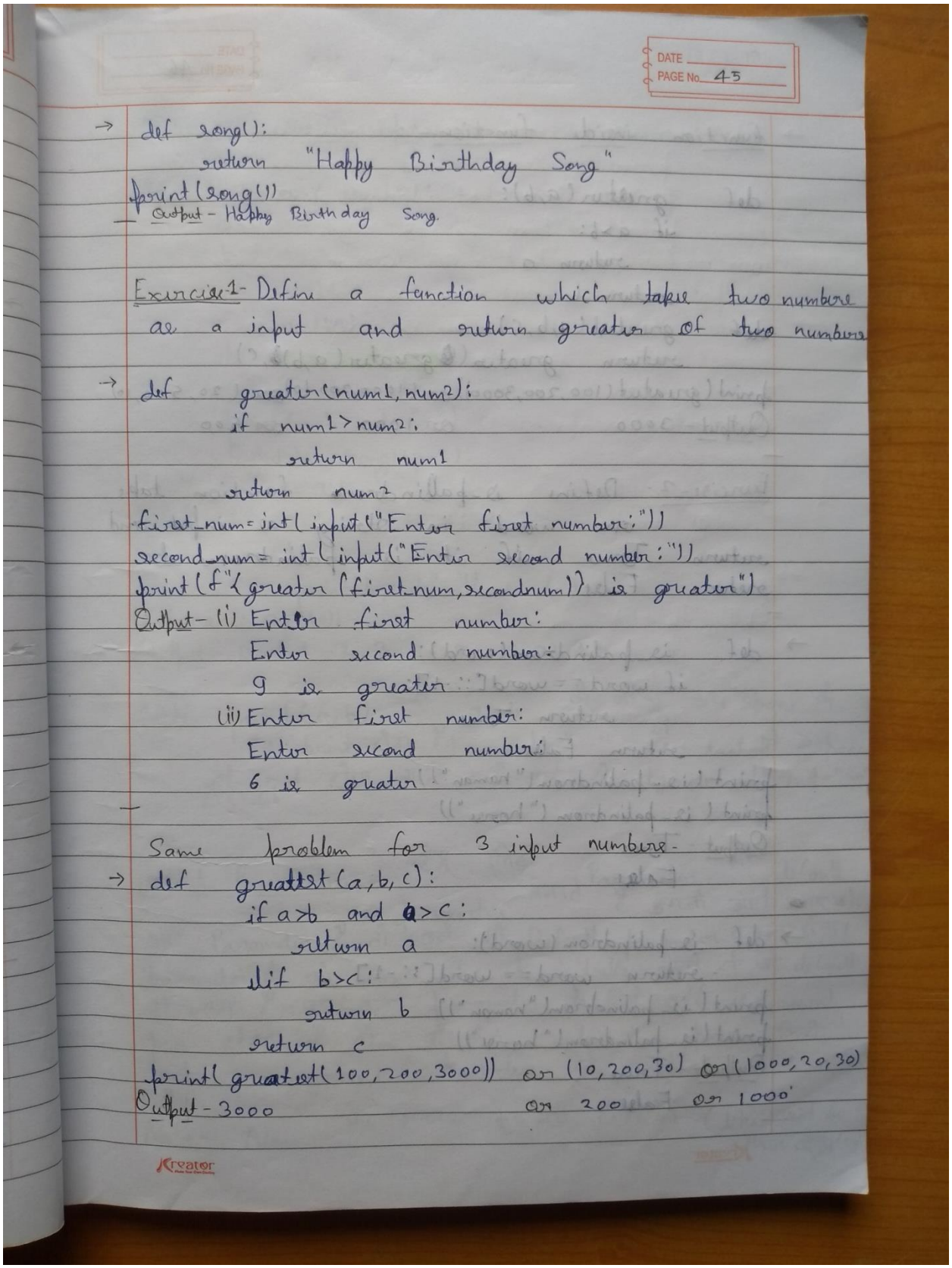
```
def odd_even(num):
    if num%2 == 0:
        return "Even Number"
    return "Odd Number"
number = int(input("Enter a number: "))
print(odd_even(number))
```

Output - (i) Enter a number:
Even Number
(ii) Enter a number
Odd Number

Note - एक बार return होने के बाद function में आगे के code execute नहीं होता।

```
def is_even(num):
    return num%2 == 0
number = int(input("Enter a number: "))
print(is_even(number))
```

Output - (i) Enter a number: 10
True
(ii) Enter a number: 25
False



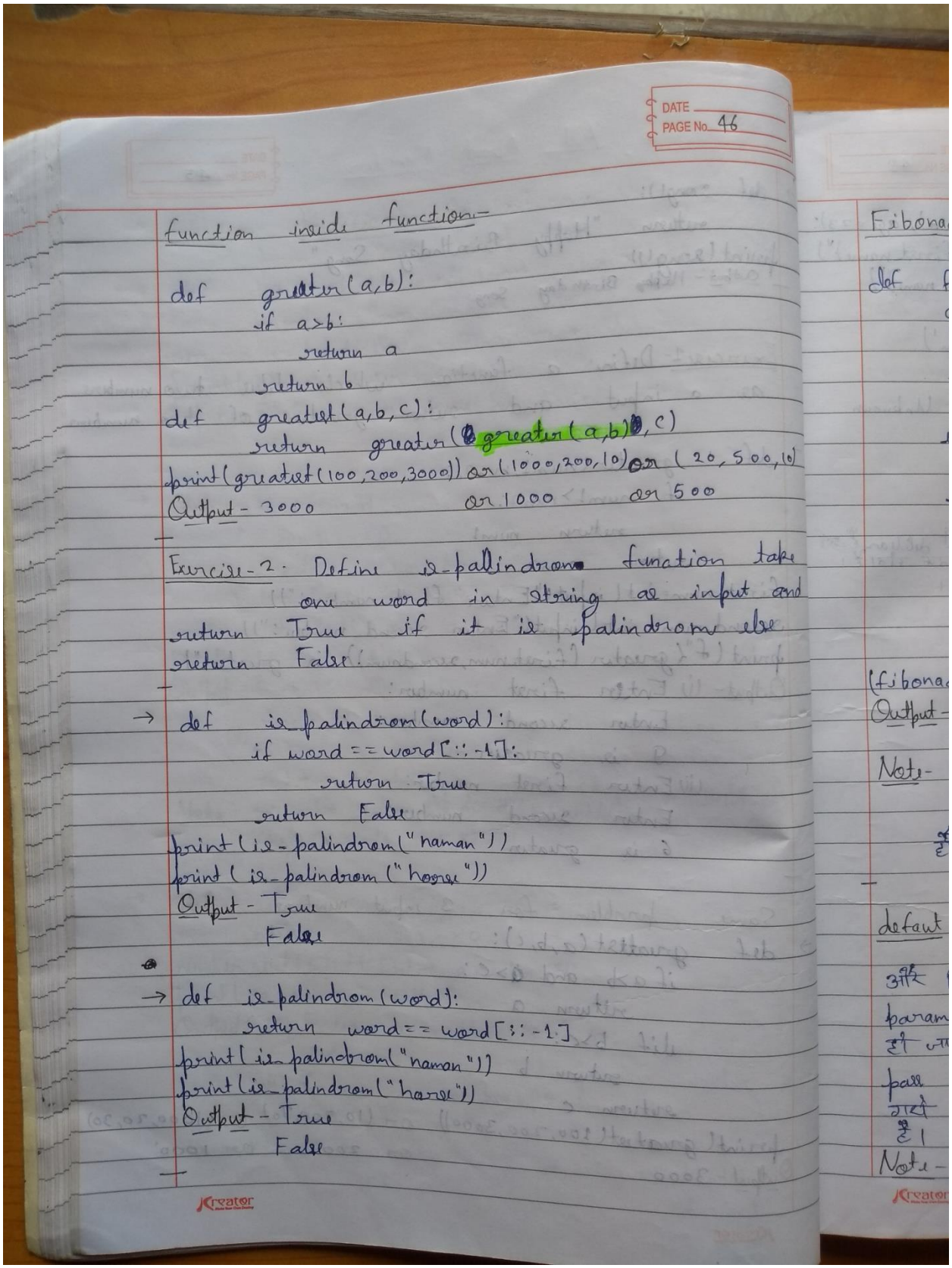
```
→ def song():
      return "Happy Birthday Song"
print(song())
```
Output - Happy Birthday Song.

Exercise 1- Define a function which take two numbers as a input and return greater of two numbers

```
→ def greater(num1, num2):
      if num1 > num2:
          return num1
      return num2
first_num = int(input("Enter first number:"))
second_num = int(input("Enter second number:"))
print(f"{greater(first_num, secondnum)} is greater")
```
Output - (i) Enter first number:
Enter second number:
9 is greater
(ii) Enter first number:
Enter second number:
6 is greater

Same problem for 3 input numbers-

```
→ def greatest(a, b, c):
      if a > b and a > c:
          return a
      elif b > c:
          return b
      return c
print(greatest(100, 200, 3000)) or (10, 200, 30) or (1000, 20, 30)
```
Output - 3000 or 200 or 1000



DATE
PAGE No. 46

Function inside function:-

```
def greater(a,b):
    if a>b:
        return a
    return b
def greatest(a,b,c):
    return greater(greater(a,b),c)
print(greatest(100,200,3000)) or (1000,200,10) or (20,500,10)
```

Output - 3000 or 1000 or 500

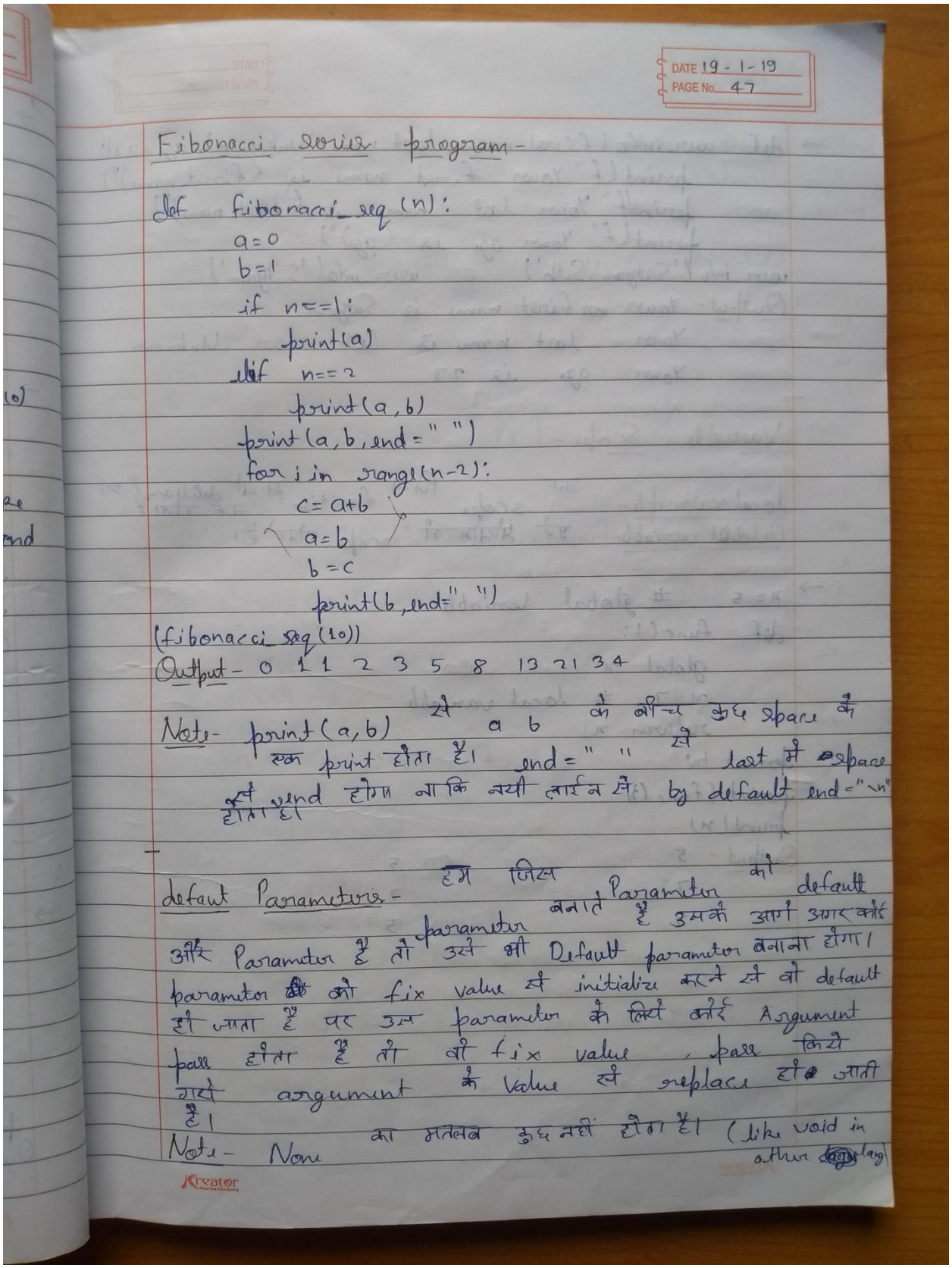
Exercise-2. Define is_palindrom function take one word in string as input and return True if it is palindrom else return False.

```
def is_palindrom(word):
    if word == word[::-1]:
        return True
    return False
print(is_palindrom("naman"))
print(is_palindrom("house"))
```

Output - True
False

```
def is_palindrom(word):
    return word == word[::-1]
print(is_palindrom("naman"))
print(is_palindrom("house"))
```

Output - True
False



DATE 19-1-19
PAGE No. 47

Fibonacci series program -

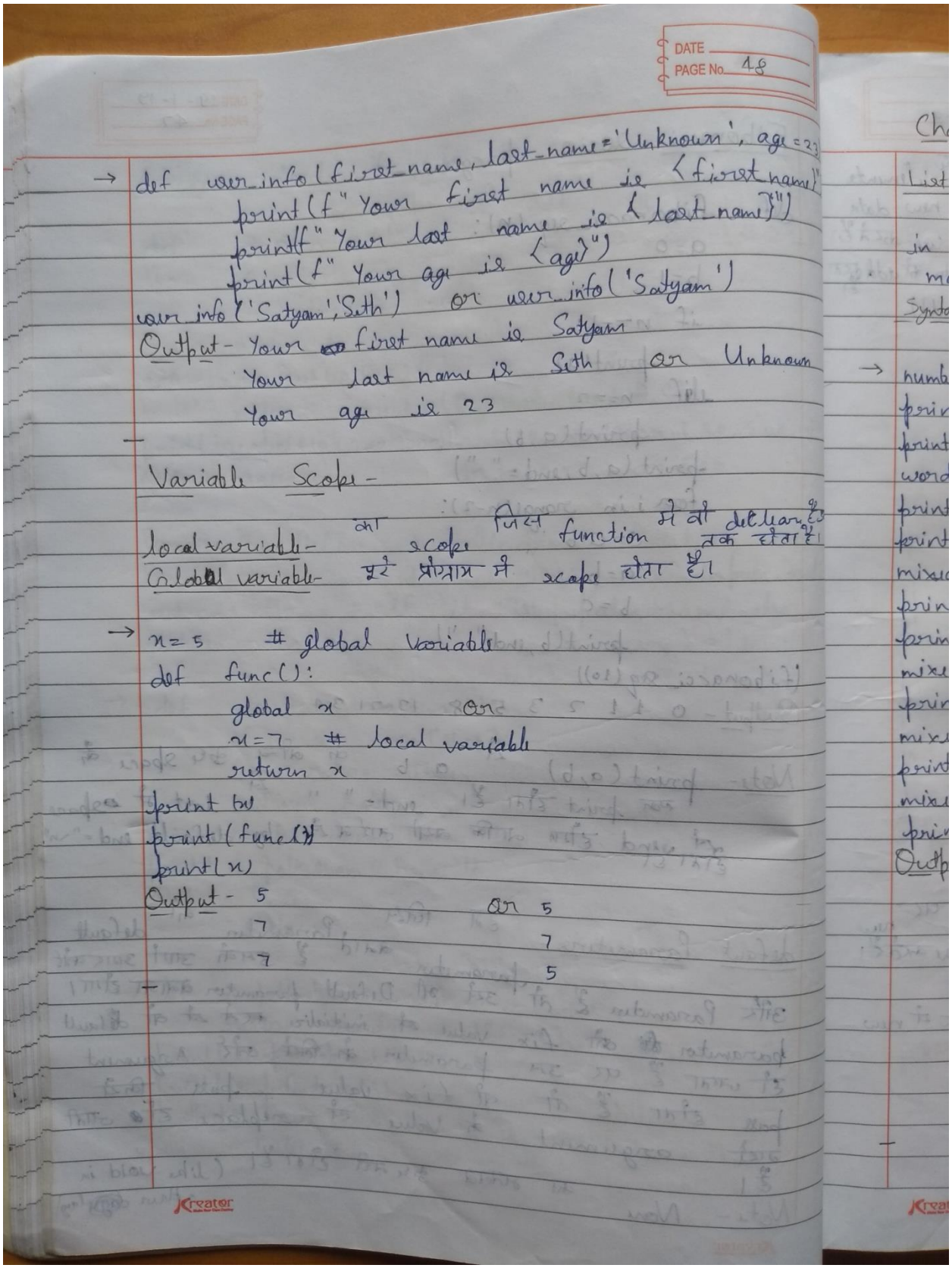
```
def fibonacci_seq(n):
    a=0
    b=1
    if n==1:
        print(a)
    elif n==2
        print(a,b)
    print(a,b,end=" ")
    for i in range(n-2):
        c=a+b
        a=b
        b=c
        print(b,end=" ")
fibonacci_seq(10)
```

Output - 0 1 1 2 3 5 8 13 21 34

Note - print(a,b) से a b के बीच कुछ space के एक print होता है। end=" " से last में space की जगह end होगा ना कि नयी लाईन से by default end="\n" होता है।

default Parameters - हम जिस Paramter को default बनाते है उसके आगे अगर कोई और Paramter है तो उसे भी Default paramter बनाना होगा। paramter को fix value से initialize करने से वो default हो जाता है पर उस paramter के लिये कोई Argument pass होता है तो वो fix value, pass किये गये argument के value से replace हो जाती है।

Note - None का मतलब कुछ नहीं होता है। (like void in other language)



```
def user_info(first_name, last_name='Unknown', age=23):
    print(f"Your first name is {first_name}")
    printf"Your last name is {last_name}")
    print(f"Your age is {age}")
user_info('Satyam','Seth')  or user_info('Satyam')
```

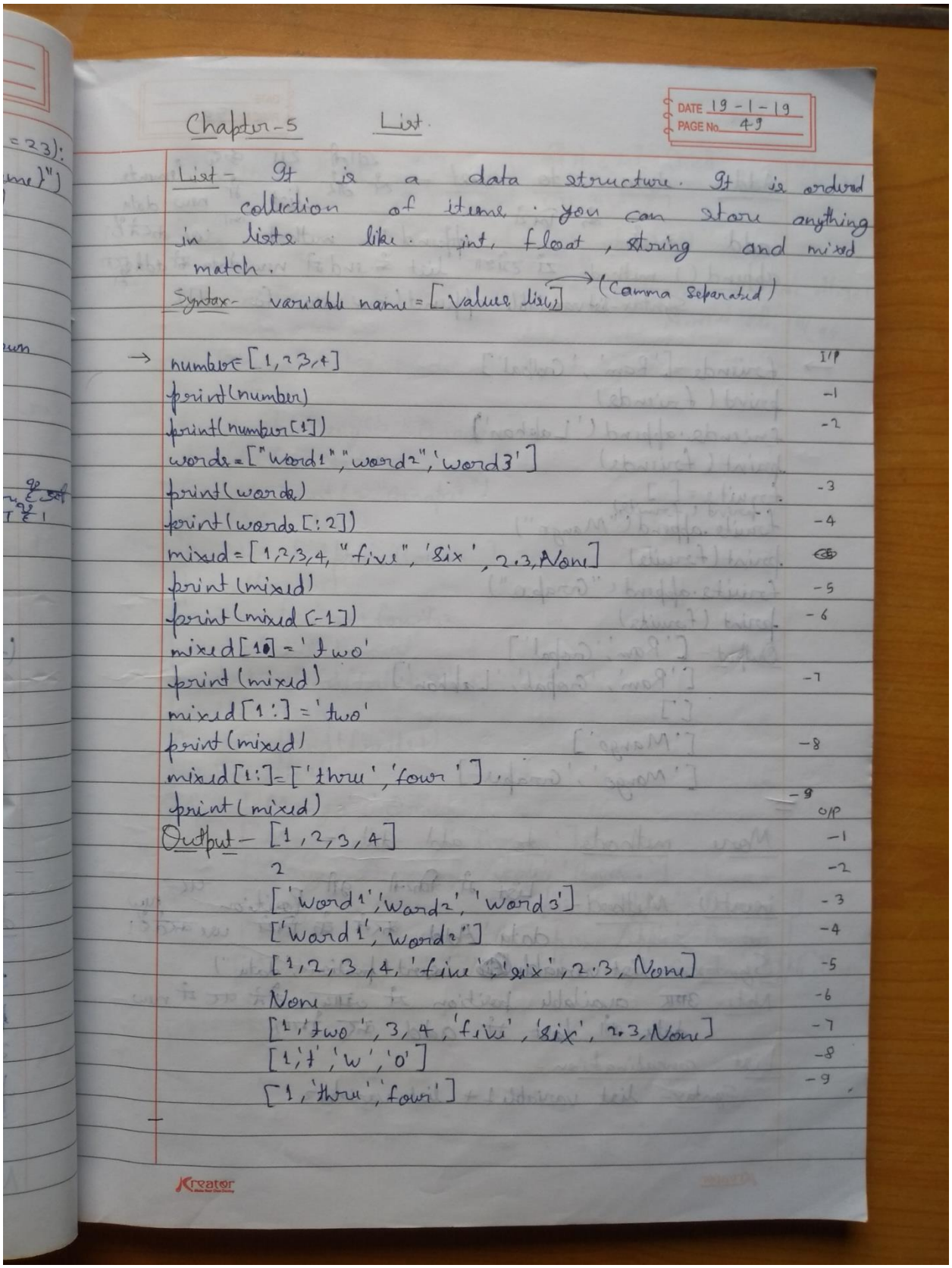
Output - Your first name is Satyam
Your last name is Seth or Unknown
Your age is 23

Variable Scope -

local variable - का scope जिस function में वो declare हुए तक होता है।
Global variable - पूरे प्रोग्राम में scope होता है।

```
x = 5    # global variable
def func():
    global x    or
    x = 7    # local variable
    return x
print(x)
print(func())
print(x)
```

Output - 5 or 5
7 7
7 5



Chapter-5 List

DATE 19-1-19
PAGE No. 49

List - It is a data structure. It is ordered collection of items. You can store anything in lists like int, float, string and mixed match.

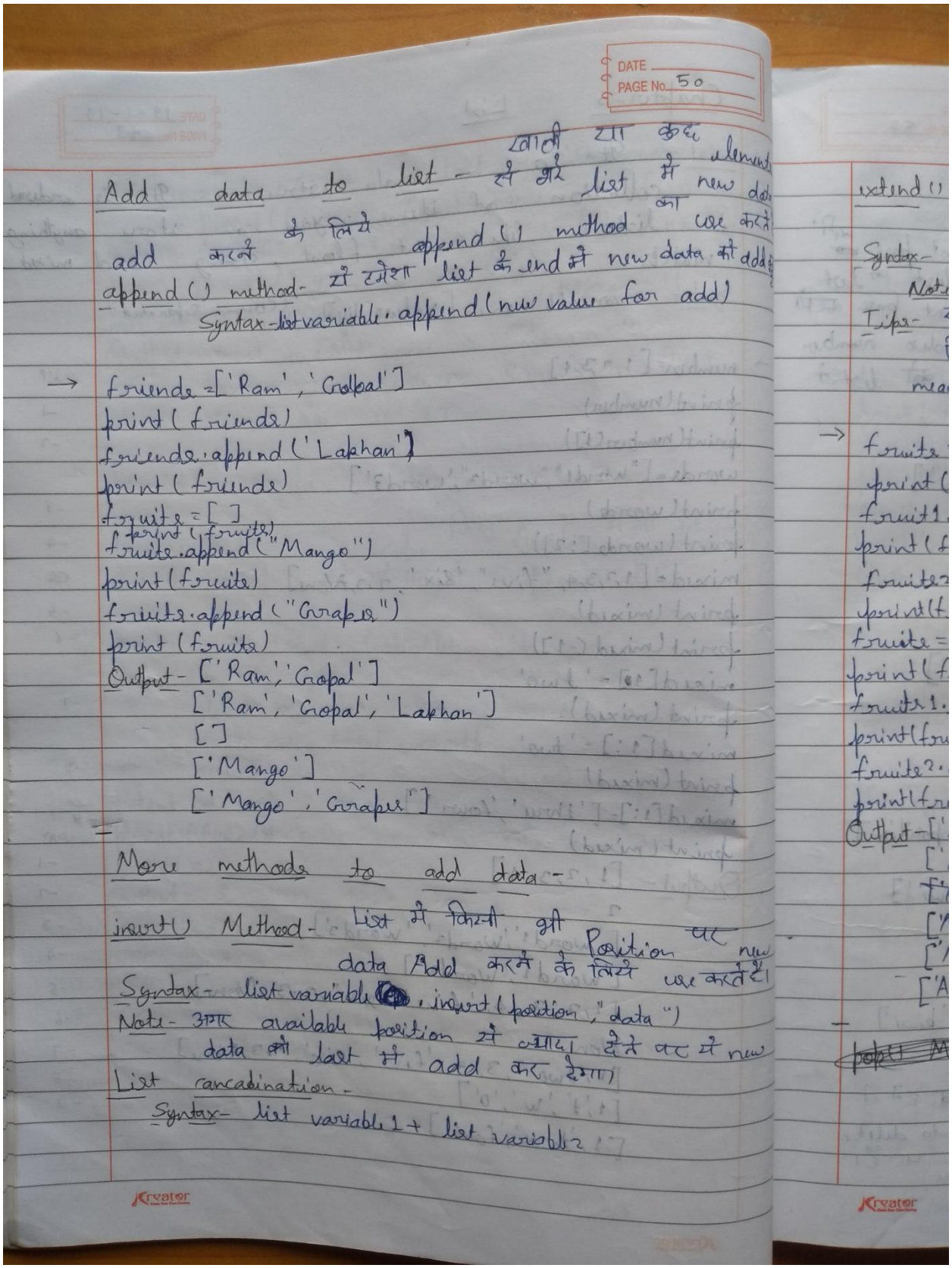
Syntax - variable name = [values list] → (Comma Separated)

```
→ number=[1,2,3,4]                                  I/P
  print(number)                                      -1
  print(number[1])                                   -2
  words=["word1","word2",'word3']
  print(words)                                       -3
  print(words[:2])                                   -4
  mixed=[1,2,3,4,"five",'six',2.3,None]
  print(mixed)                                       -5
  print(mixed[-1])                                   -6
  mixed[1]='two'
  print(mixed)                                       -7
  mixed[1:]='two'
  print(mixed)                                       -8
  mixed[1:]=['thru','four']
  print(mixed)                                       -9
```

Output - O/P

```
[1,2,3,4]                                            -1
2                                                    -2
['word1','word2','word3']                            -3
['word1','word2']                                    -4
[1,2,3,4,'five','six',2.3,None]                      -5
None                                                 -6
[1,'two',3,4,'five','six',2.3,None]                  -7
[1,'t','w','o']                                      -8
[1,'thru','four']                                    -9
```

Kreator



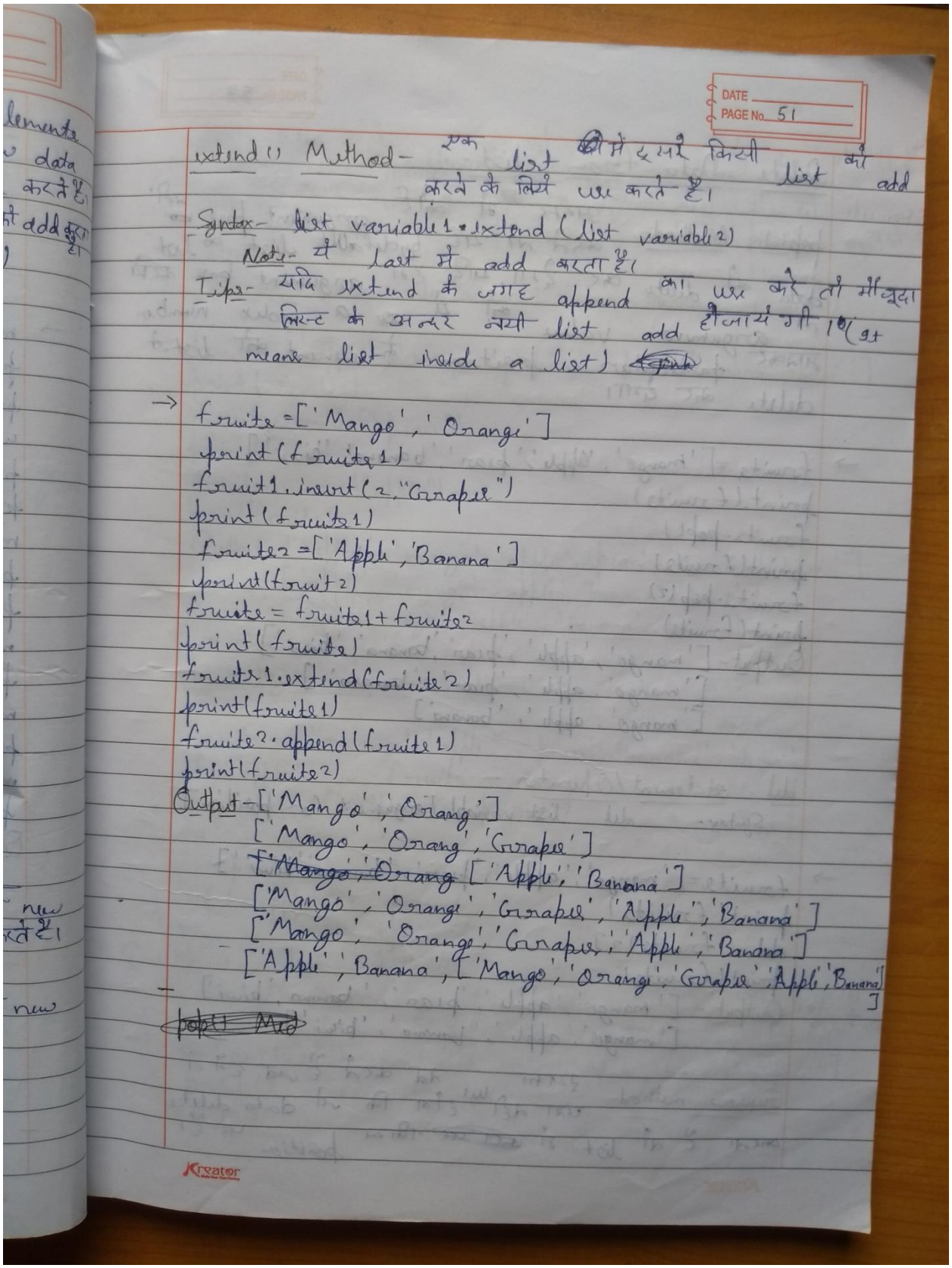
Add data to list - खाली या कुछ elements वाली list में new data add करने के लिये append() method का use करते हैं।
append() method - यह हमेशा list के end में new data को add करता है।
Syntax - list variable.append(new value for add)

→
```
friends=['Ram','Gopal']
print(friends)
friends.append('Lakhan')
print(friends)
fruits=[]
print(fruits)
fruits.append("Mango")
print(fruits)
fruits.append("Grapes")
print(fruits)
```
Output - ['Ram','Gopal']
['Ram','Gopal','Lakhan']
[]
['Mango']
['Mango','Grapes']

More methods to add data -

insert() Method - List में किसी भी Position पर new data Add करने के लिये use करते हैं।
Syntax - list variable.insert(position,"data")
Note - अगर available position से ज्यादा देते तब यह new data को last में add कर देगा।

List concatination -
Syntax - list variable 1 + list variable 2



extend() Method- एक list में दूसरे किसी list को add करने के लिये use करते हैं।

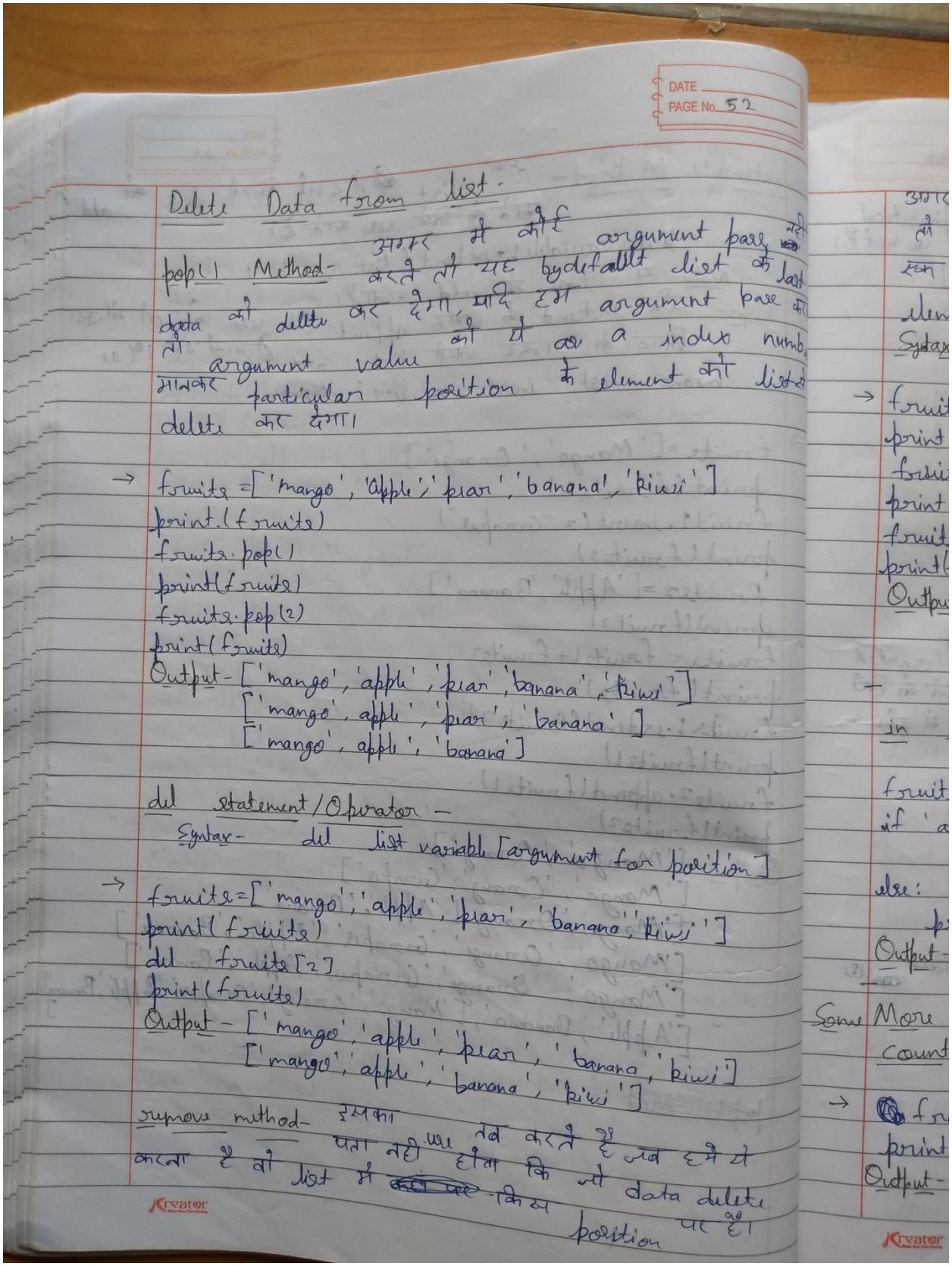
Syntax- list variable1.extend(list variable2)

Note- ये last में add करता है।

Tips- यदि extend के जगह append का use करें तो मौजूदा लिस्ट के अन्दर नयी list add हो जायेगी।(list means list inside a list)

```
fruits=['Mango','Orangi']
print(fruits1)
fruit1.insert(2,"Grapes")
print(fruits1)
fruits2=['Apple','Banana']
print(fruit2)
fruits=fruits1+fruits2
print(fruits)
fruits1.extend(fruits2)
print(fruits1)
fruits2.append(fruits1)
print(fruits2)
```

Output- ['Mango','Orang']
['Mango','Orang','Grapes']
~~['Mango','Orang']~~ ['Apple','Banana']
['Mango','Orangi','Grapes','Apple','Banana']
['Mango','Orangi','Grapes','Apple','Banana']
['Apple','Banana',['Mango','Orangi','Grapes','Apple','Banana']]



DATE
PAGE No. 52

Delete Data from list.

pop() Method- अगर हम कोई argument pass नहीं करते तो यह by default list के last data को delete कर देगा, यदि हम argument pass करते तो argument value को यह as a index number मानकर particular position के element को list से delete कर देगा।

```
→ fruits = ['mango', 'apple', 'pear', 'banana', 'kiwi']
print(fruits)
fruits.pop()
print(fruits)
fruits.pop(2)
print(fruits)
```

Output - ['mango', 'apple', 'pear', 'banana', 'kiwi']
['mango', 'apple', 'pear', 'banana']
['mango', 'apple', 'banana']

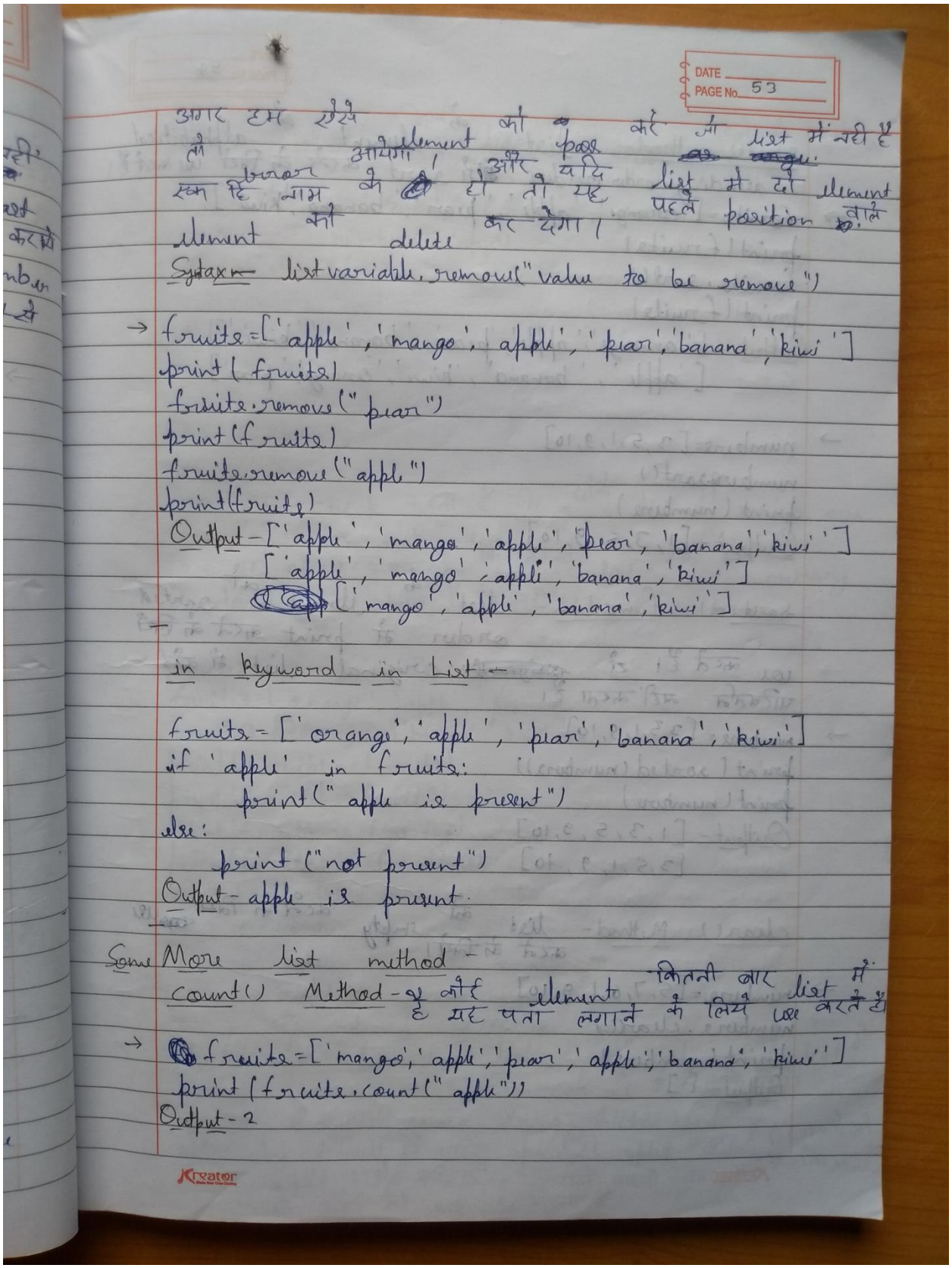
del statement / Operator -

Syntax - del list variable [argument for position]

```
→ fruits=['mango', 'apple', 'pear', 'banana', 'kiwi']
print(fruits)
del fruits[2]
print(fruits)
```

Output - ['mango', 'apple', 'pear', 'banana', 'kiwi']
['mango', 'apple', 'banana', 'kiwi']

remove method- इसका तब करते है जब हमे ये पता नहीं होता कि जो data delete करना है वो list में किस position पर है।



अगर हम ऐसे element को pass करें जो list में नहीं है तो error आयेगा। और यदि list में दो element एक ही नाम के हो तो यह पहले position वाले element को delete कर देगा।

Syntax – list variable.remove("value to be remove")

→
```
fruits=['apple','mango','apple','piar','banana','kiwi']
print(fruits)
fruits.remove("piar")
print(fruits)
fruits.remove("apple")
print(fruits)
```
Output – ['apple','mango','apple','piar','banana','kiwi']
['apple','mango','apple','banana','kiwi']
['mango','apple','banana','kiwi']

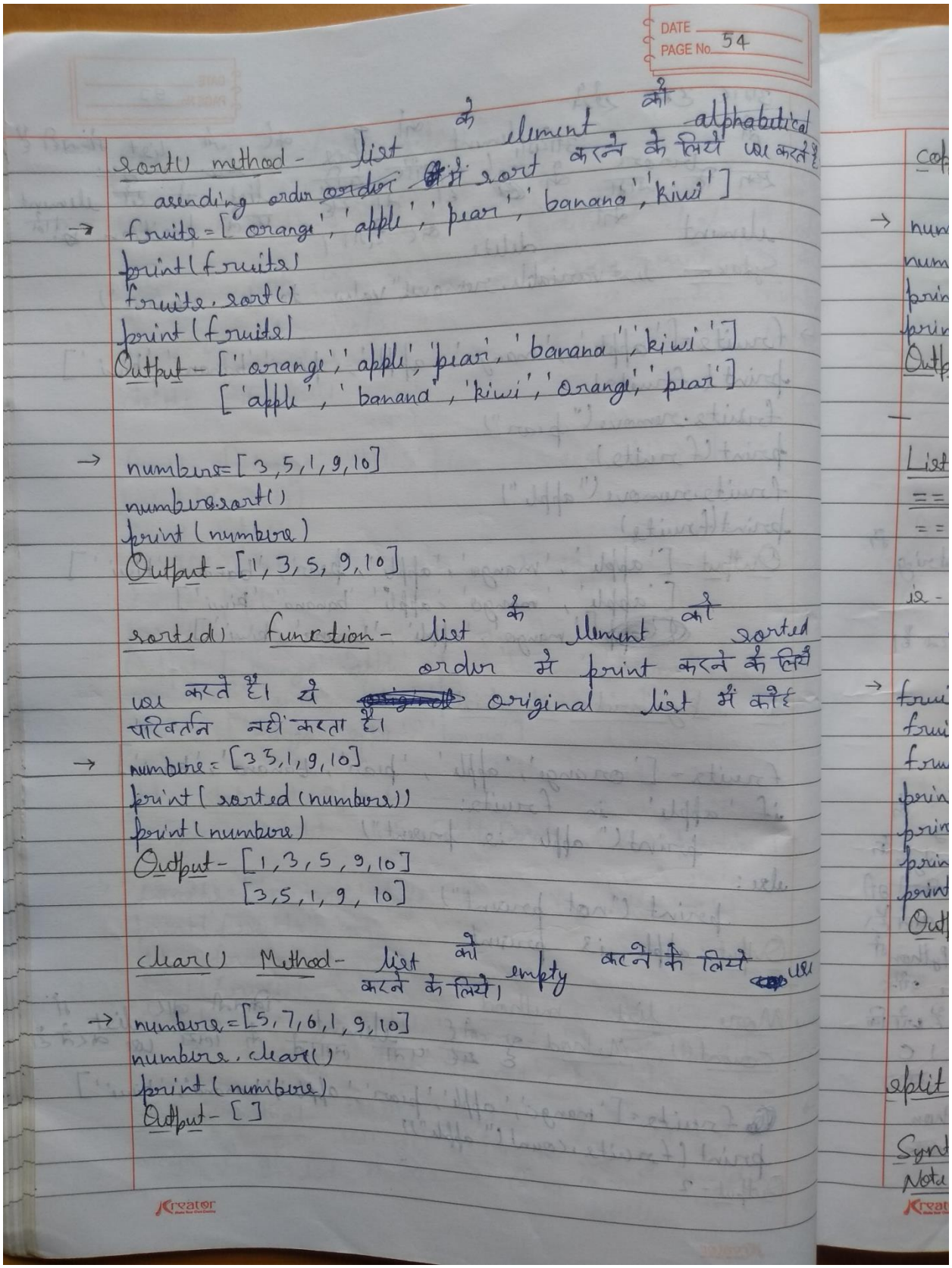
in Keyword in List –

```
fruits=['orange','apple','piar','banana','kiwi']
if 'apple' in fruits:
    print("apple is present")
else:
    print("not present")
```
Output – apple is present

Some More list method –

count() Method – यह कौन से element कितनी बार list में है यह पता लगाने के लिये use करते हैं

→
```
fruits=['mango','apple','piar','apple','banana','kiwi']
print(fruits.count("apple"))
```
Output – 2



sort() method - list के element को ascending order में sort करने के लिये use करते है

→ fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi']
print(fruits)
fruits.sort()
print(fruits)
Output - ['orange', 'apple', 'pear', 'banana', 'kiwi']
['apple', 'banana', 'kiwi', 'orange', 'pear']

→ numbers = [3, 5, 1, 9, 10]
numbers.sort()
print(numbers)
Output - [1, 3, 5, 9, 10]

sorted() function - list के element को sorted order में print करने के लिये use करते है। ये original list में कोई परिवर्तन नहीं करता है।

→ numbers = [3, 5, 1, 9, 10]
print(sorted(numbers))
print(numbers)
Output - [1, 3, 5, 9, 10]
[3, 5, 1, 9, 10]

clear() Method - list को empty करने के लिये use करने के लिये।

→ numbers = [5, 7, 6, 1, 9, 10]
numbers.clear()
print(numbers)
Output - []

DATE
PAGE No. 55

copy() Method - किसी list को copy करने के लिये use करते हैं।

→
```
numbers = [1,2,3,4,5,6]
numbers_copy = numbers.copy()
print(numbers)
print(numbers_copy)
```
Output - [1,2,3,4,5,6]
[1,2,3,4,5,6]

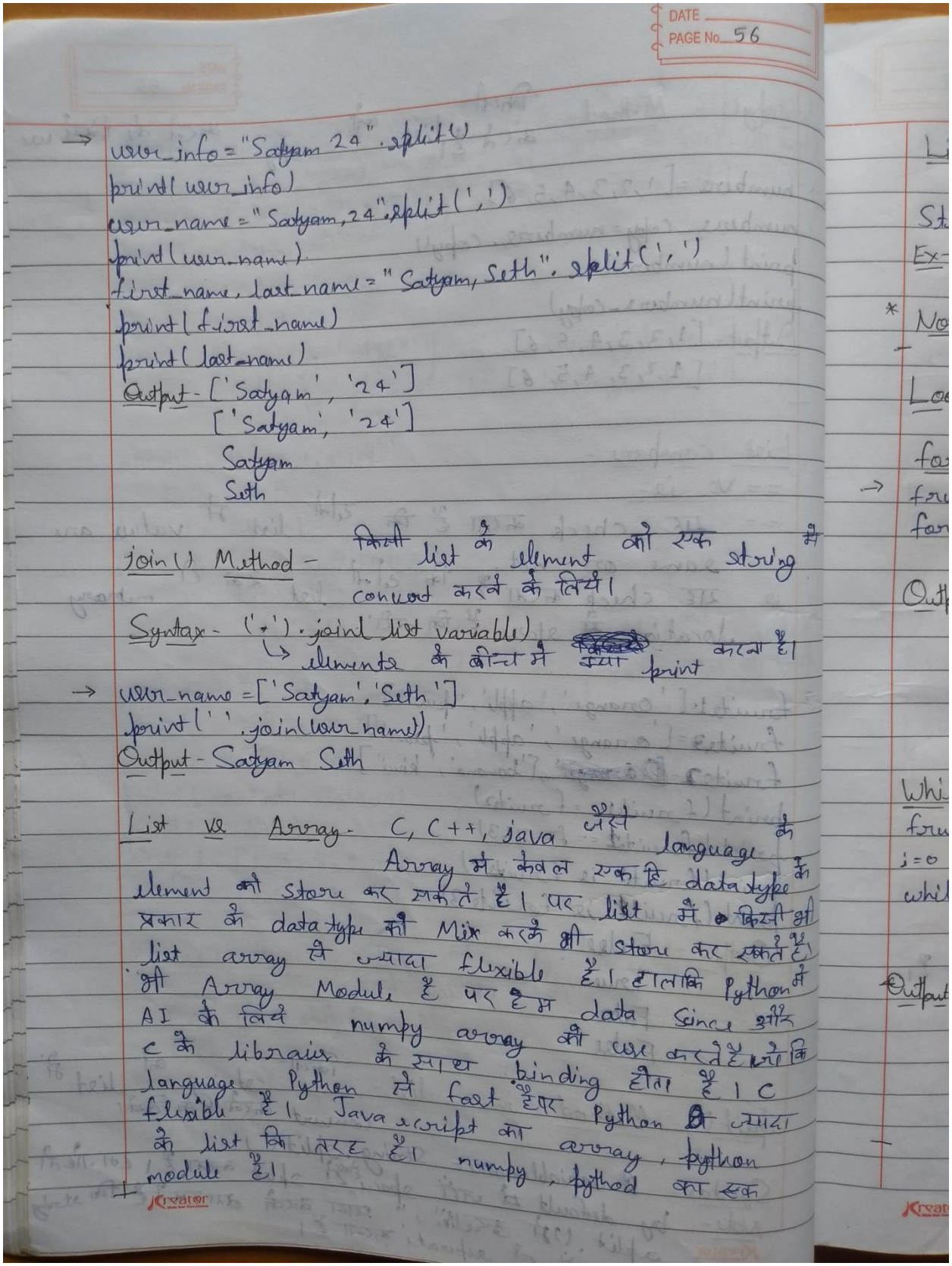
List compare -
== vs is -
== यह check करता है कि दोनों list में value are same or not.
is - यह check करता है कि दोनों list एक ही memory location में store है कि नहीं।

→
```
fruits1 = ['orange', 'appli', 'pear']
fruits3 = ['orange', 'appli', 'pear']
fruits2 = ['banana', 'kiwi']
print(fruits1 == fruits2)
print(fruits1 == fruits3)
print(fruits1 is fruits2)
print(fruits1 is fruits3)
```
Output - False
True
False
False

split() Method in list - किसी string को list में convert करने के लिये।

Syntax - variablename = String.split()

Note - by default तो यहाँ space से split करता है। हम चाहें तो split() में कुछ भी, " पास करके बता सकते हैं कि string को कहाँ से separate करना है।



DATE
PAGE No. 56

```
user_info = "Satyam 24".split()
print(user_info)
user_name = "Satyam,24".split(',')
print(user_name)
first_name, last_name = "Satyam, Seth".split(',')
print(first_name)
print(last_name)
```

Output - ['Satyam', '24']
['Satyam', '24']
Satyam
Seth

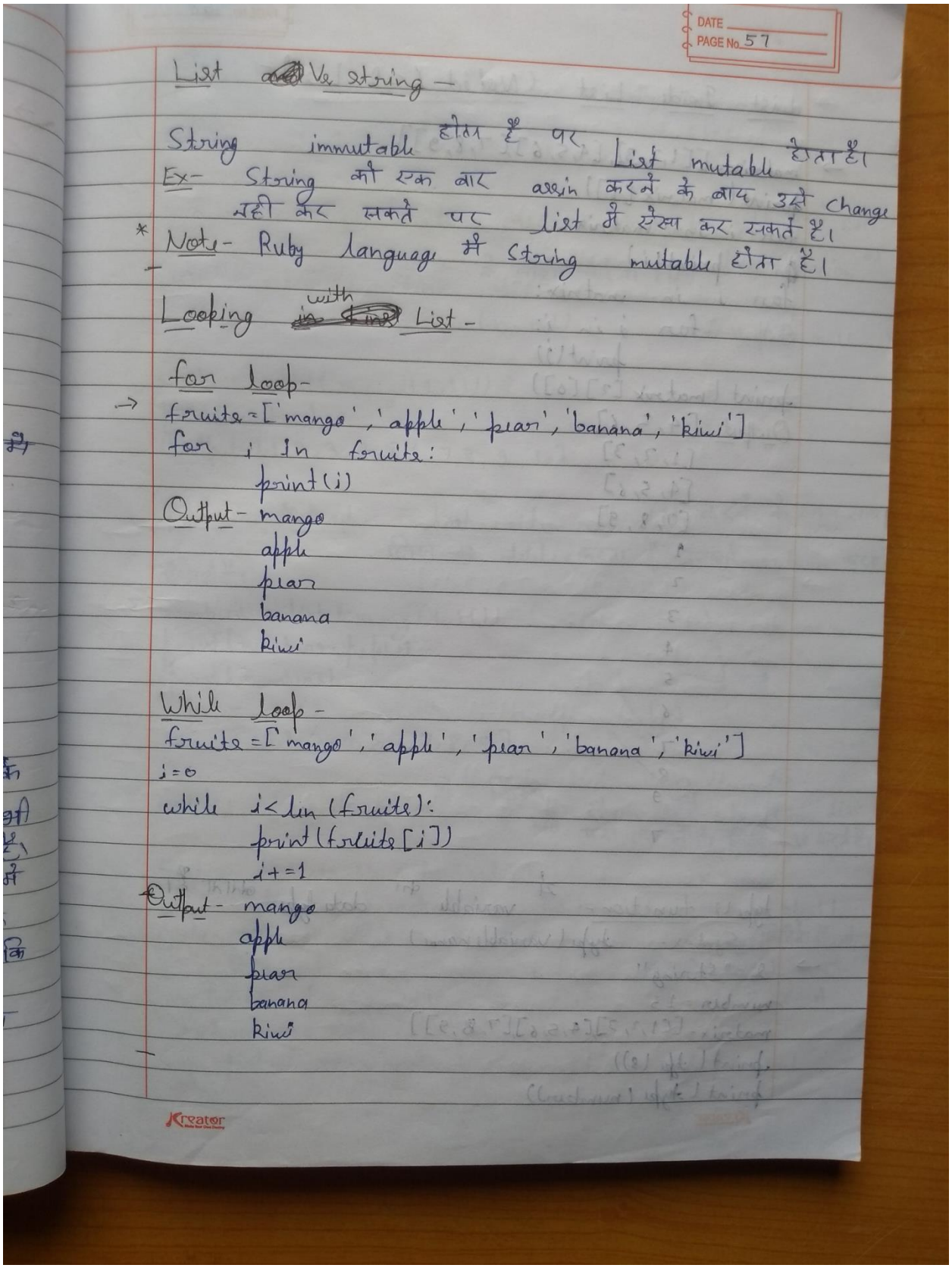
join() Method - ~~किसी~~ list के element को एक string में convert करने के लिये।

Syntax - (' ').join(list variable)
↳ elements के बीच में क्या print करना है।

```
user_name = ['Satyam', 'Seth']
print(' '.join(user_name))
```

Output - Satyam Seth

List vs Array - C, C++, java जैसे language के Array में केवल एक ही data type के element को store कर सकते हैं। पर list में किसी भी प्रकार के data type को Mix करके भी store कर सकते हैं। list array से ज्यादा flexible है। हालांकि Python में भी Array Module है पर हम data Science और AI के लिये numpy array को use करते हैं क्योंकि C के libraies के साथ binding होता है। C language Python से fast है पर Python ज्यादा flexible है। Java script का array, python के list कि तरह है। numpy, python का एक module है।



List Vs String –

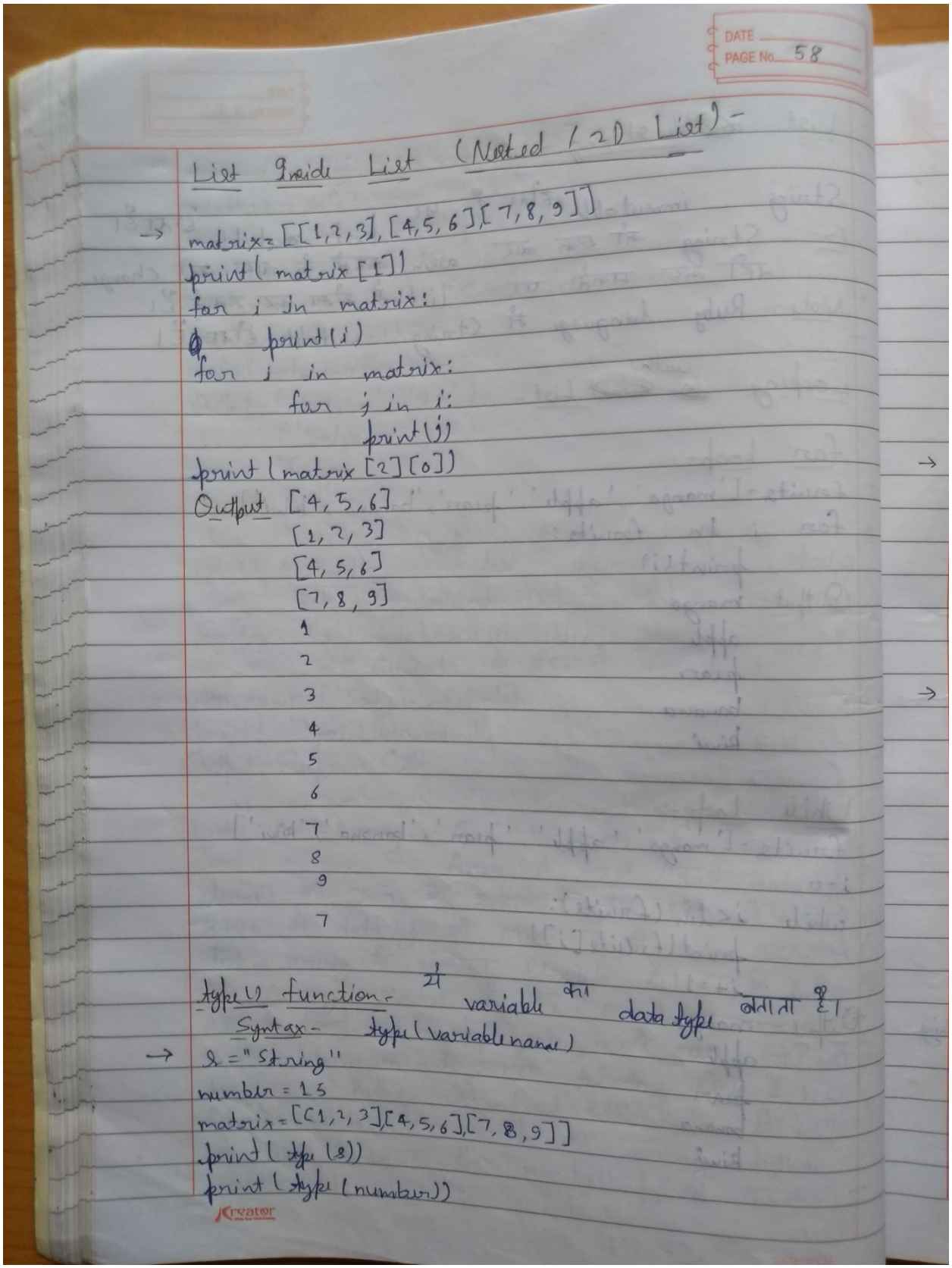
String immutable होता है पर List mutable होता है।
Ex- String को एक बार assin करने के बाद उसे change नहीं कर सकते पर list में ऐसा कर सकते है।
* Note- Ruby language में String mutable होता है।

Looping with List –

for loop-

→
```
fruits=['mango','apple','pear','banana','kiwi']
for i in fruits:
    print(i)
```
Output- mango
apple
pear
banana
kiwi

While loop -
```
fruits=['mango','apple','pear','banana','kiwi']
i=0
while i<len(fruits):
    print(fruits[i])
    i+=1
```
Output- mango
apple
pear
banana
kiwi



DATE
PAGE No. 58

List Inside List (Nested / 2D List) -

→ matrix = [[1,2,3],[4,5,6],[7,8,9]]
print(matrix[1])
for i in matrix:
    print(i)
for i in matrix:
    for j in i:
        print(j)
print(matrix[2][0])
Output [4,5,6]
[1,2,3]
[4,5,6]
[7,8,9]
1
2
3
4
5
6
7
8
9
7

type() function - किसी variable का data type बताता है।
Syntax - type(variable name)
→ s = "String"
number = 15
matrix = [[1,2,3],[4,5,6],[7,8,9]]
print(type(s))
print(type(number))

DATE
PAGE No. 59

print(type(matrix))

More about lists-

Generate List with range() Method- इसके लिये हमें list() function का use range function के साथ करना होगा।

Syntax- list(range(n, m)) (for n to m-1)

→
```
numbers = list(range(1,11))
print(numbers)
```
Output - [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

pop() Method- last item को delete करने के साथ-साथ जिसे delete करता है उसको return करता है।

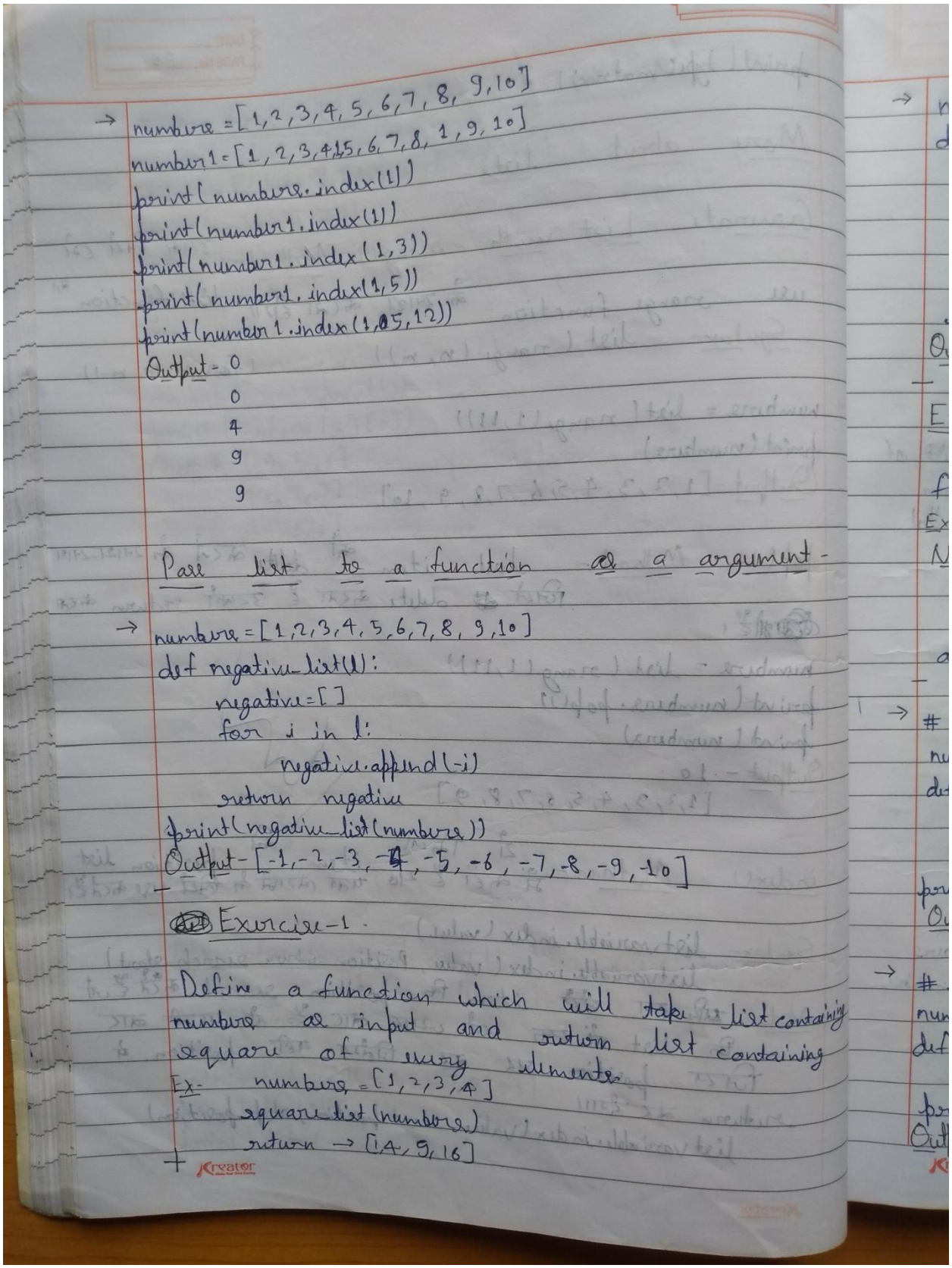
→
```
numbers = list(rang(1,11))
print(numbers.pop())
print(numbers)
```
Output - 10
[1, 2, 3, 4, 5, 6, 7, 8, 9]

index() Method- यें किसी item को list में कहाँ है यह position पता लगाने के लिये use करते है।

Syntax- listvariable.index(value)
listvariable.index(value, Position where search start)

Note- यदि हम ऐसे item कि position search करते हैं जो कि list में एक से ज्यादा बार है तो पहली बार जिस position पर वह मिलेगा वही position ये return कर देगा।

listvariable.index(value, start position, stop position)



```
numbers = [1,2,3,4,5,6,7,8,9,10]
number1 = [1,2,3,4,5,6,7,8,1,9,10]
print(numbers.index(1))
print(number1.index(1))
print(number1.index(1,3))
print(number1.index(1,5))
print(number1.index(1,05,12))
```

Output - 0
0
4
9
9

Pass list to a function as a argument -

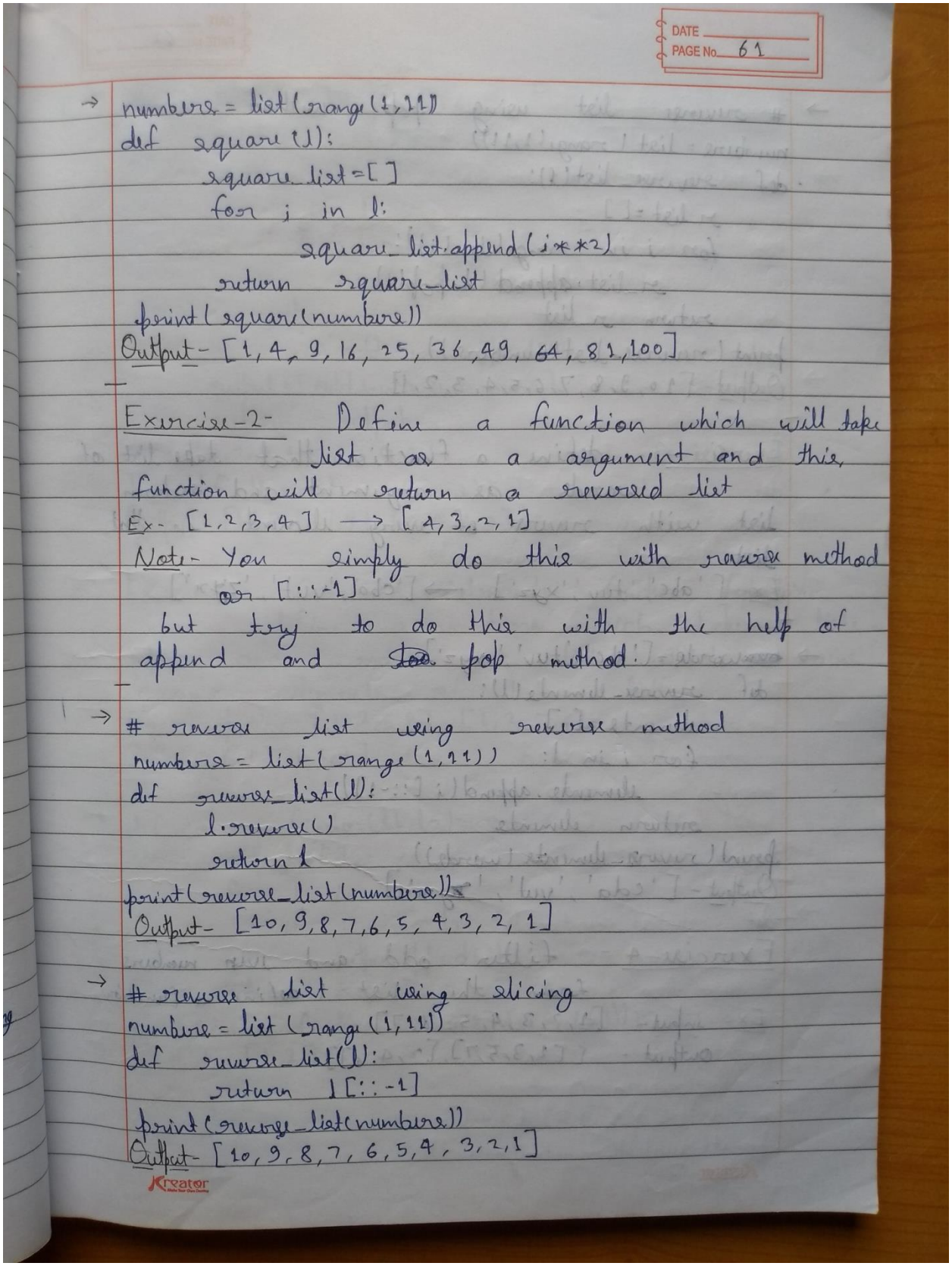
```
numbers = [1,2,3,4,5,6,7,8,9,10]
def negative_list(l):
    negative = []
    for i in l:
        negative.append(-i)
    return negative
print(negative_list(numbers))
```

Output - [-1,-2,-3,-4,-5,-6,-7,-8,-9,-10]

Exercise - 1.

Define a function which will take list containing numbers as input and return list containing square of every elements.

Ex- numbers = [1,2,3,4]
square_list(numbers)
return → [1,4,9,16]



DATE
PAGE No. 61

→
```
numbers = list(range(1,11))
def square(l):
    square_list=[]
    for i in l:
        square_list.append(i**2)
    return square_list
print(square(numbers))
```
Output- [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

—

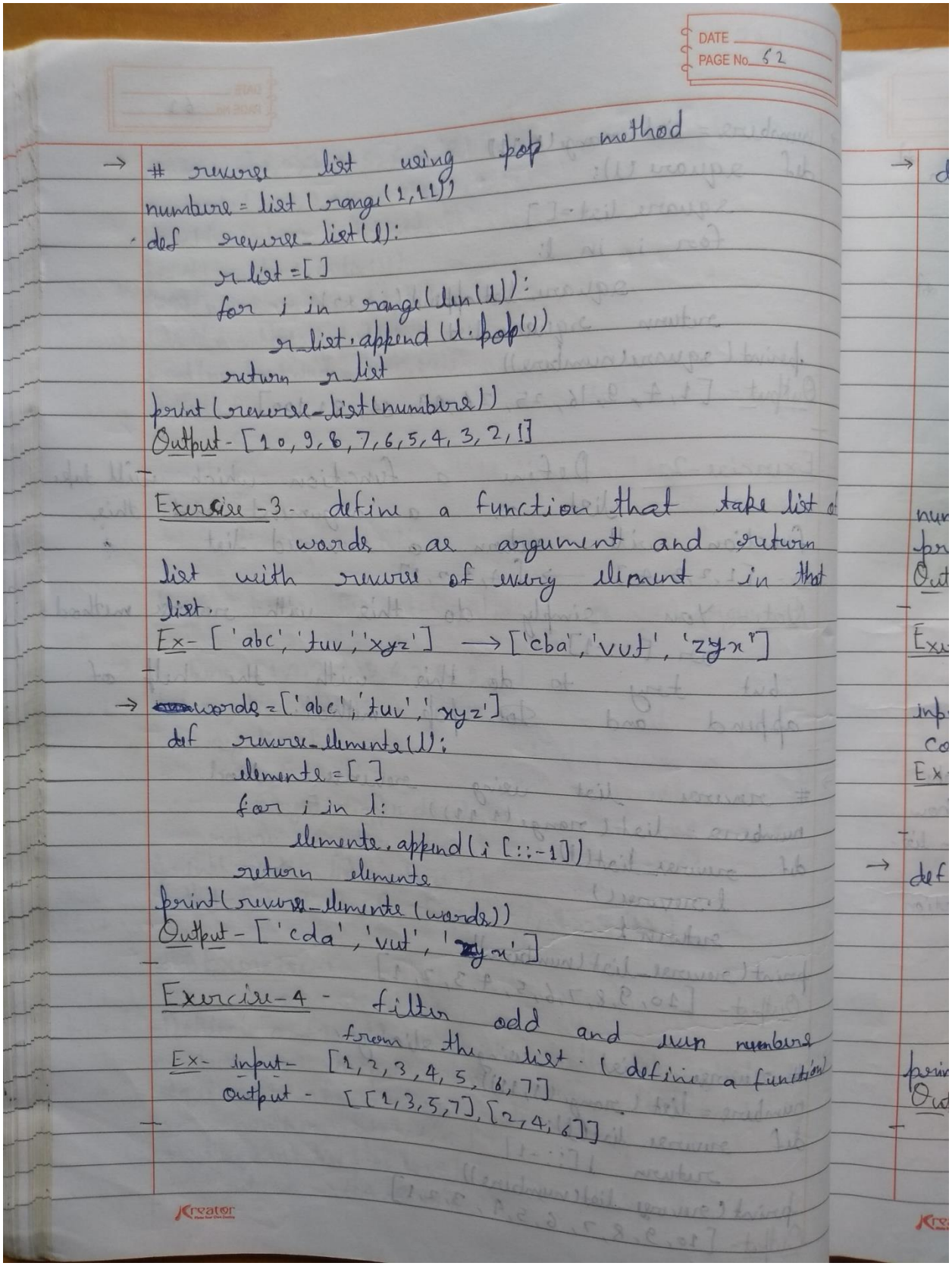
Exercise-2- Define a function which will take list as a argument and this function will return a reversed list
Ex- [1,2,3,4] → [4,3,2,1]
Note- You simply do this with reverse method or [::-1]
but try to do this with the help of append and ~~too~~ pop method.

—

→
```
# reverse list using reverse method
numbers = list(range(1,11))
def reverse_list(l):
    l.reverse()
    return l
print(reverse_list(numbers))
```
Output- [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

→
```
# reverse list using slicing
numbers = list(range(1,11))
def reverse_list(l):
    return l[::-1]
print(reverse_list(numbers))
```
Output- [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

Kreator



```
# reverse list using pop method
numbers = list(range(1,11))
def reverse_list(l):
    r_list = []
    for i in range(len(l)):
        r_list.append(l.pop())
    return r_list
print(reverse_list(numbers))
```

Output - [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

Exercise - 3. define a function that take list of words as argument and return list with reverse of every element in that list.

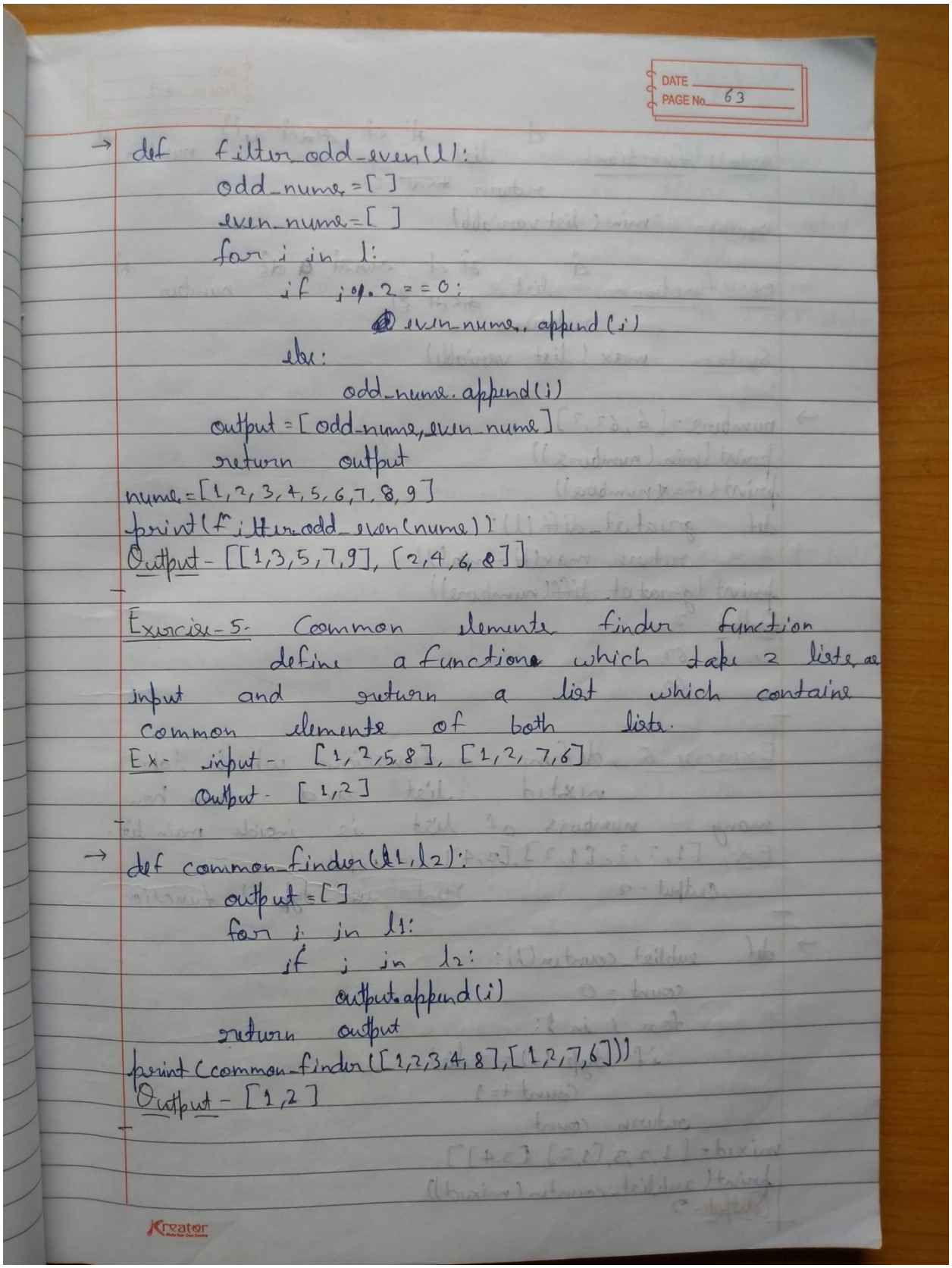
Ex - ['abc', 'tuv', 'xyz'] → ['cba', 'vut', 'zyx']

```
words = ['abc', 'tuv', 'xyz']
def reverse_elements(l):
    elements = []
    for i in l:
        elements.append(i[::-1])
    return elements
print(reverse_elements(words))
```

Output - ['cba', 'vut', 'zyx']

Exercise - 4 - filter odd and even numbers from the list. (define a function)

Ex - Input - [1, 2, 3, 4, 5, 6, 7]
Output - [[1, 3, 5, 7], [2, 4, 6]]



```
→ def filter_odd_even(l):
      odd_nums=[]
      even_nums=[]
      for i in l:
          if i%2==0:
              even_nums.append(i)
          else:
              odd_nums.append(i)
      output=[odd_nums,even_nums]
      return output
nums=[1,2,3,4,5,6,7,8,9]
print(filter_odd_even(nums))
```

Output - [[1,3,5,7,9],[2,4,6,8]]

Exercise-5- Common elements finder function

define a function which take 2 lists as input and return a list which contains common elements of both lists.

Ex- input - [1,2,5,8], [1,2,7,6]

Output - [1,2]

```
→ def common_finder(l1,l2):
      output=[]
      for i in l1:
          if i in l2:
              output.append(i)
      return output
print(common_finder([1,2,3,4,8],[1,2,7,6]))
```

Output - [1,2]

DATE
PAGE No. 64

min() function- दे list में से सबसे छोटे number को return करता है।

Syntax- min (list variable)

max() function- दे list में से सबसे बड़े number को return करता है।

Syntax- max (list variable)

```
numbers=[6,62,2]
print(min(numbers))
print(max(numbers))
def greatest_diff(l):
    return max(l)-min(l)
print(greatest_diff(numbers))
```

Output - 2
62
60

Exercise-6 define a function which take nested list and return how many numbers of list is inside main list.
Ex- [1,2,3,[1,2],[3,4]] - input
Output - 2
Hint- use type() function

```
def sublist_counter(l):
    count = 0
    for i in l:
        if type(i)==list:
            count+=1
    return count
mixed=[1,2,3,[1,2],[3,4]]
print(sublist_counter(mixed))
```

Output - 2

Chapter-6 Tuples

DATE 20-1-19
PAGE No. 65

Tuples introduction- Tuples is a data structure. tuple can store any data type. Tuples are immutable, once tuple is created you can't update data inside tuple. Tuples are faster than list.
इसका use तब करते हैं जब हमें पता है कि हमारा data fix रहने वाला है।

Syntax- variablename = ('value', 'value2', — ,'Valuen')
Ex- days = ('monday', 'tuesday')

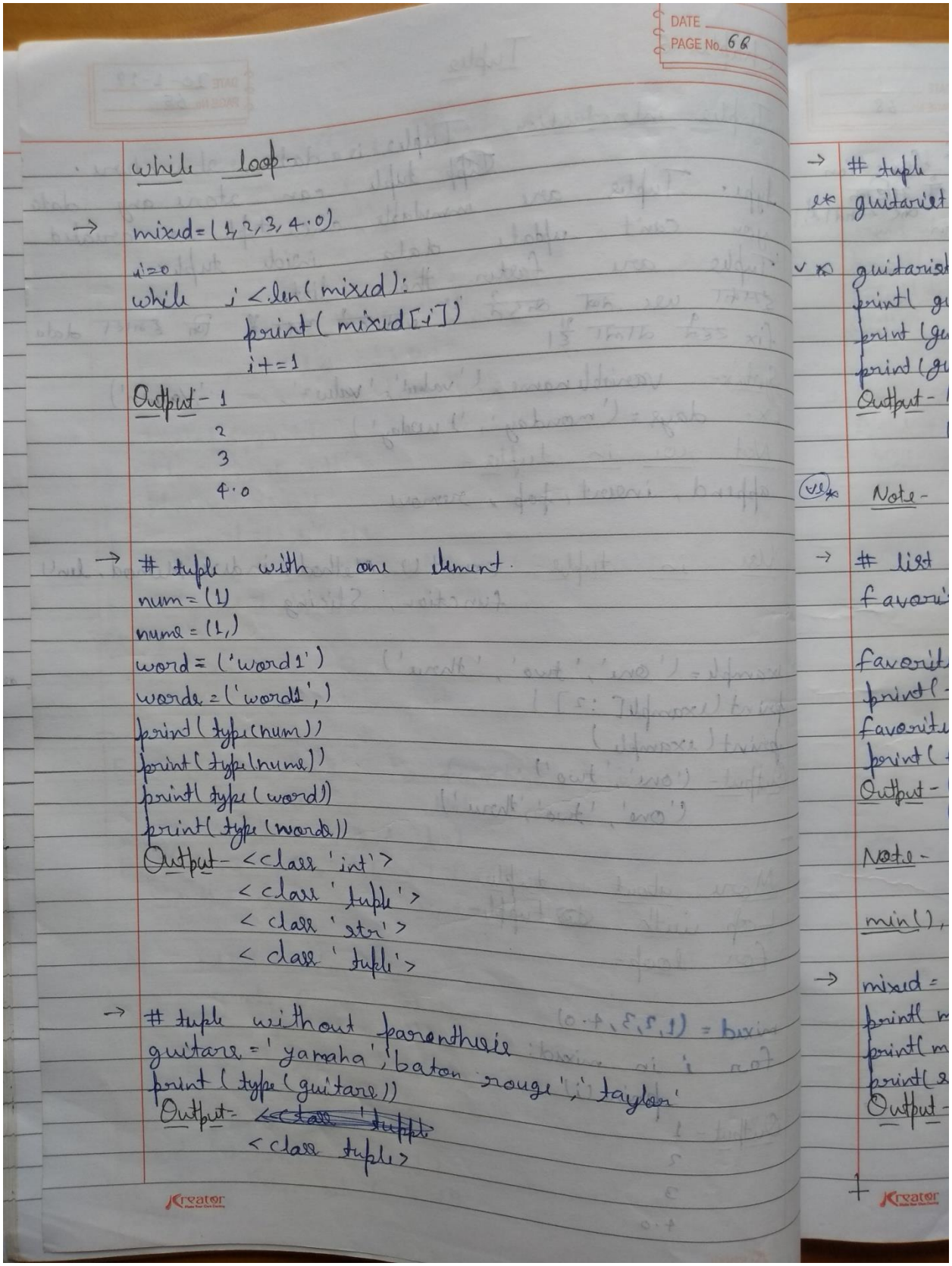
Not use in tuples -
append, insert, pop, remove

Use in tuples- count() Method, index() Method, len() function, Slicing

→ example = ('one', 'two', 'three')
print(example[:2])
print(example)
Output- ('one', 'two')
('one', 'two', 'three')

More about tuples-
Loop with tuples-
for loop-

→ mixed = (1,2,3,4.0)
for i in mixed:
print(i)
Output- 1
2
3
4.0



while loop -

→ mixed = (1, 2, 3, 4.0)

i = 0
while i < len(mixed):
    print(mixed[i])
    i += 1

Output - 1
2
3
4.0

→ # tuple with one element.
num = (1)
num2 = (1,)
word = ('word1')
word2 = ('word1',)
print(type(num))
print(type(num2))
print(type(word))
print(type(word2))
Output - <class 'int'>
<class 'tuple'>
<class 'str'>
<class 'tuple'>

→ # tuple without parenthesis
guitars = 'yamaha', 'baton rouge', 'taylor'
print(type(guitars))
Output - <class 'tuple'>

→ # tuple
guitarist

guitarist
print(gu
print(gu
print(gu
Output -

Note -

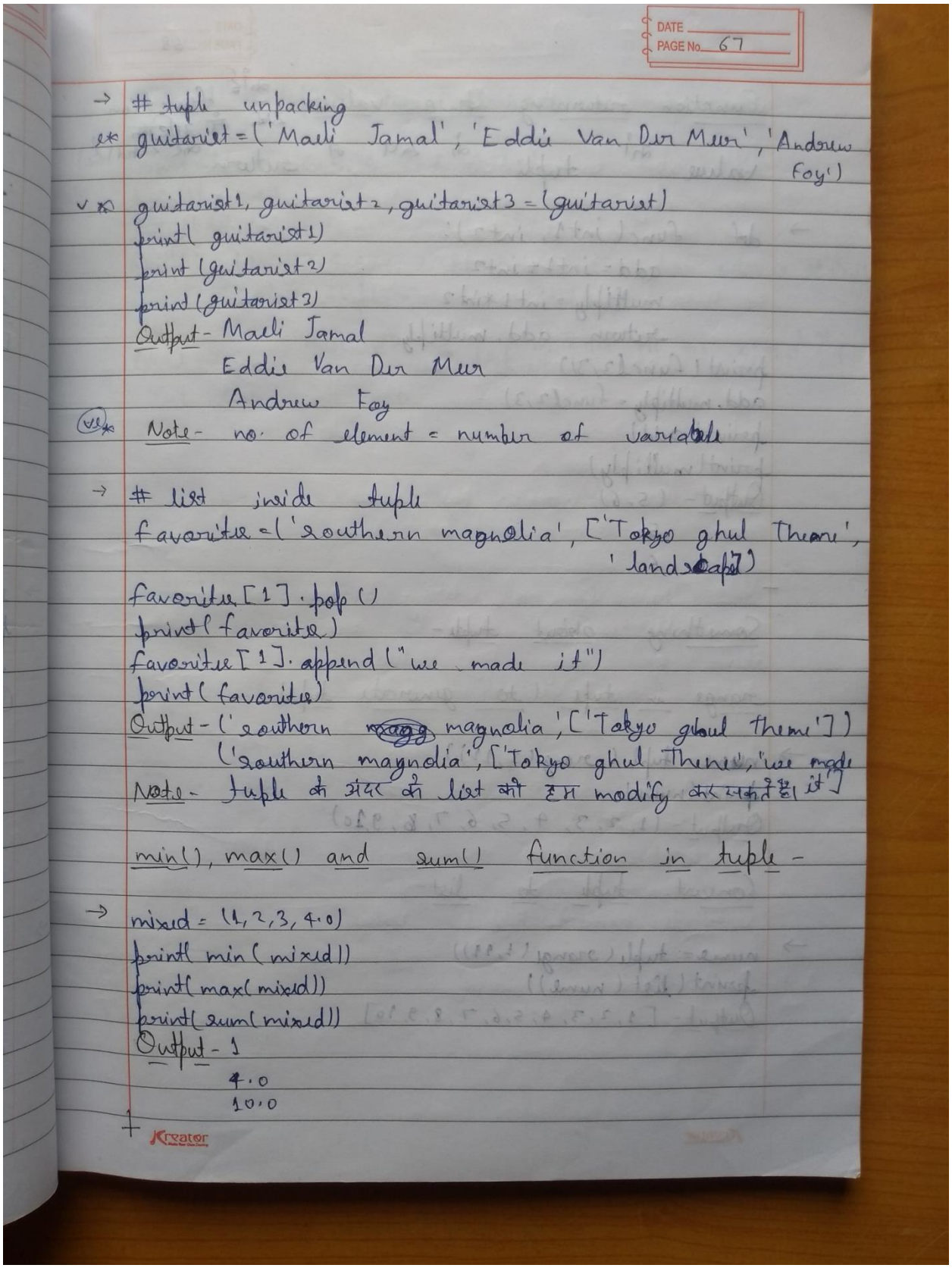
→ # list
favori

favorit
print(
favorite
print(
Output -

Note -

min(),

→ mixed =
print(m
print(m
print(m
Output -



DATE
PAGE No. 67

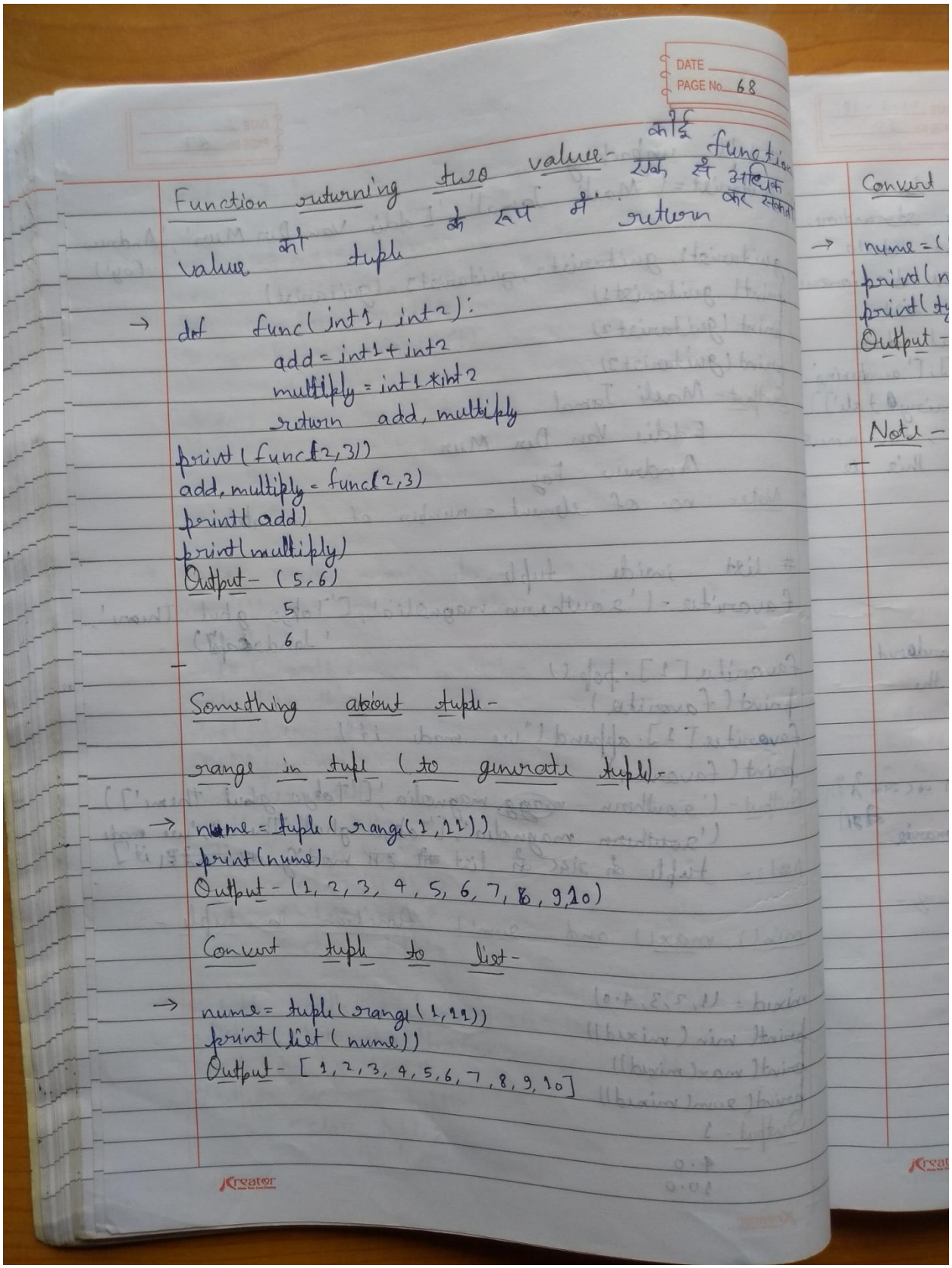
→ # tuple unpacking
** guitarist = ('Mael Jamal', 'Eddie Van Der Meer', 'Andrew Foy')

✓ ** guitarist1, guitarist2, guitarist3 = (guitarist)
print(guitarist1)
print(guitarist2)
print(guitarist3)
Output - Mael Jamal
Eddie Van Der Meer
Andrew Foy
** Note - no. of element = number of variable

→ # list inside tuple
favorite = ('southern magnolia', ['Tokyo ghul Theme', 'landscape'])
favorite[1].pop()
print(favorite)
favorite[1].append("we made it")
print(favorite)
Output - ('southern magnolia', ['Tokyo ghul theme'])
('southern magnolia', ['Tokyo ghul Theme', 'we made it']
Note - tuple के अंदर की list को हम modify कर सकते है।

min(), max() and sum() function in tuple -

→ mixed = (1, 2, 3, 4.0)
print(min(mixed))
print(max(mixed))
print(sum(mixed))
Output - 1
4.0
10.0



DATE
PAGE No. 68

Function returning two value- एक से अधिक function का return value को tuple के रूप में.

```
def func(int1, int2):
    add = int1 + int2
    multiply = int1 * int2
    return add, multiply
print(func(2,3))
add, multiply = func(2,3)
print(add)
print(multiply)
```

Output- (5,6)
5
6

Something about tuple-

range in tuple (to generate tuple)-

```
name = tuple(range(1,11))
print(name)
```

Output- (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

Convert tuple to list-

```
name = tuple(range(1,11))
print(list(name))
```

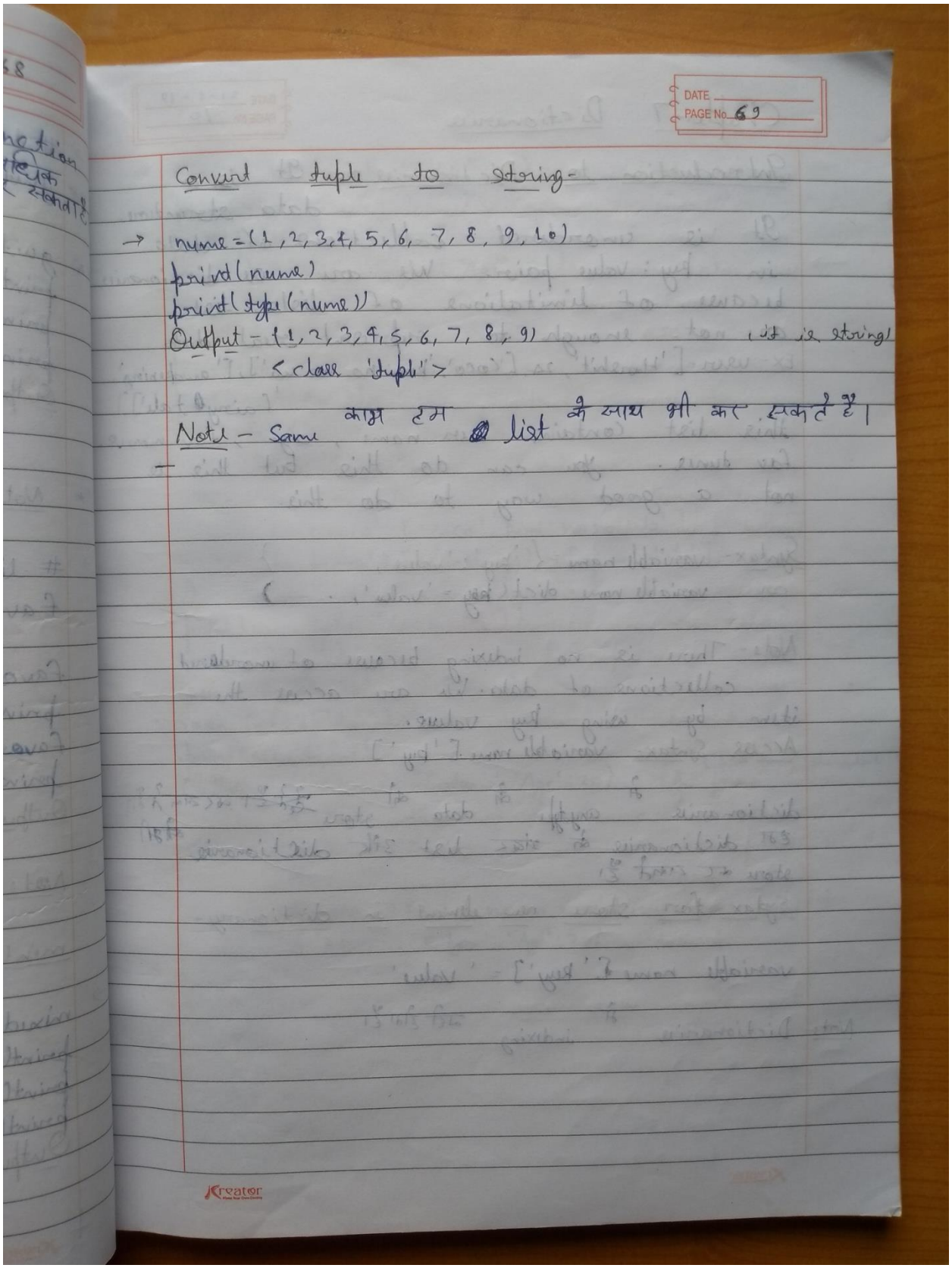
Output- [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Convert

```
name = (
print(n
print(ty
```

Output -

Note -

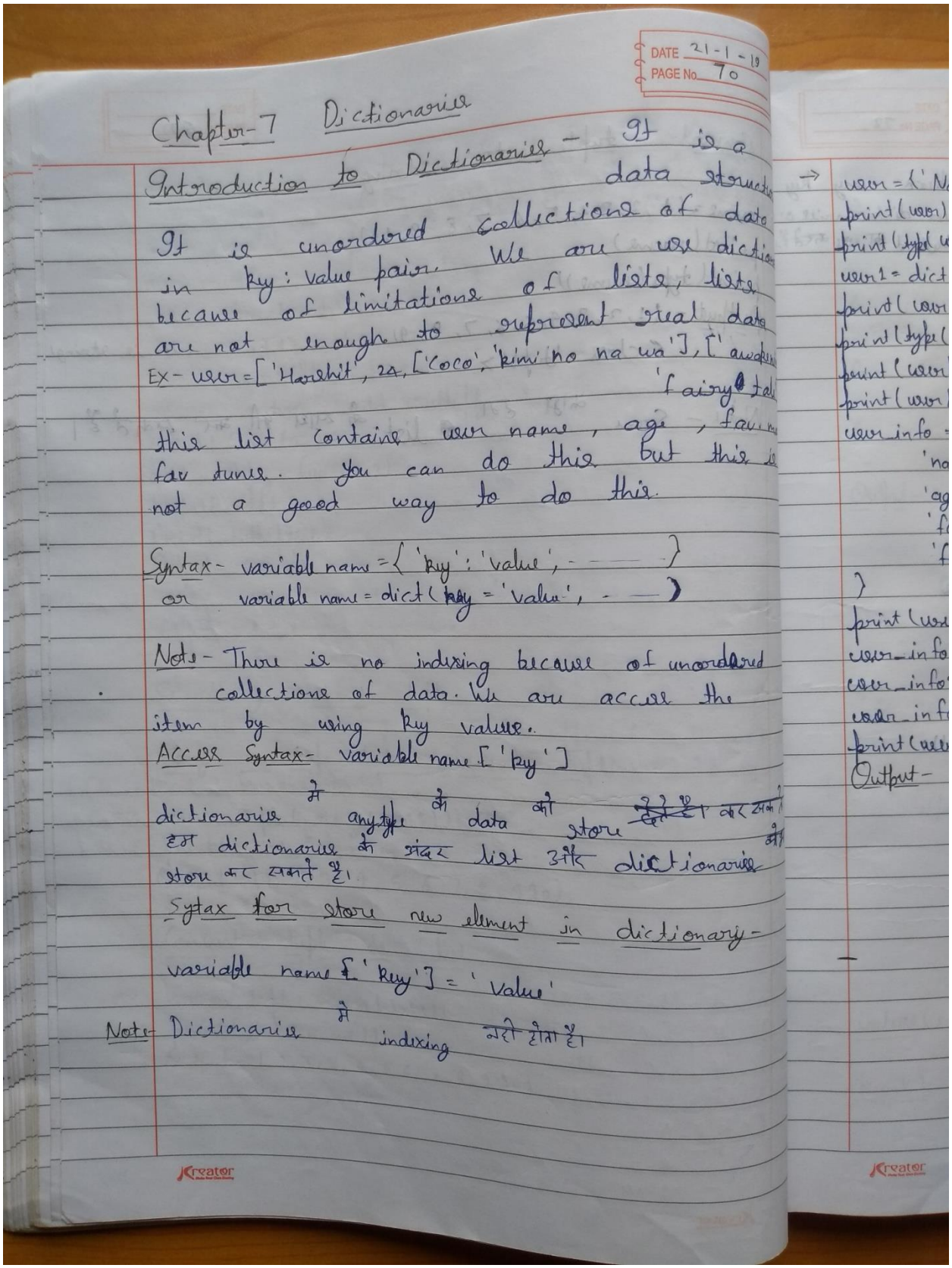


DATE
PAGE No. 69

Convert tuple to string -

→ nums = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
print(nums)
print(type(nums))
Output - (1, 2, 3, 4, 5, 6, 7, 8, 9) (it is string)
<class 'tuple'>

Note - Same काम हम list के साथ भी कर सकते हैं।



DATE 21-1-19
PAGE No. 70

Chapter-7 Dictionaries

Introduction to Dictionaries – It is a data structure. It is unordered collections of data in key : value pair. We are use dictionaries because of limitations of lists, lists are not enough to represent real data.

Ex- user = ['Harshit', 24, ['coco', 'kimi no na wa'], ['awaken', 'fairy tail']]

this list contains user name, age, fav. movies, fav tunes. You can do this but this is not a good way to do this.

Syntax- variable name = { 'key' : 'value', - - - - }
or variable name = dict(key = 'value', - - -)

Note- There is no indexing because of unordered collections of data. We are access the item by using key values.

Access Syntax- variable name ['key']

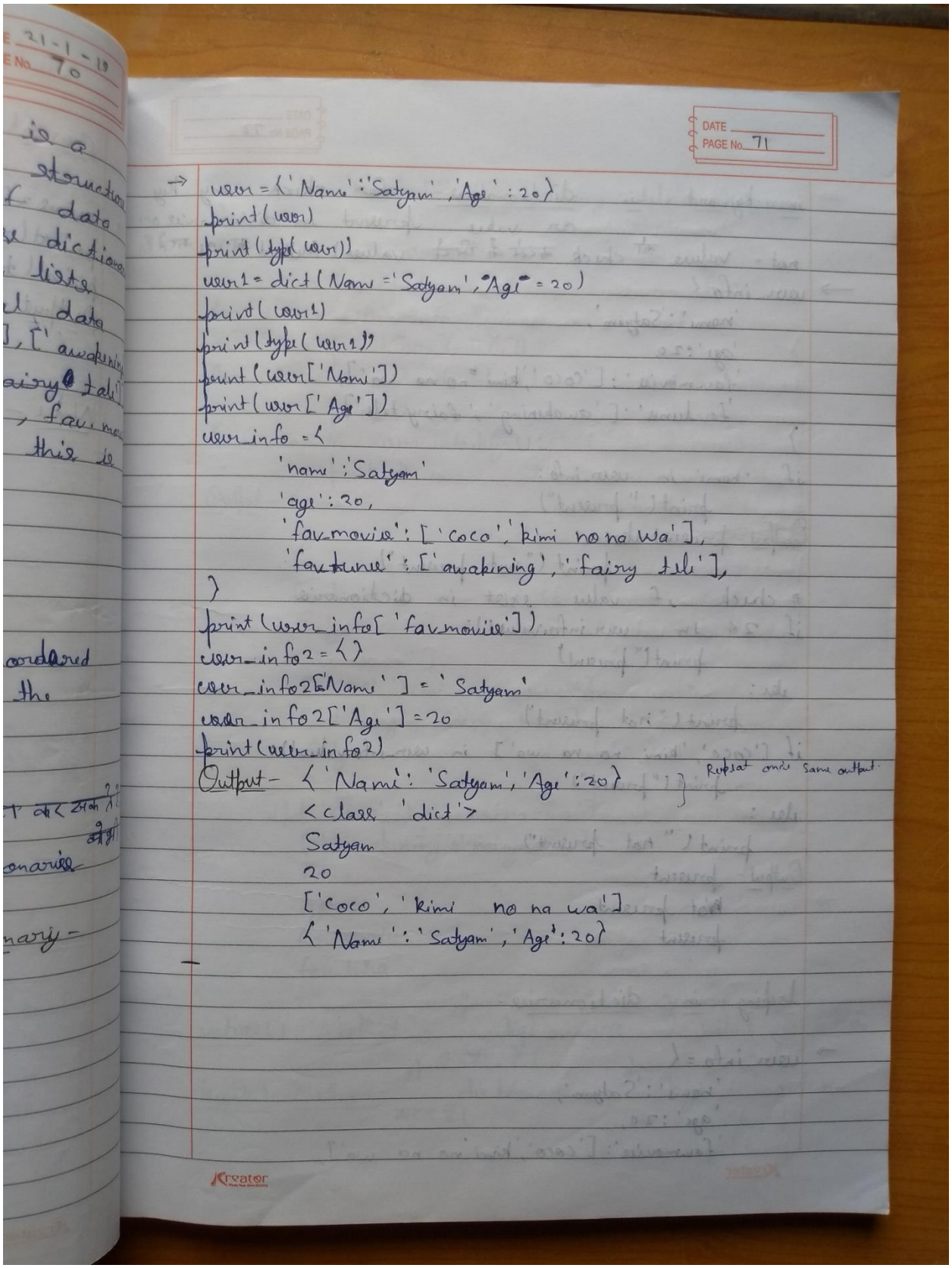
dictionaries में anytype के data को store कर सकते हैं। इन dictionaries के अंदर list और dictionaries भी store कर सकते हैं।

Sytax for store new element in dictionary-

variable name ['key'] = 'value'

Note- Dictionaries में indexing नहीं होता है।

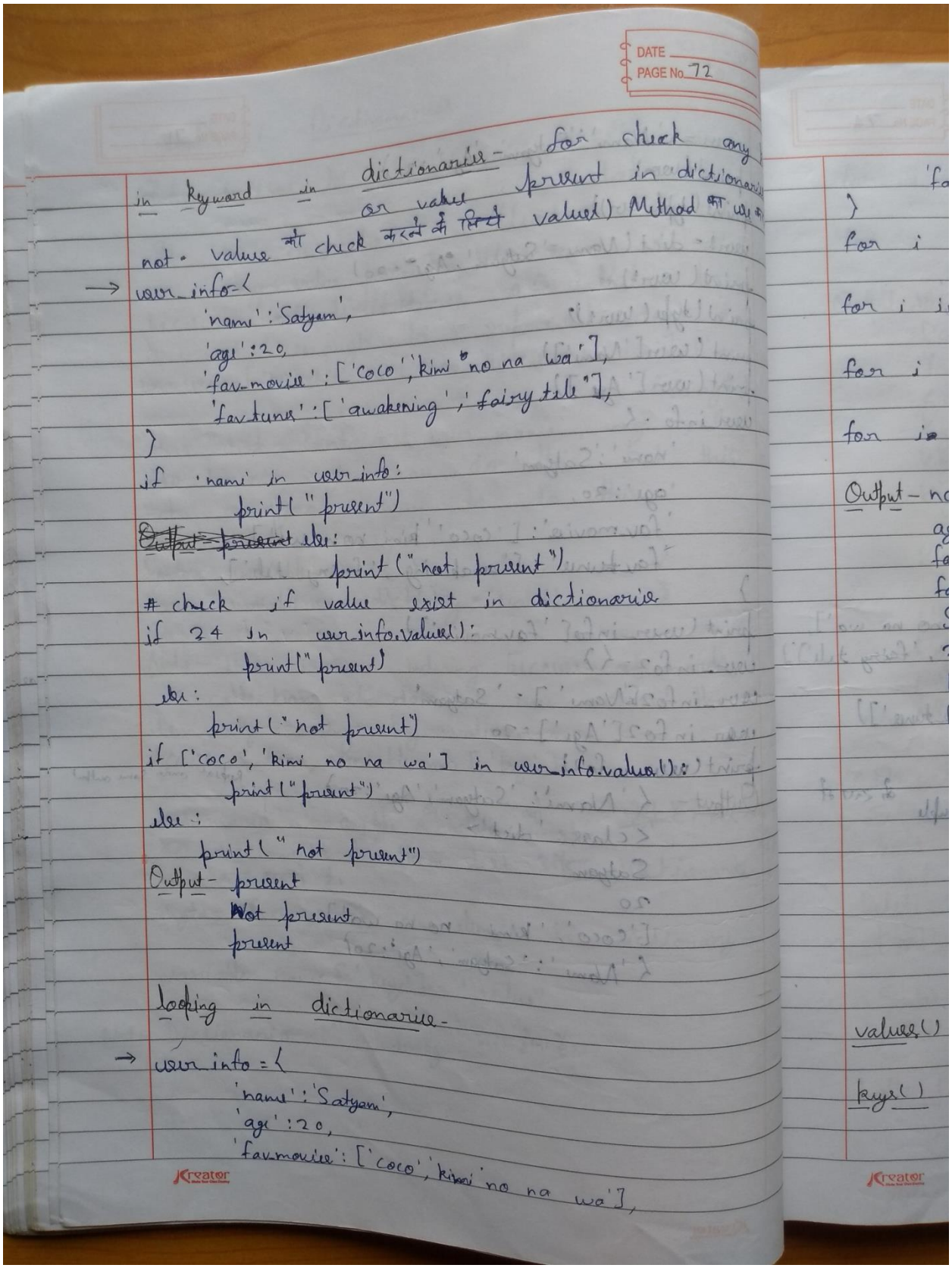
user = {'N
print(user)
print(type
user1 = dict
print(user
print(type(
print(user
print(user
user_info =
'no
'ag
'f
'f
}
print(use
user_info
user_info
user_info
print(use
Output-



```
user = {'Name':'Satyam', 'Age':20}
print(user)
print(type(user))
user1 = dict(Name='Satyam', Age=20)
print(user1)
print(type(user1))
print(user['Name'])
print(user['Age'])
user_info = {
    'name':'Satyam'
    'age':20,
    'fav_movie':['coco','kimi no na wa'],
    'fav_tune':['awakining','fairy tali'],
}
print(user_info['fav_movie'])
user_info2 = {}
user_info2['Name'] = 'Satyam'
user_info2['Age'] = 20
print(user_info2)
```

Output - {'Name': 'Satyam', 'Age':20}
<class 'dict'>
Satyam
20
['coco', 'kimi no na wa']
{'Name': 'Satyam', 'Age': 20}

Repeat once same output



in Keyword in dictionaries - for check any key or value present in dictionaries not. value को check करने के लिए value() Method का use

→ user_info = {
'name': 'Satyam',
'age': 20,
'fav_movie': ['coco', 'kimi no na wa'],
'fav_tune': ['awakening', 'fairy tale'],
}

```
if 'name' in user_info:
    print("present")
else:
    print("not present")
# check if value exist in dictionaries
if 24 in user_info.values():
    print("present")
else:
    print("not present")
if ['coco', 'kimi no na wa'] in user_info.values():
    print("present")
else:
    print("not present")
```

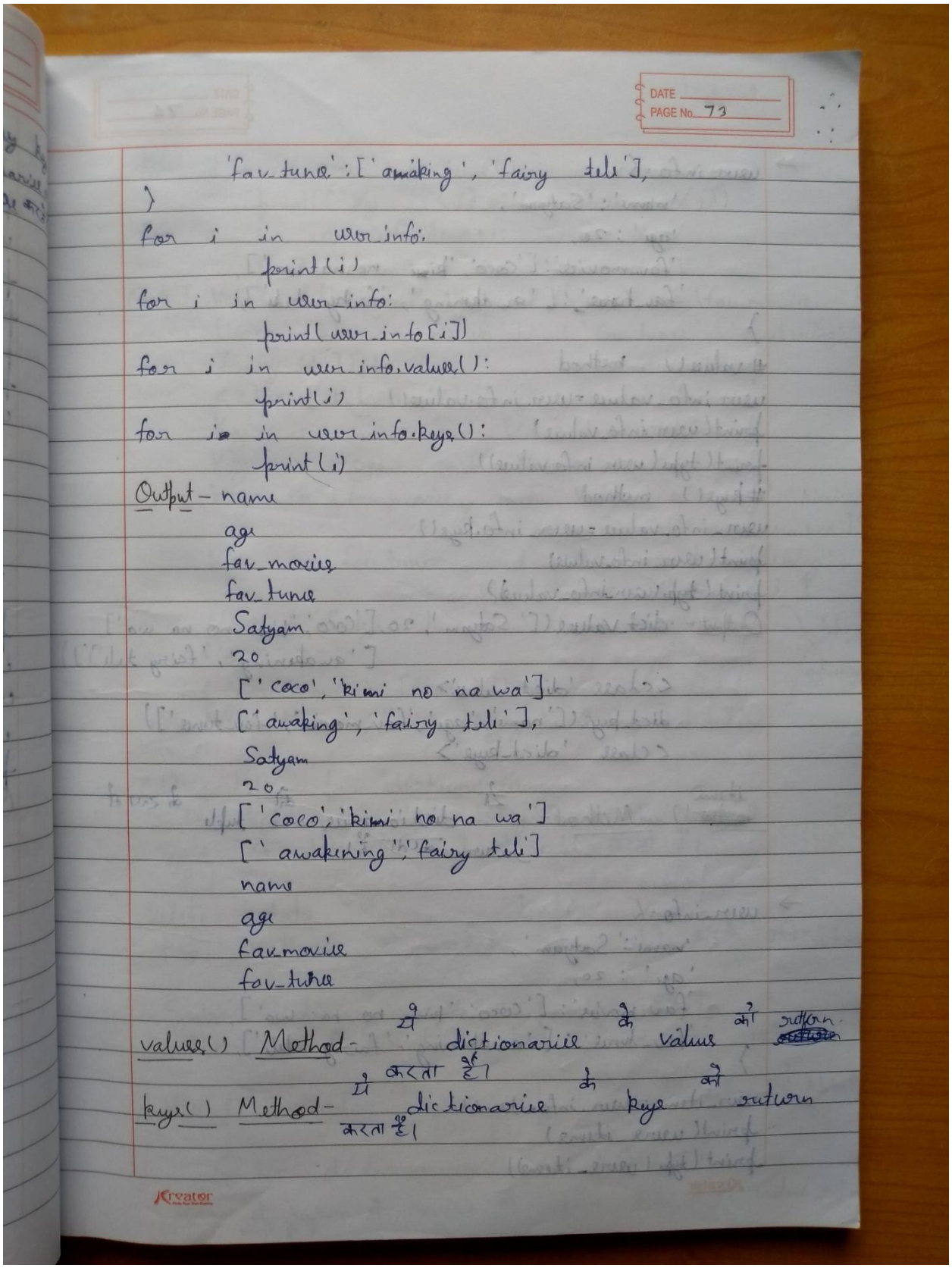
Output - present
Not present
present

looping in dictionaries -

→ user_info = {
'name': 'Satyam',
'age': 20,
'fav_movie': ['coco', 'kimi no na wa'],

values()

keys()

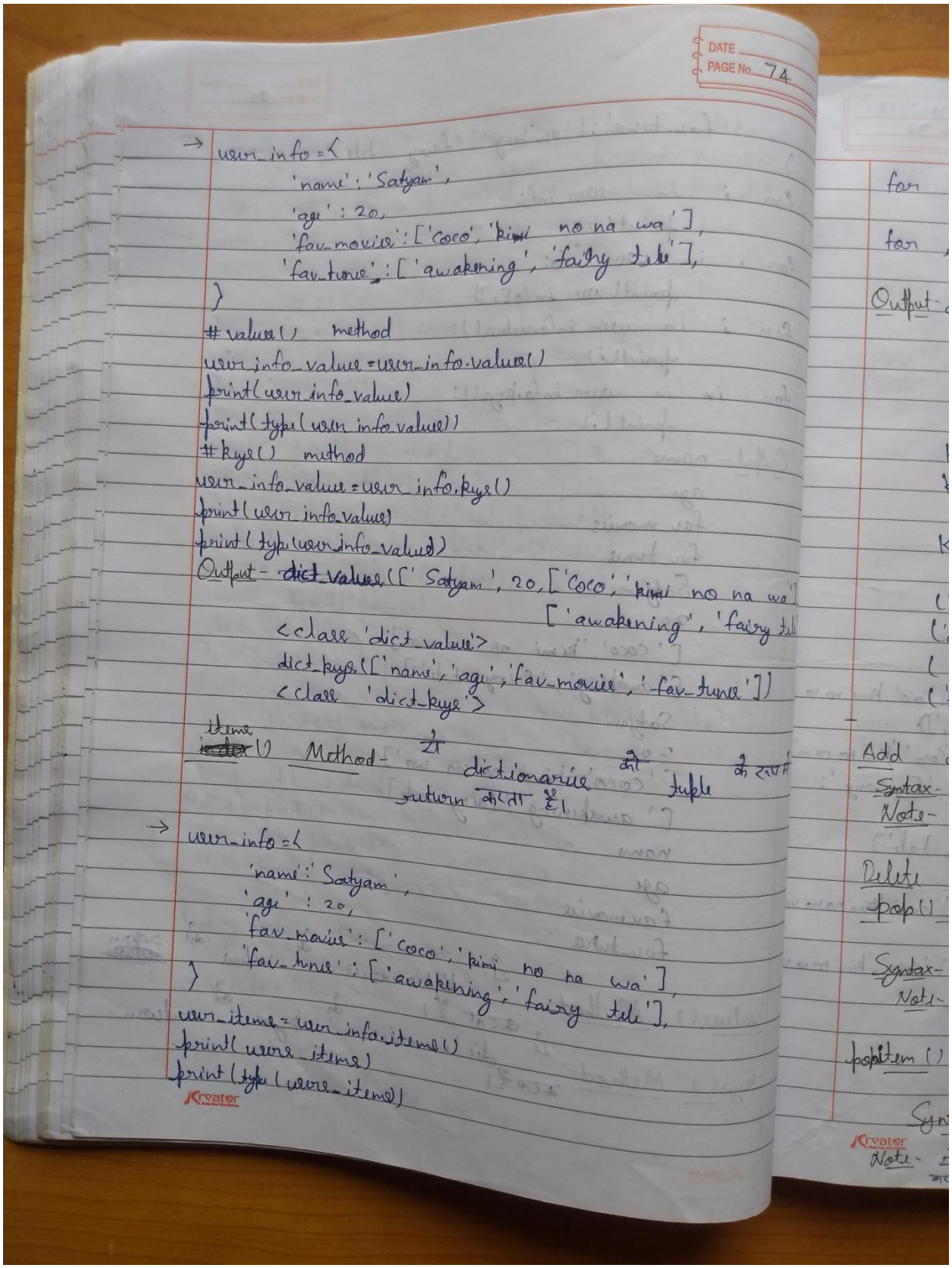


```
    'fav_tune':['awaking', 'fairy tele'],
}
for i in user_info:
    print(i)
for i in user_info:
    print(user_info[i])
for i in user_info.values():
    print(i)
for i in user_info.keys():
    print(i)
```

Output - name
age
fav_movies
fav_tune
Satyam
20
['coco', 'kimi no na wa']
['awaking', 'fairy tele'],
Satyam
20
['coco', 'kimi no na wa']
['awakening', 'fairy tele']
name
age
fav_movie
fav_tune

values() Method - यह dictionaries के values को return करता है।

keys() Method - यह dictionaries के keys को return करता है।



```
user_info = {
    'name': 'Satyam',
    'age': 20,
    'fav_movies': ['coco', 'kimi no na wa'],
    'fav_tunes': ['awakening', 'fairy tale'],
}
# values() method
user_info_value = user_info.values()
print(user_info_value)
print(type(user_info_value))
# keys() method
user_info_value = user_info.keys()
print(user_info_value)
print(type(user_info_value))
```

Output- dict_values(['Satyam', 20, ['coco', 'kimi no na wa'], ['awakening', 'fairy tale']])
<class 'dict_values'>
dict_keys(['name', 'age', 'fav_movies', 'fav_tunes'])
<class 'dict_keys'>

items() Method- It dictionaries को tuple के रूप में return करता है।

```
user_info = {
    'name': 'Satyam',
    'age': 20,
    'fav_movies': ['coco', 'kimi no na wa'],
    'fav_tunes': ['awakening', 'fairy tale'],
}
user_items = user_info.items()
print(user_items)
print(type(user_items))
```

Creator

for

for

Output-

Add

Syntax-

Note-

Delete

pop()

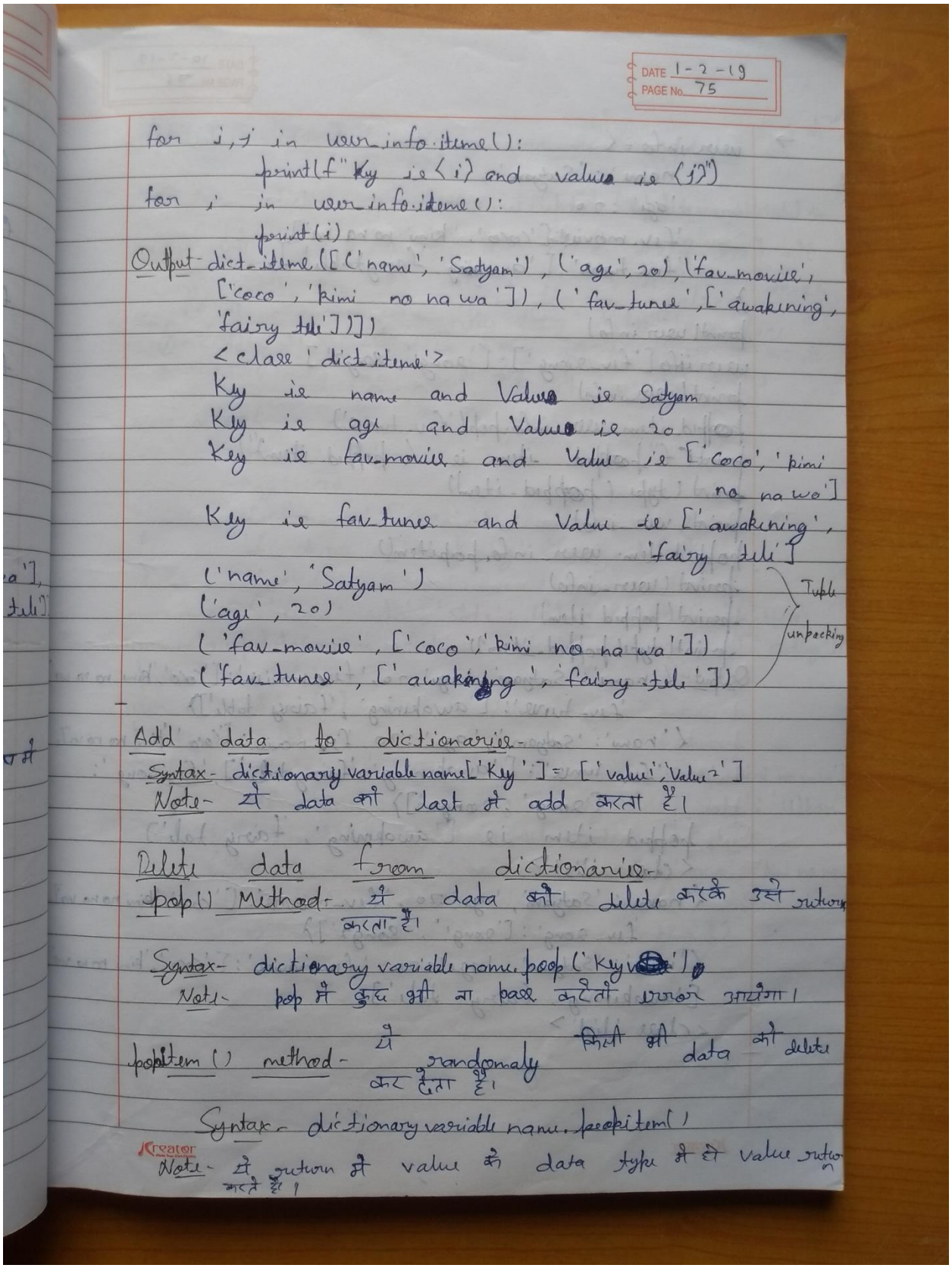
Syntax-

Note-

popitem()

Creator

Note-



DATE 1-2-19
PAGE No. 75

```
for i,j in user_info.items():
    print(f"Key is {i} and value is {j}")
for i in user_info.items():
    print(i)
```

Output- dict_items([('name', 'Satyam'), ('age', 20), ('fav_movies', ['coco', 'kimi no na wa']), ('fav_tunes', ['awakening', 'fairy tale'])])

```
<class 'dict_items'>
```

Key is name and Value is Satyam
Key is age and Value is 20
Key is fav_movies and Value is ['coco', 'kimi no na wa']
Key is fav_tunes and Value is ['awakening', 'fairy tale']

('name', 'Satyam')
('age', 20)
('fav_movies', ['coco', 'kimi no na wa'])
('fav_tunes', ['awakening', 'fairy tale'])

Tuple unpacking

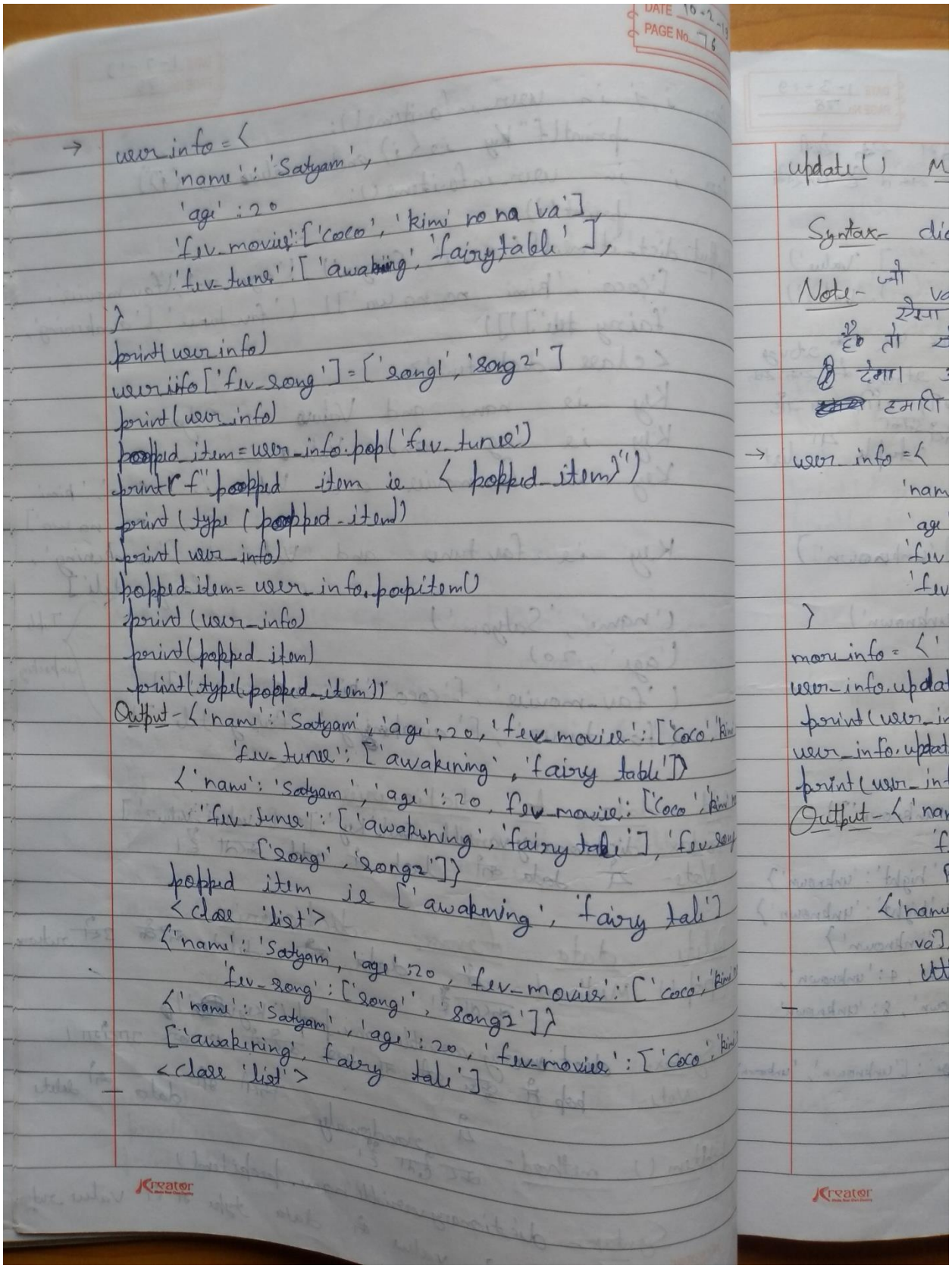
Add data to dictionaries-
Syntax- dictionary variable name['Key'] = ['value', 'value2']
Note- यह data को last में add करता है।

Delete data from dictionaries-
pop() Method- यह data को delete करके उसे return करता है।

Syntax- dictionary variable name.pop('Key')
Note- pop में कुछ भी न pass करेंगे तो error आएगा।

popitem() method- यह किसी भी data को delete randomaly कर देता है।

Syntax- dictionary variable name.popitem()
Note- यह return में value को data type में दो value return करते हैं।



```
user_info = {
    'name' : 'Satyam',
    'age' : 20
    'fav_movies': ['coco', 'kimi no na wa'],
    'fav_tunes' : ['awakening', 'fairytale' ],
}
print(user_info)
user_info['fav_song'] = ['song1', 'song2']
print(user_info)
popped_item = user_info.pop('fav_tunes')
print(f"popped item is {popped_item}")
print(type(popped_item))
print(user_info)
popped_item = user_info.popitem()
print(user_info)
print(popped_item)
print(type(popped_item))
```

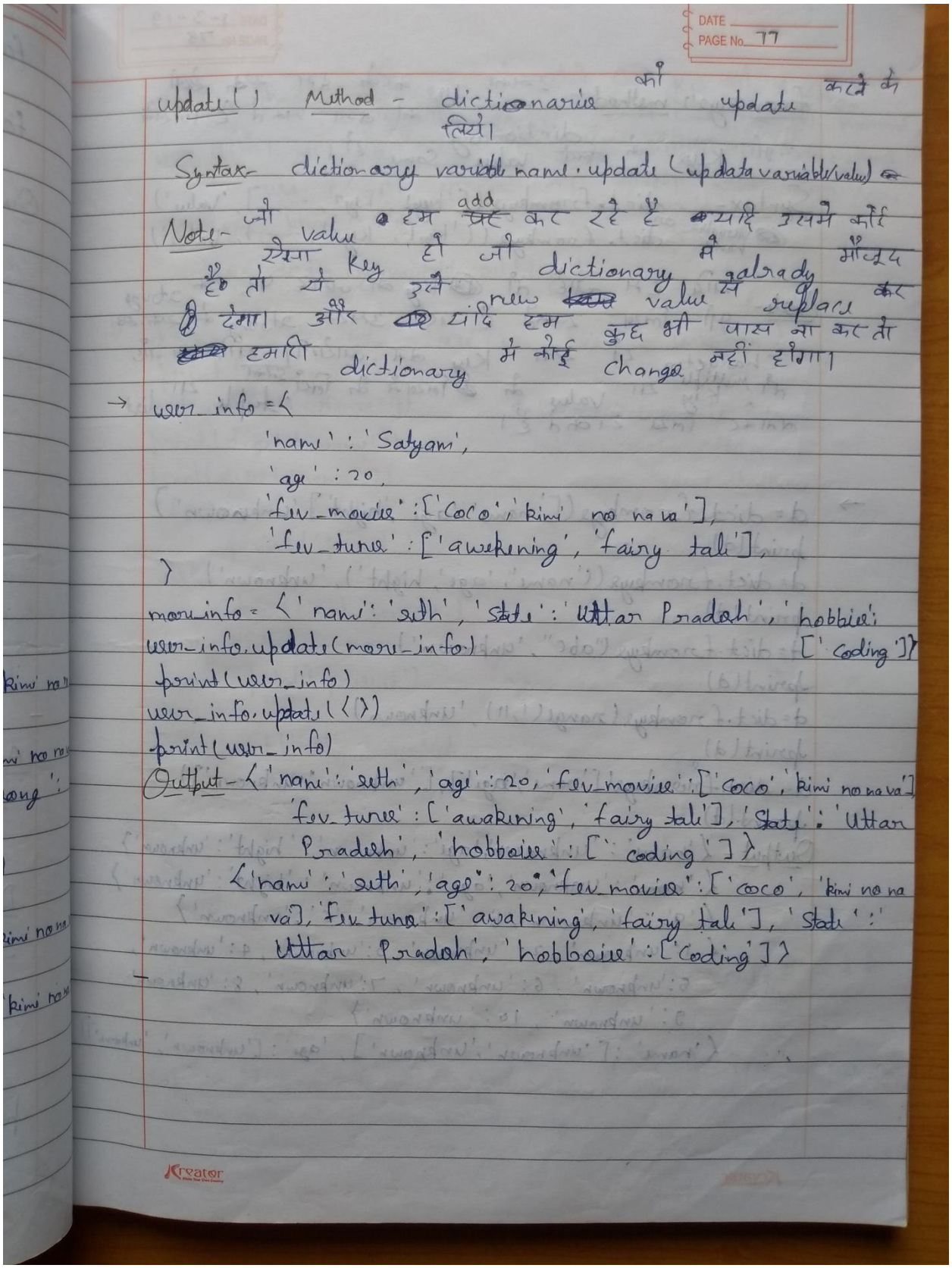
Output - {'name': 'Satyam', 'age': 20, 'fav_movies': ['coco', 'kimi
'fav_tunes': ['awakening', 'fairy tale']}
{'name': 'Satyam', 'age': 20, 'fav_movies': ['coco', 'kimi
'fav_tunes': ['awakening', 'fairy tale'], 'fav_song':
['song', 'song2']}
popped item is ['awakening', 'fairy tale']
<class 'list'>
{'name': 'Satyam', 'age': 20, 'fav_movies': ['coco', 'kimi
'fav_song': ['song', 'song2']}
{'name': 'Satyam', 'age': 20, 'fav_movies': ['coco', 'kimi
['awakening', 'fairy tale']
<class 'list'>

update() M

Syntax- di

Note- जो va
देगा
हमारी

user_info = {
'nam
'age
'fav
'fav
}
more_info = {'
user_info.upda
print(user_in
user_info.upda
print(user_in
Output - {'na
f
{'nam
va]



update() Method - dictionaries को update करने के लिये।

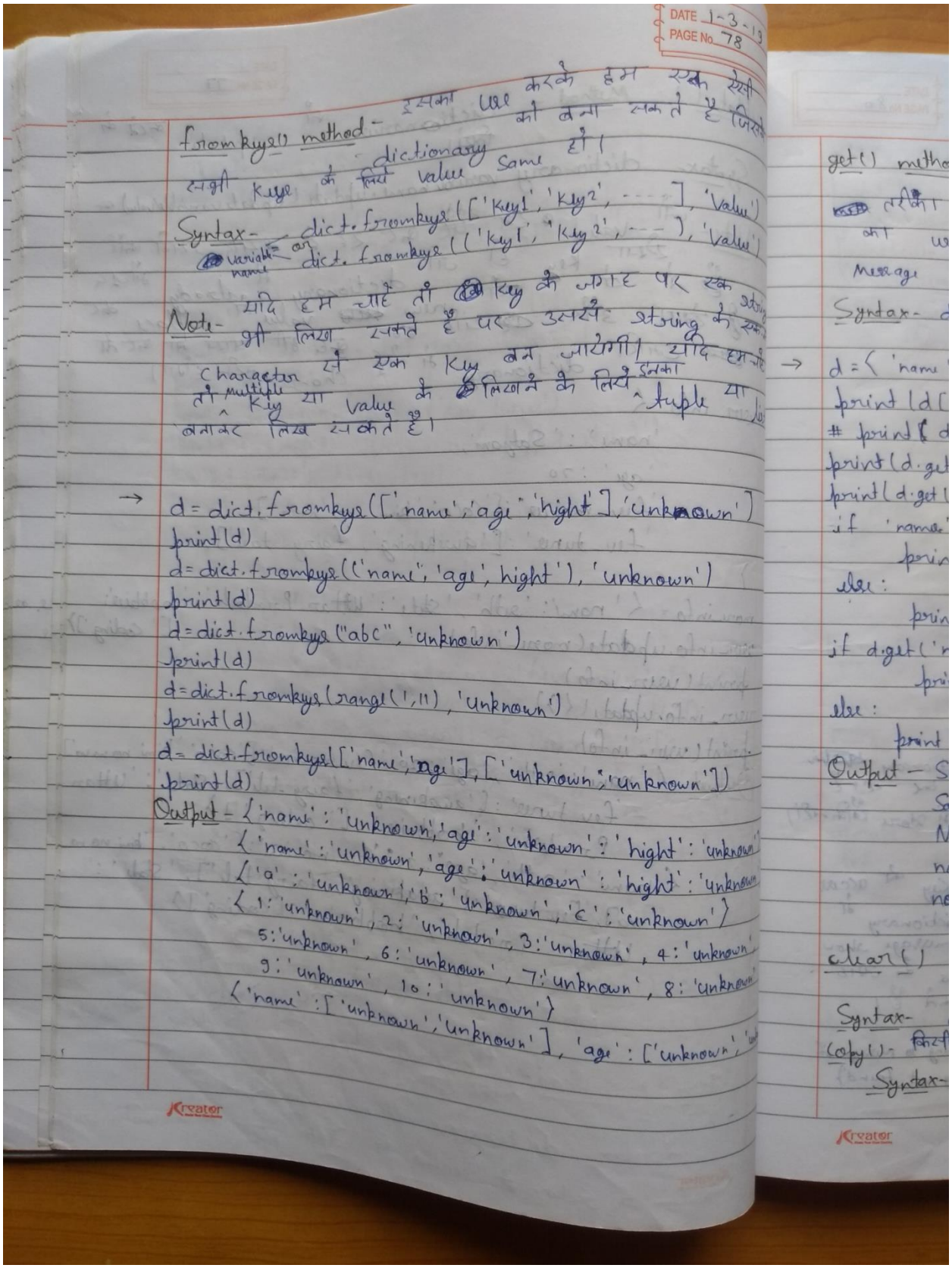
Syntax- dictionary variable name.update(updata variable/value)

Note- जो value हम add कर रहे हैं यदि उसमें कोई ऐसा key हो जो dictionary में पहले से मौजूद है तो वह इसे new value से replace कर देगा। और यदि हम कुछ भी पास ना करें तो हमारी dictionary में कोई change नहीं होगा।

→ user_info = {
'name' : 'Satyam',
'age' : 20,
'fav_movie' : ['coco', 'kimi no na wa'],
'fav_tune' : ['awakening', 'fairy tali'],
}

more_info = {'name': 'seth', 'state': 'Uttar Pradesh', 'hobbies': ['coding']}
user_info.update(more_info)
print(user_info)
user_info.update({})
print(user_info)

Output - {'name': 'seth', 'age': 20, 'fav_movie': ['coco', 'kimi no na wa'], 'fav_tune': ['awakening', 'fairy tali'], 'State': 'Uttar Pradesh', 'hobbies': ['coding']}
{'name': 'seth', 'age': 20, 'fav_movie': ['coco', 'kimi no na wa'], 'fav_tune': ['awakening', 'fairy tali'], 'State': 'Uttar Pradesh', 'hobbies': ['coding']}



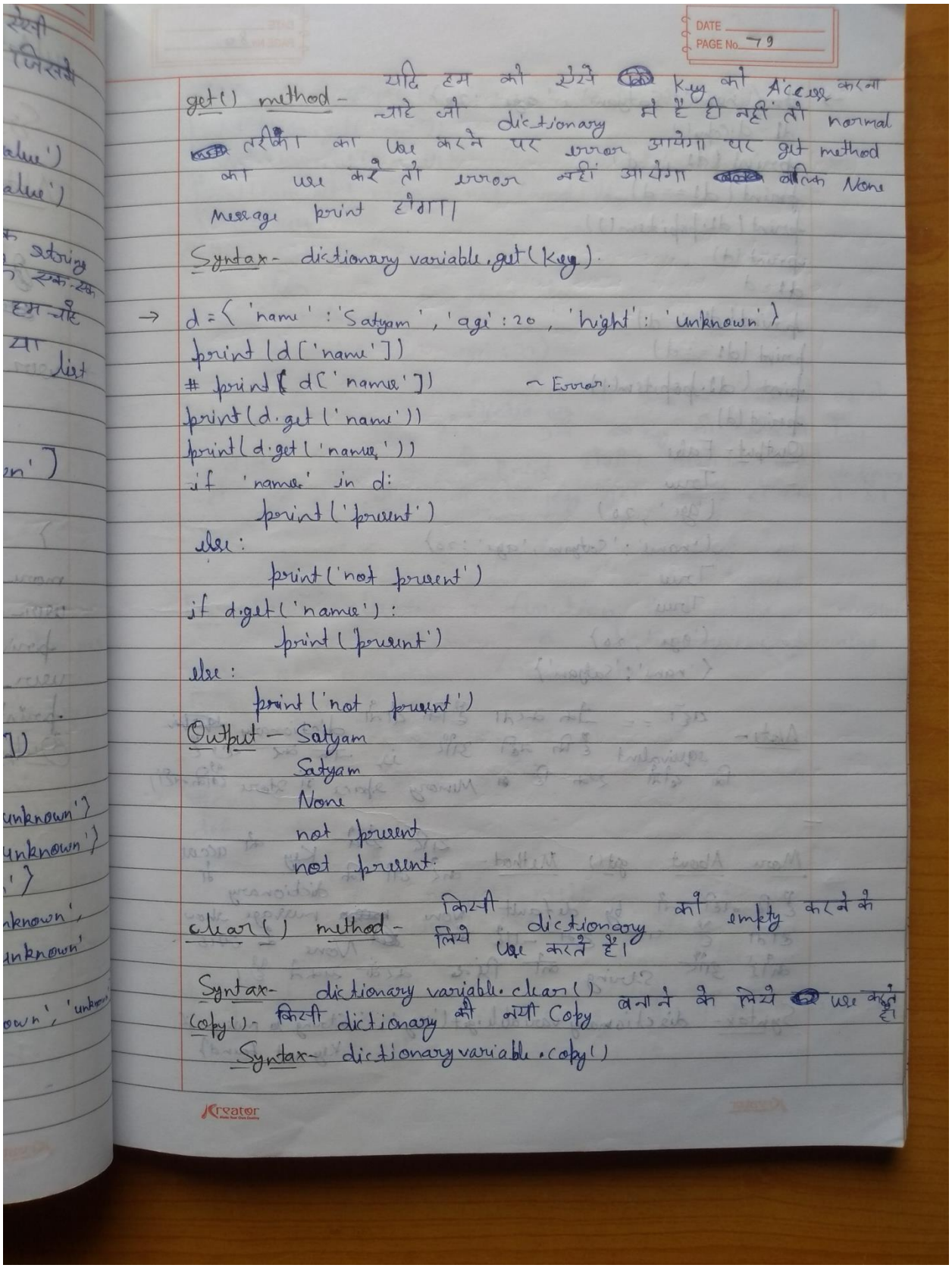
fromkeys() method - इसका use करके हम एक ऐसी dictionary को बना सकते है जिसमे सभी keys के लिये value same हो।

Syntax - dict.fromkeys(['key1', 'key2', ---], 'value')
variable name = or dict.fromkeys(('key1', 'key2', ---), 'value')

Note - यदि हम चाहें तो key के जगह पर एक string भी लिख सकते है पर उसमे string के हर character से एक key बन जायेगी। यदि हमें multiple key या value के लिखने के लिये इसका tuple या list बनाकर लिख सकते है।

→ d = dict.fromkeys(['name', 'age', 'hight'], 'unknown')
print(d)
d = dict.fromkeys(('name', 'age', 'hight'), 'unknown')
print(d)
d = dict.fromkeys("abc", 'unknown')
print(d)
d = dict.fromkeys(range(1,11), 'unknown')
print(d)
d = dict.fromkeys(['name', 'age'], ['unknown', 'unknown'])
print(d)

Output - {'name': 'unknown', 'age': 'unknown', 'hight': 'unknown'}
{'name': 'unknown', 'age': 'unknown', 'hight': 'unknown'}
{'a': 'unknown', 'b': 'unknown', 'c': 'unknown'}
{1: 'unknown', 2: 'unknown', 3: 'unknown', 4: 'unknown', 5: 'unknown', 6: 'unknown', 7: 'unknown', 8: 'unknown', 9: 'unknown', 10: 'unknown'}
{'name': ['unknown', 'unknown'], 'age': ['unknown', 'unknown']}



get() method - यदि हम को ऐसे किसी key को Access करना चाहे जो dictionary में है ही नहीं तो normal तरीका का use करने पर error आयेगा पर get method का use करें तो error नहीं आयेगा बल्कि None Message print होगा।।

Syntax- dictionary variable.get(key)

```
d={'name':'Satyam','age':20,'hight':'unknown'}
print(d['name'])
# print(d['names'])          ~ Error.
print(d.get('name'))
print(d.get('names'))
if 'names' in d:
    print('present')
else:
    print('not present')
if d.get('names'):
    print('present')
else:
    print('not present')
```

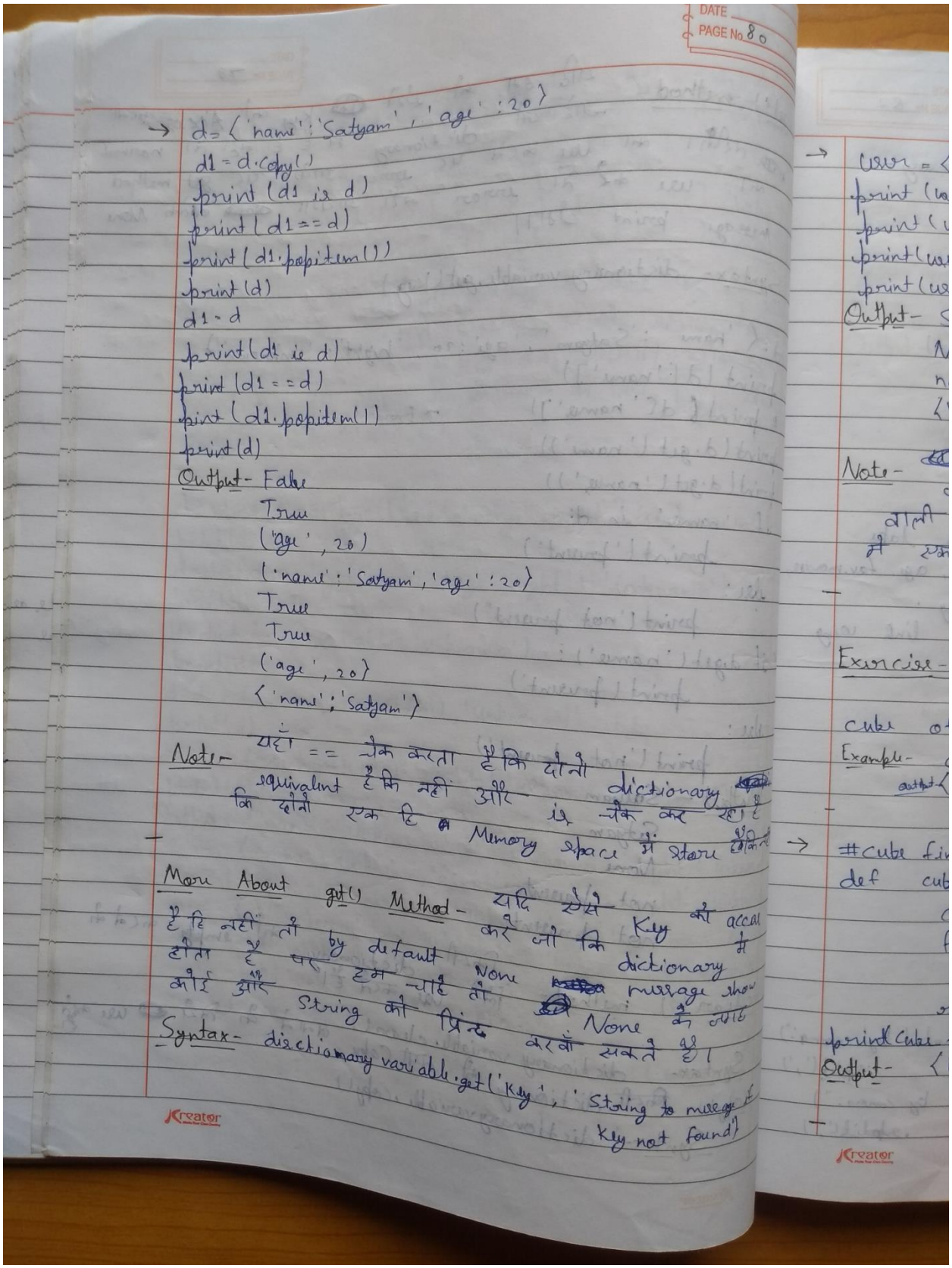
Output - Satyam
Satyam
None
not present
not present

clear() method - किसी dictionary को empty करने के लिये use करते हैं।

Syntax- dictionary variable.clear()

copy() - किसी dictionary की नयी Copy बनाने के लिये use करते हैं

Syntax- dictionary variable.copy()



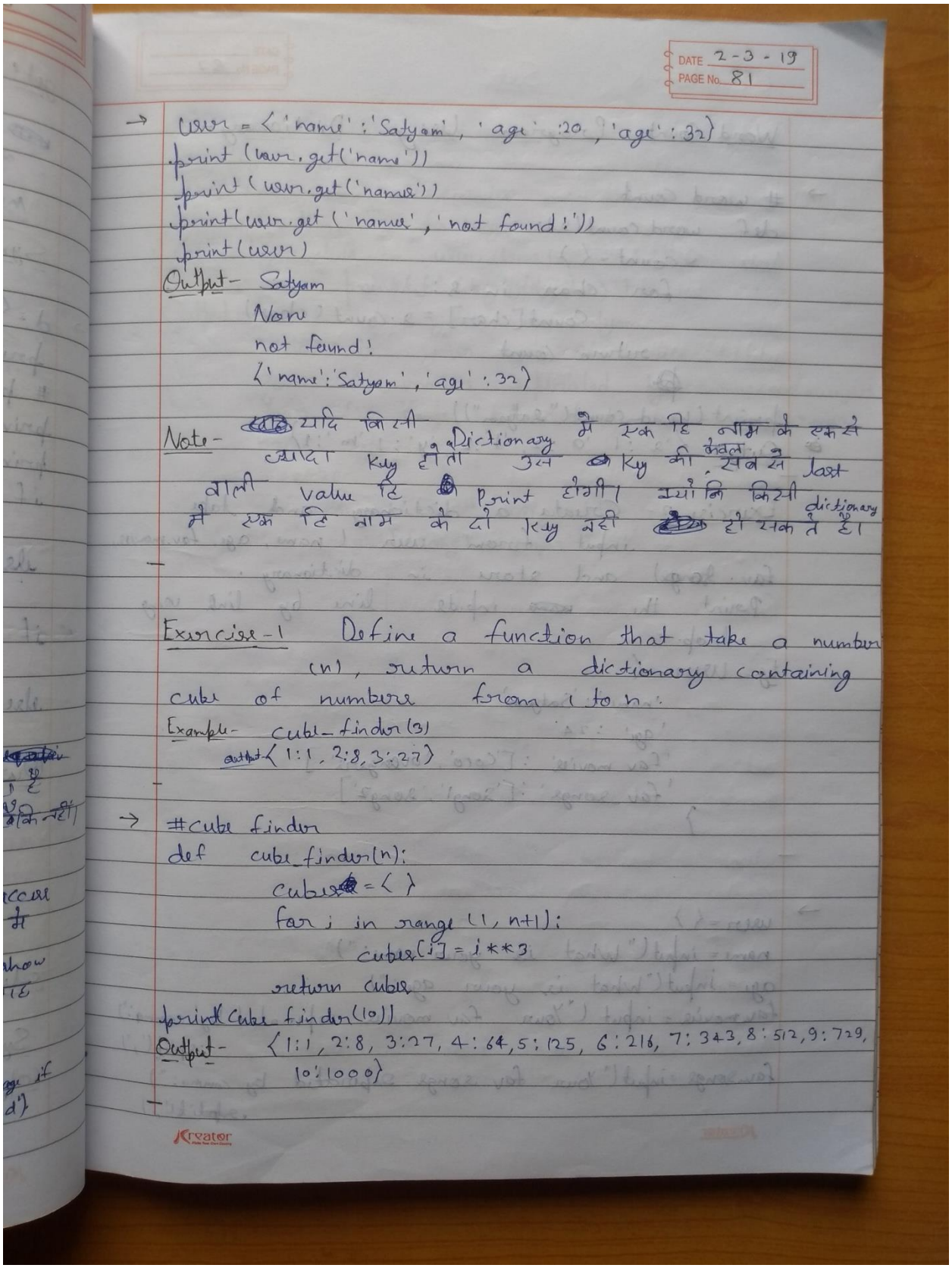
```
d={'name':'Satyam','age':20}
d1=d.copy()
print(d1 is d)
print(d1==d)
print(d1.popitem())
print(d)
d1=d
print(d1 is d)
print(d1==d)
print(d1.popitem())
print(d)
```

Output- False
True
('age',20)
{'name':'Satyam','age':20}
True
True
('age',20)
{'name':'Satyam'}

Note- यहाँ == चेक करता है कि दोनों dictionary equivalent है कि नहीं और is चेक करता है कि दोनों एक ही Memory space में store होकर

More About get() Method- यदि ऐसे Key को access कर लें कि dictionary में है कि नहीं तो by default None message show होता है पर हम चाहें तो None के जगह कोई और String को प्रिंट करवा सकते हैं।

Syntax- disctionary variable.get('Key','String to message if Key not found')



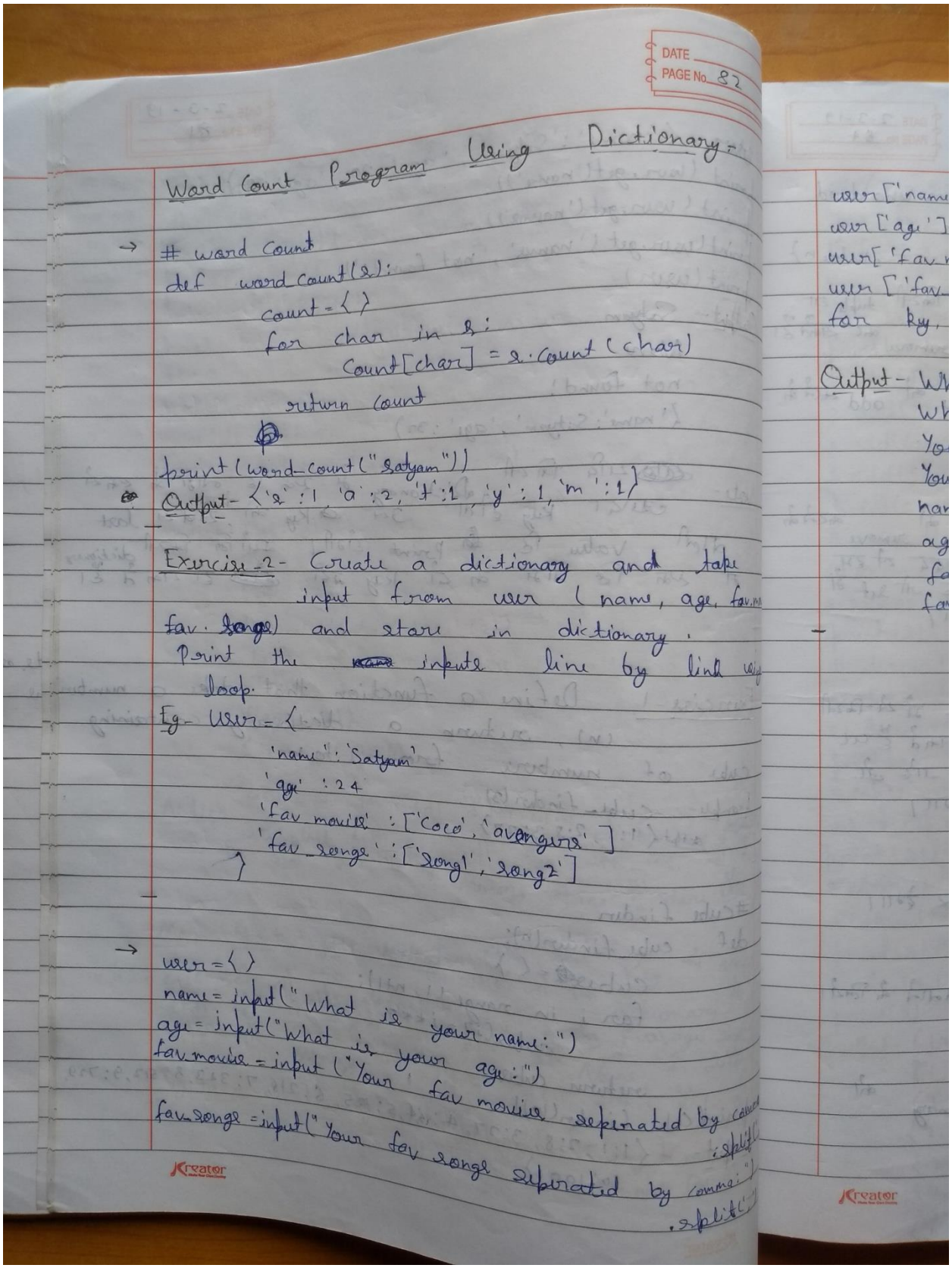
DATE 2-3-19
PAGE No. 81

→
```
user = {'name':'Satyam', 'age':20, 'age':32}
print(user.get('name'))
print(user.get('names'))
print(user.get('names','not found!'))
print(user)
```
Output- Satyam
None
not found!
{'name':'Satyam', 'age':32}

Note- यदि किसी Dictionary में एक ही नाम के एक से ज्यादा key होती उस key की केवल एक से last वाली value ही print होगी। क्योंकि किसी dictionary में एक ही नाम के दो key नहीं हो सकते है।

Exercise-1 Define a function that take a number (n), return a dictionary containing cube of numbers from 1 to n.
Example- cube_finder(3)
output {1:1, 2:8, 3:27}

→
```
#cube finder
def cube_finder(n):
    cubes = {}
    for i in range(1, n+1):
        cubes[i] = i**3
    return cubes
print(Cube_finder(10))
```
Output- {1:1, 2:8, 3:27, 4:64, 5:125, 6:216, 7:343, 8:512, 9:729, 10:1000}



DATE
PAGE No. 82

Word Count Program Using Dictionary :-

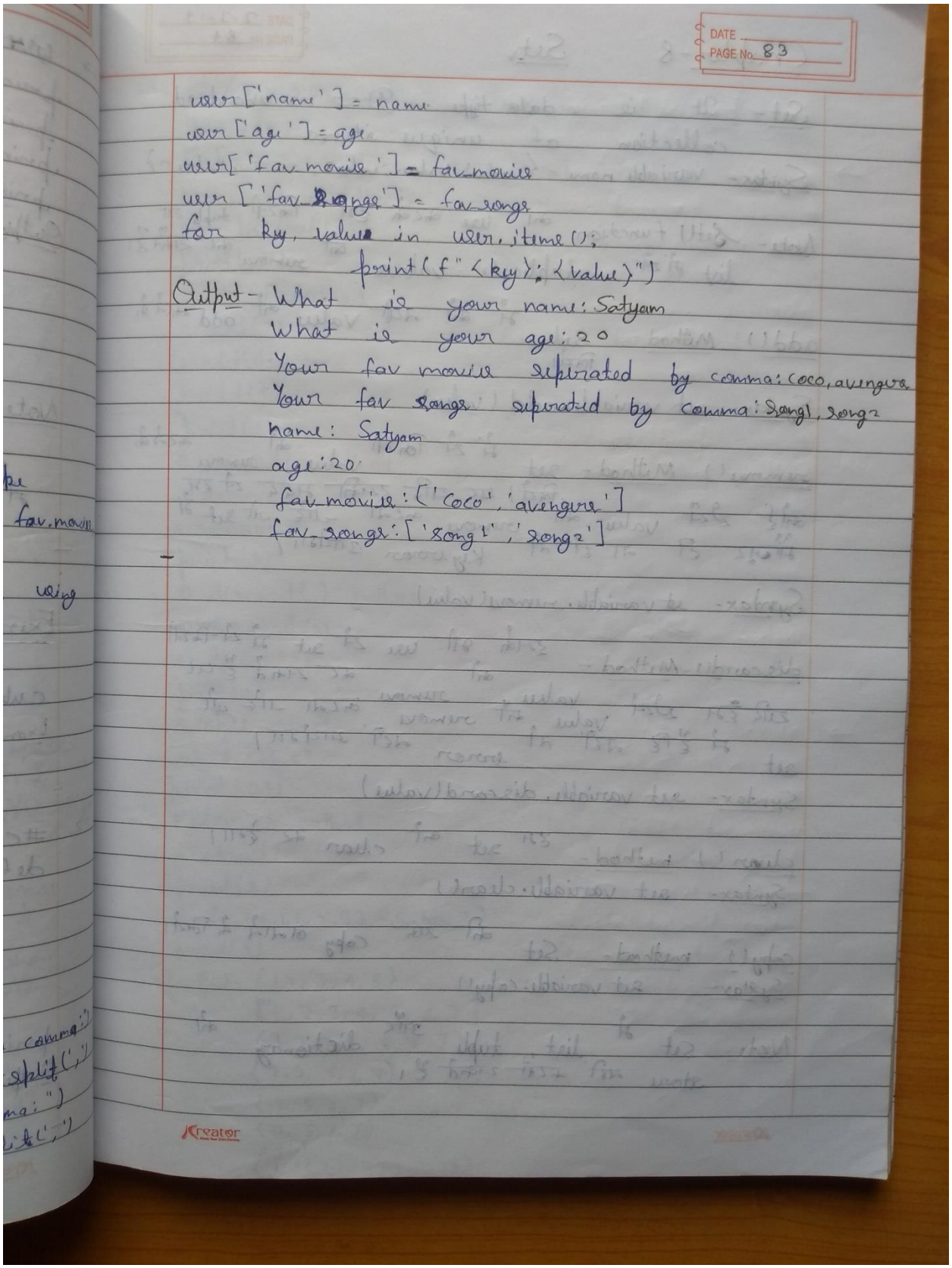
→
```
# word count
def word_count(s):
    count = { }
    for char in s:
        count[char] = s.count(char)
    return count

print(word_count("satyam"))
```

Output - {'s' : 1, 'a' : 2, 't' : 1, 'y' : 1, 'm' : 1}

Exercise - 2 - Create a dictionary and take input from user (name, age, fav. movies, fav. songs) and store in dictionary. Print the inputs line by line using loop.

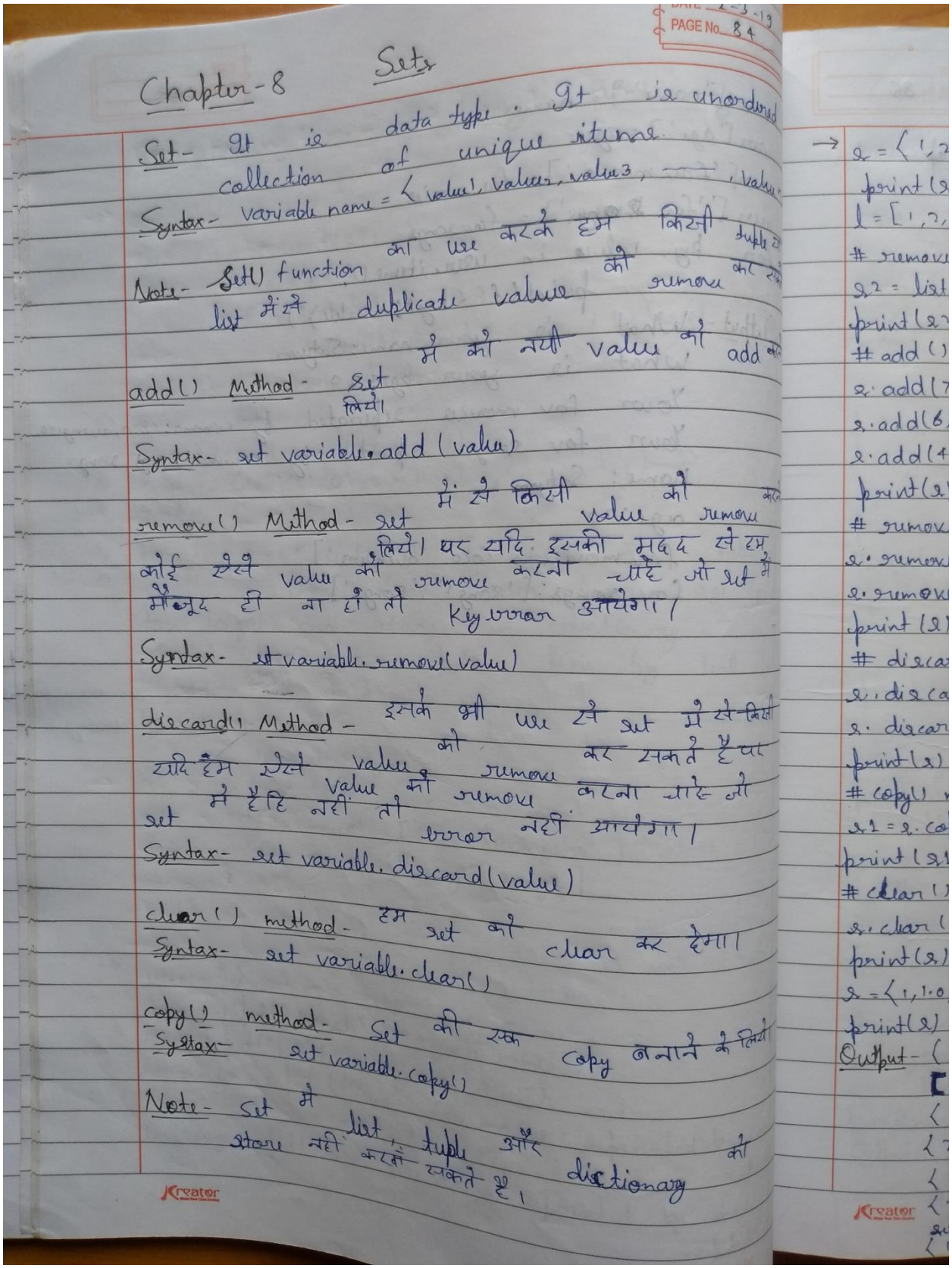
Eg - User = {
    'name' : 'Satyam'
    'age' : 24
    'fav movies' : ['coco', 'avengers']
    'fav_songs' : ['song1', 'song2']
    }

→
```
user = { }
name = input("What is your name:")
age = input("What is your age:")
fav_movies = input("Your fav movies seperated by comma:").split(',')
fav_songs = input("Your fav songs seperated by comma:").split(',')
```



```
user['name'] = name
user['age'] = age
user['fav_movies'] = fav_movies
user['fav_songs'] = fav_songs
for key, value in user.items():
        print(f"{key}: {value}")
```

Output - What is your name: Satyam
What is your age: 20
Your fav movies seperated by comma: coco, avengers
Your fav songs seperated by comma: song1, song2
name: Satyam
age: 20
fav_movies: ['coco', 'avengers']
fav_songs: ['song1', 'song2']



Chapter-8 Sets

Set- It is data type. It is unordered collection of unique items.

Syntax- variable name = { value1, value2, value3, ——, valueN }

Note- Set() function का use करके हम किसी tuple या list में से duplicate value को remove कर सकते हैं।

add() Method- Set में नयी value को add करने के लिये।

Syntax- set variable.add(value)

remove() Method- set में से किसी value को remove करने के लिये। पर यदि इसको प्रयोग से हम कोई ऐसे value को remove करना चाहें जो set में मौजूद ही ना हो तो KeyError आयेगा।

Syntax- set variable.remove(value)

discard() Method- इसके भी use से set में से किसी को remove कर सकते हैं पर यदि हम ऐसे value को remove करना चाहें जो set में है ही नहीं तो error नहीं आयेगा।

Syntax- set variable.discard(value)

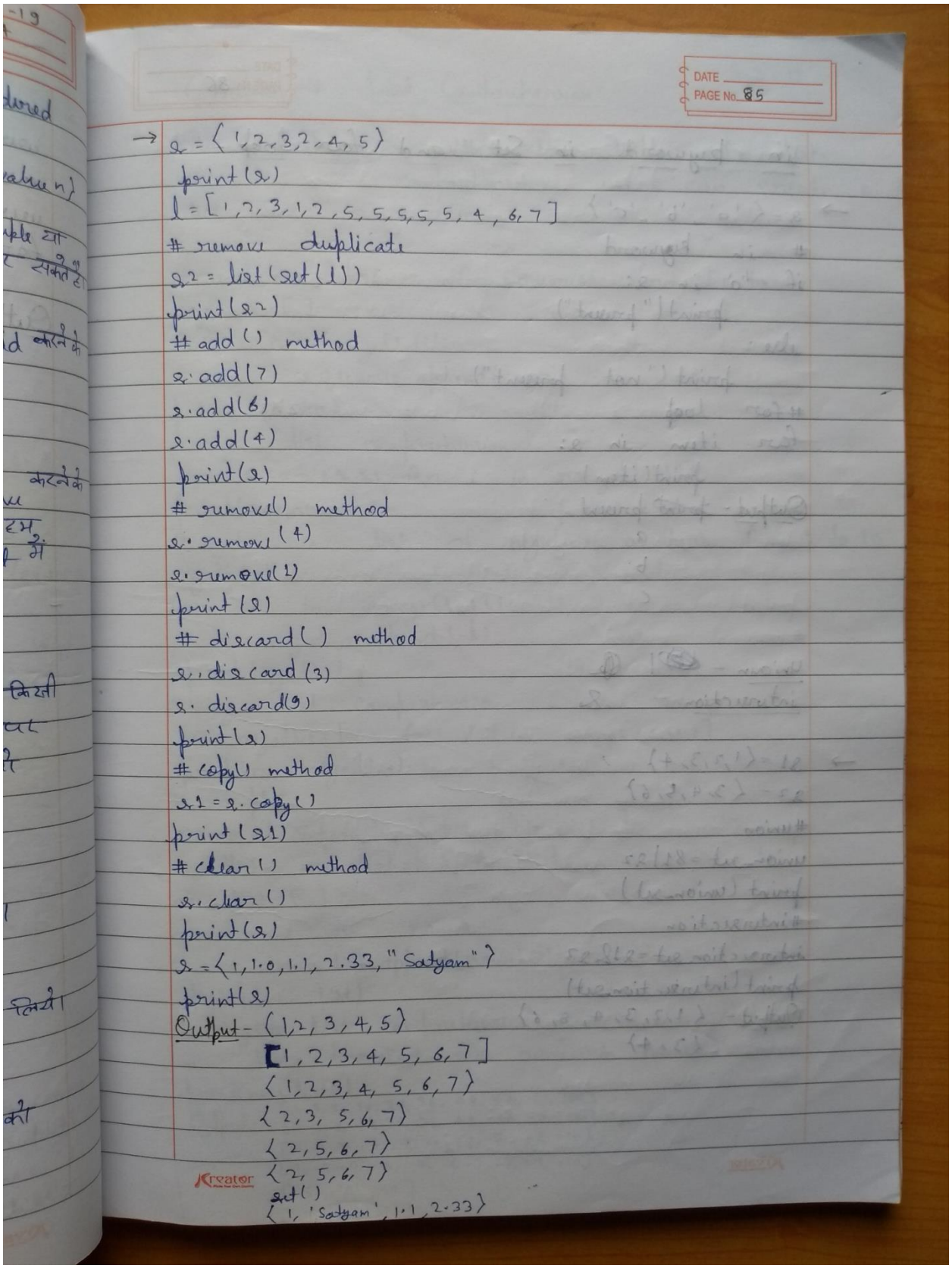
clear() method- हम set को clear कर देगा।

Syntax- set variable.clear()

copy() method- Set की एक copy बनाने के लिये

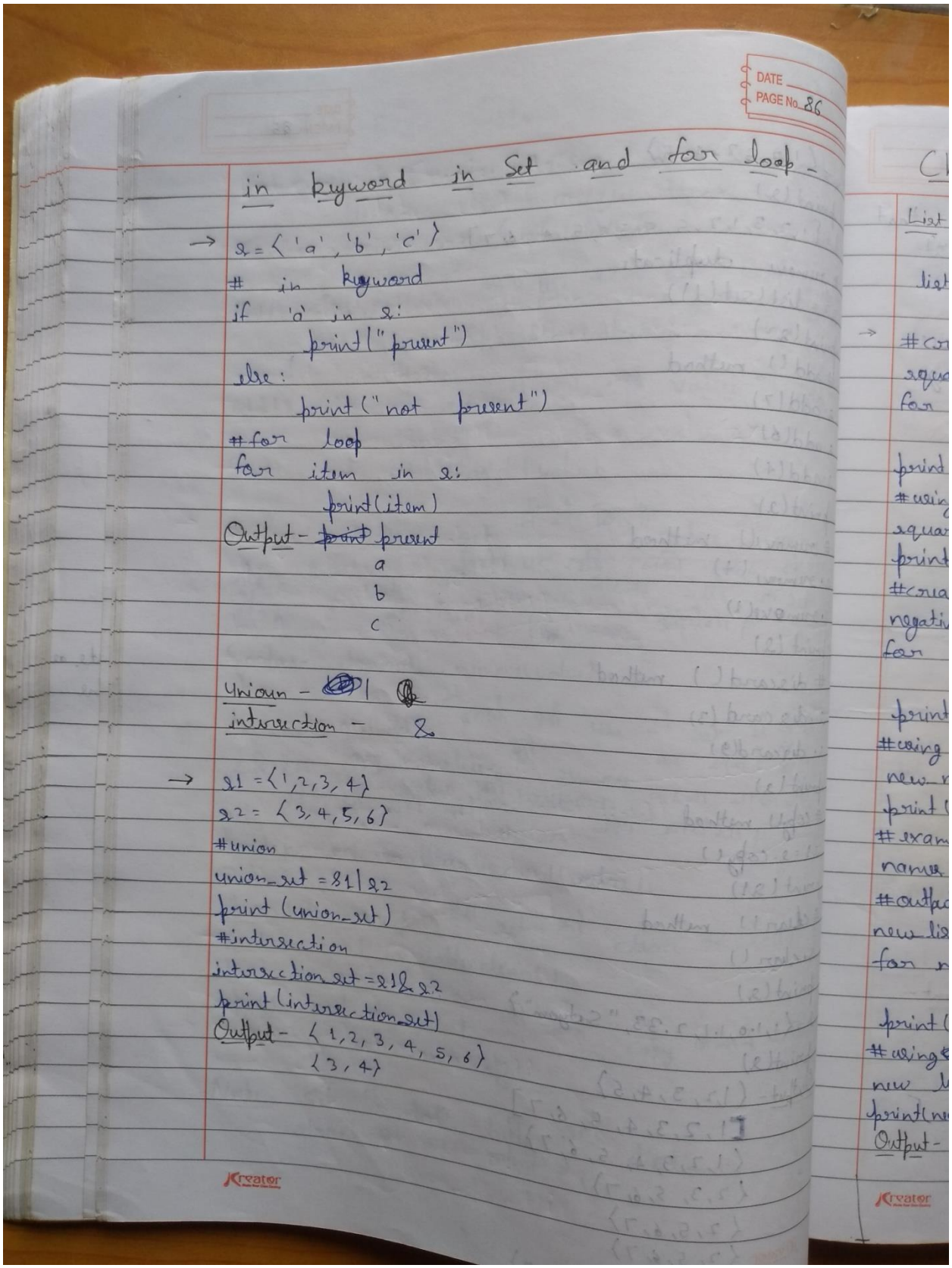
Syntax- set variable.copy()

Note- Set में list, tuple और dictionary को store नहीं कर सकते हैं।



```
s = {1,2,3,2,4,5}
print(s)
l = [1,2,3,1,2,5,5,5,5,5,4,6,7]
# remove duplicate
s2 = list(set(l))
print(s2)
# add() method
s.add(7)
s.add(6)
s.add(4)
print(s)
# remove() method
s.remove(4)
s.remove(1)
print(s)
# discard() method
s.discard(3)
s.discard(9)
print(s)
# copy() method
s1 = s.copy()
print(s1)
# clear() method
s.clear()
print(s)
s = {1,1.0,1.1,2.33,"Satyam"}
print(s)
```

Output -
```
{1,2,3,4,5}
[1,2,3,4,5,6,7]
{1,2,3,4,5,6,7}
{2,3,5,6,7}
{2,5,6,7}
{2,5,6,7}
set()
{1,'Satyam',1.1,2.33}
```

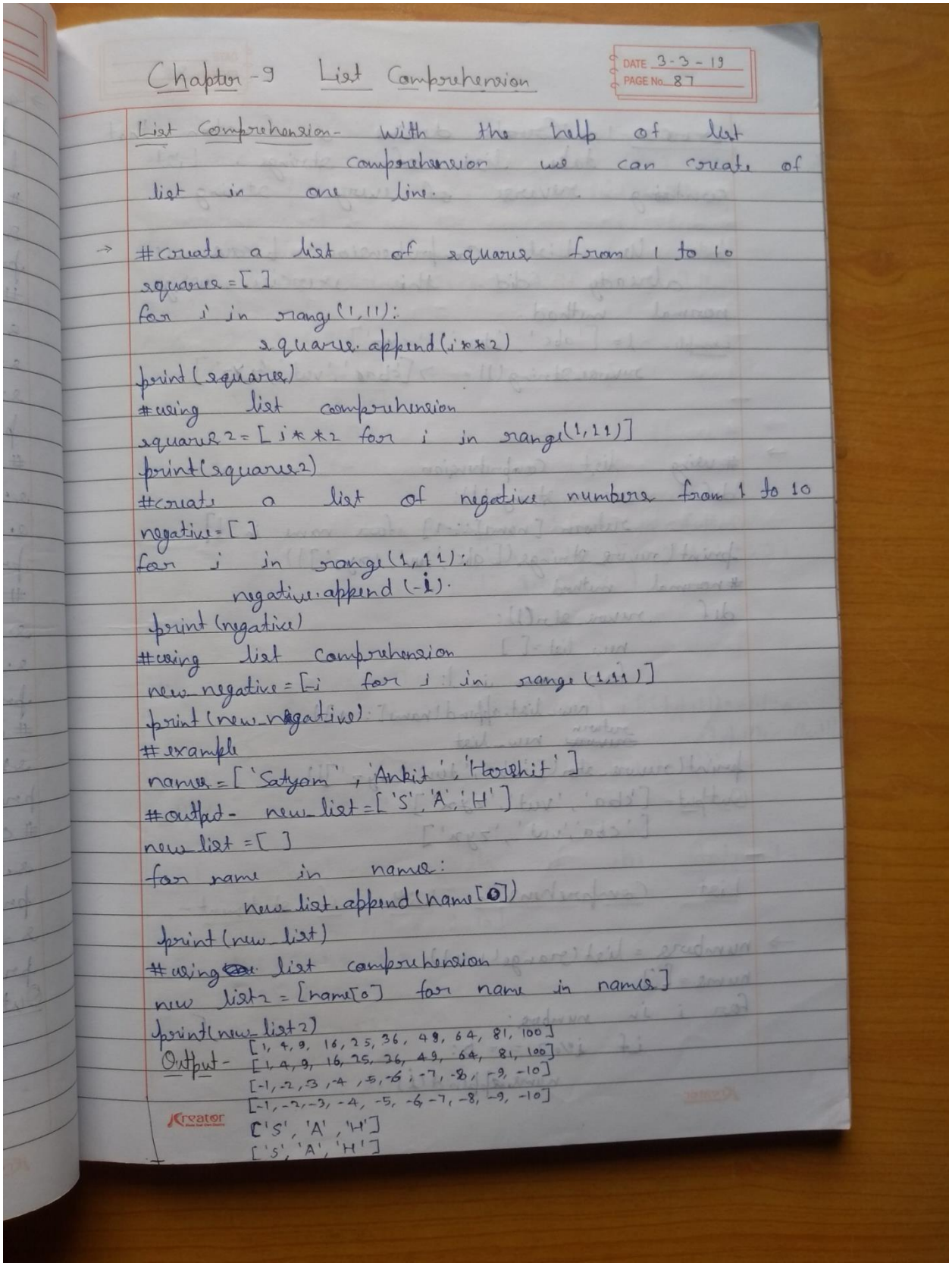


in keyword in Set and for loop -

→ s = {'a', 'b', 'c'}
# in keyword
if 'a' in s:
    print("present")
else:
    print("not present")
#for loop
for item in s:
    print(item)
Output - present
a
b
c

Union - |
intersection - &

→ s1 = {1,2,3,4}
s2 = {3,4,5,6}
#union
union_set = s1|s2
print(union_set)
#intersection
intersection_set = s1&s2
print(intersection_set)
Output - {1,2,3,4,5,6}
{3,4}



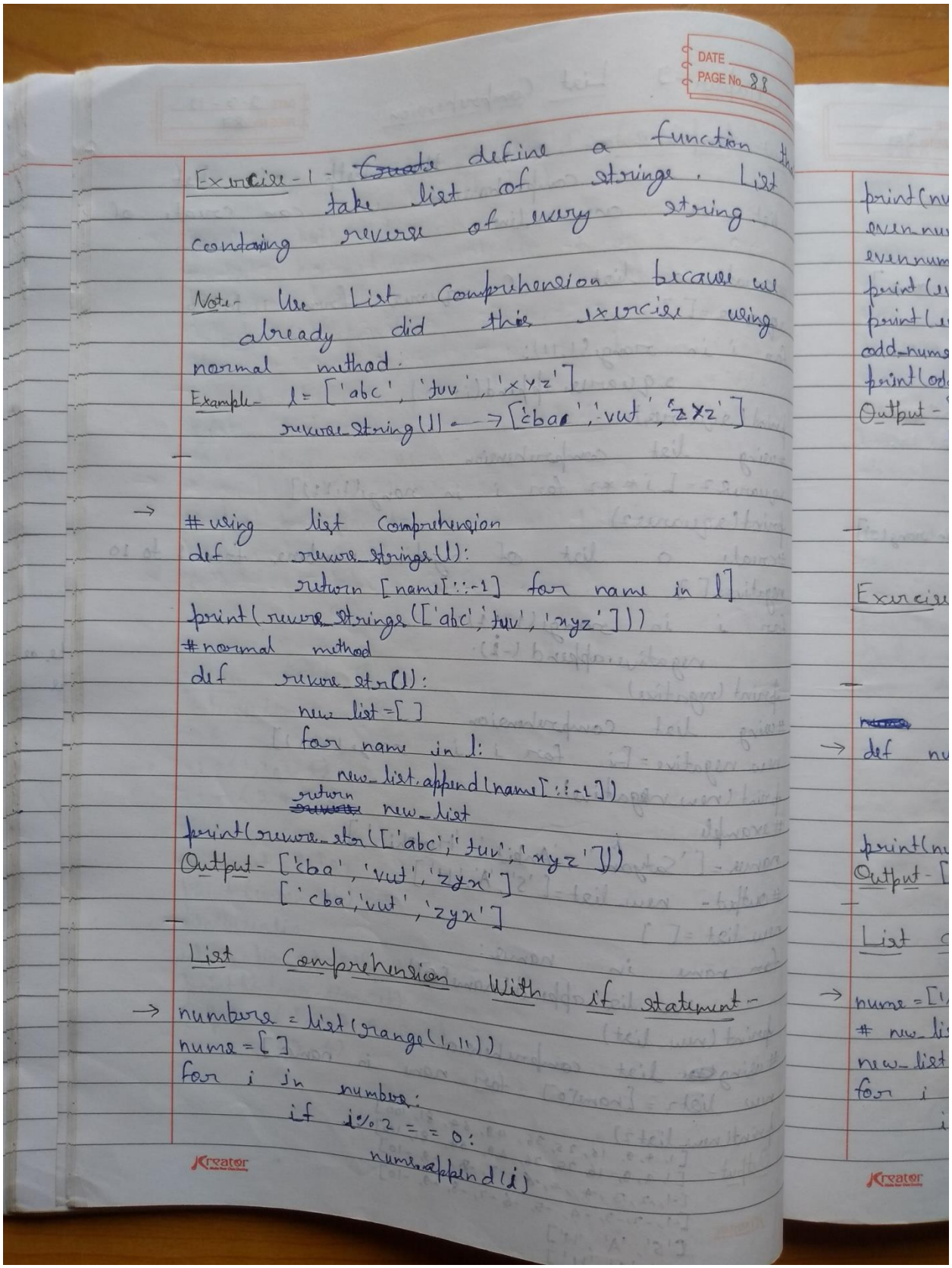
Chapter - 9 List Comprehension

DATE 3-3-19
PAGE No. 87

List Comprehension - With the help of list comprehension we can create of list in one line.

→
```
#create a list of squares from 1 to 10
squares = [ ]
for i in range(1,11):
        squares.append(i**2)
print(squares)
#using list comprehension
squares2 = [i**2 for i in range(1,11)]
print(squares2)
#create a list of negative numbers from 1 to 10
negative = [ ]
for i in range(1,11):
        negative.append(-i)
print(negative)
#using list comprehension
new_negative = [-i for i in range(1,11)]
print(new_negative)
#example
names = ['Satyam', 'Ankit', 'Harshit']
#output - new_list = ['S', 'A', 'H']
new_list = [ ]
for name in names:
        new_list.append(name[0])
print(new_list)
#using list comprehension
new_list2 = [name[0] for name in names]
print(new_list2)
```

Output -
```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
[-1, -2, -3, -4, -5, -6, -7, -8, -9, -10]
[-1, -2, -3, -4, -5, -6, -7, -8, -9, -10]
['S', 'A', 'H']
['S', 'A', 'H']
```



Exercise-1 - Create define a function that take list of strings. List containing reverse of every string.

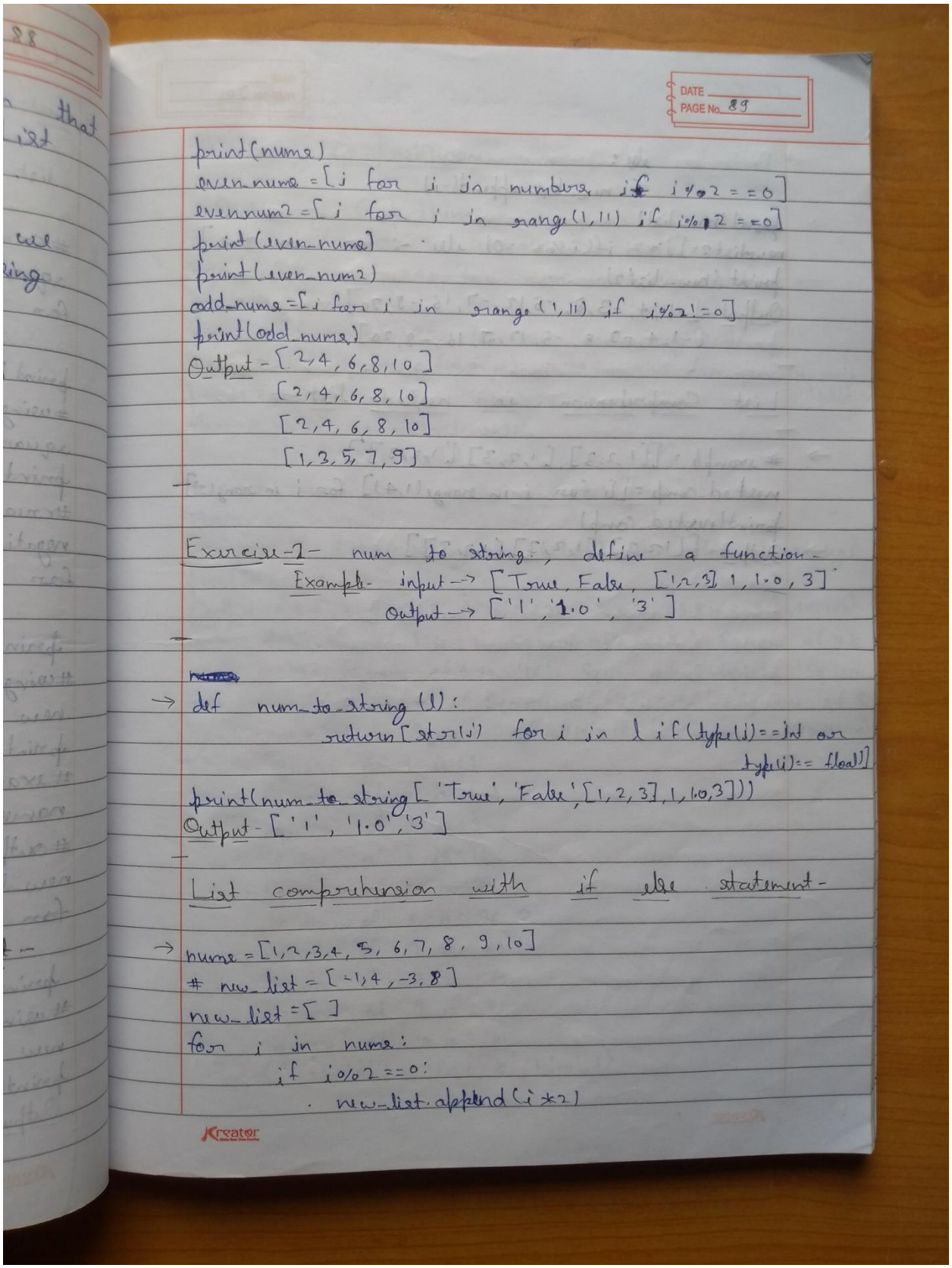
Note- Use List Comprehension because we already did this exercise using normal method.

Example- l = ['abc', 'tuv', 'xyz']
reverse_string(l) ---> ['cba', 'vut', 'zxz']

→ # using list comprehension
def reverse_strings(l):
    return [name[::-1] for name in l]
print(reverse_strings(['abc', 'tuv', 'xyz']))
# normal method
def reverse_str(l):
    new_list = []
    for name in l:
        new_list.append(name[::-1])
    return new_list
print(reverse_str(['abc', 'tuv', 'xyz']))
Output- ['cba', 'vut', 'zyx']
['cba', 'vut', 'zyx']

List Comprehension With if statement -

→ numbers = list(range(1, 11))
nums = []
for i in numbers:
    if i%2 == 0:
        nums.append(i)



```
print(nums)
even_nums = [i for i in numbers if i%2 == 0]
even_num2 = [i for i in range(1,11) if i%2 == 0]
print(even_nums)
print(even_num2)
odd_nums = [i for i in range(1,11) if i%2 != 0]
print(odd_nums)
```

Output - [2,4,6,8,10]
[2,4,6,8,10]
[2,4,6,8,10]
[1,3,5,7,9]

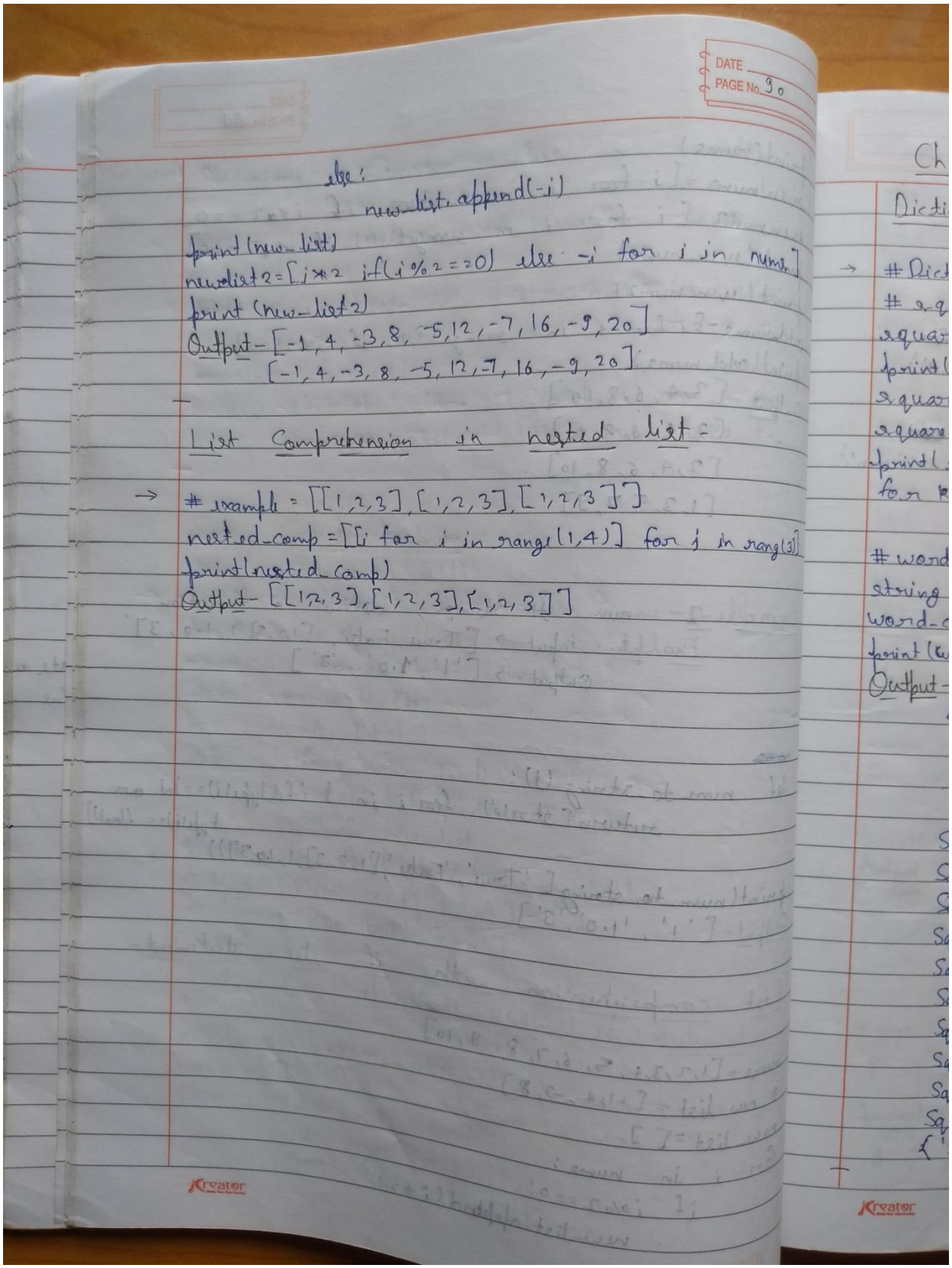
Exercise-1- num to string, define a function-
Example- input --> [True, False, [1,2,3], 1, 1.0, 3]
Output --> ['1', '1.0', '3']

```
def num_to_string(l):
    return [str(i) for i in l if (type(i)==int or type(i)== float)]
print(num_to_string['True', 'False', [1,2,3], 1, 1.0, 3]))
```

Output - ['1', '1.0', '3']

List comprehension with if else statement-

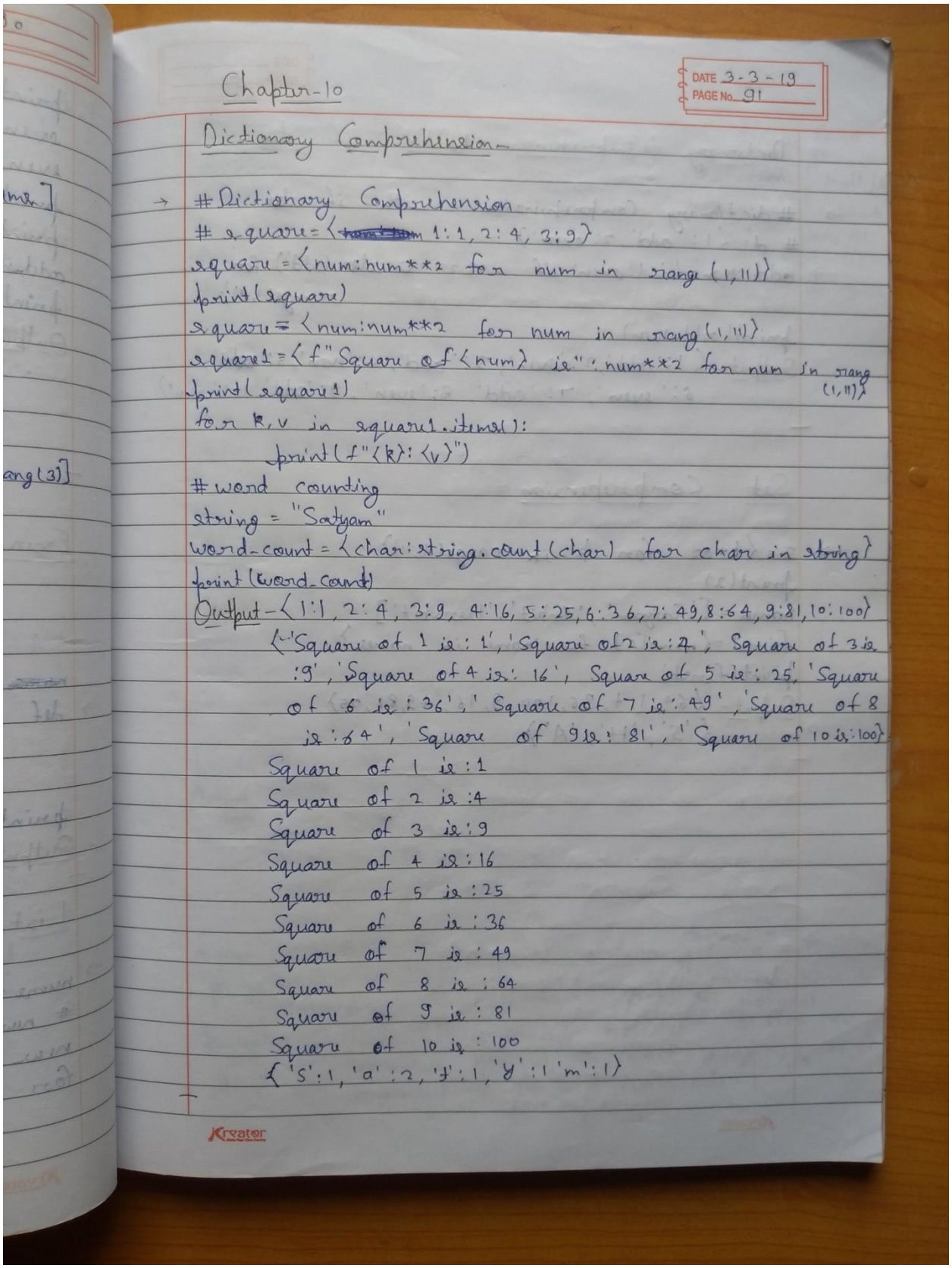
```
nums = [1,2,3,4,5,6,7,8,9,10]
# new_list = [-1,4,-3,8]
new_list = [ ]
for i in nums:
    if i%2 == 0:
        new_list.append(i*2)
```



else:
    new_list.append(-i)
print(new_list)
newlist2=[i*2 if(i%2==0) else -i for i in nums]
print(new_list2)
Output- [-1,4,-3,8,-5,12,-7,16,-9,20]
[-1,4,-3,8,-5,12,-7,16,-9,20]

List Comprehension in nested list:

# example = [[1,2,3],[1,2,3],[1,2,3]]
nested_comp=[[i for i in range(1,4)] for j in range(3)]
print(nested_comp)
Output- [[1,2,3],[1,2,3],[1,2,3]]



Chapter-10

DATE 3-3-19
PAGE No. 91

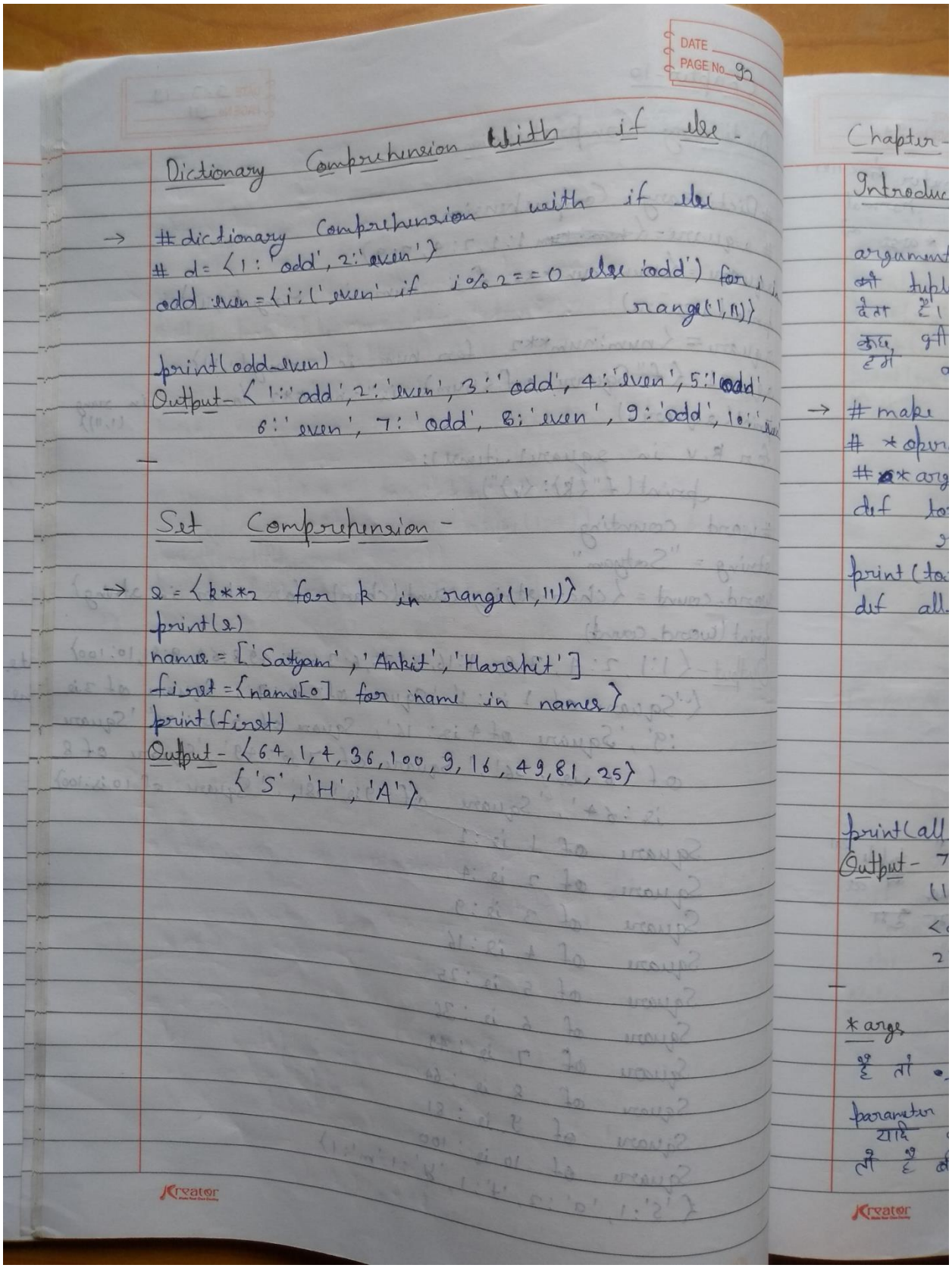
Dictionary Comprehension-

```
# Dictionary Comprehension
# square = {1:1, 2:4, 3:9}
square = {num:num**2 for num in range(1,11)}
print(square)
square = {num:num**2 for num in range(1,11)}
square1 = {f"Square of {num} is":num**2 for num in range(1,11)}
print(square1)
for k,v in square1.items():
    print(f"{k}:{v}")
# word counting
string = "Satyam"
word_count = {char:string.count(char) for char in string}
print(word_count)
```

Output - {1:1, 2:4, 3:9, 4:16, 5:25, 6:36, 7:49, 8:64, 9:81, 10:100}

{'Square of 1 is:1', 'Square of 2 is:4', Square of 3 is:9', 'Square of 4 is:16', Square of 5 is:25', 'Square of 6 is:36', 'Square of 7 is:49', 'Square of 8 is:64', 'Square of 9 is:81', 'Square of 10 is:100}

Square of 1 is:1
Square of 2 is:4
Square of 3 is:9
Square of 4 is:16
Square of 5 is:25
Square of 6 is:36
Square of 7 is:49
Square of 8 is:64
Square of 9 is:81
Square of 10 is:100
{'S':1, 'a':2, 't':1, 'y':1 'm':1}



Dictionary Comprehension with if else -

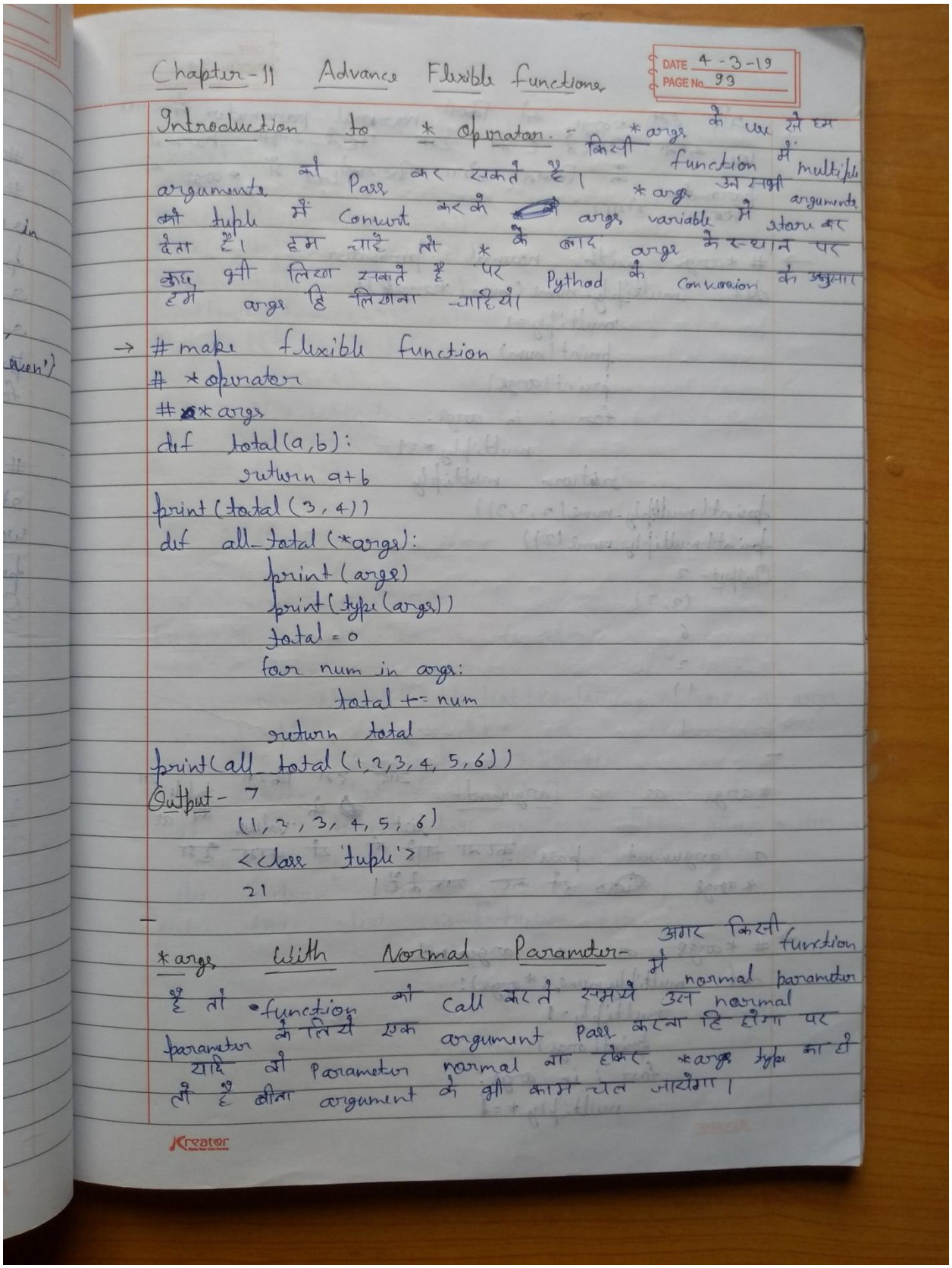
→ # dictionary Comprehension with if else
# d = {1: 'odd', 2: 'even'}
odd_even = {i: ('even' if i%2==0 else 'odd') for i in range(1,11)}

print(odd_even)
Output - {1: 'odd', 2: 'even', 3: 'odd', 4: 'even', 5: 'odd', 6: 'even', 7: 'odd', 8: 'even', 9: 'odd', 10: ...}

—

Set Comprehension -

→ s = {k**2 for k in range(1,11)}
print(s)
name = ['Satyam', 'Ankit', 'Harshit']
first = {name[0] for name in names}
print(first)
Output - {64, 1, 4, 36, 100, 9, 16, 49, 81, 25}
{'S', 'H', 'A'}



Chapter-11 Advance Flexible functions

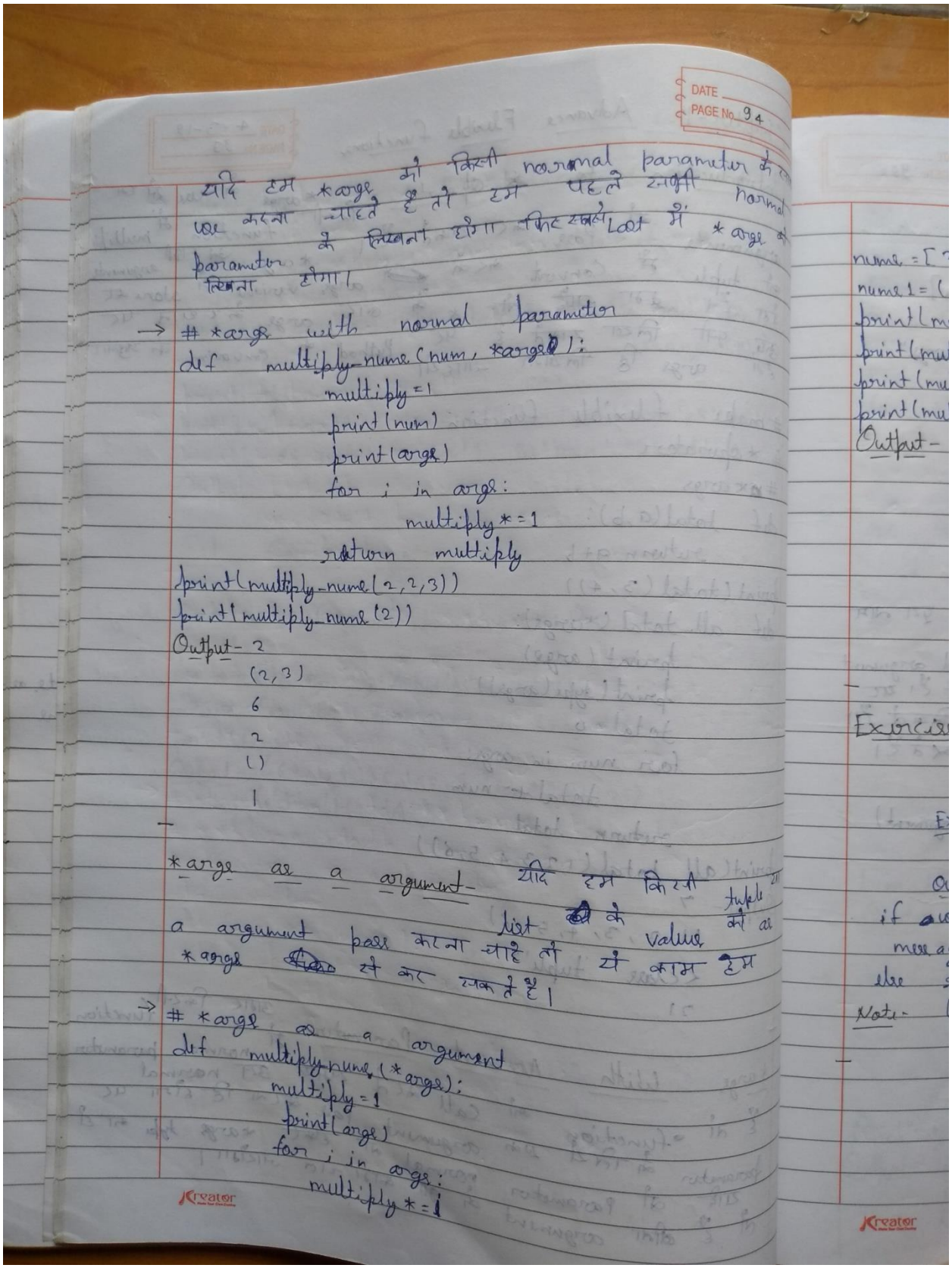
DATE 4-3-19
PAGE No. 93

Introduction to * Operator - किसी function में multiple arguments को pass कर सकते हैं। *args के use से हम उन सभी arguments को tuple में convert कर के args variable में store कर देता है। हम चाहें तो * के बाद args के स्थान पर कुछ भी लिख सकते हैं पर Pythod के convension के अनुसार हमें args ही लिखना चाहिये।

```
→ # make flexible function
  # *operator
  # **args
  def total(a,b):
      return a+b
  print(total(3,4))
  def all_total(*args):
      print(args)
      print(type(args))
      total = 0
      for num in args:
          total += num
      return total
  print(all_total(1,2,3,4,5,6))
```

Output - 7
(1, 2, 3, 4, 5, 6)
<class 'tuple'>
21

*args With Normal Paramter - अगर किसी function में normal paramter है तो function को call करते समय उस normal parameter के लिये एक argument pass करना ही होगा पर यदि वो parameter normal न होकर *args type का है तो हैं कितना argument के भी काम चल जायेगा।

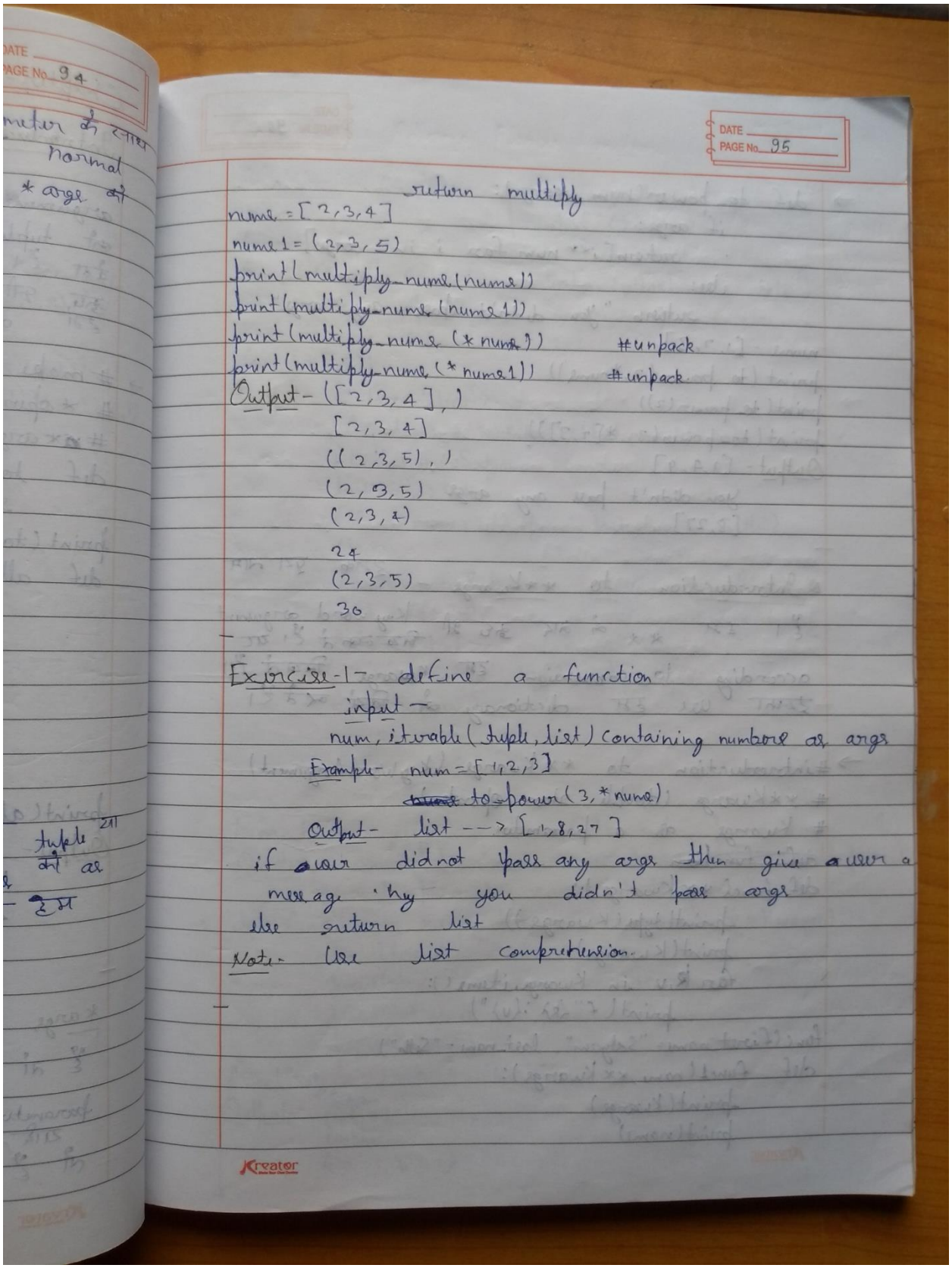


यदि हम *args में किसी normal parameter के use करना चाहते है तो हम पहले उसकी normal parameter के लिखना होगा फिर सबसे Last में *args को लिखना होगा।

→ # *args with normal paramater
def multiply_nums(num, *args):
    multiply = 1
    print(num)
    print(args)
    for i in args:
        multiply *= i
    return multiply
print(multiply_nums(2,2,3))
print(multiply_nums(2))
Output- 2
(2,3)
6
2
()
1

*args as a argument- यदि हम किसी list के value को tuple को as a argument pass करना चाहे तो यह काम हम *args से कर सकते है।

→ # *args as a argument
def multiply_nums(*args):
    multiply = 1
    print(args)
    for i in args:
        multiply *= i

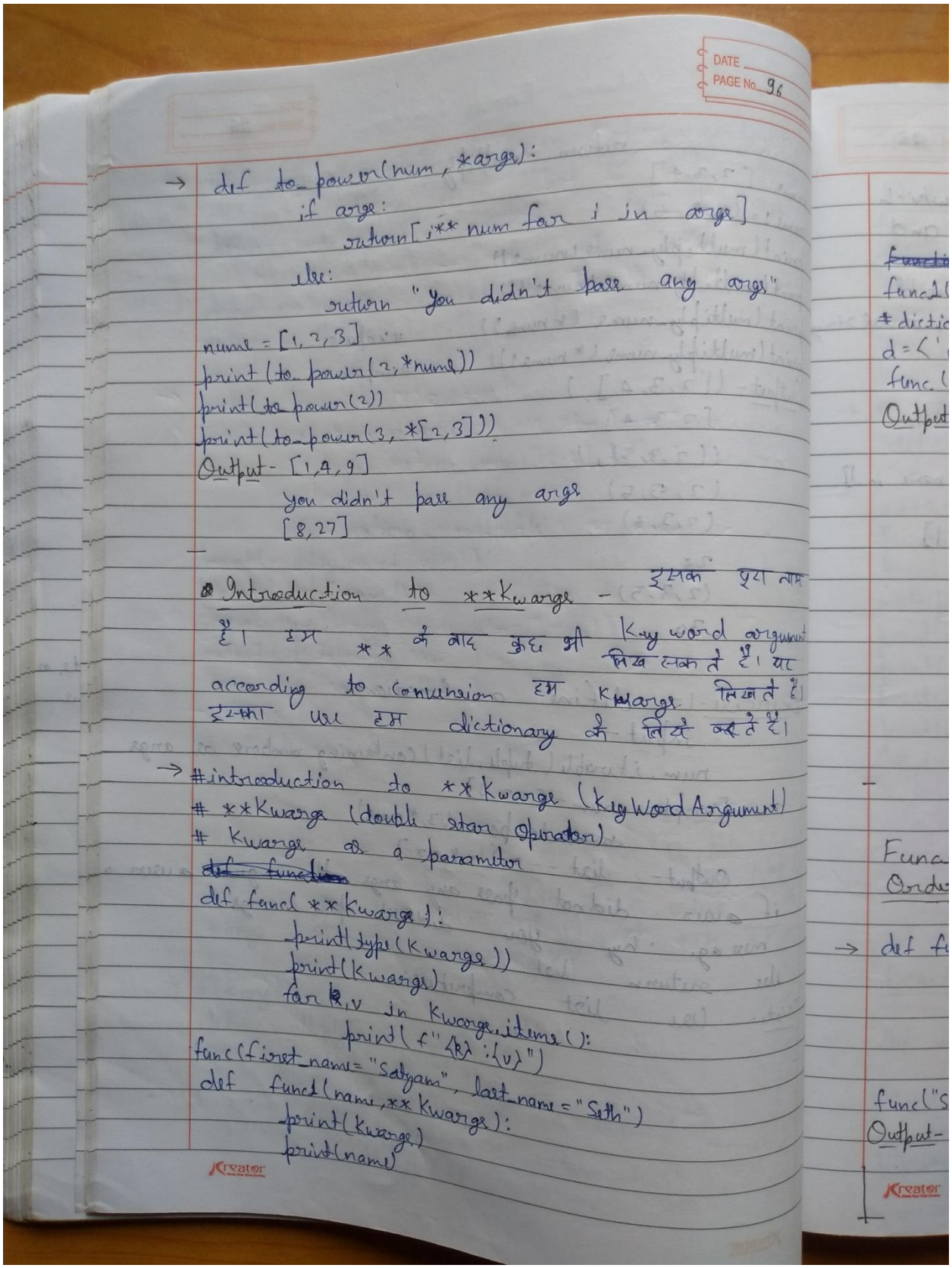


return multiply

```
nums = [2,3,4]
nums1 = (2,3,5)
print(multiply_nums(nums))
print(multiply_nums(nums1))
print(multiply_nums(*nums))      #unpack
print(multiply_nums(*nums1))     #unpack
```

Output - ([2,3,4],)
[2,3,4]
((2,3,5),)
(2,3,5)
(2,3,4)
24
(2,3,5)
30

Exercise-1 - define a function
input -
num, iterable (tuple, list) containing numbers as args
Example - num = [1,2,3]
~~time~~ to_power(3, *nums)
Output - list --> [1,8,27]
if user did not pass any args then give a user a message 'hey you didn't pass args'
else return list
Note - Use list comprehension.

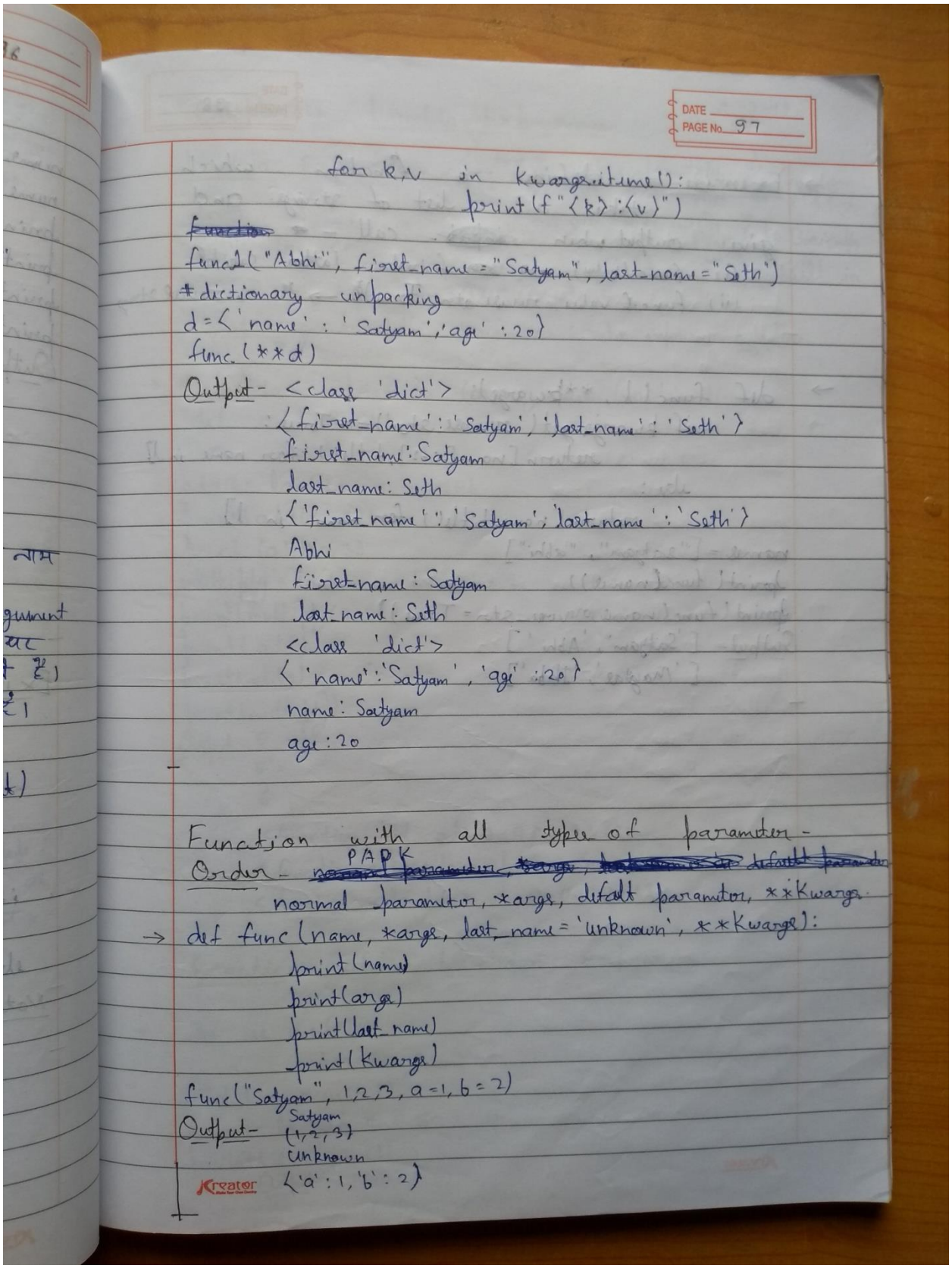


```
def to_power(num, *args):
    if args:
        return [i**num for i in args]
    else:
        return "you didn't pass any args"
nums = [1,2,3]
print(to_power(2,*nums))
print(to_power(2))
print(to_power(3,*[2,3]))
```

Output- [1,4,9]
you didn't pass any args
[8,27]

* Introduction to **kwargs - इसका पूरा नाम है। इस ** के बाद कुछ भी Key word argument लिख सकते हैं। यह according to convension हम kwargs लिखते हैं। इसका use हम dictionary के लिये करते हैं।

```
#introduction to **kwargs (KeyWord Argument)
# **kwargs (double star Operator)
# kwargs as a paramter
def funcl **kwargs ):
    print(type(kwargs))
    print(kwargs)
    for k,v in kwargs.items():
        print(f"{k} :{v}")
funcl(first_name="Satyam", last_name="Seth")
def funcl(name,**kwargs):
    print(kwargs)
    print(name)
```



```
for k,v in kwargs.items():
    print(f"{k}:{v}")
```

~~function~~

```
func1("Abhi", first_name="Satyam", last_name="Seth")
```

* dictionary unpacking

```
d={'name':'Satyam','age':20}
func(**d)
```

Output- <class 'dict'>
{first_name':'Satyam','last_name':'Seth'}
first_name: Satyam
last_name: Seth
{'first_name':'Satyam','last_name':'Seth'}
Abhi
first_name: Satyam
last_name: Seth
<class 'dict'>
{'name':'Satyam','age':20}
name: Satyam
age: 20

Function with all types of paramter -

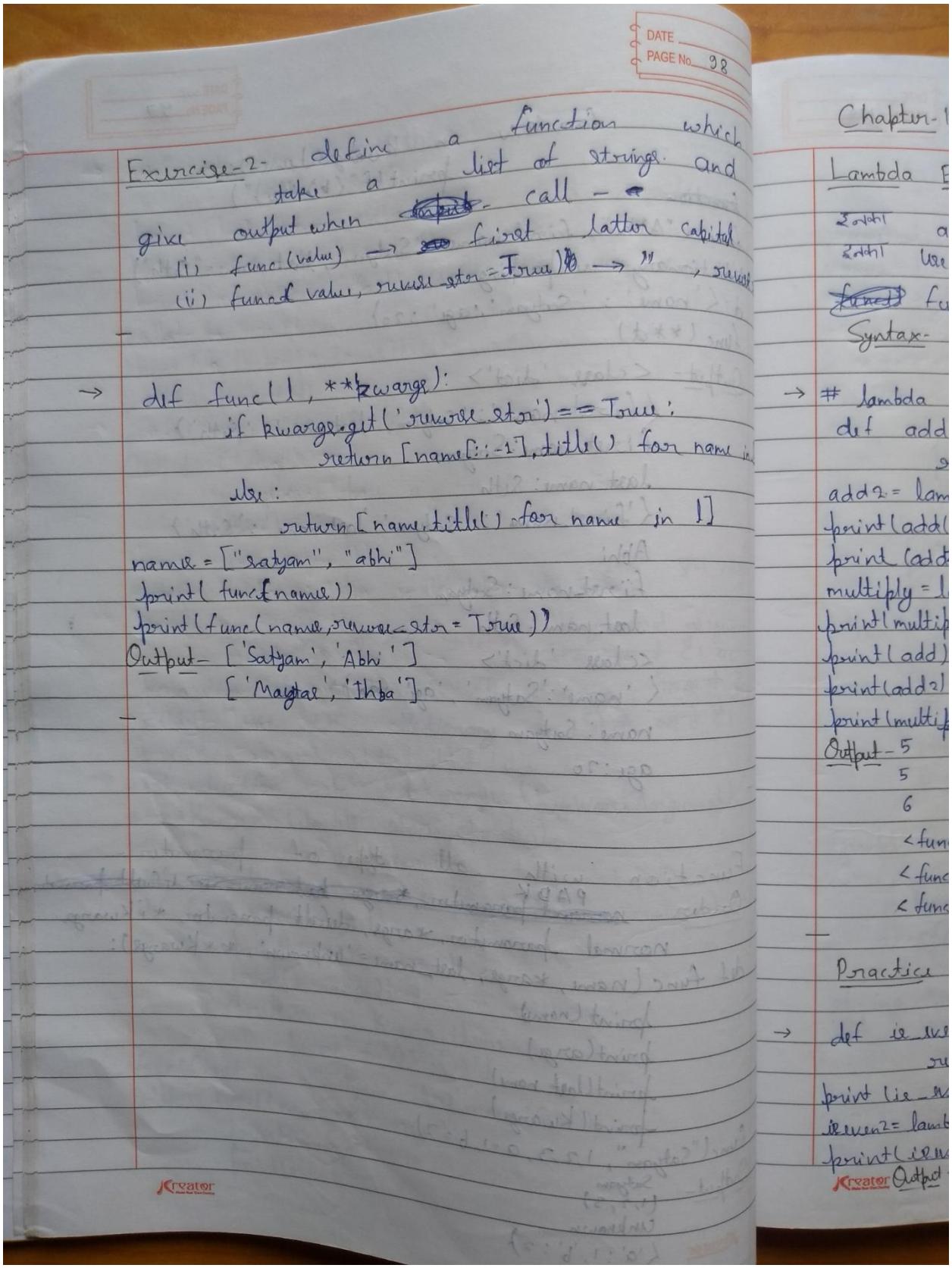
Order - ~~normal paramter, *args, default paramter~~ PAPK

normal paramter, *args, dfault paramter, **kwargs

→

```
def func(name, *args, last_name='unknown', **kwargs):
    print(name)
    print(args)
    print(last_name)
    print(kwargs)
func("Satyam", 1,2,3, a=1, b=2)
```

Output- Satyam
(1,2,3)
unknown
{'a':1,'b':2}



Exercise-2- define a function which take a list of strings and give output when ~~input~~ call -
(i) func(value) → ~~xxx~~ first letter capital.
(ii) func(value, reverse_str=True) → " , reverse

```
def func(l, **kwargs):
    if kwargs.get('reverse_str')==True:
        return [name[::-1].title() for name in l]
    else:
        return [name.title() for name in l]
name=["satyam", "abhi"]
print(func(name))
print(func(name, reverse_str=True))
```

Output- ['Satyam', 'Abhi']
['Mayrat', 'Ihba']

Chapter-

Lambda

Syntax-

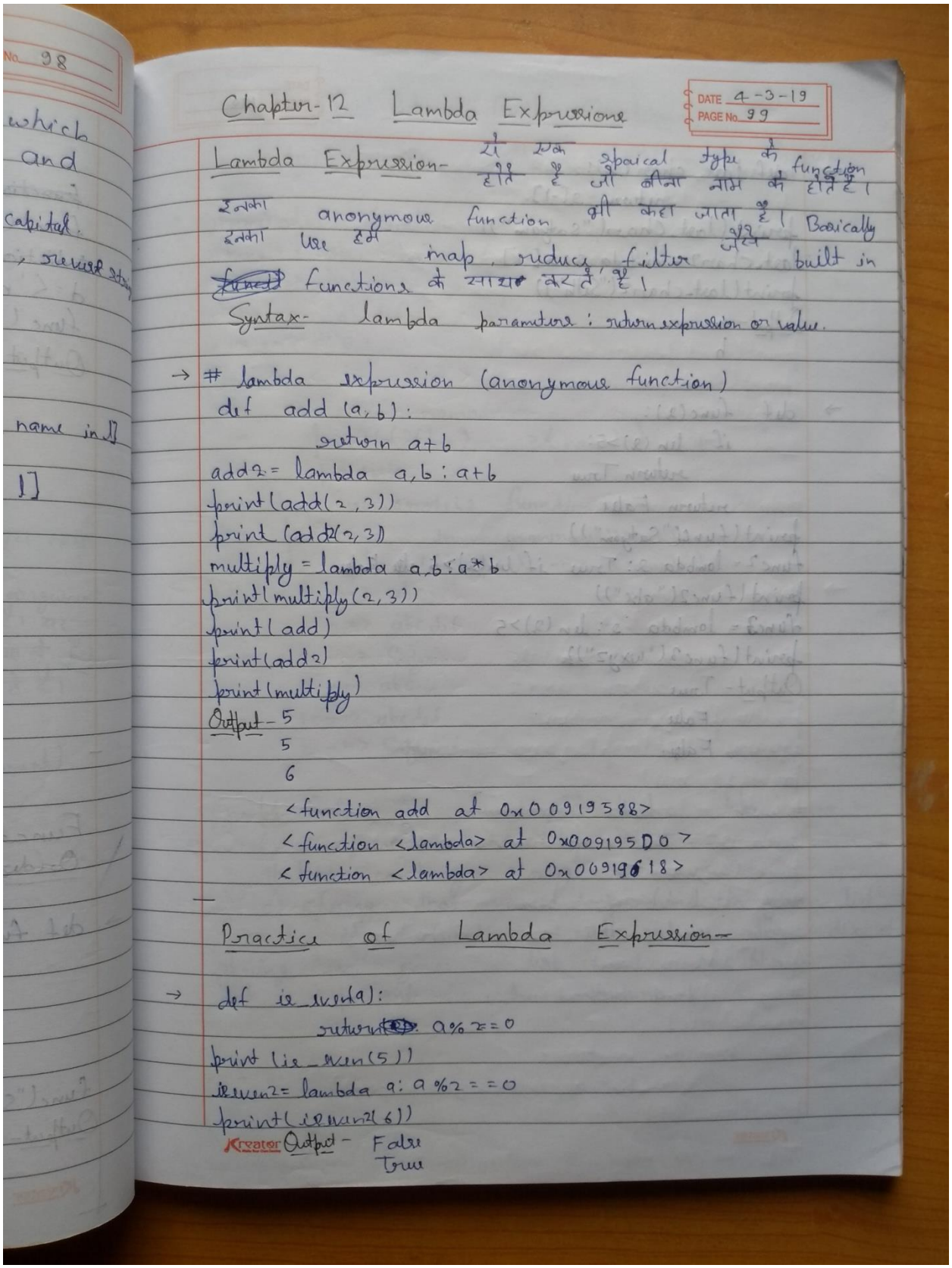
```
# lambda
def add
add2 = lam
print(add(
print(add
multiply = l
print(multip
print(add)
print(add2)
print(multi
```

Output- 5
5
6
< fun
< func
< func

Practice

```
def is_
print(is_
is_even2 = lam
print(is_
```

Output



Chapter-12 Lambda Expressions

DATE 4-3-19
PAGE No. 99

Lambda Expression- ये एक special type के function होते हैं जो बिना नाम के होते हैं। इनको anonymous function भी कहा जाता है। Basically इनका use हम map, reduce, filter जैसे built in functions के साथ करते हैं।

Syntax- lambda paramters : return expression or value.

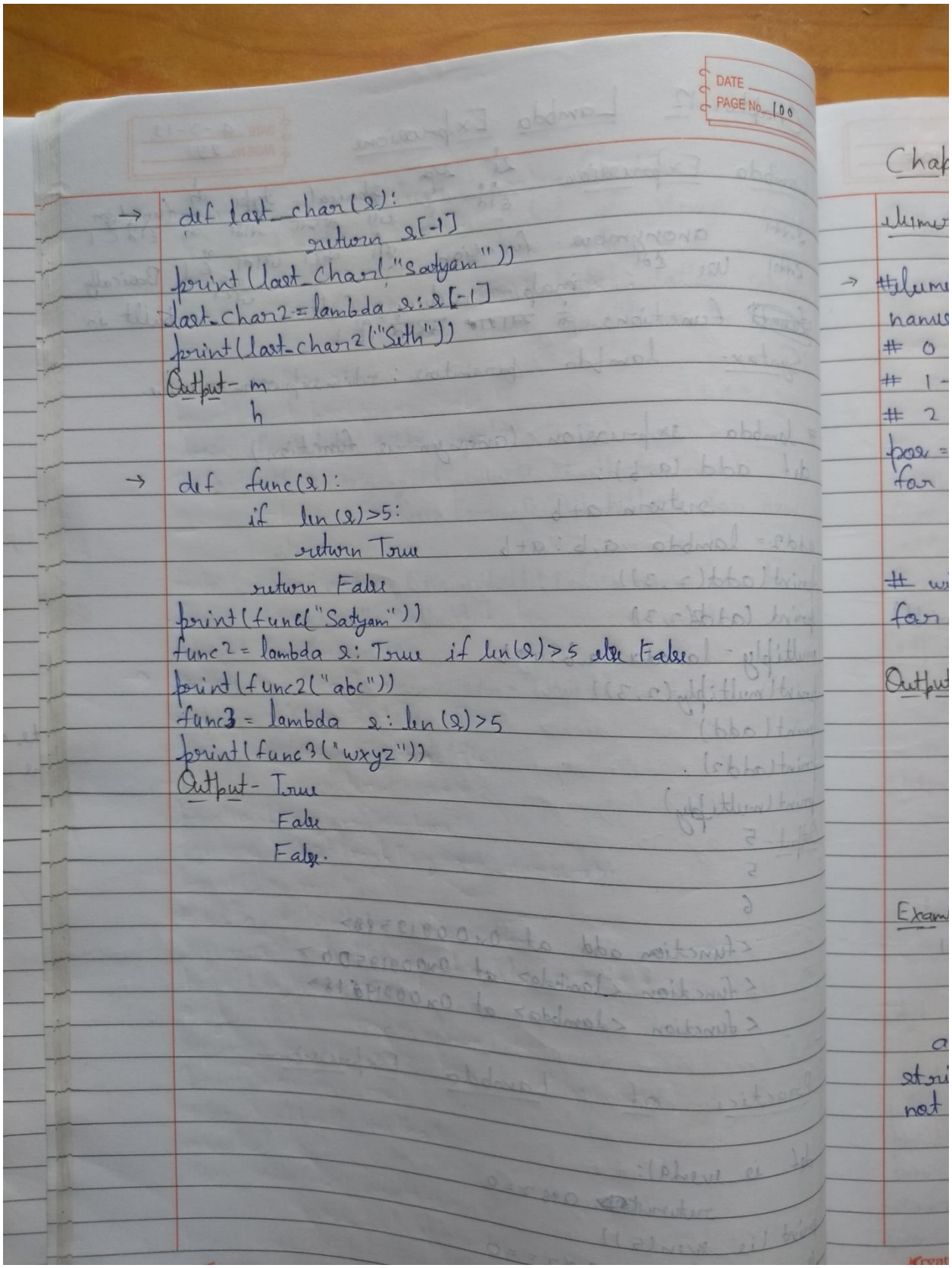
```
# lambda expression (anonymous function)
def add(a,b):
    return a+b
add2 = lambda a,b: a+b
print(add(2,3))
print(add2(2,3))
multiply = lambda a,b: a*b
print(multiply(2,3))
print(add)
print(add2)
print(multiply)
```

Output-
```
5
5
6
<function add at 0x00919588>
<function <lambda> at 0x009195D0>
<function <lambda> at 0x00919618>
```

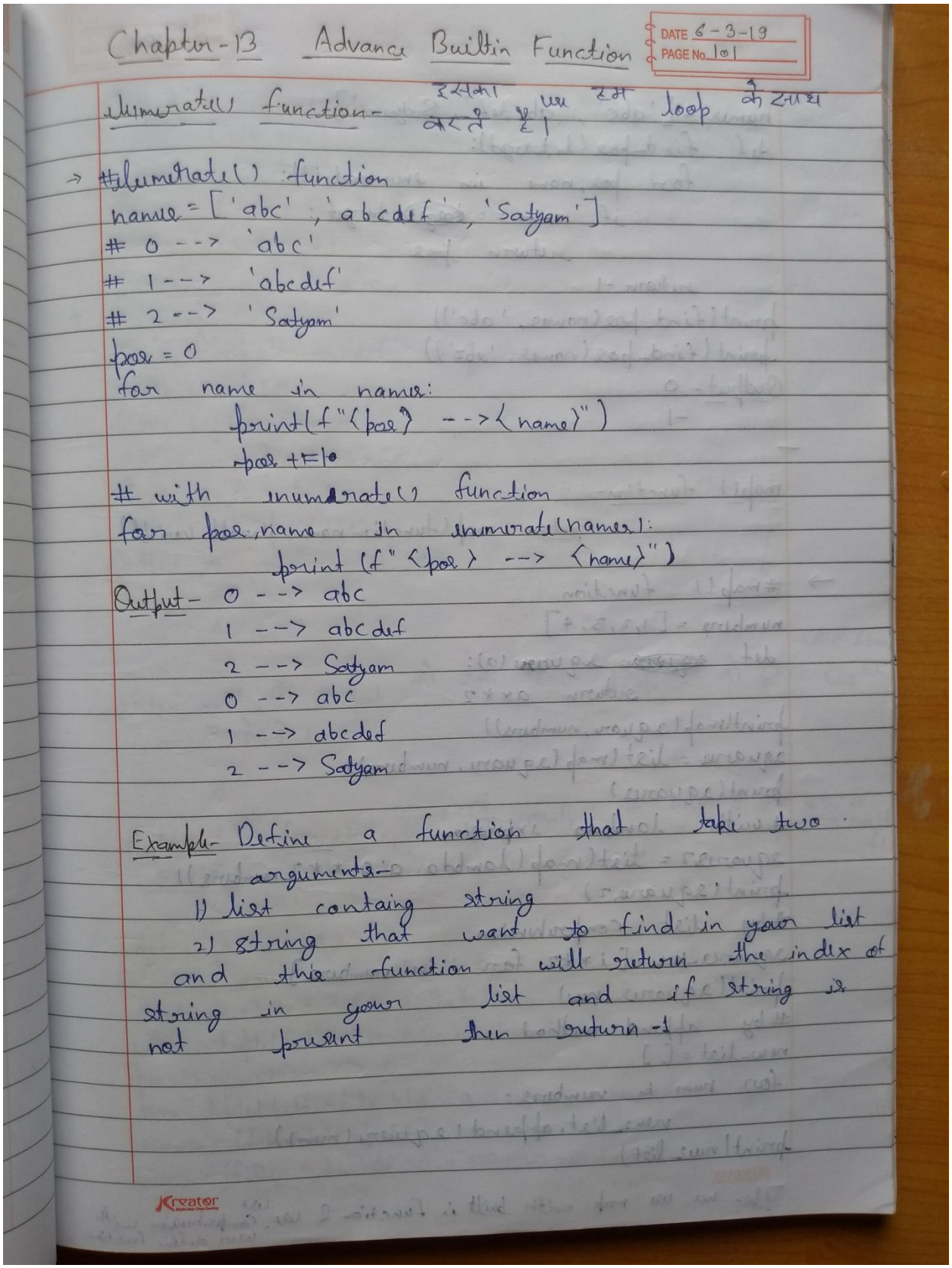
Practice of Lambda Expression-

```
def is_even(a):
    return a%2==0
print(is_even(5))
ieven2 = lambda a: a%2==0
print(ieven2(6))
```

Output- False
True



```
def last_char(s):
    return s[-1]
print(last_char("satyam"))
last_char2 = lambda s: s[-1]
print(last_char2("Seth"))
Output- m
        h

def func(s):
    if len(s)>5:
        return True
    return False
print(func("Satyam"))
func2 = lambda s: True if len(s)>5 else False
print(func2("abc"))
func3 = lambda s: len(s)>5
print(func3("wxyz"))
Output- True
        False
        False.
```



Chapter-13 Advance Builtin Function

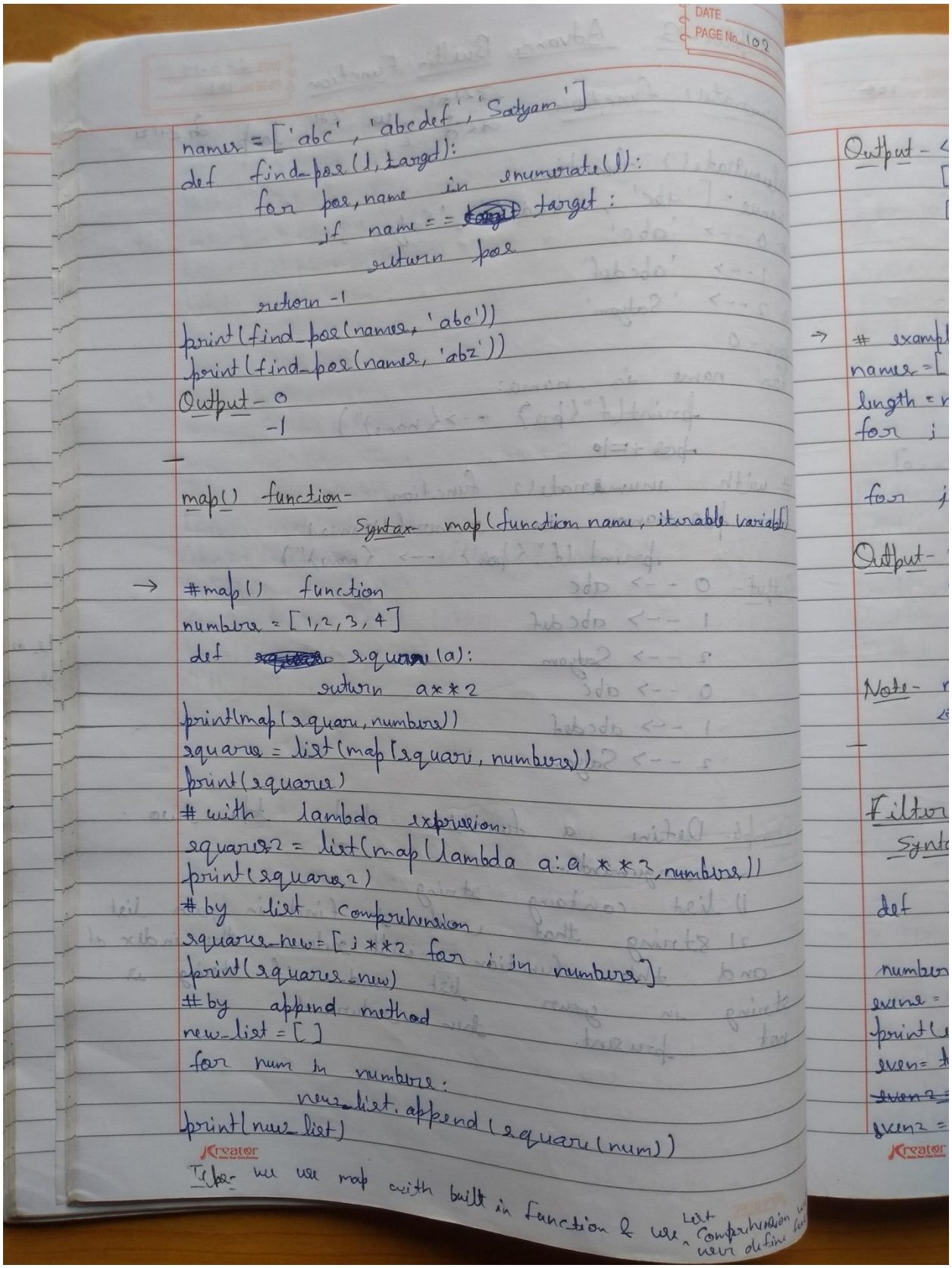
DATE 6-3-19
PAGE No. 101

enumerate() function- इसका use तब करते हैं जब loop के साथ

→ #Enumerate() function
names = ['abc', 'abcdef', 'Satyam']
# 0 --> 'abc'
# 1 --> 'abcdef'
# 2 --> 'Satyam'
pos = 0
for name in names:
    print(f"{pos} --> {name}")
    pos += 1
# with enumerate() function
for pos, name in enumerate(names):
    print(f"{pos} --> {name}")

Output- 0 --> abc
1 --> abcdef
2 --> Satyam
0 --> abc
1 --> abcdef
2 --> Satyam

Example- Define a function that take two arguments:-
1) list containg string
2) string that want to find in your list and this function will return the index of string in your list and if string is not present then return -1



```
names = ['abc', 'abcdef', 'Satyam']
def find_pos(l, target):
    for pos, name in enumerate(l):
        if name == target:
            return pos
    return -1
print(find_pos(names, 'abc'))
print(find_pos(names, 'abz'))
```

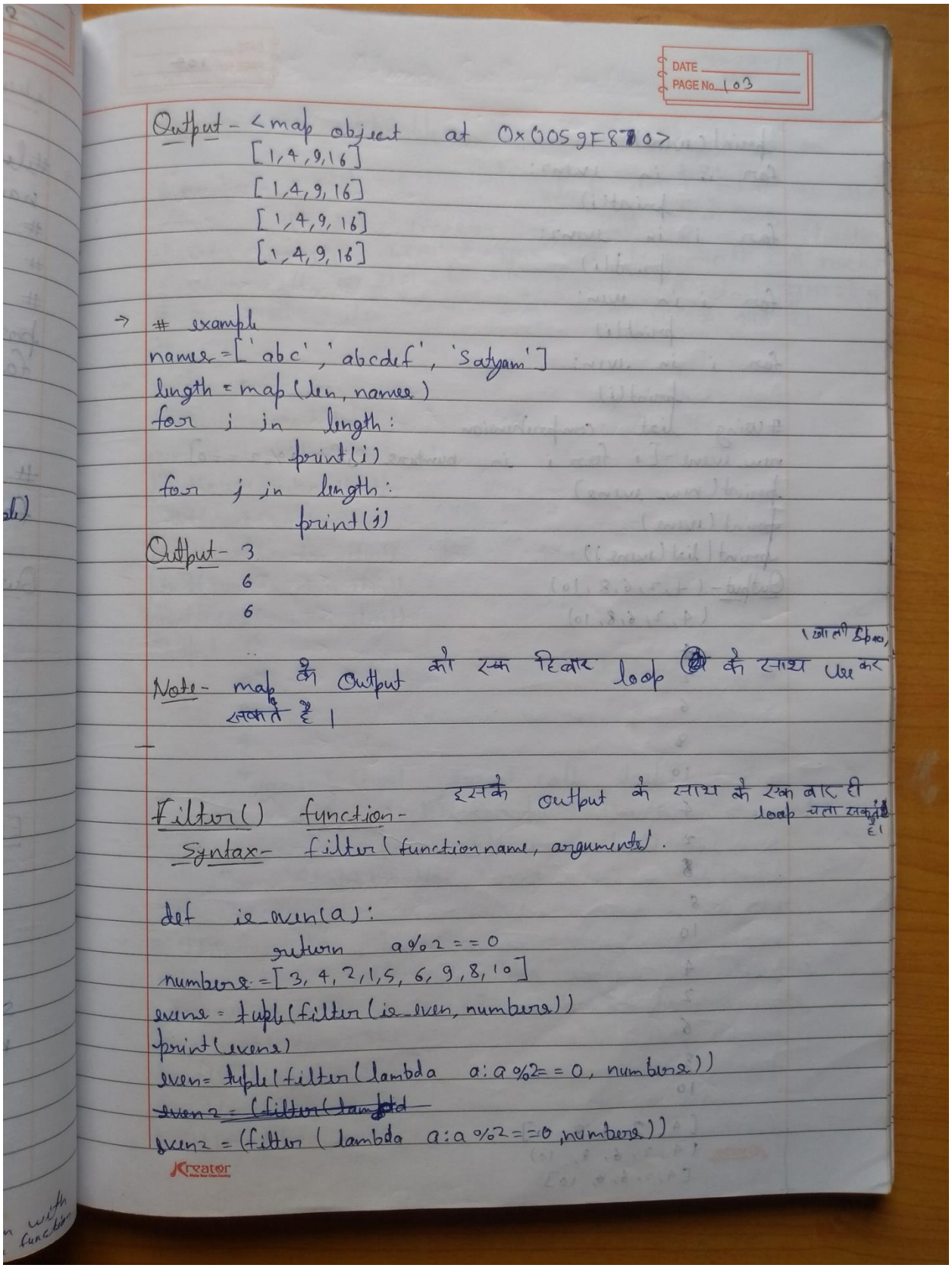
Output - 0
-1

map() function -

Syntax - map(function name, iterable variable)

```
#map() function
numbers = [1,2,3,4]
def square(a):
    return a**2
print(map(square, numbers))
square = list(map(square, numbers))
print(squares)
# with lambda expression.
squares2 = list(map(lambda a: a**2, numbers))
print(squares2)
# by list comprehension
square_new = [i**2 for i in numbers]
print(squares_new)
# by append method
new_list = []
for num in numbers:
    new_list.append(square(num))
print(new_list)
```

Tips - we use map with built in function & use List Comprehension in user define



Output - <map object at 0x0059F870>
[1,4,9,16]
[1,4,9,16]
[1,4,9,16]
[1,4,9,16]

→ # example
names = ['abc', 'abcdef', 'satyam']
length = map(len, names)
for i in length:
    print(i)
for j in length:
    print(j)

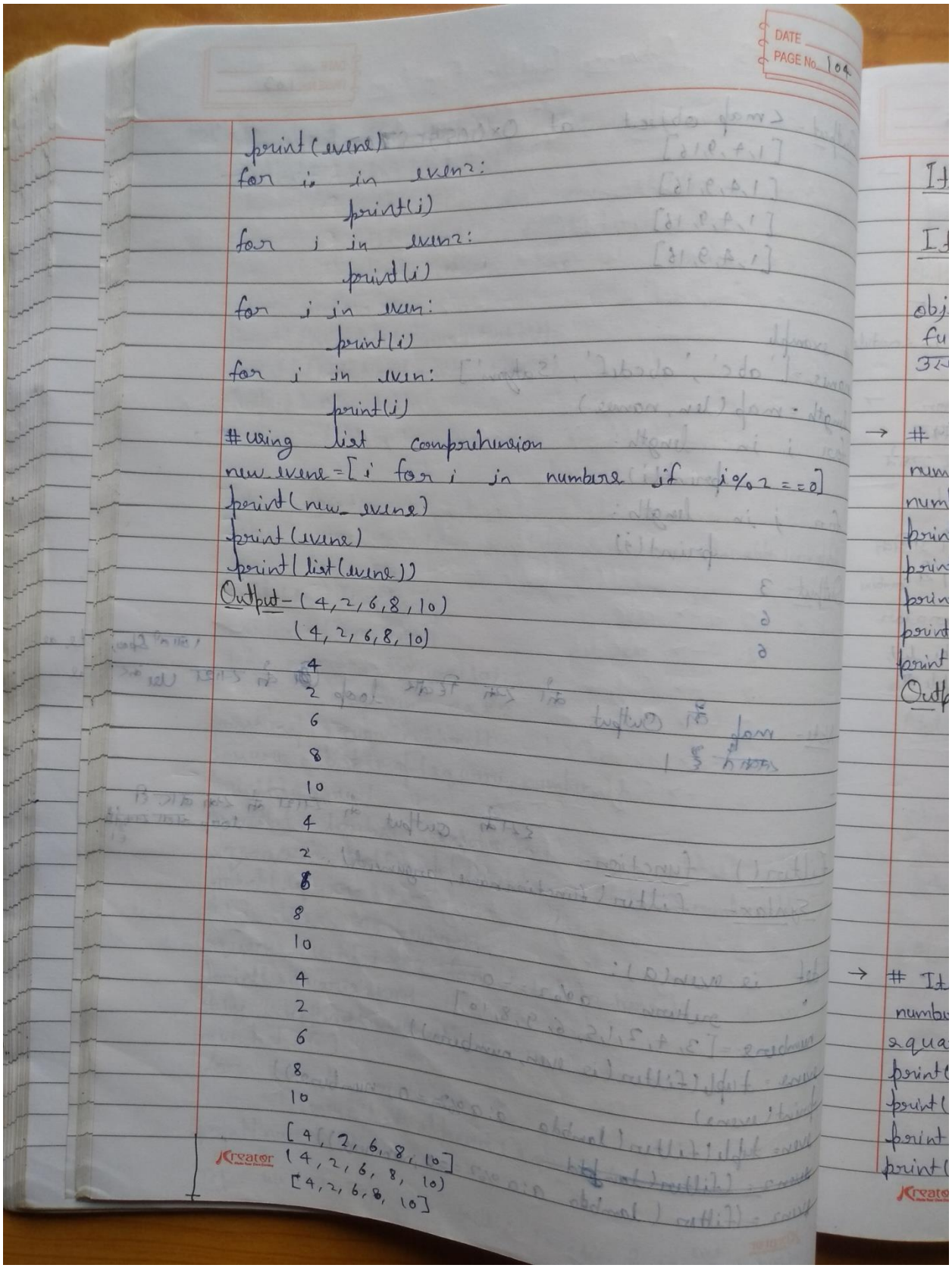
Output - 3
6
6

Note - map का Output को एक बार loop के साथ use कर सकते हैं।

Filter() function - इसके output के साथ के एक बार ही loop चला सकते हैं।

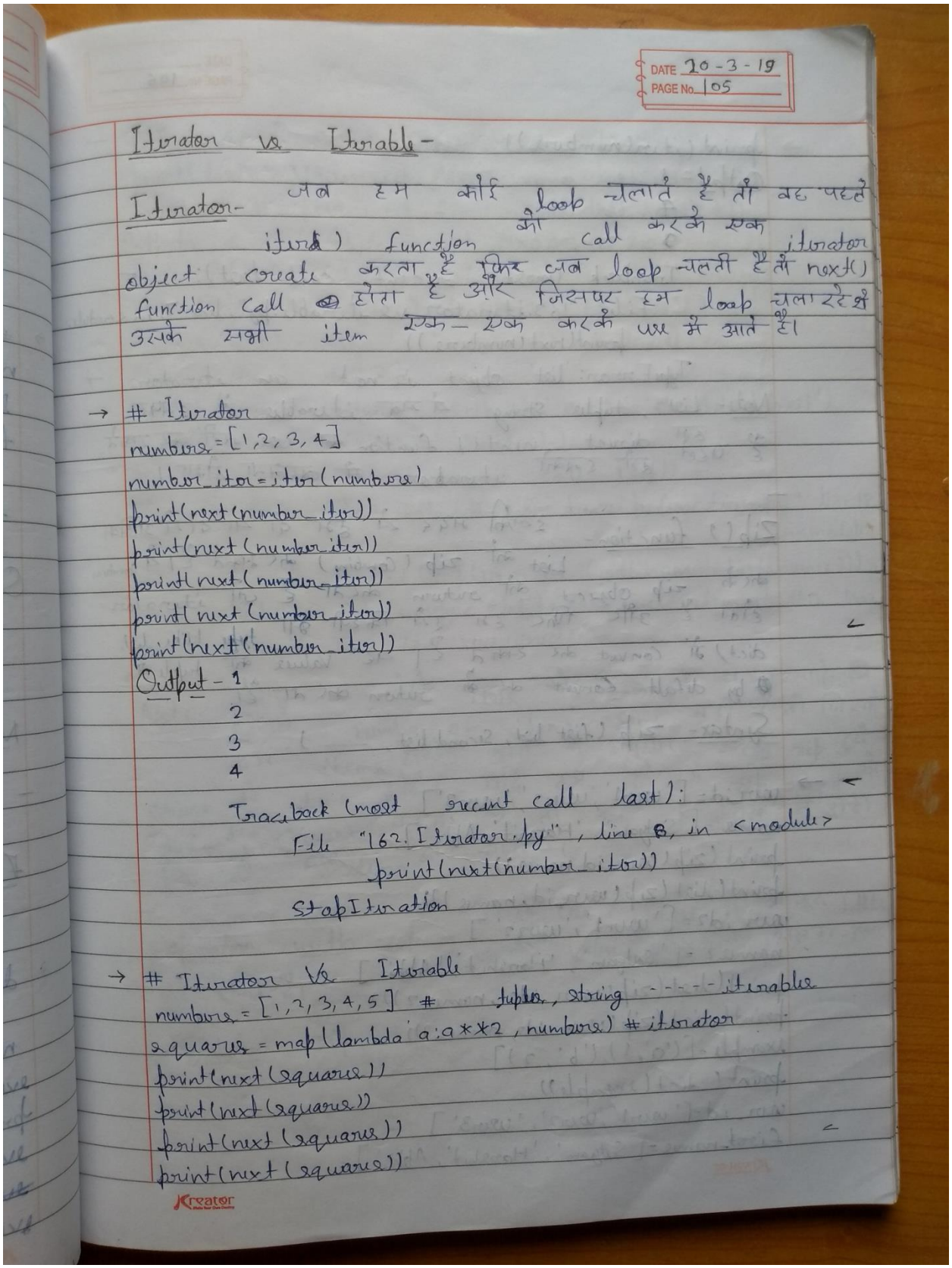
Syntax - filter(function name, arguments).

def is_even(a):
    return a%2==0
numbers = [3,4,2,1,5,6,9,8,10]
evens = tuple(filter(is_even, numbers))
print(evens)
even = tuple(filter(lambda a: a%2==0, numbers))
~~even2 = (filter(lambda~~
even2 = (filter(lambda a: a%2==0, numbers))



```
print(evens)
for i in evens:
        print(i)
for i in evens:
        print(i)
for i in evens:
        print(i)
for i in evens:
        print(i)
# using list comprehension
new_evens = [i for i in numbers if i%2==0]
print(new_evens)
print(evens)
print(list(evens))
```

Output - (4, 2, 6, 8, 10)
(4, 2, 6, 8, 10)
4
2
6
8
10
4
2
6
8
10
4
2
6
8
10
[4, 2, 6, 8, 10]
(4, 2, 6, 8, 10)
[4, 2, 6, 8, 10]



DATE 20-3-19
PAGE No. 105

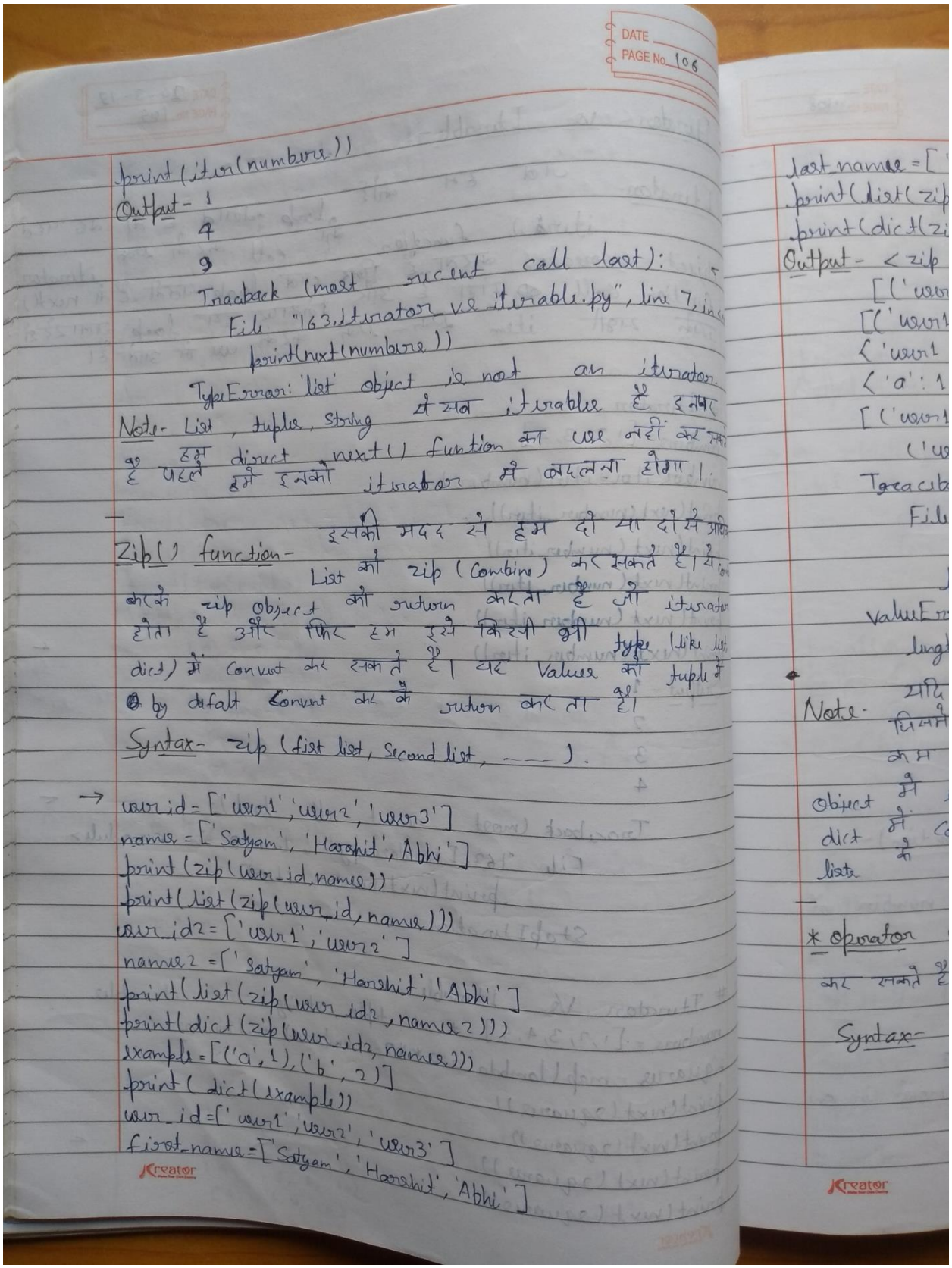
Iterator vs Iterable -

Iterator - जब हम कोई loop चलाते हैं तो वह पहले उसे iter() function को call करके एक iterator object create करता है फिर जब loop चलती है तो next() function call होता है और जिसपर हम loop चला रहे हैं उसके सभी item एक-एक करके use में आते हैं।

→ # Iterator
numbers = [1, 2, 3, 4]
number_itor = itor(numbers)
print(next(number_itor))
print(next(number_itor))
print(next(number_itor))
print(next(number_itor))
print(next(number_itor))

Output - 1
2
3
4
Traceback (most recent call last):
File "162. Iterator.py", line 8, in <module>
print(next(number_itor))
StopIteration

→ # Iterator Vs Iterable
numbers = [1, 2, 3, 4, 5] # tuples, string ----- iterables
squares = map(lambda a: a**2, numbers) # iterator
print(next(squares))
print(next(squares))
print(next(squares))
print(next(squares))

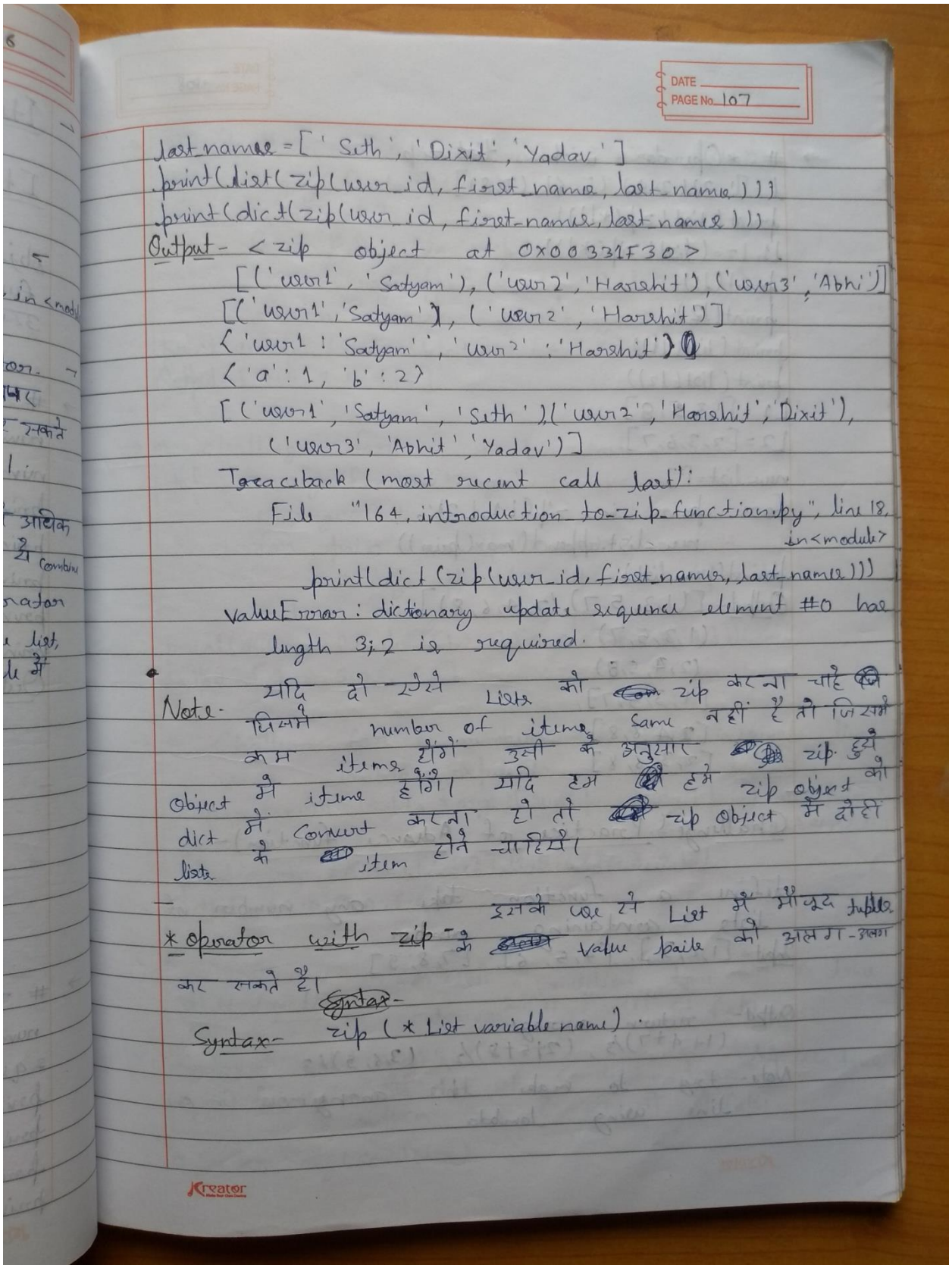


print(iter(numbers))
Output- 1
4
9
Traceback (most recent call last):
File "163.Iterator_vs_iterable.py", line 7, in
print(next(numbers))
TypeError: 'list' object is not an iterator.
Note- List, tuple, string सब iterable है इनमें
है पहले हम direct next() function का use नहीं कर सक
हमें इनको iterator में बदलना होगा।

Zip() function- इसकी मदद से हम दो या दो से अधि
List को zip (combine) कर सकते है। ये
करके zip object को return करता है जो iterator
होता है और फिर हम इसे किसी भी type (like list
dict) में convert कर सकते है। यह Values को tuple में
by default convert करके return करता है।
Syntax- zip (first list, second list, ---).

→ user_id = ['user1', 'user2', 'user3']
name = ['Satyam', 'Harshit', 'Abhi']
print(zip(user_id, name))
print(list(zip(user_id, name)))
user_id2 = ['user1', 'user2']
name2 = ['Satyam', 'Harshit', 'Abhi']
print(list(zip(user_id2, name2)))
print(dict(zip(user_id2, name2)))
example = [('a', 1), ('b', 2)]
print(dict(example))
user_id = ['user1', 'user2', 'user3']
first_name = ['Satyam', 'Harshit', 'Abhi']

last_name = [
print(dict(zip
print(dict(zi
Output- <zip
[('user
[('user1
{'user1
{'a': 1
[('user1
('us
Traceb
Fil
ValueE
leng
Note- यदि
object में
dict में
के
list.
* Operator
कर सकते है
Syntax-



```
last_names = ['Seth', 'Dixit', 'Yadav']
print(list(zip(user_id, first_name, last_name)))
print(dict(zip(user_id, first_names, last_names)))
```

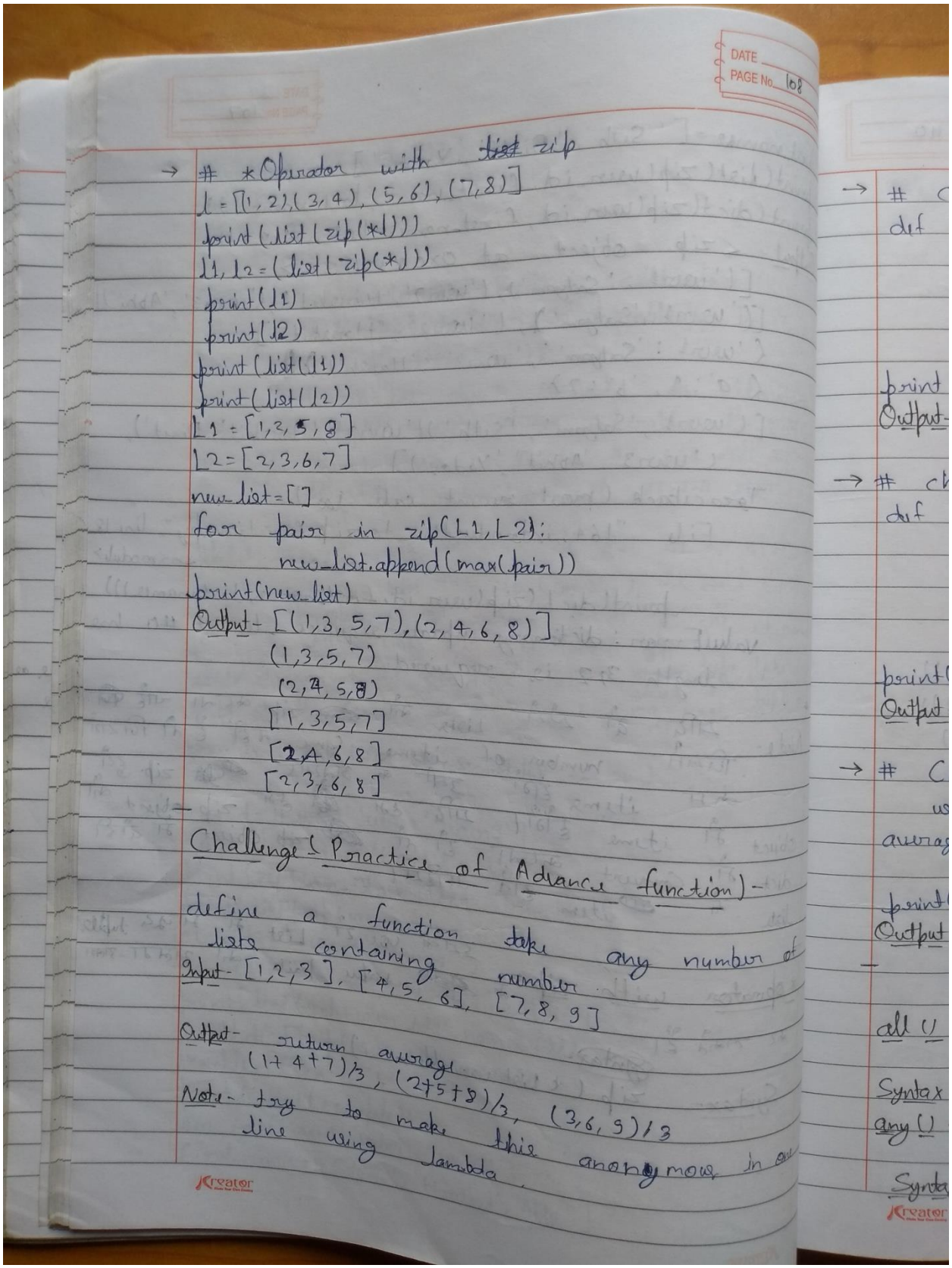
Output - <zip object at 0x00331F30>

```
[('user1', 'Satyam'), ('user2', 'Harshit'), ('user3', 'Abhi')]
[('user1', 'Satyam'), ('user2', 'Harshit')]
{'user1': 'Satyam', 'user2': 'Harshit'}
{'a': 1, 'b': 2}
[('user1', 'Satyam', 'Seth'), ('user2', 'Harshit', 'Dixit'),
 ('user3', 'Abhi', 'Yadav')]
Traceback (most recent call last):
  File "164. introduction to zip function.py", line 18, in <module>
    print(dict(zip(user_id, first_names, last_names)))
ValueError: dictionary update sequence element #0 has length 3; 2 is required.
```

Note - यदि दो या दो से अधिक Lists को zip करना चाहें जिसमें number of items same नहीं है तो जिसमें कम items होंगे उसी के अनुसार zip हुई object में items होंगे। यदि हम को zip object को dict में convert करना हो तो zip object में दो ही lists के item होने चाहिये।

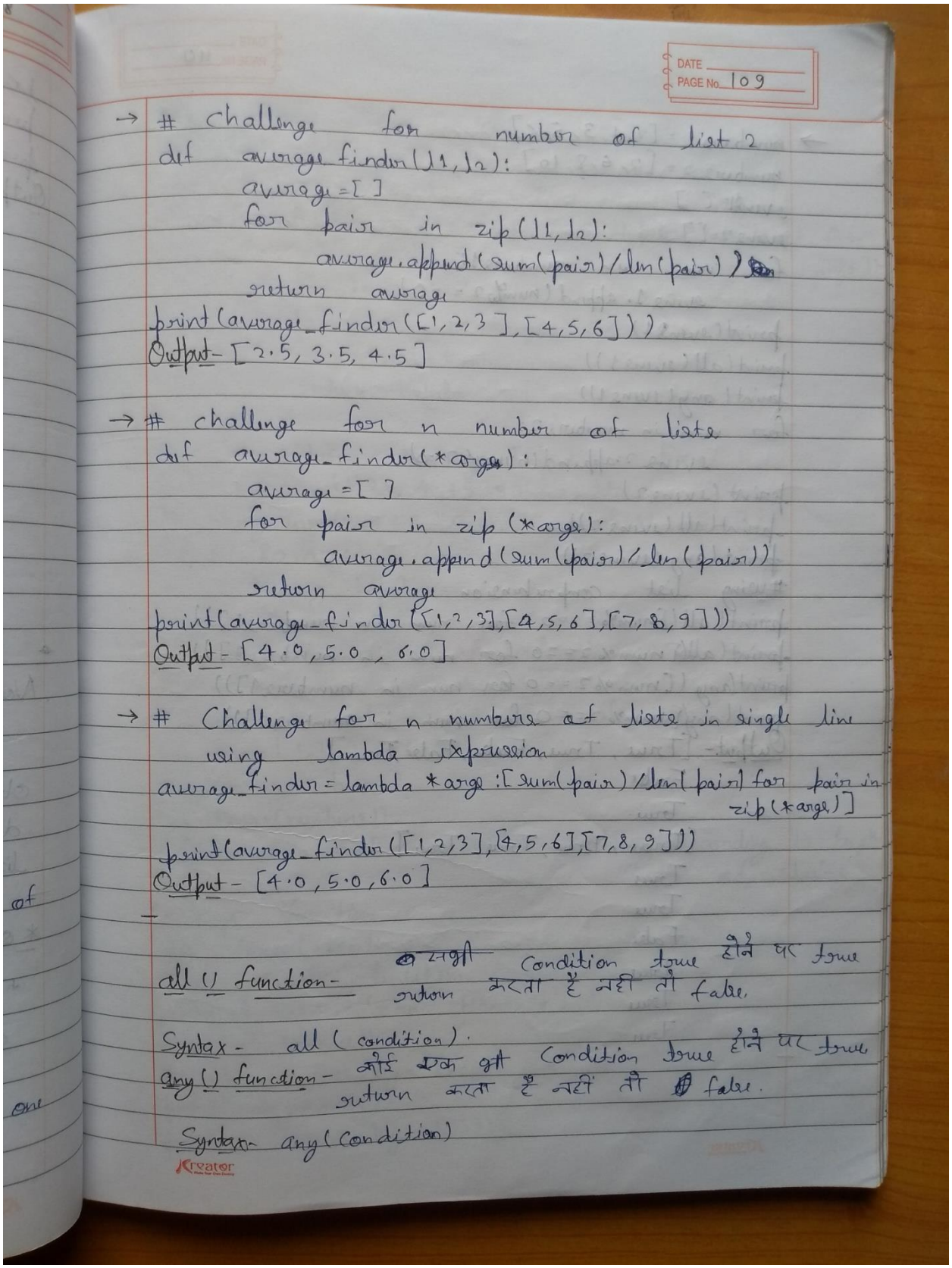
*operator with zip - इसको use से List में मौजूद tuples के value pairs को अलग-अलग कर सकते हैं।

Syntax - zip(*List variable name)



→ # *Operator with ~~list~~ zip
l = [(1,2),(3,4),(5,6),(7,8)]
print(list(zip(*l)))
l1, l2 = (list(zip(*l)))
print(l1)
print(l2)
print(list(l1))
print(list(l2))
L1 = [1,3,5,8]
L2 = [2,3,6,7]
new_list = []
for pair in zip(L1,L2):
    new_list.append(max(pair))
print(new_list)
Output - [(1,3,5,7),(2,4,6,8)]
(1,3,5,7)
(2,4,5,8)
[1,3,5,7]
[2,4,6,8]
[2,3,6,8]

Challenge (Practice of Advance function) -
define a function take any number of lists containing number
Input - [1,2,3], [4,5,6], [7,8,9]
Output - return average
(1+4+7)/3, (2+5+8)/3, (3+6+9)/3
Note - try to make this anonymous in one line using lambda



→ # challenge for number of list 2
```
def average_finder(l1, l2):
    average = []
    for pair in zip(l1, l2):
        average.append(sum(pair)/len(pair))
    return average
print(average_finder([1,2,3],[4,5,6]))
```
Output - [2.5, 3.5, 4.5]

→ # challenge for n number of lists
```
def average_finder(*args):
    average = []
    for pair in zip(*args):
        average.append(sum(pair)/len(pair))
    return average
print(average_finder([1,2,3],[4,5,6],[7,8,9]))
```
Output - [4.0, 5.0, 6.0]

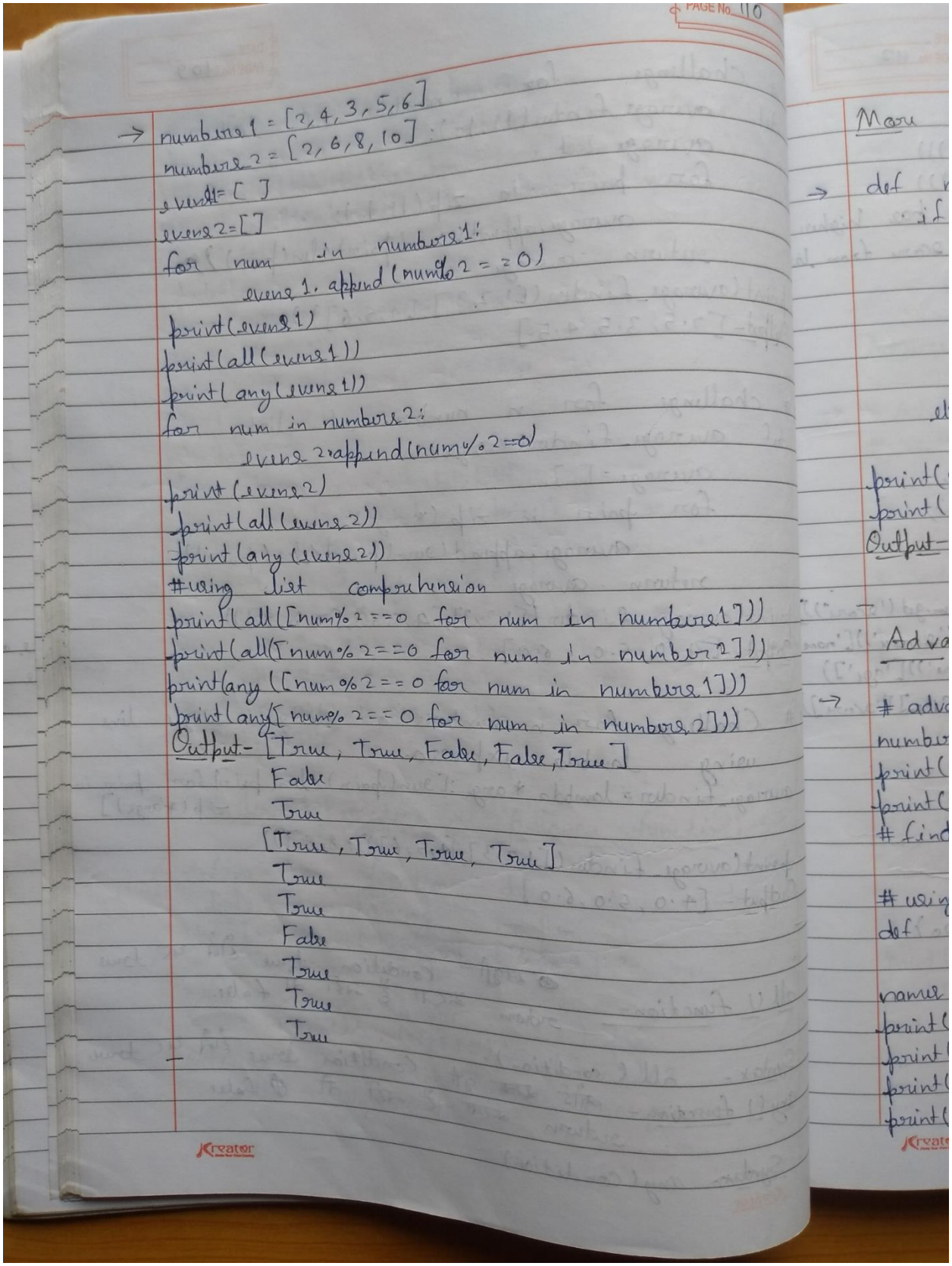
→ # Challenge for n numbers of lists in single line using lambda expression
```
average_finder = lambda *arge :[sum(pair)/len(pair) for pair in zip(*args)]
print(average_finder([1,2,3],[4,5,6],[7,8,9]))
```
Output - [4.0, 5.0, 6.0]

all() function - Condition true होने पर true return करता है नहीं तो false.

Syntax - all(condition).

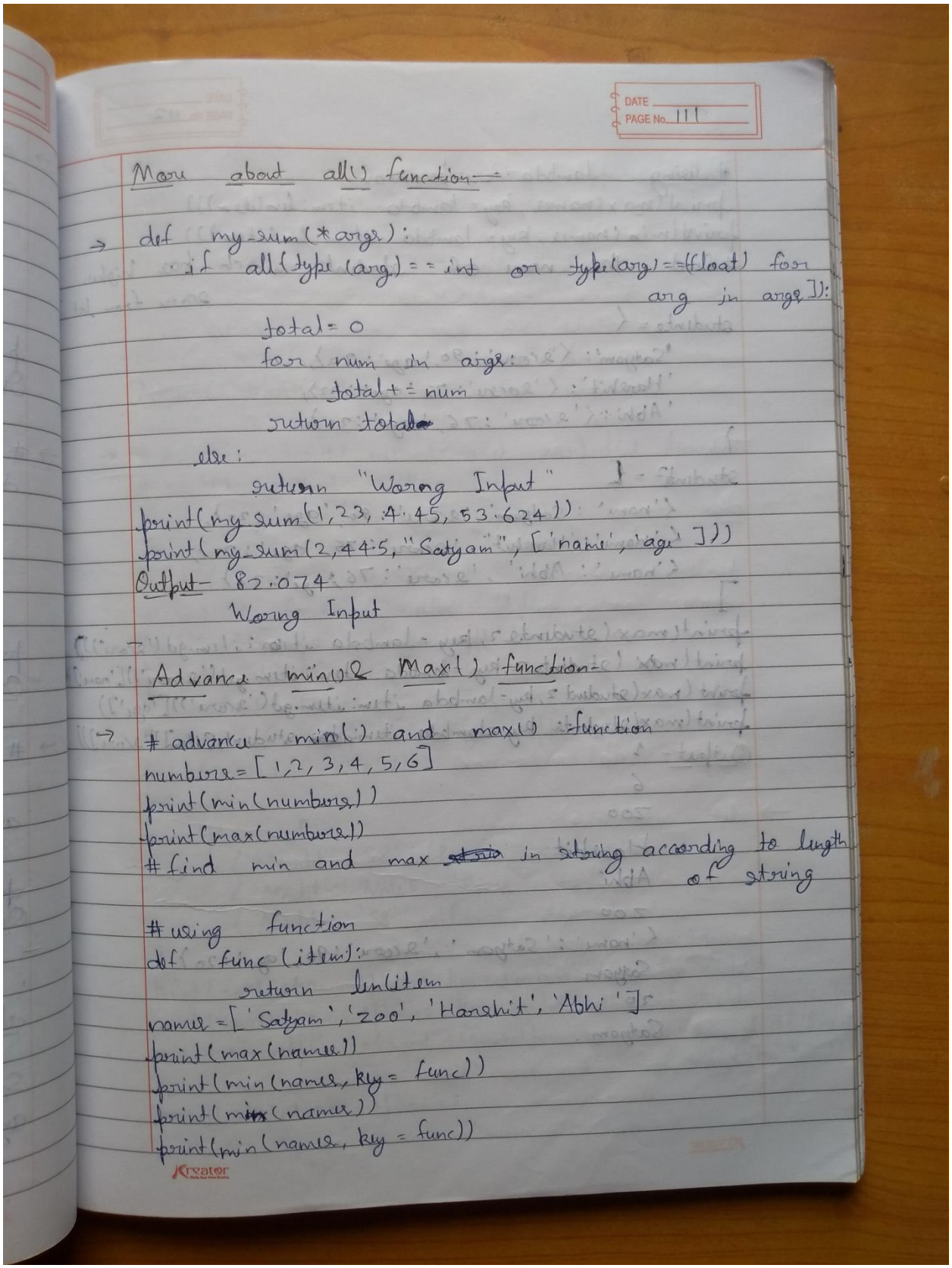
any() function - कोई एक भी Condition true होने पर true return करता है नहीं तो false.

Syntax - any(condition)



```
numbers1 = [2,4,3,5,6]
numbers2 = [2,6,8,10]
evens1 = []
evens2 = []
for num in numbers1:
    evens1.append(num%2==0)
print(evens1)
print(all(evens1))
print(any(evens1))
for num in numbers2:
    evens2.append(num%2==0)
print(evens2)
print(all(evens2))
print(any(evens2))
#using list comprehension
print(all([num%2==0 for num in numbers1]))
print(all([num%2==0 for num in numbers2]))
print(any([num%2==0 for num in numbers1]))
print(any([num%2==0 for num in numbers2]))
```

Output-
```
[True, True, False, False, True]
False
True
[True, True, True, True]
True
True
False
True
True
True
```

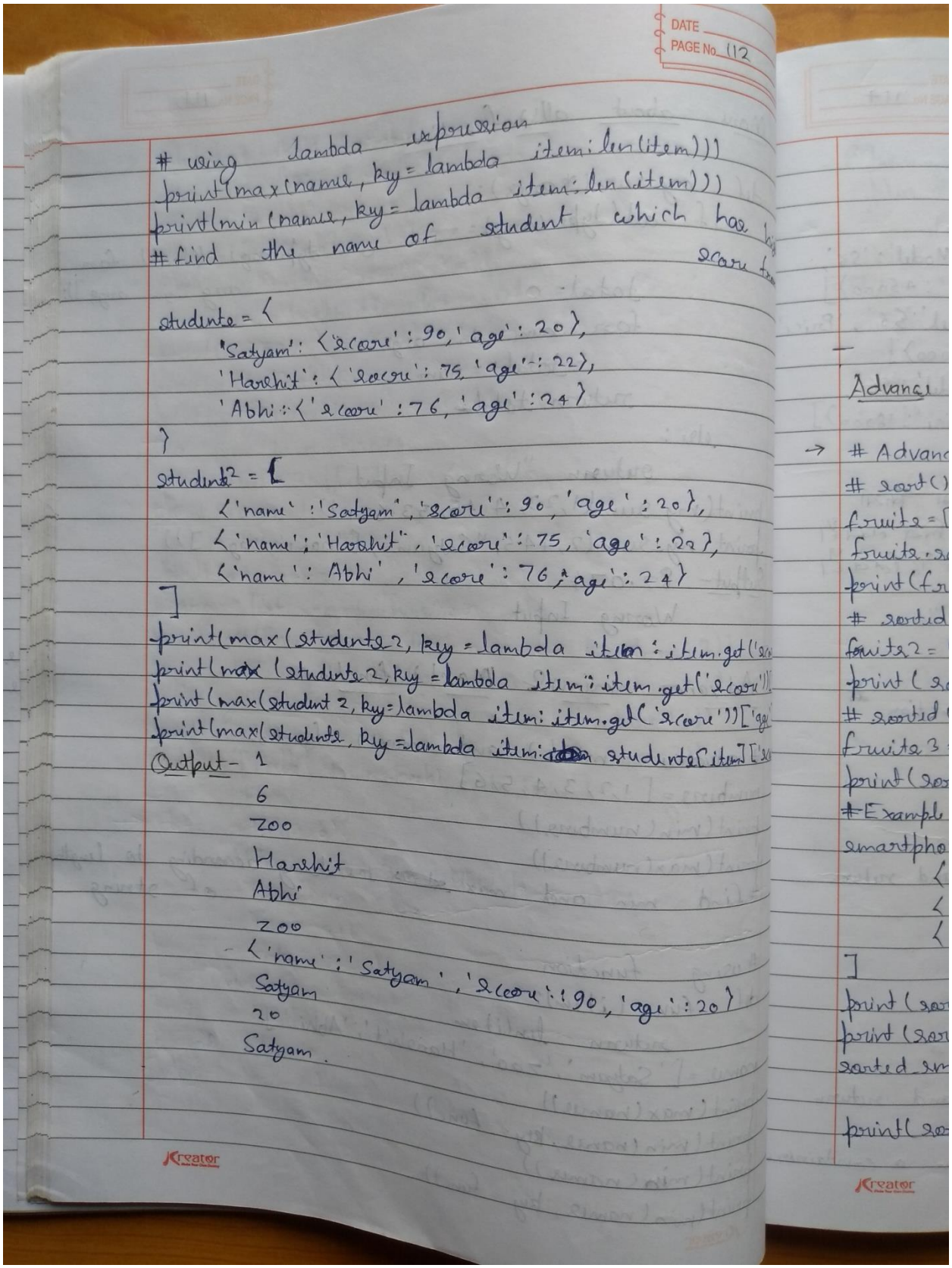


More about all() function:-

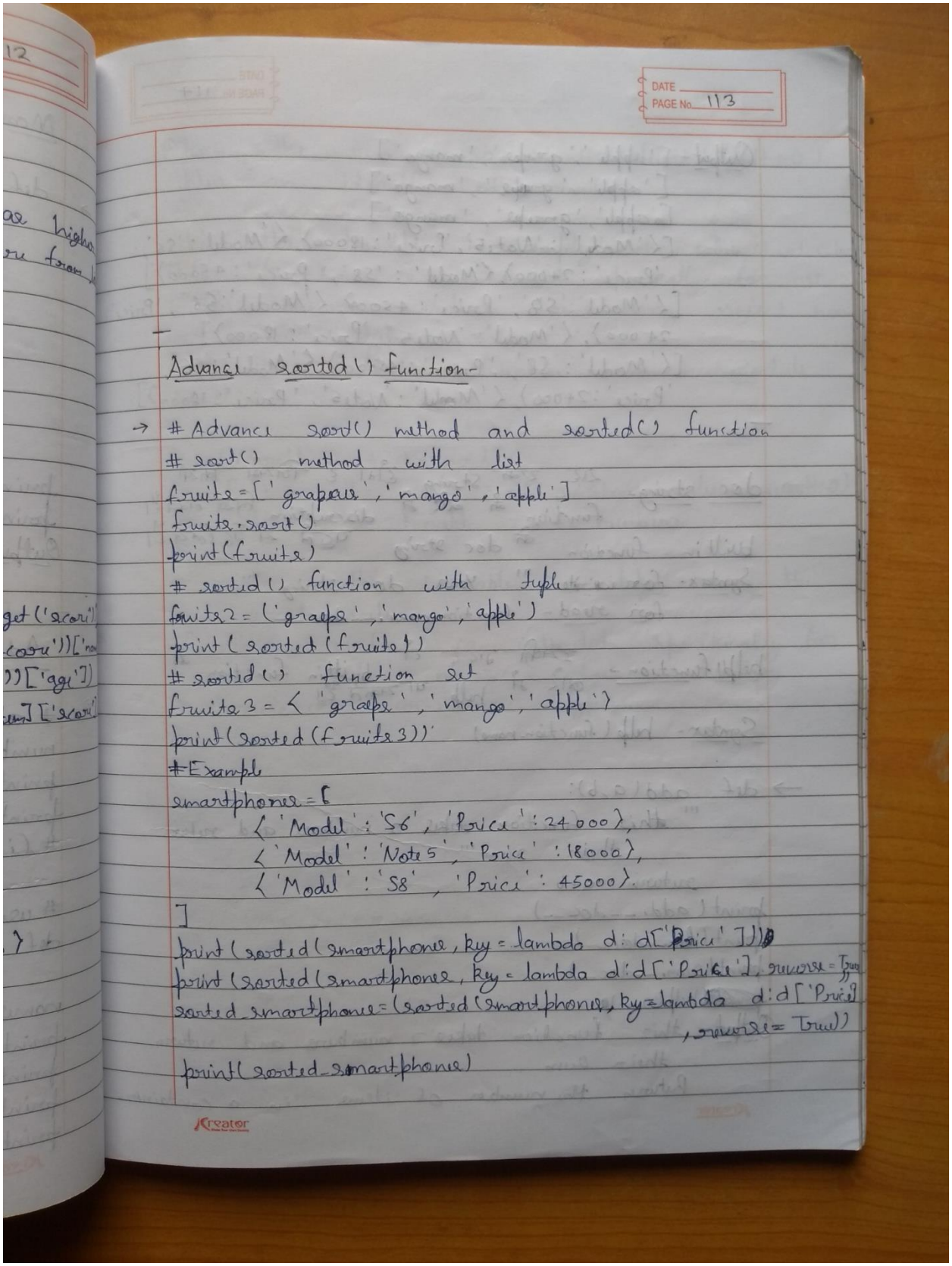
→
```
def my_sum(*args):
    if all(type(arg)==int or type(arg)==float for arg in args):
        total=0
        for num in args:
            total+=num
        return total
    else:
        return "Worong Input"
print(my_sum(1,23,4.45,53.624))
print(my_sum(2,44.5,"Satyam",['name','age']))
```
Output- 82.074
Woorng Input

Advance min() & max() function:-

→
```
# advance min() and max() function
numbers=[1,2,3,4,5,6]
print(min(numbers))
print(max(numbers))
# find min and max in string according to lngth of string

# using function
def func(item):
    return len(item)
names=['Satyam','zoo','Harshit','Abhi']
print(max(names))
print(min(names,key=func))
print(min(names))
print(min(names,key=func))
```

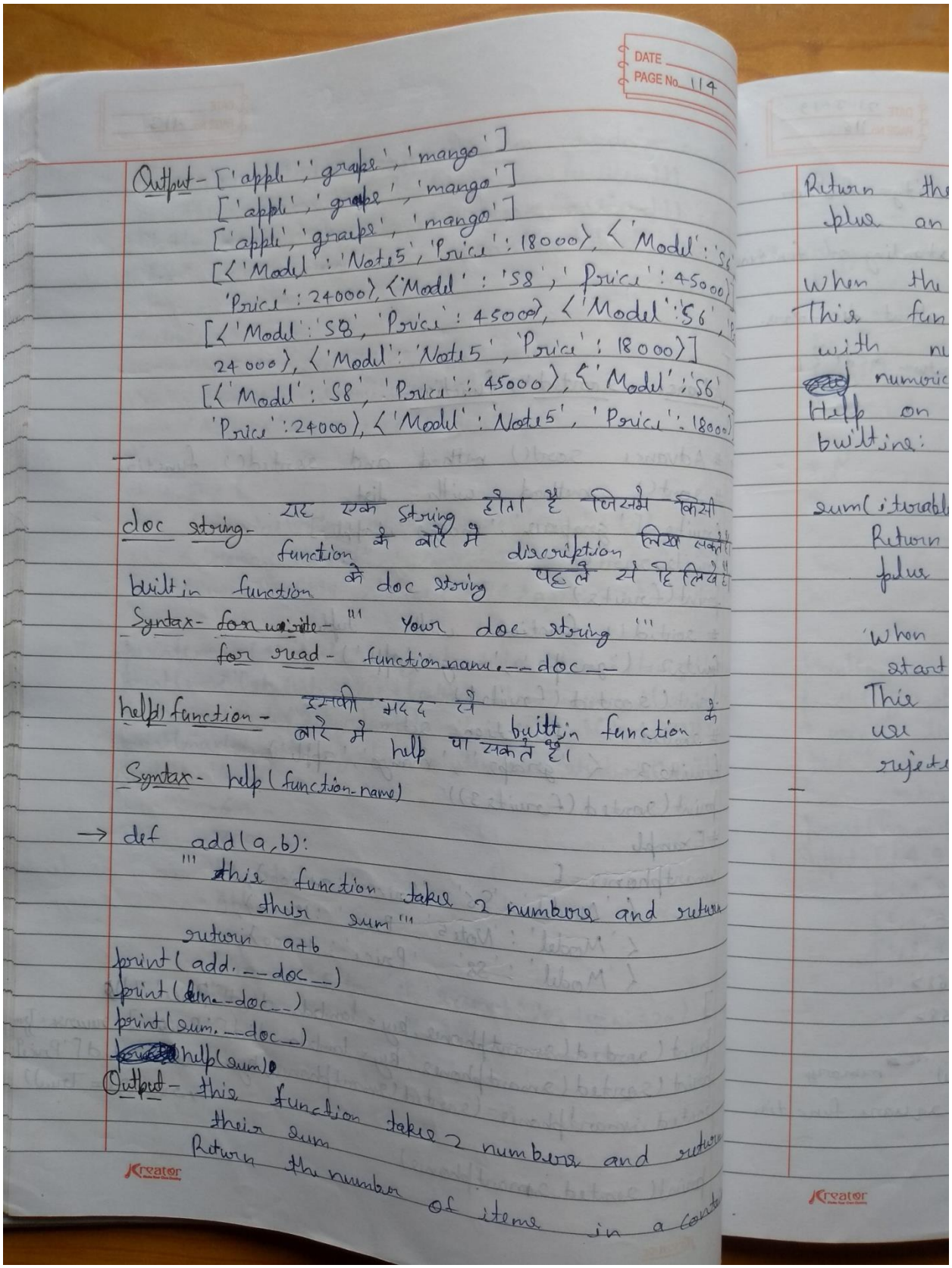


```
# using lambda expression
print(max(name, key = lambda item: len(item)))
print(min(name, key = lambda item: len(item)))
# find the name of student which has
                                    score
students = {
    'Satyam': {'score': 90, 'age': 20},
    'Harshit': {'score': 75, 'age': 22},
    'Abhi': {'score': 76, 'age': 24}
}
student2 = [
    {'name': 'Satyam', 'score': 90, 'age': 20},
    {'name': 'Harshit', 'score': 75, 'age': 22},
    {'name': Abhi', 'score': 76, 'age': 24}
]
print(max(students2, key = lambda item : item.get('sc
print(max(students2, key = lambda item: item.get('score'))
print(max(student 2, key = lambda item: item.get('score'))['age'
print(max(students, key = lambda item: students[item]['s
Output - 1
         6
         200
         Harshit
         Abhi
         200
         {'name': 'Satyam', 'score': 90, 'age': 20}
         Satyam
         20
         Satyam
```



Advance sorted() function-

```
# Advance sort() method and sorted() function
# sort() method with list
fruits = ['grapes', 'mango', 'apple']
fruits.sort()
print(fruits)
# sorted() function with tuple
fruits2 = ('grapes', 'mango', 'apple')
print(sorted(fruits))
# sorted() function set
fruits3 = {'grapes', 'mango', 'apple'}
print(sorted(fruits3))
#Example
smartphones = [
    {'Model': 'S6', 'Price': 24000},
    {'Model': 'Note 5', 'Price': 18000},
    {'Model': 'S8', 'Price': 45000}
]
print(sorted(smartphones, key = lambda d: d['Price']))
print(sorted(smartphones, key = lambda d: d['Price'], reverse = True))
sorted_smartphones = (sorted(smartphones, key=lambda d: d['Price']
                             , reverse = True))

print(sorted_smartphones)
```



Output- ['apple', 'grapes', 'mango']
['apple', 'grapes', 'mango']
['apple', 'grapes', 'mango']
[{'Model': 'Note5', 'Price': 18000}, {'Model': 'S8', 'Price': 24000}, {'Model': 'S8', 'Price': 45000}]
[{'Model': 'S8', 'Price': 45000}, {'Model': 'S6', 'Price': 24000}, {'Model': 'Note5', 'Price': 18000}]
[{'Model': 'S8', 'Price': 45000}, {'Model': 'S6', 'Price': 24000}, {'Model': 'Note5', 'Price': 18000}]

doc string- यह एक String होता है जिसमें किसी function के बारे में discription लिख सकते है built in function के doc string पहले से ही लिखे है

Syntax- for write- """ Your doc string """
for read- function_name.__doc__

help() function- इसकी मदद से built in function के बारे में help पा सकते है।

Syntax- help(function_name)

```
def add(a,b):
    """this function takes 2 numbers and returns
       their sum"""
    return a+b
print(add.__doc__)
print(sum.__doc__)
print(sum.__doc__)
help(sum)
```

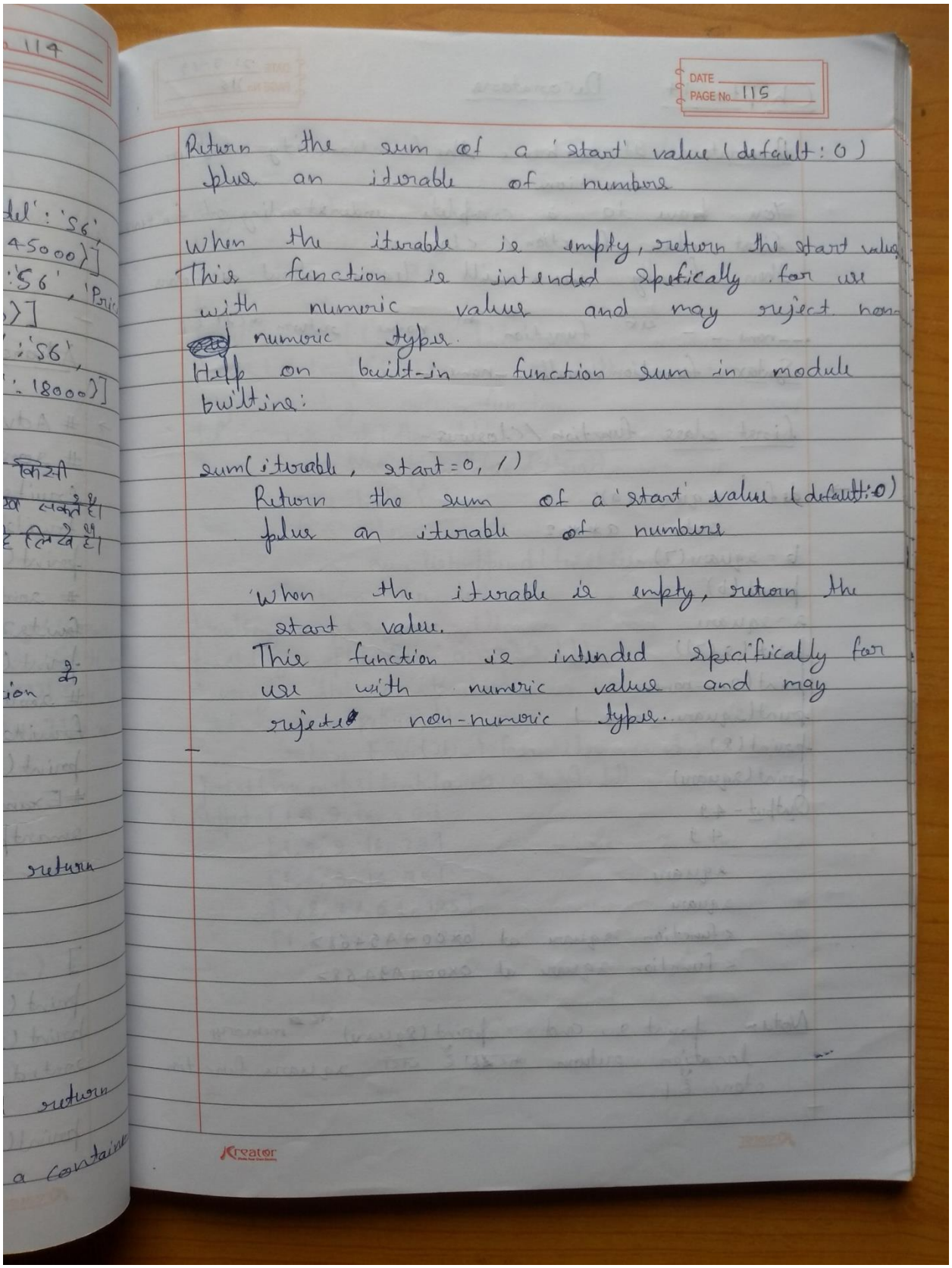
Output- this function takes 2 numbers and returns their sum
Return the number of items in a conta

Return the
plus an

When the
This fun
with nu
numeric
Help on
built-in:

sum(iterable
Return
plus

'when
start
This
use
rejects



DATE
PAGE No. 115

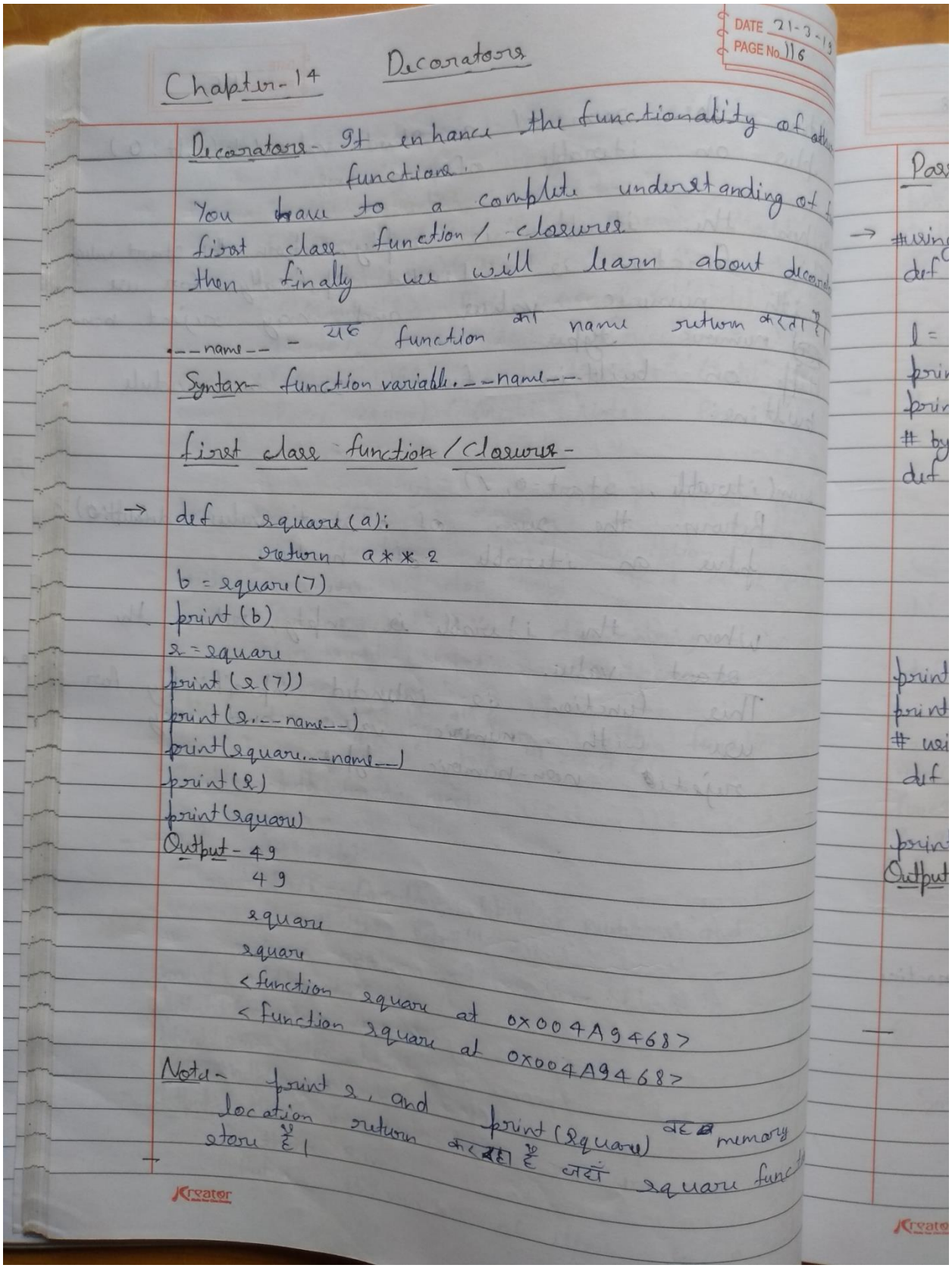
Return the sum of a 'start' value (default: 0) plus an iterable of numbers

When the iterable is empty, return the start value. This function is intended specifically for use with numeric values and may reject non-numeric types.

Help on built-in function sum in module builtins:

sum(iterable, start=0, /)
Return the sum of a 'start' value (default: 0) plus an iterable of numbers

When the iterable is empty, return the start value.
This function is intended specifically for use with numeric values and may reject non-numeric types.



Chapter-14 Decorators

DATE 21-3-19
PAGE No. 116

Decorators - It enhance the functionality of other functions.
You have to a complete understanding of 1. first class function / closures
then finally we will learn about decorators

__name__ - यह function का name return करता है।

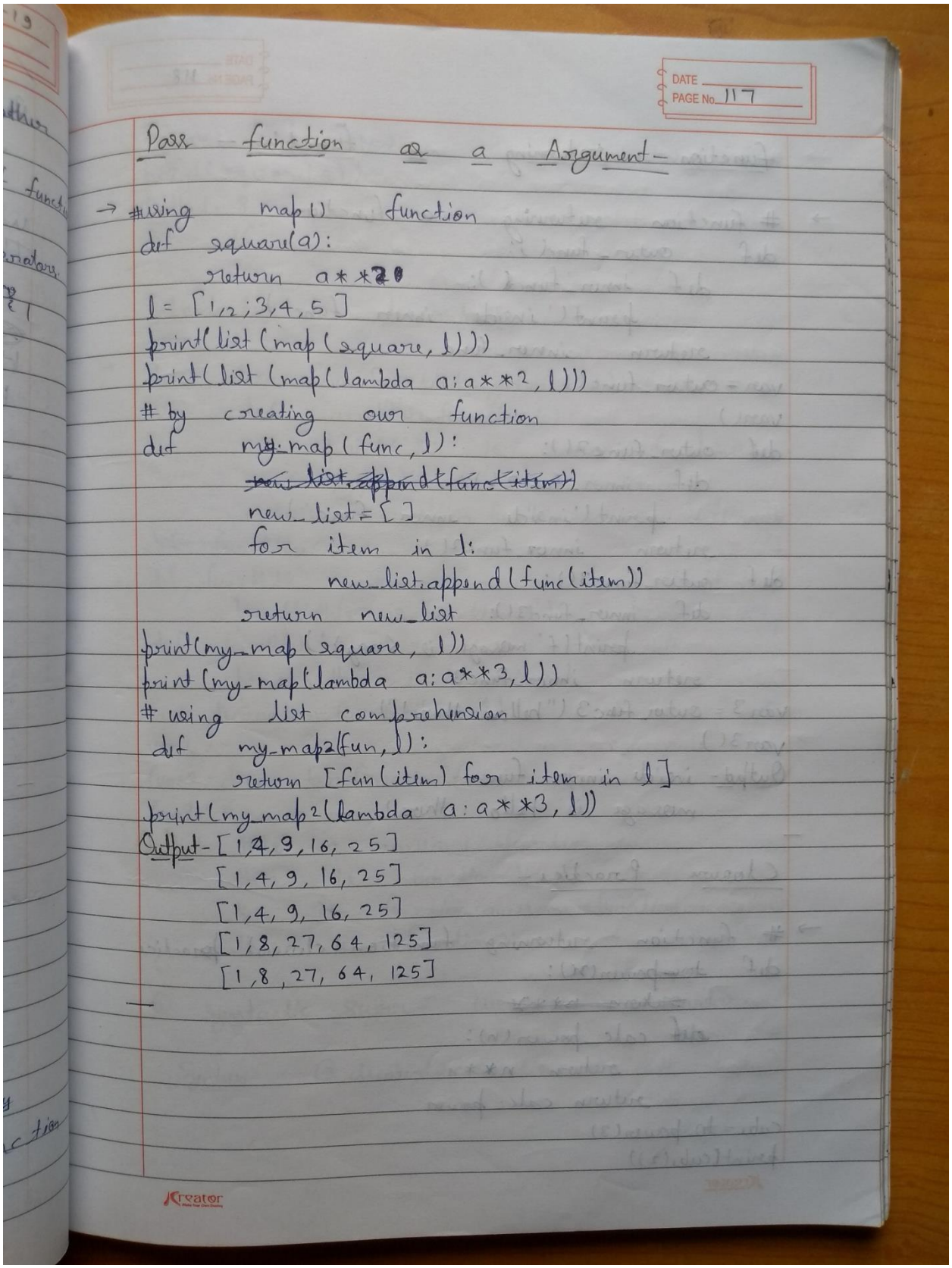
Syntax - function variable.__name__

first class function / closures -

```
def square(a):
    return a**2
b = square(7)
print(b)
s = square
print(s(7))
print(s.__name__)
print(square.__name__)
print(s)
print(square)
```

Output - 49
49
square
square
<function square at 0x004A94687>
<function square at 0x004A94687>

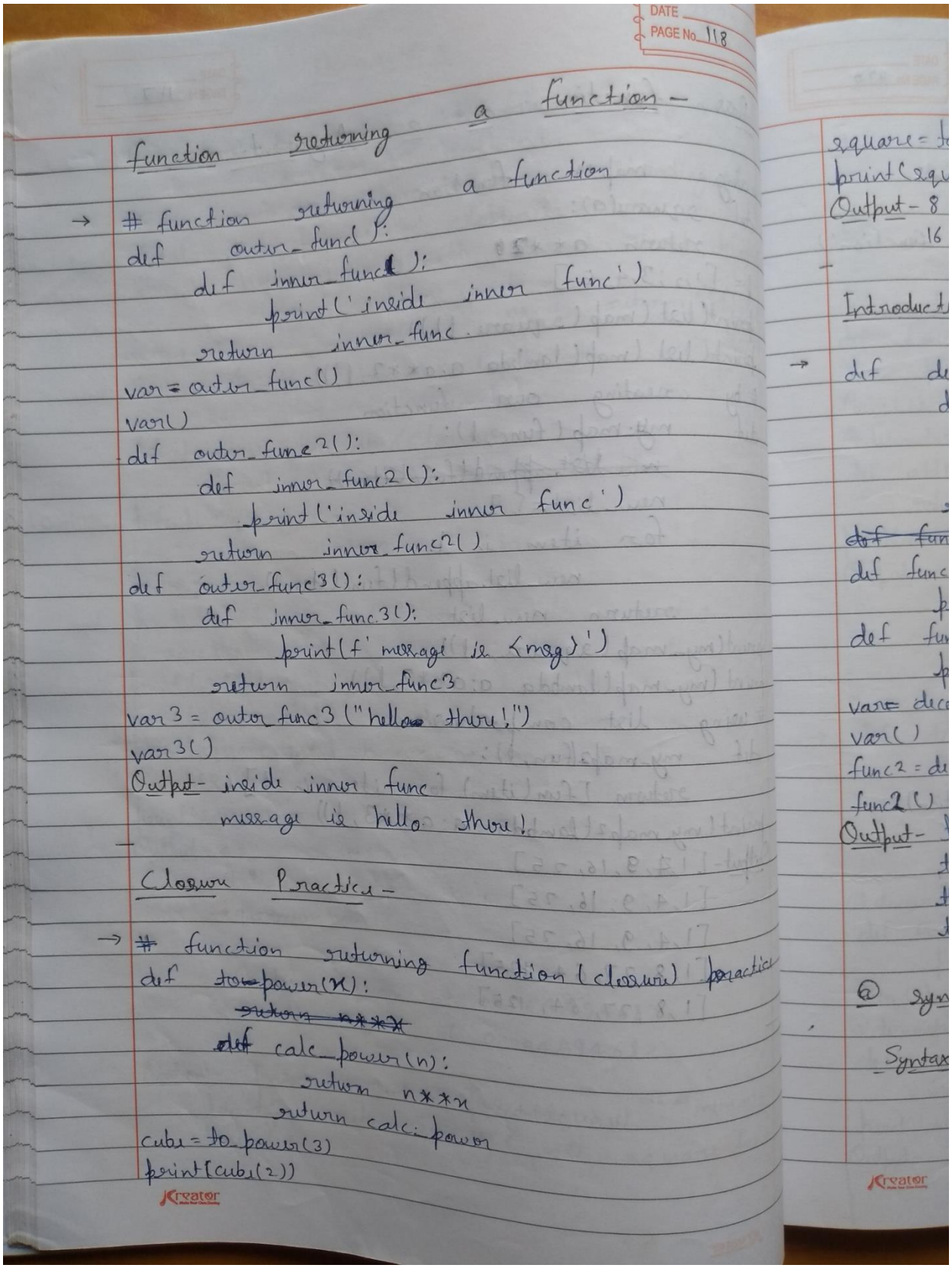
Note - print s, and print(square) दोनों memory location return करते हैं जहाँ square function store है।



DATE
PAGE No. 117

Pass function as a Argument-

→ #using map() function
```
def square(a):
    return a**2
l = [1,2,3,4,5]
print(list(map(square, l)))
print(list(map(lambda a:a**2, l)))
# by creating our function
def my_map(func, l):
    new_list = []
    for item in l:
        new_list.append(func(item))
    return new_list
print(my_map(square, l))
print(my_map(lambda a:a**3, l))
# using list comprehension
def my_map2(fun, l):
    return [fun(item) for item in l]
print(my_map2(lambda a:a**3, l))
```
Output- [1,4,9,16,25]
[1,4,9,16,25]
[1,4,9,16,25]
[1,8,27,64,125]
[1,8,27,64,125]



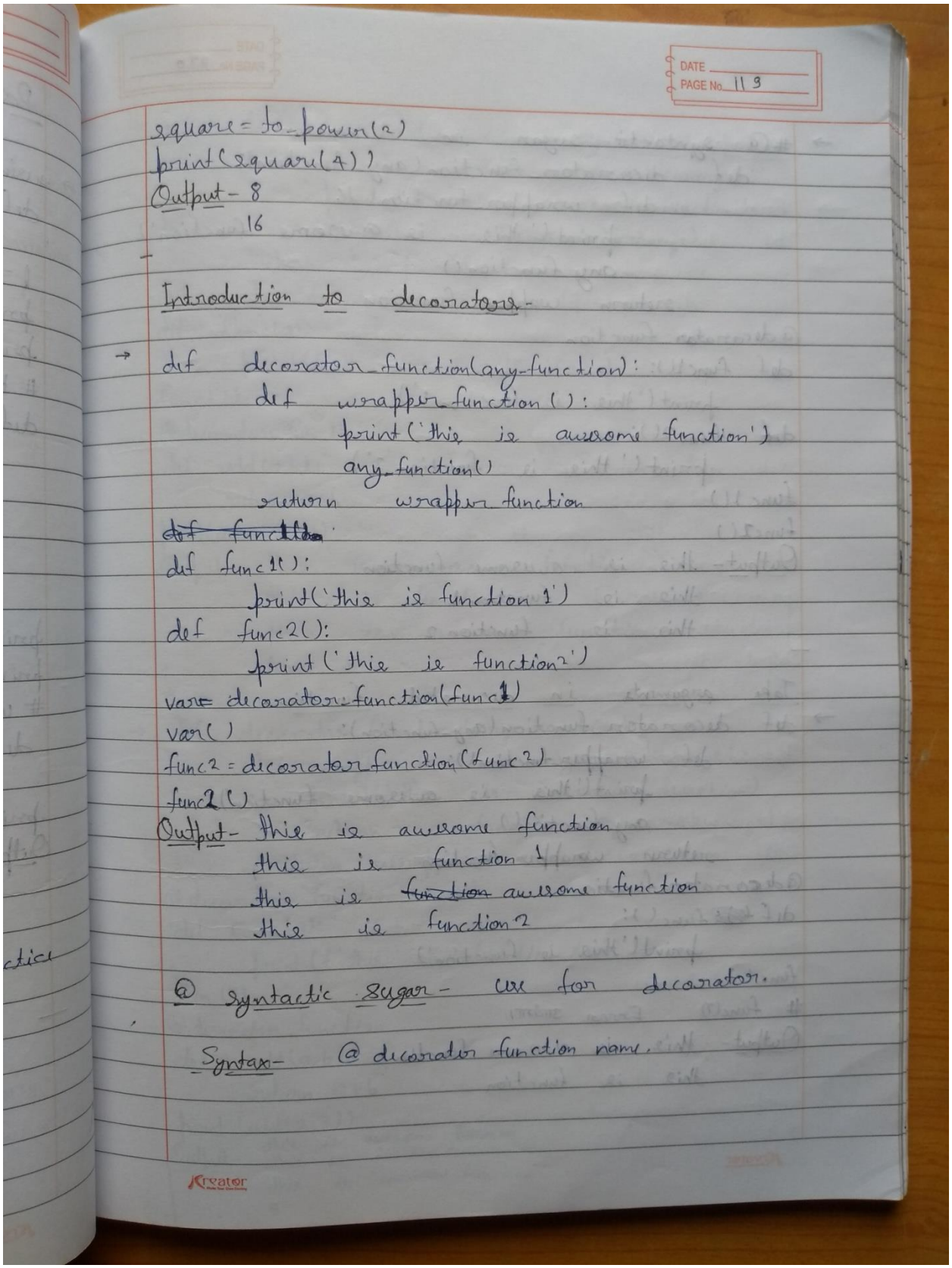
Function returning a function –

```
# function returning a function
def outer_func():
    def inner_func():
        print('inside inner func')
    return inner_func
var = outer_func()
var()
def outer_func2():
    def inner_func2():
        print('inside inner func')
    return inner_func2()
def outer_func3():
    def inner_func3():
        print(f'message is {msg}')
    return inner_func3
var3 = outer_func3("hello there!")
var3()
```

Output- inside inner func
message is hello there!

Closure Practice –

```
# function returning function (closure) practice
def to_power(x):
    def calc_power(n):
        return n***n
    return calc_power
cube = to_power(3)
print(cube(2))
```



```
square = to_power(2)
print(square(4))
Output - 8
         16
```

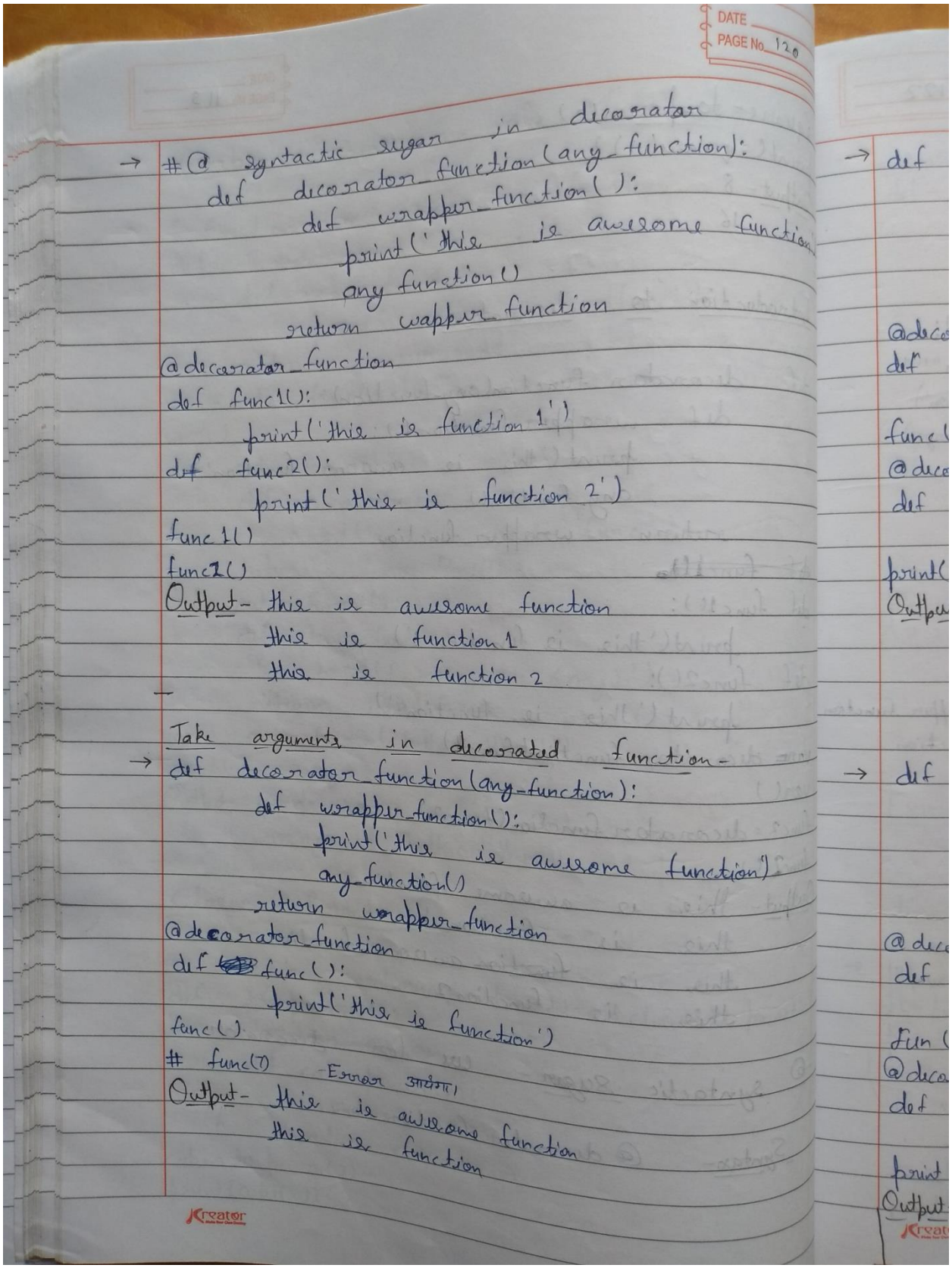
Introduction to decorators -

```
def decorator_function(any_function):
    def wrapper_function():
        print('this is awsome function')
        any_function()
    return wrapper_function
def func1():
    print('this is function 1')
def func2():
    print('this is function 2')
var = decorator_function(func1)
var()
func2 = decorator_function(func2)
func2()
```

Output - this is awsome function
this is function 1
this is awsome function
this is function 2

@ syntactic sugar - use for decorator.

Syntax - @ decorator function name.



@ syntactic sugar in decorator

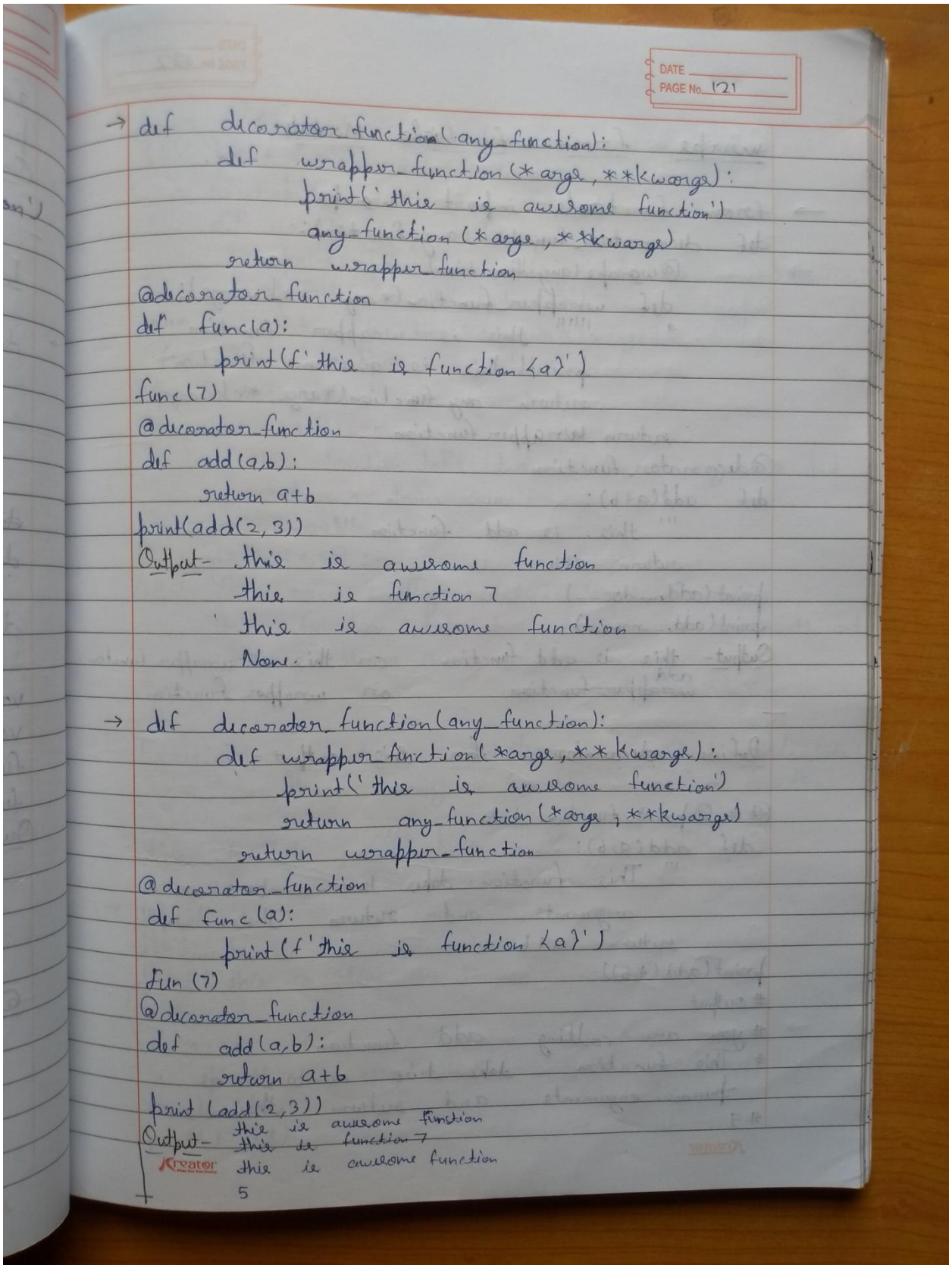
```
def decorator_function(any_function):
    def wrapper_function():
        print('this is awesome function')
        any_function()
    return wapper_function
@decorator_function
def func1():
    print('this is function 1')
def func2():
    print('this is function 2')
func1()
func2()
```

Output- this is awesome function
this is function 1
this is function 2

Take arguments in decorated function-

```
def decorator_function(any_function):
    def wrapper_function():
        print('this is awesome function')
        any_function()
    return wapper_function
@decorator_function
def func():
    print('this is function')
func()
# func(7)  Error आएगा।
```

Output- this is awesome function
this is function

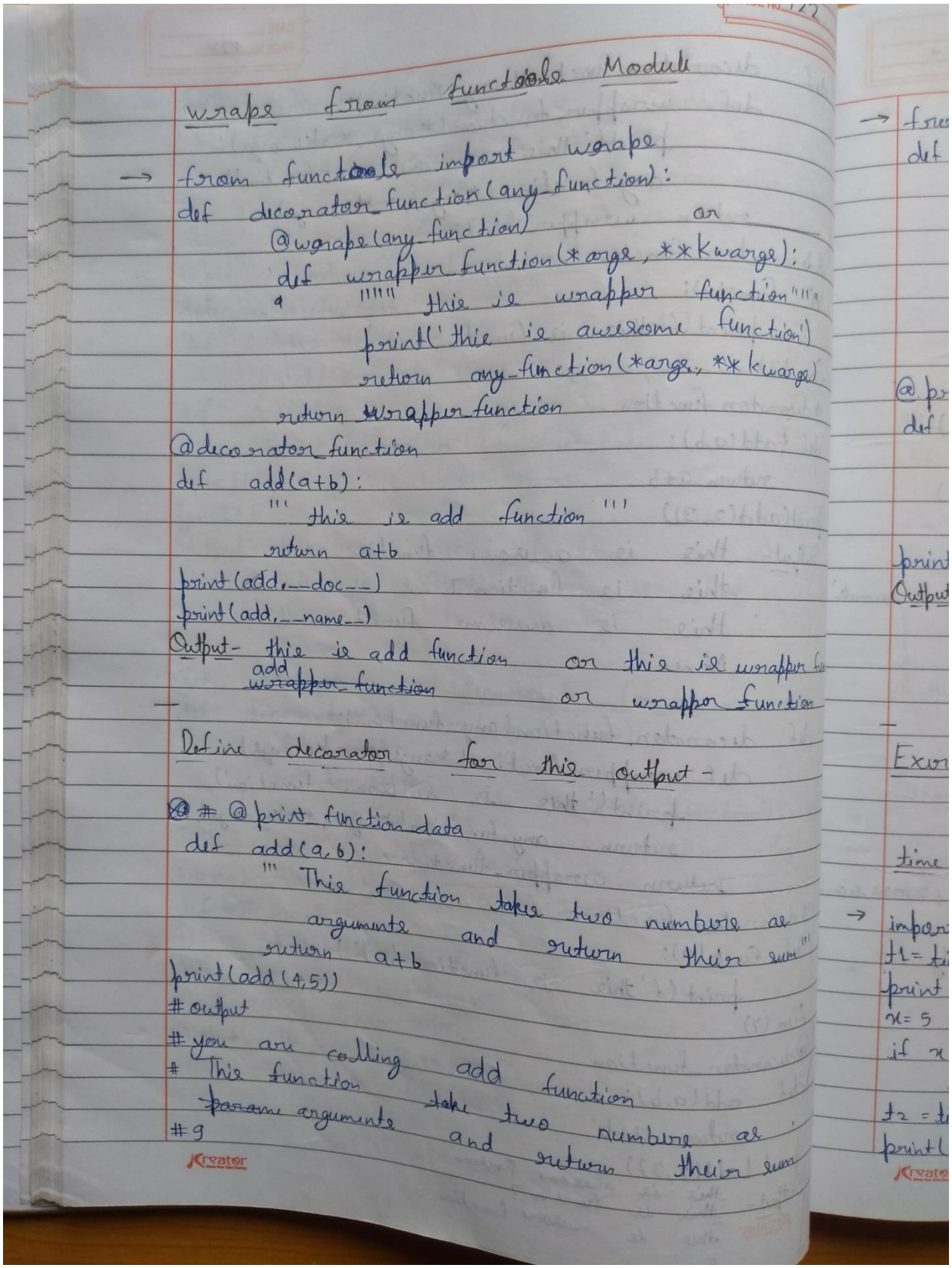


```
→ def decorator_function(any_function):
      def wrapper_function(*args, **kwargs):
          print('this is awesome function')
          any_function(*args, **kwargs)
      return wrapper_function
@decorator_function
def func(a):
    print(f'this is function {a}')
func(7)
@decorator_function
def add(a,b):
    return a+b
print(add(2,3))
```

Output- this is awesome function
this is function 7
this is awesome function
None.

```
→ def decorator_function(any_function):
      def wrapper_function(*args, **kwargs):
          print('this is awesome function')
          return any_function(*args, **kwargs)
      return wrapper_function
@decorator_function
def func(a):
    print(f'this is function {a}')
fun(7)
@decorator_function
def add(a,b):
    return a+b
print(add(2,3))
```

Output- this is awesome function
this is function 7
this is awesome function
5



wraps from functools Module

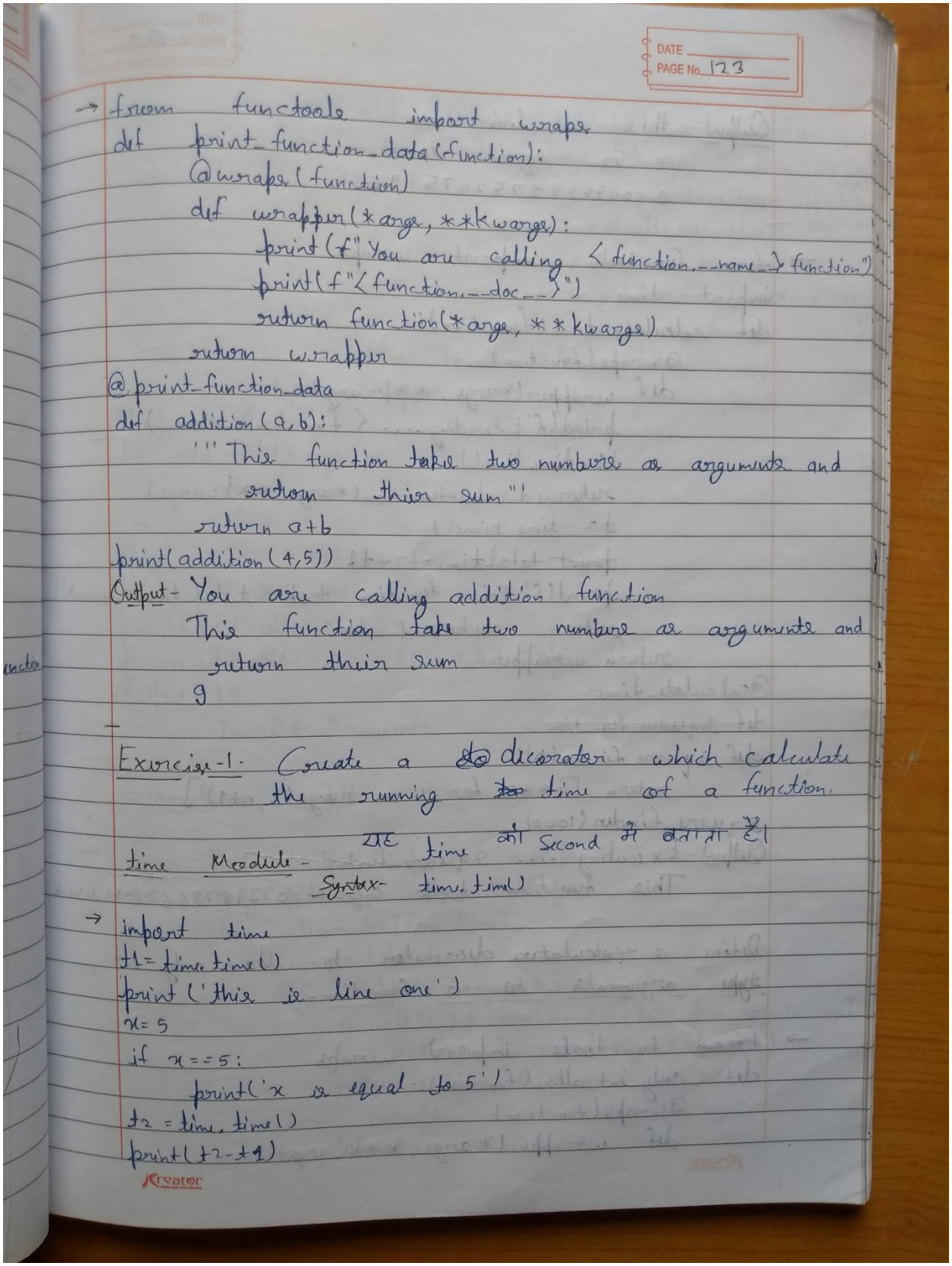
```
from functools import wraps
def decorator_function(any_function):
    @wraps(any_function)
    def wrapper_function(*args, **kwargs):
        """ this is wrapper function"""
        print('this is awesome function')
        return any_function(*args, **kwargs)
    return wrapper_function
@decorator_function
def add(a,b):
    ''' this is add function '''
    return a+b
print(add.__doc__)
print(add.__name__)
```

Output- this is add function
add

or this is wrapper function
or wrapper_function

Define decorator for this output -

```
# @print_function_data
def add(a,b):
    ''' This function takes two numbers as
        arguments and return their sum'''
    return a+b
print(add(4,5))
# output
# you are calling add function
# This function take two numbers as
  parameters arguments and return their sum
#9
```



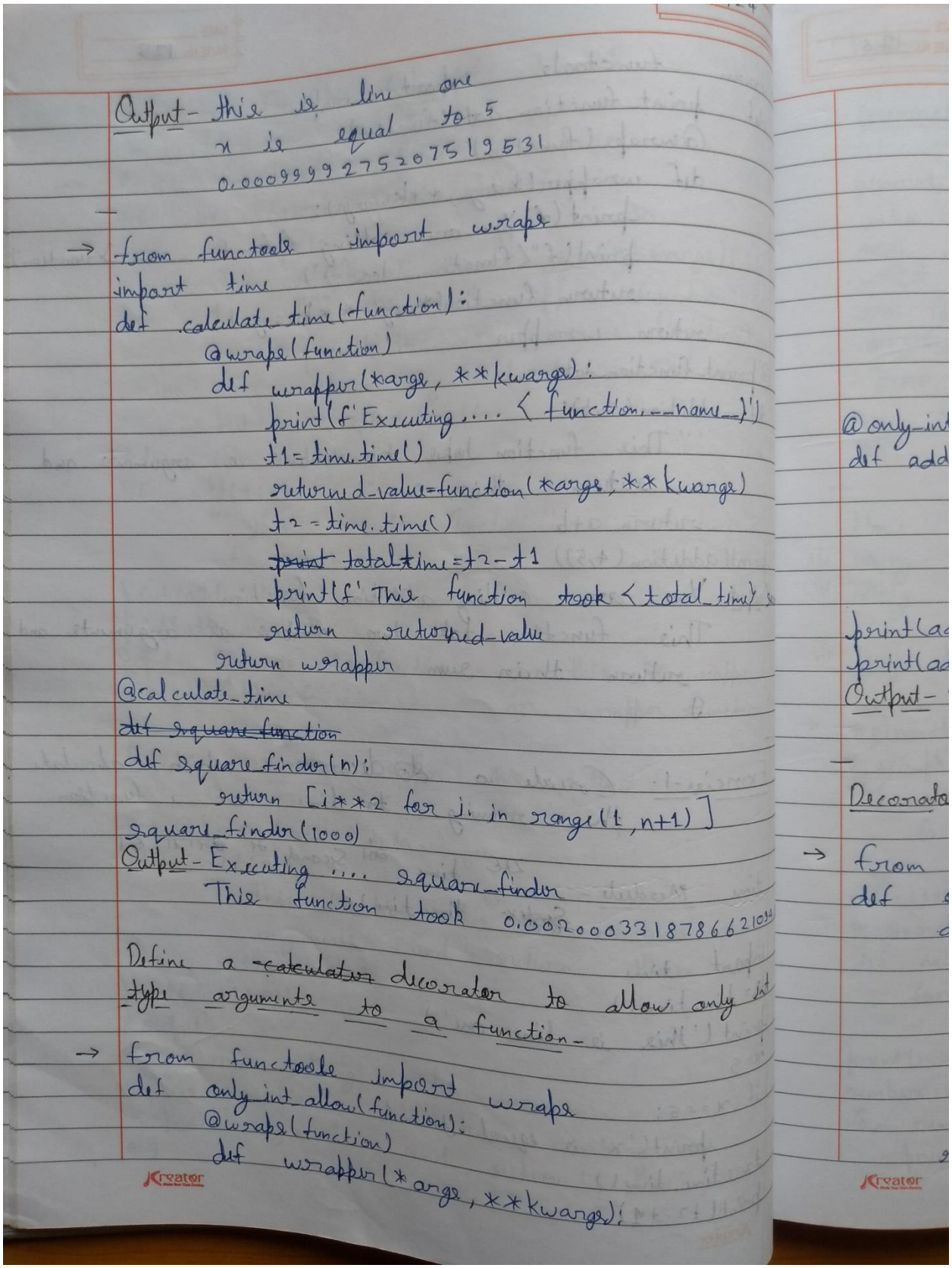
```
from functools import wraps
def print_function_data(function):
    @wraps(function)
    def wrapper(*args, **kwargs):
        print(f"You are calling <function.__name__> function")
        print(f"<function.__doc__>")
        return function(*args, **kwargs)
    return wrapper
@print_function_data
def addition(a,b):
    '''This function take two numbers as arguments and
       return their sum'''
    return a+b
print(addition(4,5))
```

Output- You are calling addition function
This function take two numbers as arguments and return their sum
9

Exercise-1. Create a decorator which calculate the running time of a function.

time Module - यह time को second में बताता है।
Syntax- time.time()

```
import time
t1 = time.time()
print('this is line one')
x = 5
if x == 5:
    print('x is equal to 5')
t2 = time.time()
print(t2-t1)
```



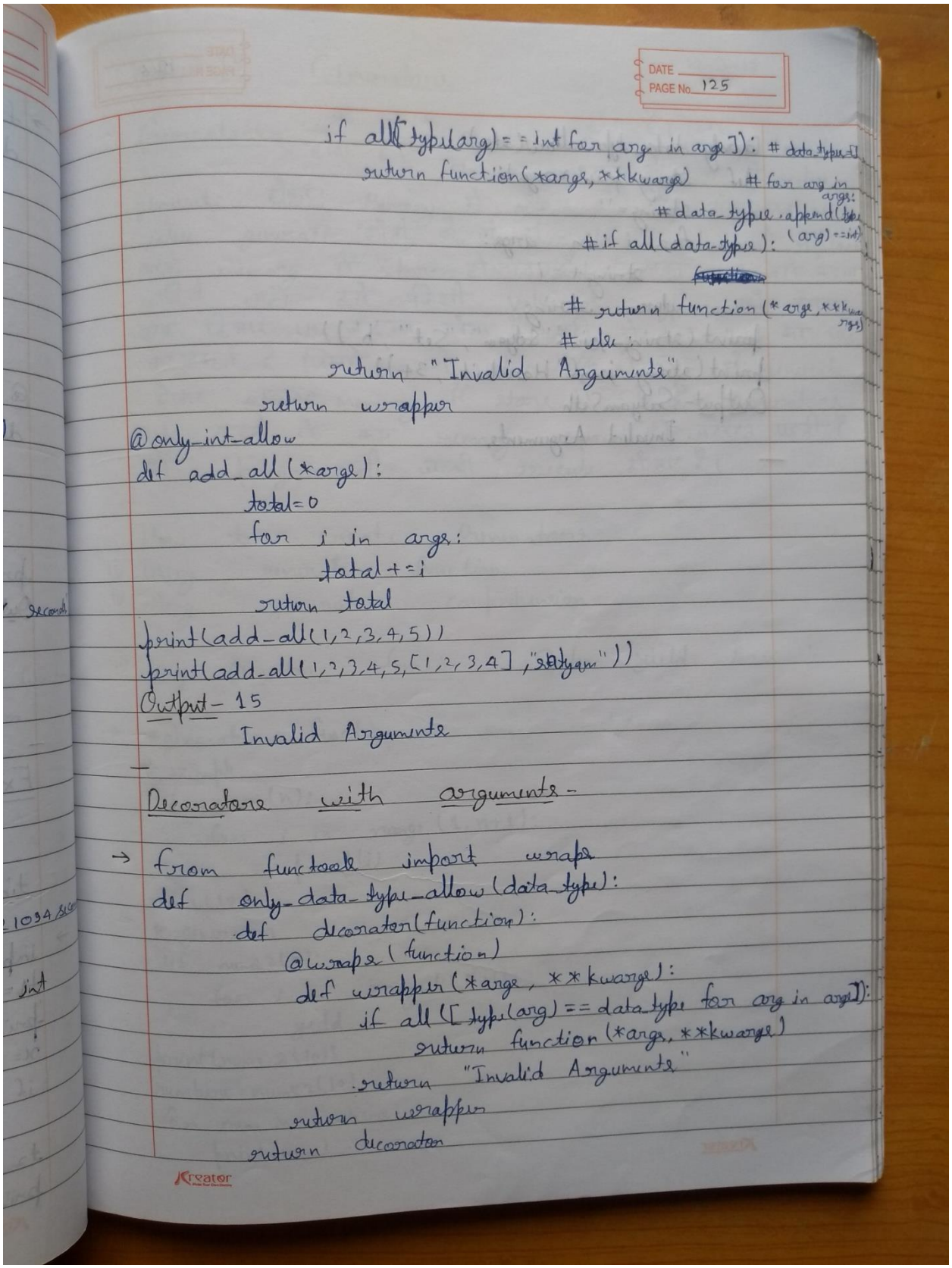
Output- this is line one
x is equal to 5
0.0009999275207519531

→
```
from functools import wraps
import time
def calculate_time(function):
    @wraps(function)
    def wrapper(*args, **kwargs):
        print(f'Executing .... {function.__name__}')
        t1 = time.time()
        returned_value = function(*args, **kwargs)
        t2 = time.time()
        total_time = t2 - t1
        print(f'This function took {total_time}')
        return returned_value
    return wrapper
@calculate_time
def square_finder(n):
    return [i**2 for i in range(1, n+1)]
square_finder(1000)
```

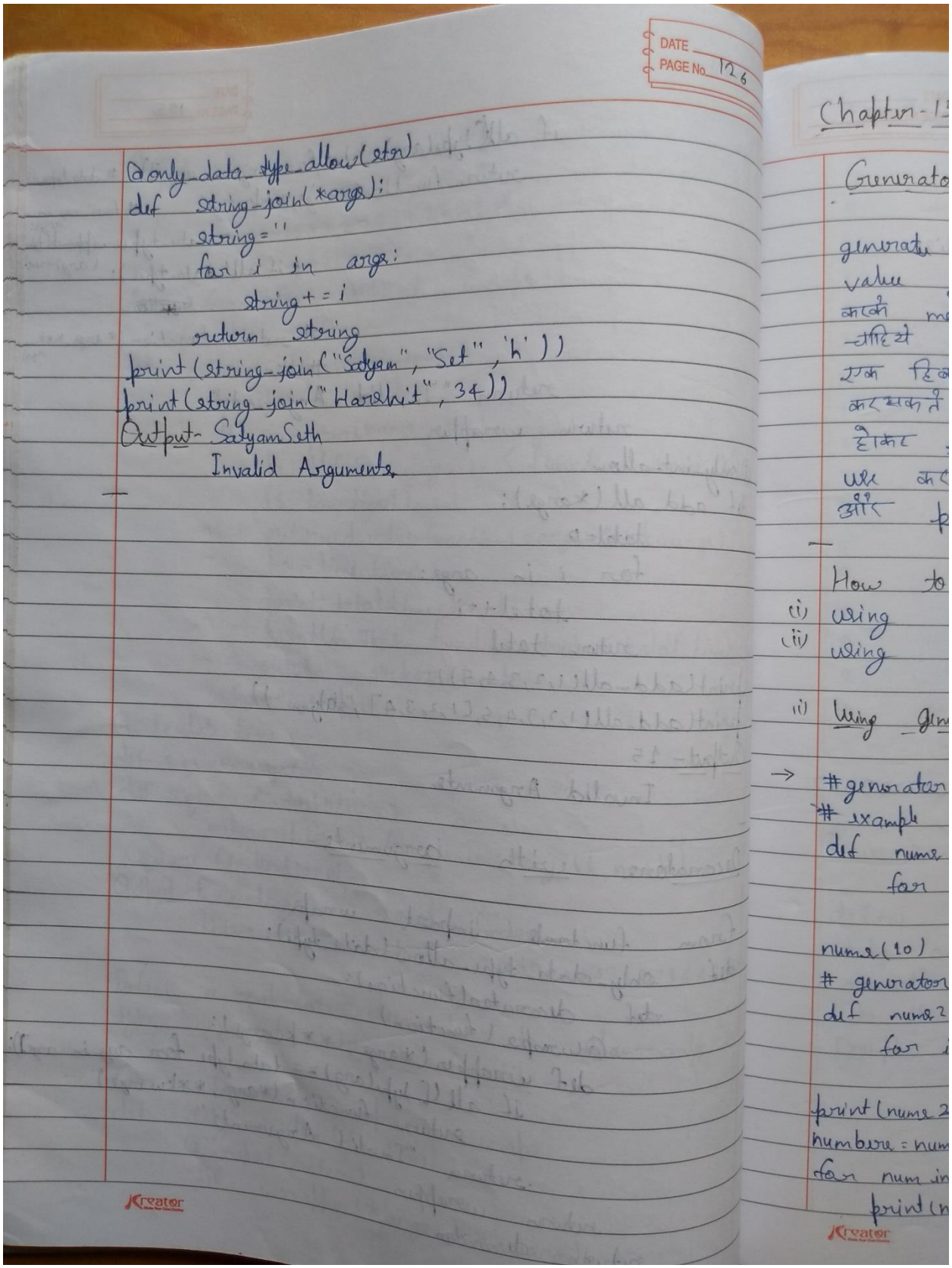
Output- Executing .... square_finder
This function took 0.0020003318786621094

Define a decorator to allow only int type arguments to a function-

→
```
from functools import wraps
def only_int_allow(function):
    @wraps(function)
    def wrapper(*args, **kwargs):
```

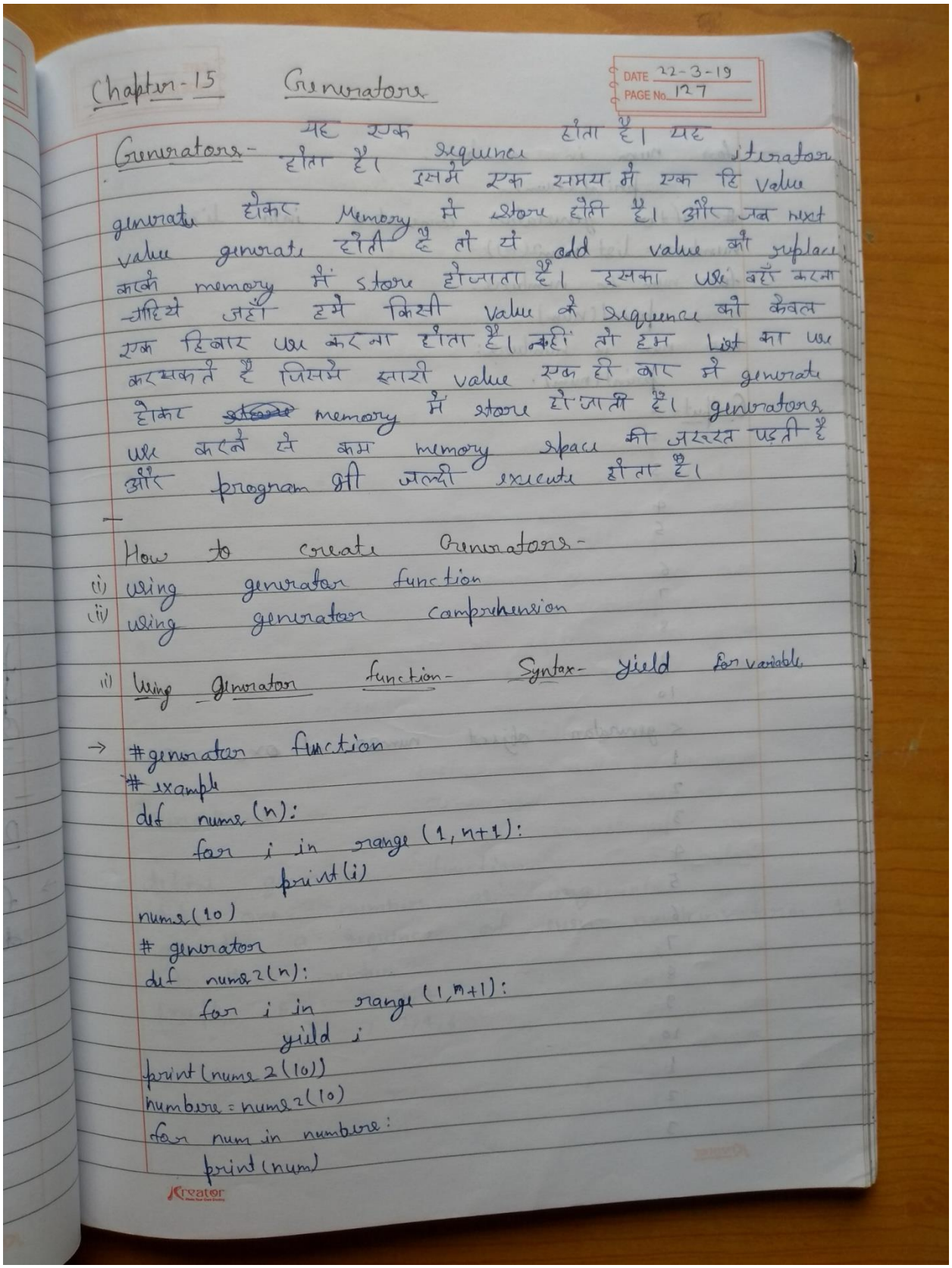


```
DATE
PAGE No. 125

        if all([type(arg) == int for arg in args]): # data_types=[]
            return function(*args, **kwargs)        # for arg in args:
                                                    #     data_type.append(type(arg))
                                                    # if all(data_type):
                                                    #     return function(*args, **kwargs)
                                                    # else:
            return "Invalid Arguments"
        return wrapper
@only_int_allow
def add_all(*args):
    total=0
    for i in args:
        total+=i
    return total
print(add_all(1,2,3,4,5))
print(add_all(1,2,3,4,5,[1,2,3,4],"satyam"))
Output- 15
        Invalid Arguments

Decorators with arguments -

→ from functools import wraps
def only_data_type_allow(data_type):
    def decorator(function):
        @wraps(function)
        def wrapper(*args, **kwargs):
            if all([type(arg) == data_type for arg in args]):
                return function(*args, **kwargs)
            return "Invalid Arguments"
        return wrapper
    return decorator
```



```
@only_data_type_allow(str)
def string_join(*args):
    string = ''
    for i in args:
        string += i
    return string
print(string_join("Satyam", "Set", 'h'))
print(string_join("Harshit", 34))
```

Output- SatyamSeth
Invalid Arguments.



Chapter-15 Generators

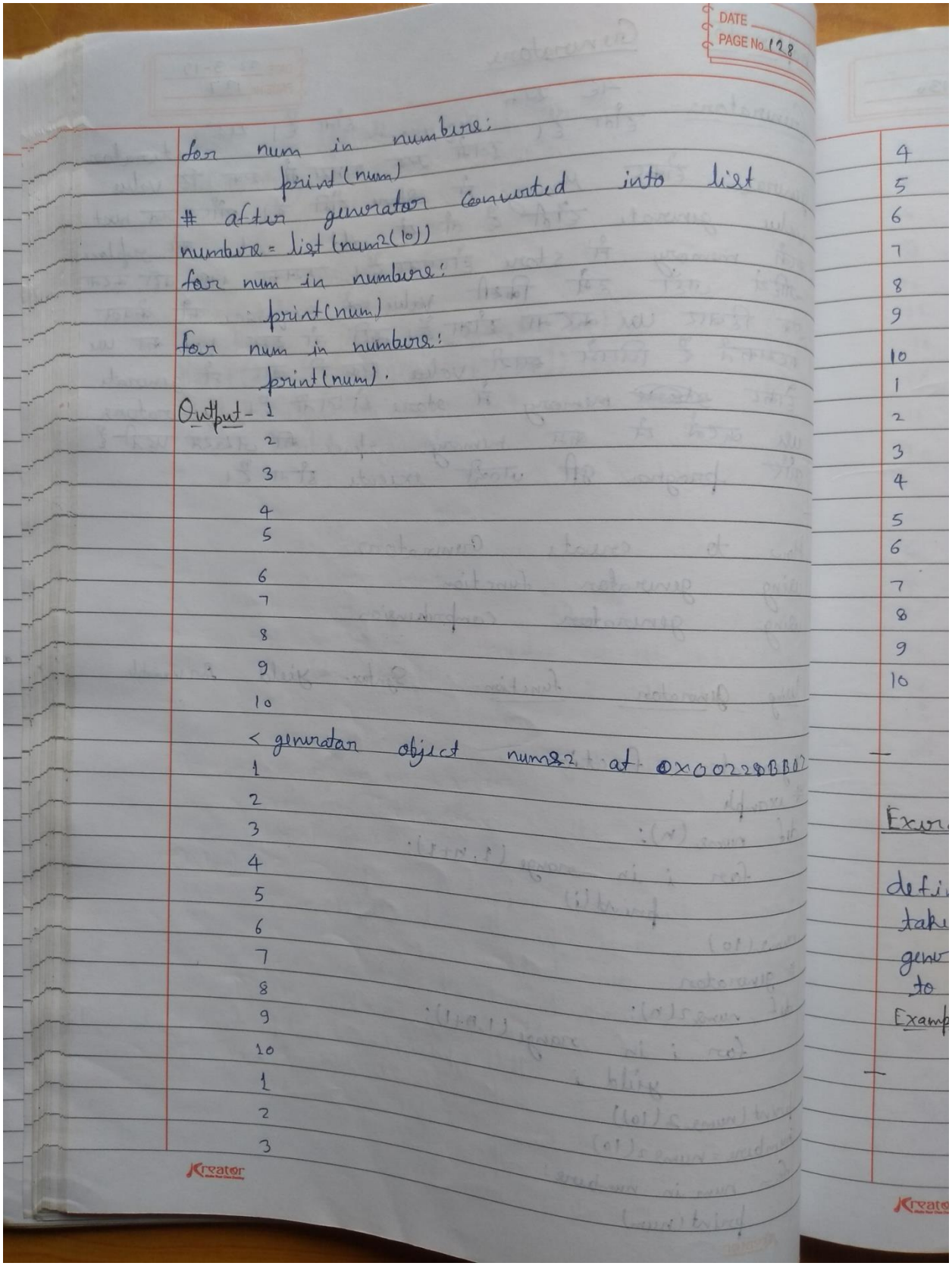
DATE 22-3-19
PAGE No. 127

Generators- यह एक sequence होता है। यह iterator होता है। इसमें एक समय में एक ही value generate होकर Memory में store होती है। और जब next value generate होती है तो ये old value को replace करके memory में store होजाता है। इसका use वहाँ करना चाहिये जहाँ हमें किसी value के sequence को केवल एक बार use करना होता है। नहीं तो हम List का use करसकते हैं जिसमें सारी value एक ही बार में generate होकर memory में store होजाती है। generators use करने से कम memory space की जरूरत पड़ती है और program भी जल्दी execute होता है।

How to create Generators-
(i) using generator function
(ii) using generator comprehension

ii) Using Generator function- Syntax- yield for variable

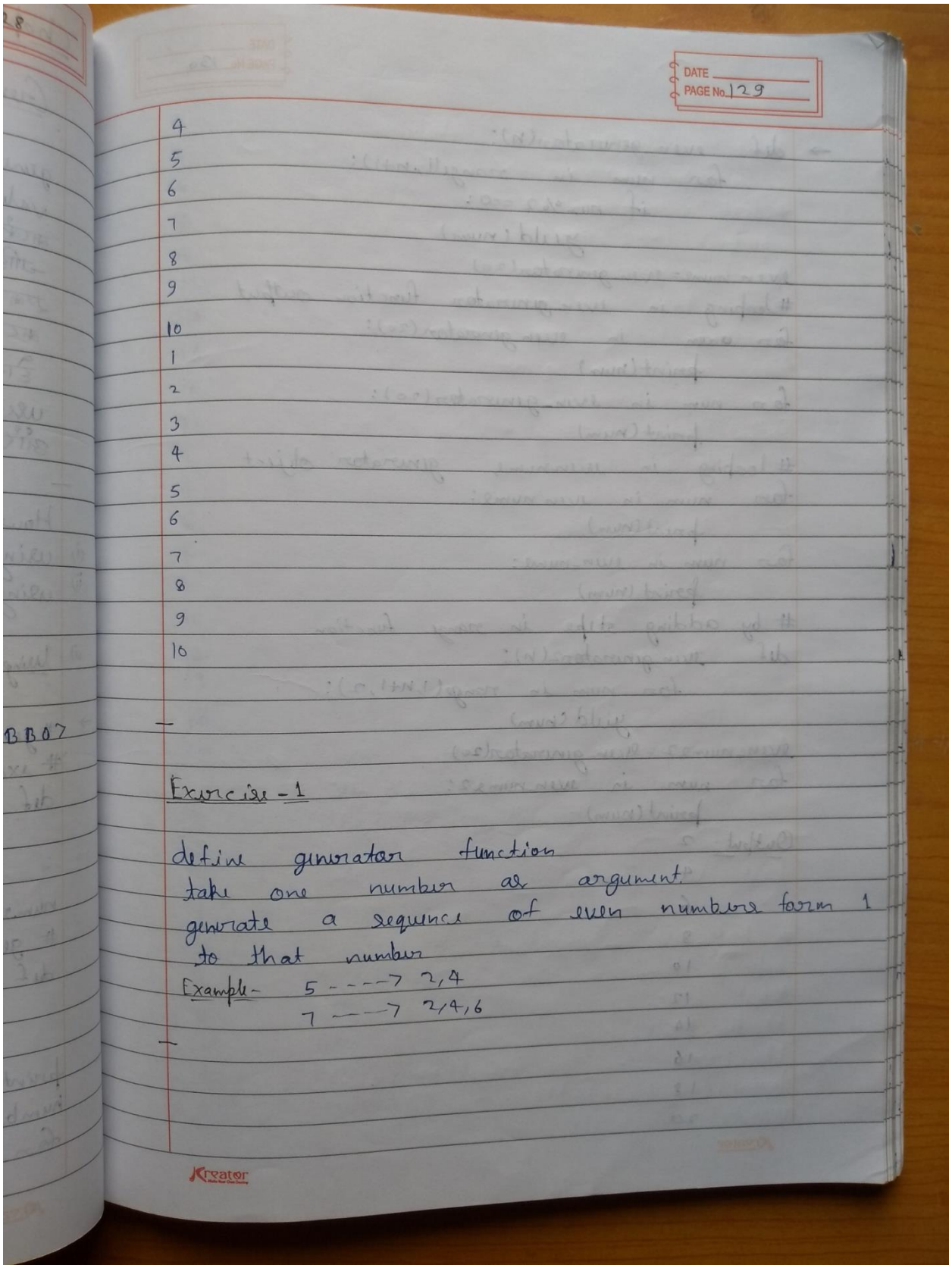
→
```
#genrator function
# example
def nums(n):
    for i in range(1,n+1):
        print(i)
nums(10)
# genrator
def nums2(n):
    for i in range(1,n+1):
        yield i
print(nums2(10))
numbers = nums2(10)
for num in numbers:
    print(num)
```

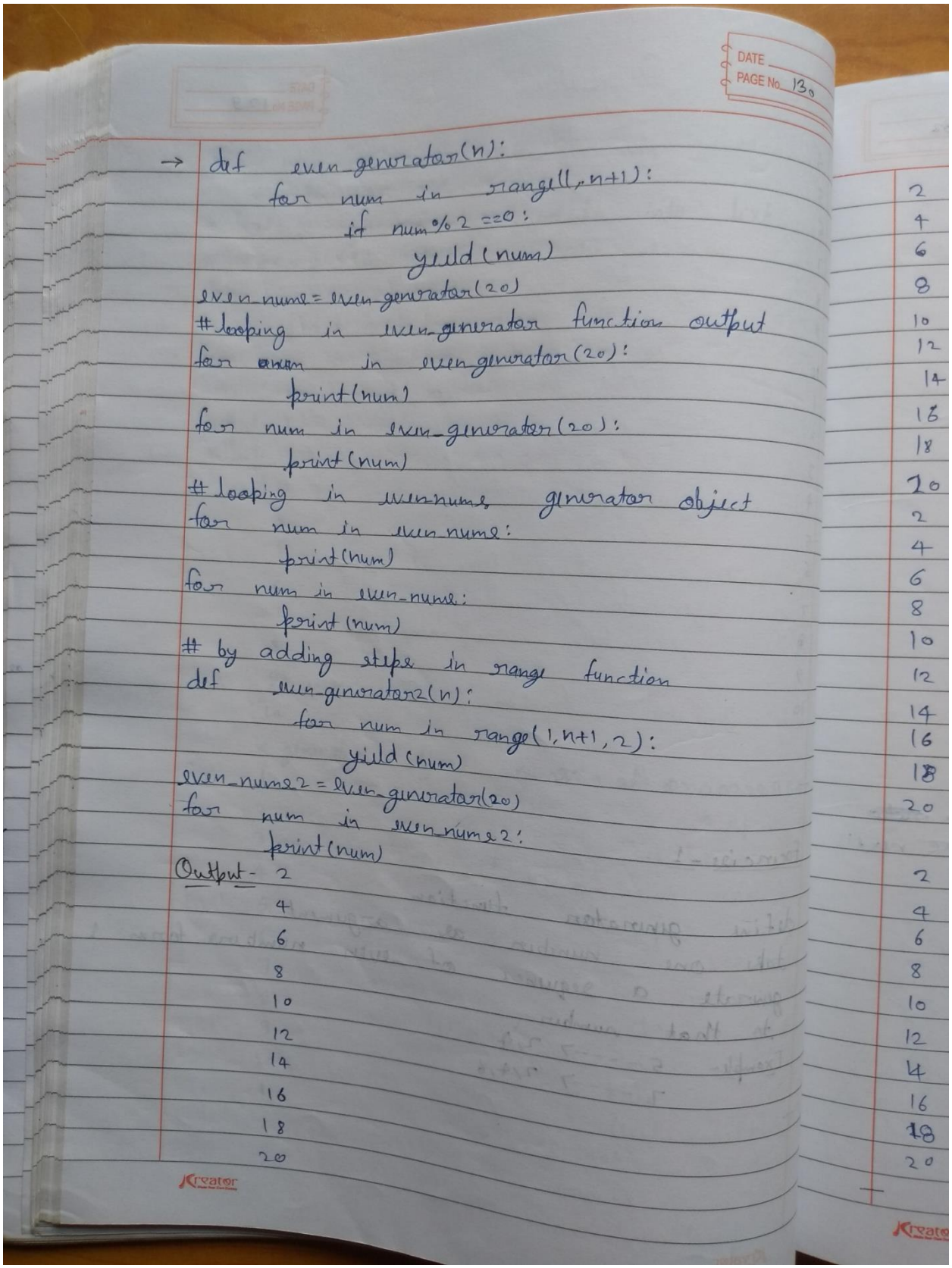


```
for num in numbers:
    print(num)
# after generator converted into list
numbers = list(num2(10))
for num in numbers:
    print(num)
for num in numbers:
    print(num).
```

Output -

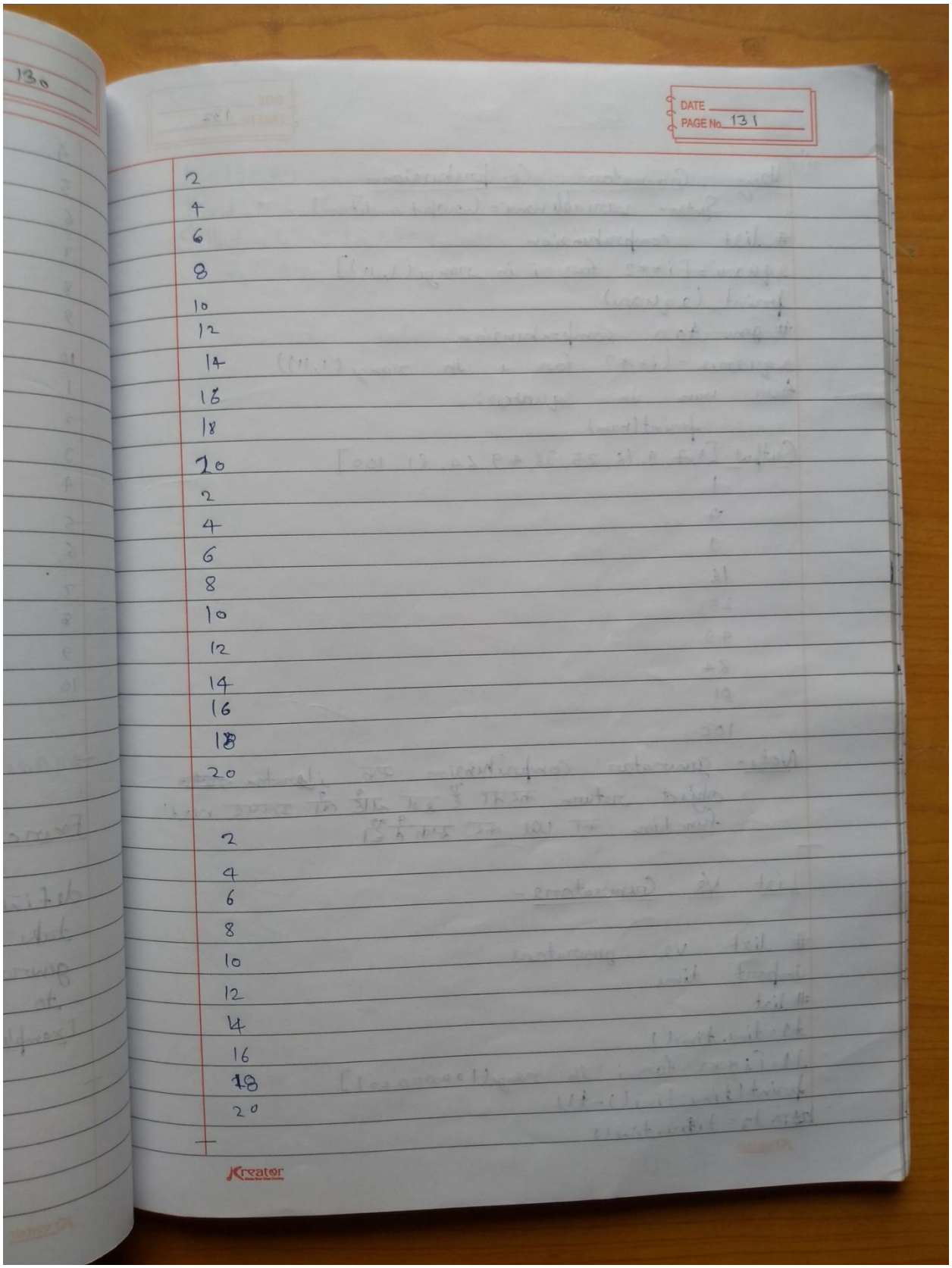
```
1
2
3
4
5
6
7
8
9
10
<generator object num22 at 0x00228BBB2
1
2
3
4
5
6
7
8
9
10
1
2
3
```

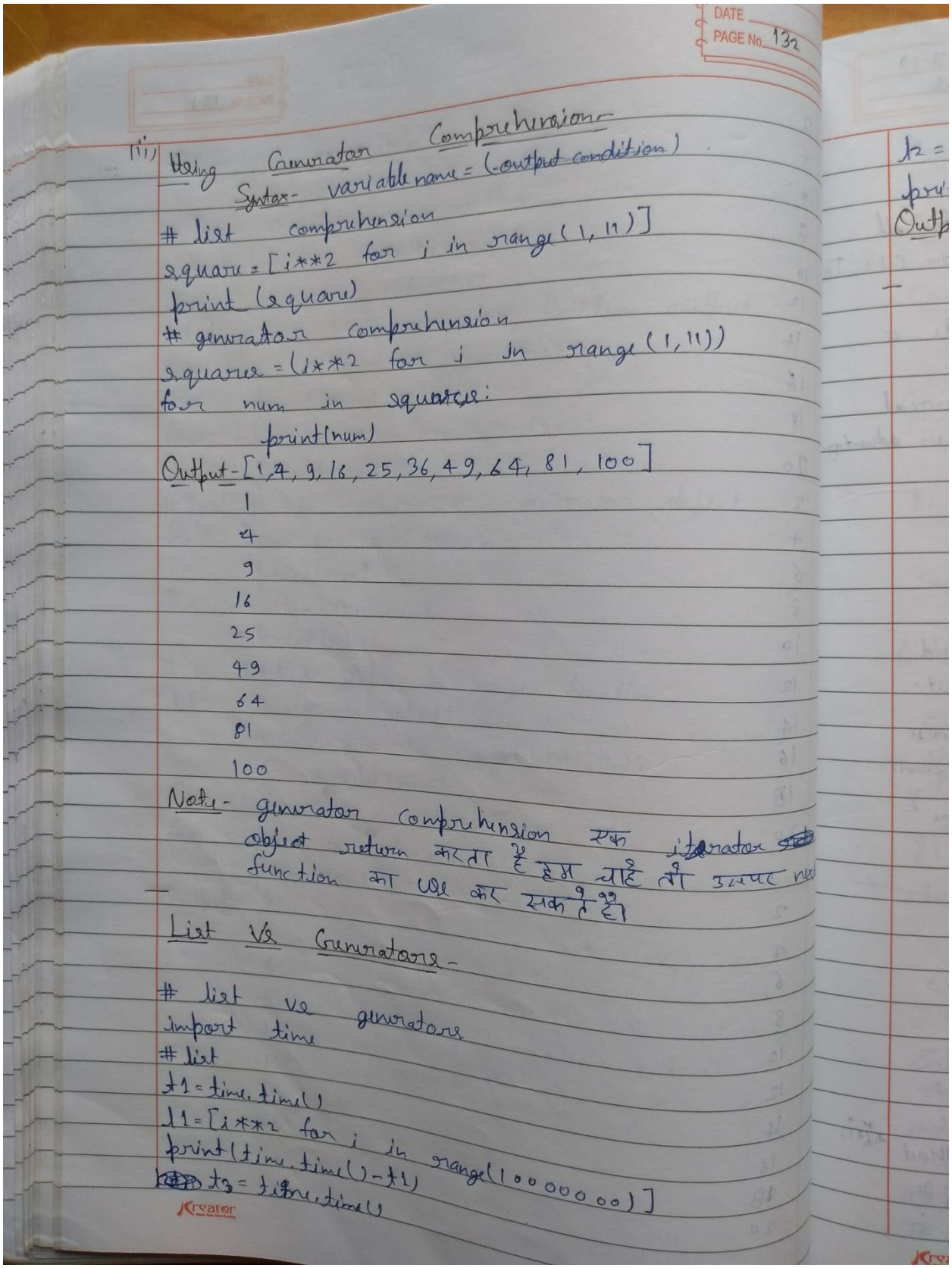

DATE
PAGE No. 129
4
5
6
7
8
9
10
1
2
3
4
5
6
7
8
9
10
BBO7
Exercise - 1
define generator function
take one number as argument
generate a sequence of even numbers form 1
to that number
Example- 5 ----> 2,4
7 ----> 2,4,6
Kreator



```
→ def even_generator(n):
      for num in range(1, n+1):
          if num%2 ==0:
              yield(num)
even_nums = even_generator(20)
#looping in even_generator function output
for even in even_generator(20):
    print(num)
for num in even_generator(20):
    print(num)
#looping in even_nums generator object
for num in even_nums:
    print(num)
for num in even_nums:
    print(num)
# by adding steps in range function
def even_generator2(n):
    for num in range(1, n+1, 2):
        yield(num)
even_nums2 = even_generator(20)
for num in even_nums2:
    print(num)
```

Output-
2
4
6
8
10
12
14
16
18
20


DATE
PAGE No. 131
2
4
6
8
10
12
14
16
18
20
2
4
6
8
10
12
14
16
18
20



(11) Using Generator Comprehension-

Syntax- variable name = (output condition)

```
# list comprehension
square = [i**2 for i in range(1,11)]
print(square)
# generator comprehension
squares = (i**2 for i in range(1,11))
for num in squares:
    print(num)
```

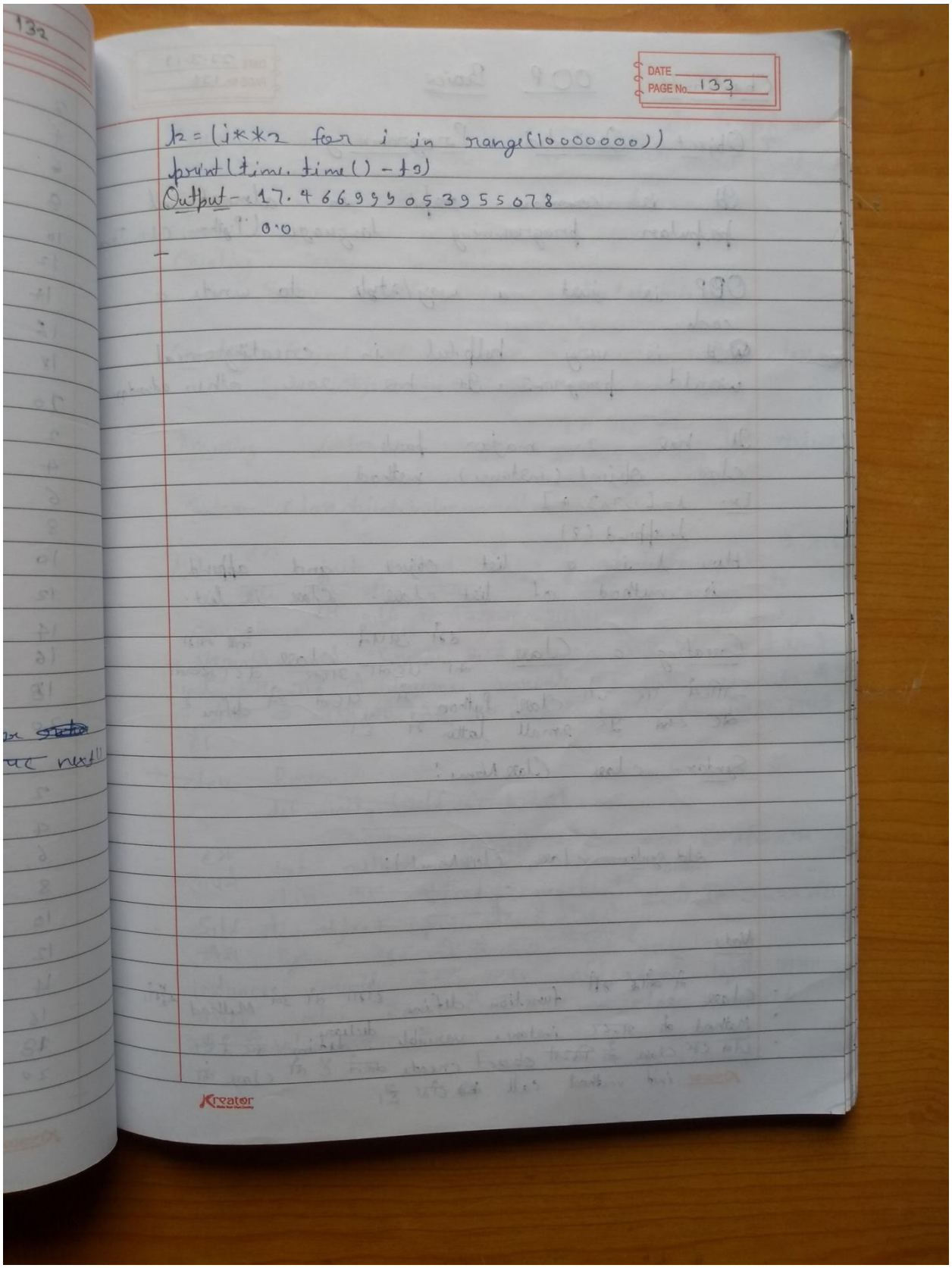
Output- [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

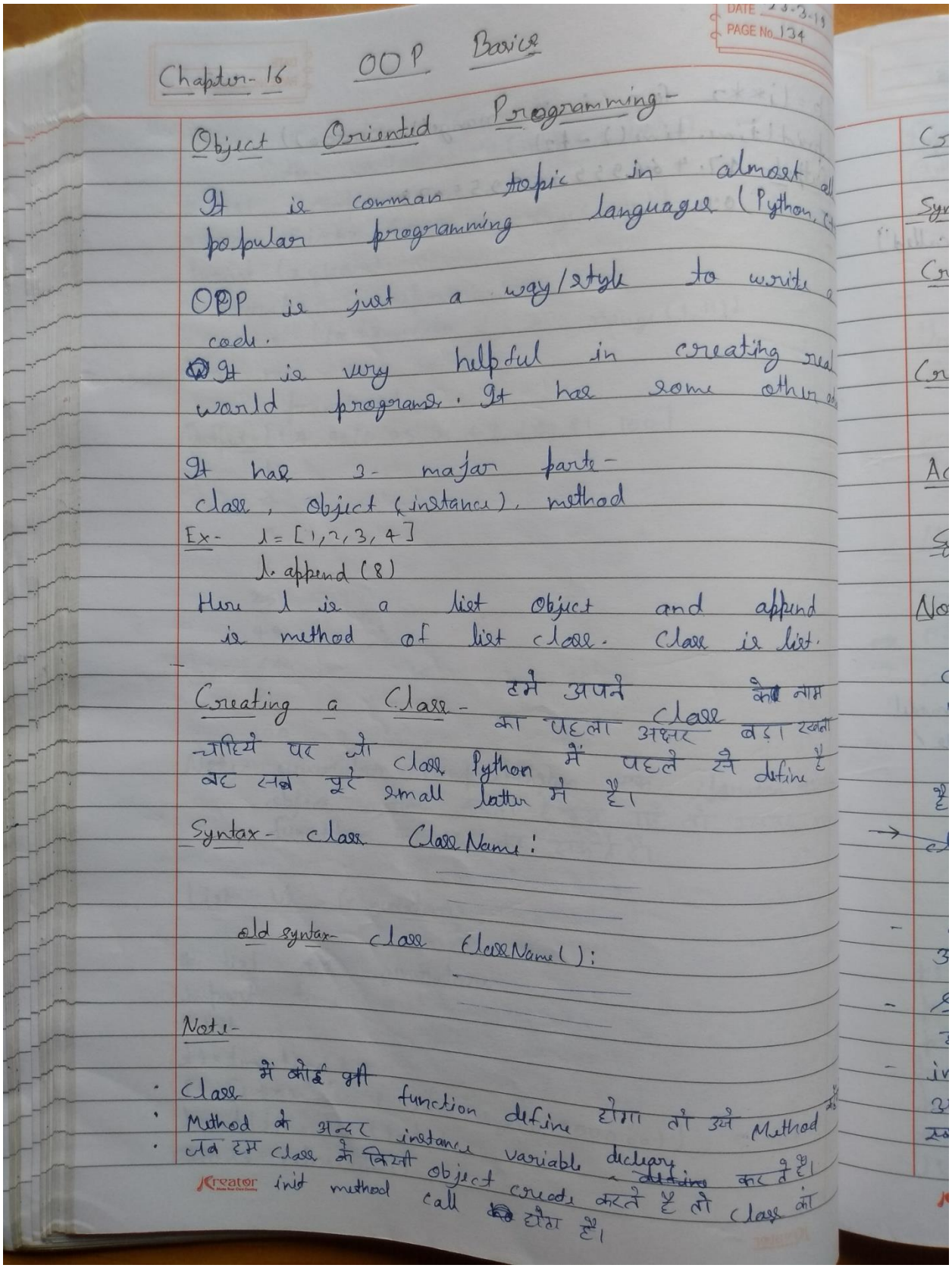
1
4
9
16
25
49
64
81
100

Note- generator comprehension एक iterator object return करता है हम चाहें तो उसपर next function का use कर सकते हैं

List Vs Generators-

```
# list vs generators
import time
# list
t1 = time.time()
l1 = [i**2 for i in range(10000000)]
print(time.time()-t1)
t2 = time.time()
```


DATE
PAGE No. 133
t2 = (i**2 for i in range(10000000))
print(time.time() - t2)
Output - 17.466995053955078
0.0
Kreator



Chapter-16 OOP Basics

DATE 23-3-19
PAGE No. 134

Object Oriented Programming-

It is comman topic in almost all popular programming languages (Python, C++

OOP is just a way/style to write a code.

It is very helpful in creating real world programs. It has some other advantages

It has 3- major parts-
class, object (instance), method

Ex- l = [1, 2, 3, 4]
l.append(8)

Here l is a list object and append is method of list class. Class is list.

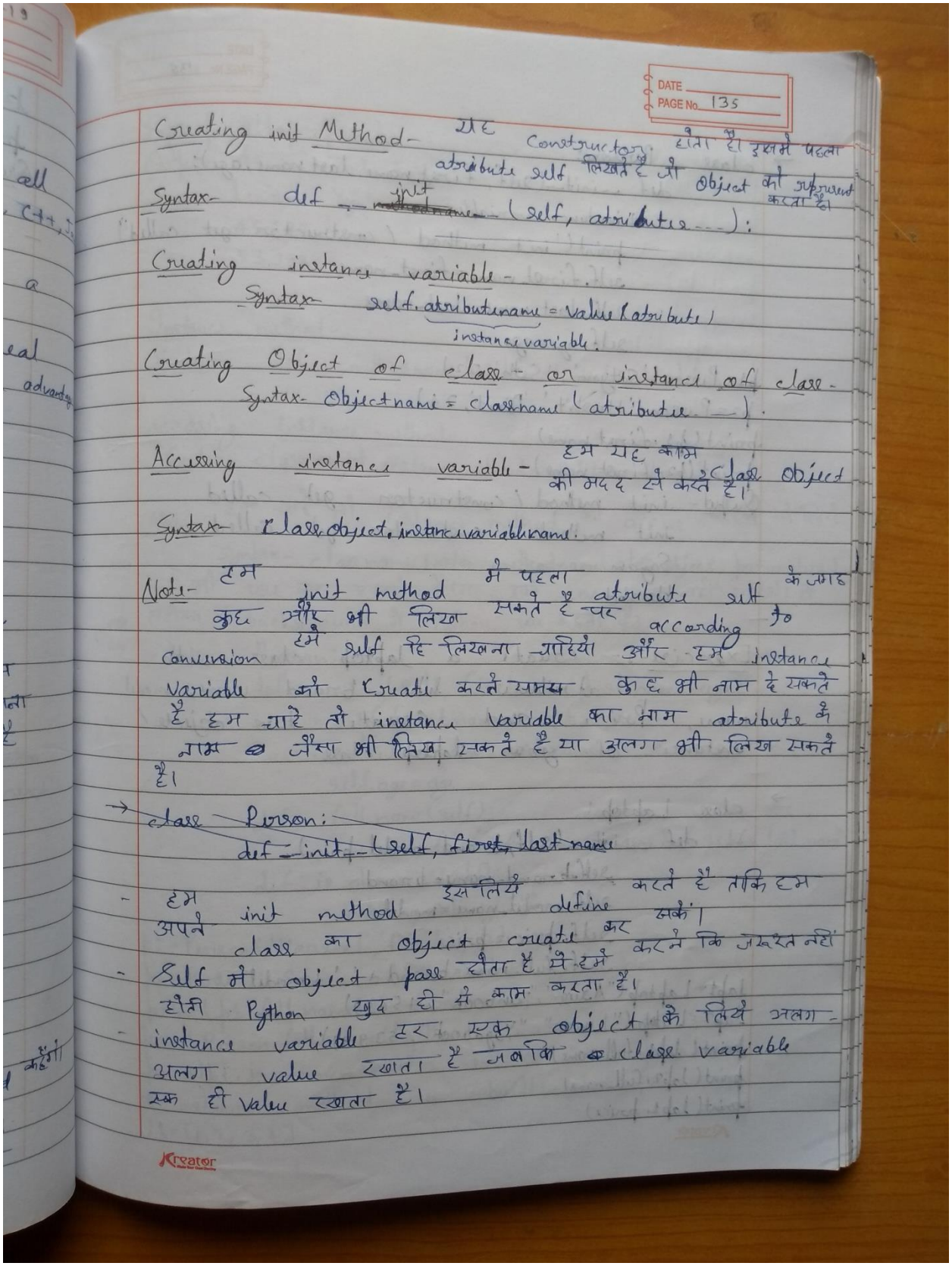
Creating a Class- हमें अपने class का पहला अक्षर बड़ा रखना चाहिये पर तो class Python में पहले से define है वह सब छोटे small letter में है।

Syntax- class ClassName:

old syntax- class ClassName():

Note-

- Class में कोई भी function define होगा तो उसे Method कहेंगे
- Method के अन्दर instance variable declare करते हैं।
- जब हम class के किसी object create करते हैं तो class का init method call होता है।



DATE
PAGE No. 135

Creating init Method- यह Constructor होता है इसमें पहला attribute self लिखते हैं जो Object को represent करता है।

Syntax- def __init__(self, attributes...):

Creating instance variable-

Syntax- self.attributename = value (attribute)
instance variable.

Creating Object of class - or instance of class-

Syntax- Objectname = classname (attributes...).

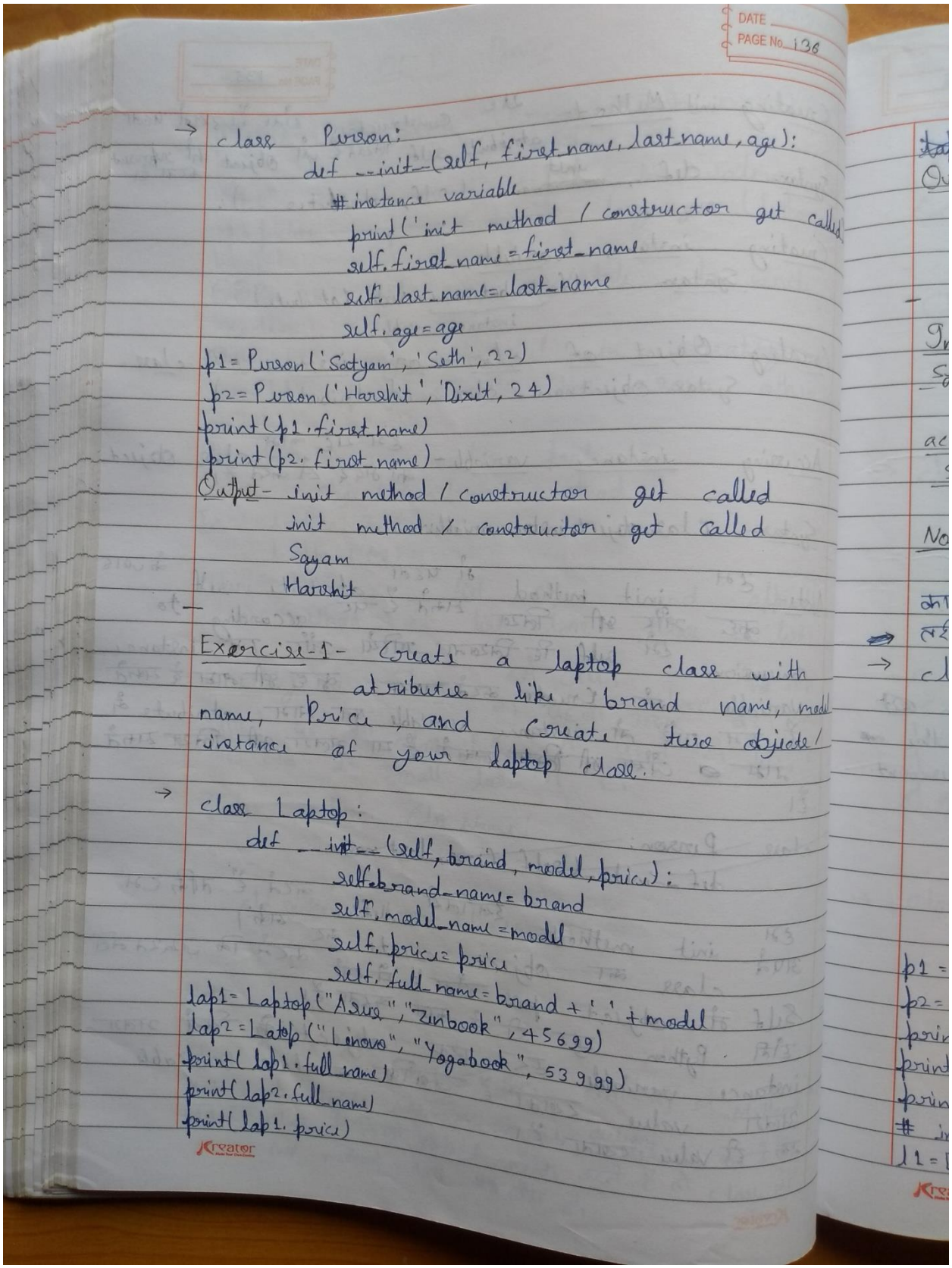
Accessing instance variable - हम यह काम class object की मदद से करते हैं।

Syntax- class object.instancevariablename.

Note- हम init method में पहला attribute self के जगह कुछ और भी लिख सकते हैं पर according to Convension हमें self ही लिखना चाहिये और हम instance Variable को Create करते समय कुछ भी नाम दे सकते हैं हम चाहे तो instance variable का नाम attribute के नाम के जैसा भी लिख सकते हैं या अलग भी लिख सकते हैं।

→ ~~class Person:~~
~~def __init__(self, first, lastname~~

- हम अपने init method इसलिये define करते हैं ताकि हम class का object create कर सकें।
- Self में object pass होता है जो हमें करने कि जरूरत नहीं होती Python खुद ही ये काम करता है।
- instance variable हर एक object के लिये अलग-अलग value रखता है जबकि class variable एक ही value रखता है।

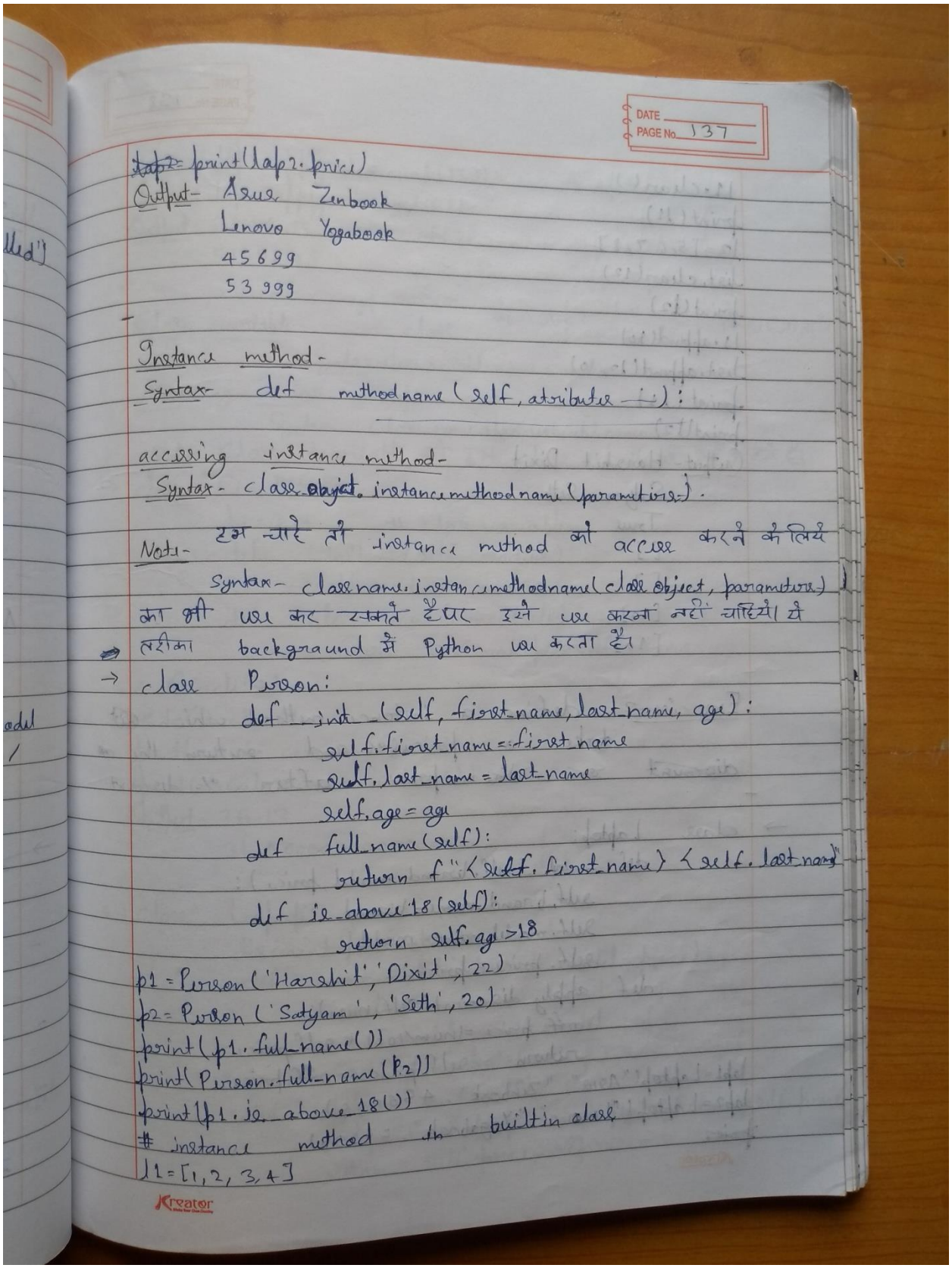


```
class Person:
    def __init__(self, first_name, last_name, age):
        # instance variable
        print('init method / constructor get called')
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
p1 = Person('Satyam', 'Seth', 22)
p2 = Person('Harshit', 'Dixit', 24)
print(p1.first_name)
print(p2.first_name)
```

Output - init method / constructor get called
init method / constructor get called
Satyam
Harshit

Exercise-1 - Create a laptop class with attributes like brand name, model name, Price, and Create two objects/ instance of your laptop class.

```
class Laptop:
    def __init__(self, brand, model, price):
        self.brand_name = brand
        self.model_name = model
        self.price = price
        self.full_name = brand + ' ' + model
lap1 = Laptop("Asus", "Zenbook", 45699)
lap2 = Laptop("Lenovo", "Yogabook", 53999)
print(lap1.full_name)
print(lap2.full_name)
print(lap1.price)
```



DATE
PAGE No. 137

~~lap2~~ = print(lap2.price)
Output- Asus Zenbook
Lenovo Yogabook
45699
53999

Instance method-
Syntax- def methodname(self, atributes ...):

accessing instance method-
Syntax- class object. instancemethodname(parameters).

Note- इस तरीके से instance method को access करने के लिये
Syntax- classname.instancemethodname(class object, parameters)
का भी use कर सकते हैं पर इसे use करना नहीं चाहिये। ये
⇒ तरीका background में Python use करता है।

→
```
class Person:
    def __init__(self, first_name, last_name, age):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
    def full_name(self):
        return f"{self.first_name} {self.last_name}"
    def is_above_18(self):
        return self.age > 18
p1 = Person('Harshit', 'Dixit', 22)
p2 = Person('Satyam', 'Seth', 20)
print(p1.full_name())
print(Person.full_name(p2))
print(p1.is_above_18())
# instance method in builtin class
l1 = [1, 2, 3, 4]
```

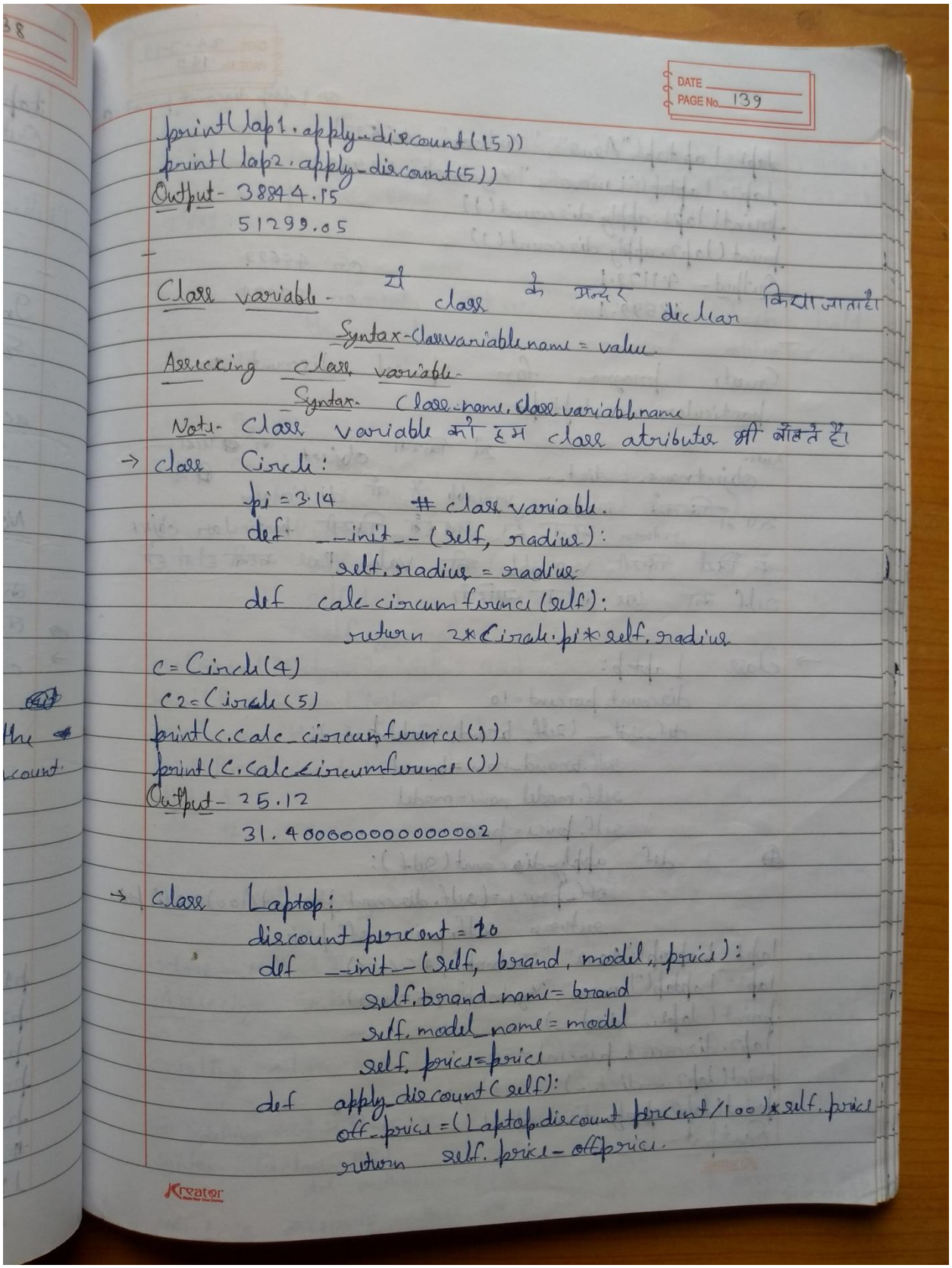
Kreator

```
l1.clear()
print(l1)
l2=[5,6,7,8]
list.clear(l2)
print(l2)
l1.append(10)
list.append(l2,10)
print(l1)
print(l2)
```

Output- Harshit Dixit
Satyam Seth
True
[]
[]
[10]
[10]

Exercise-2- define a instance method which take a price% and return the ~~discount on the~~ new price after % discount

```
class Laptop:
    def __init__(self,brand,model,price):
        self.brand_name=brand
        self.model_name=model
        self.price=price
    def apply_discount(self,num):
        off_price=(num/100)*self.price
        return self.price-off_price
lap1=Laptop("Asus","Zenbook",45999)
lap2=Laptop("Lenovo","Yogabook",53999)
```

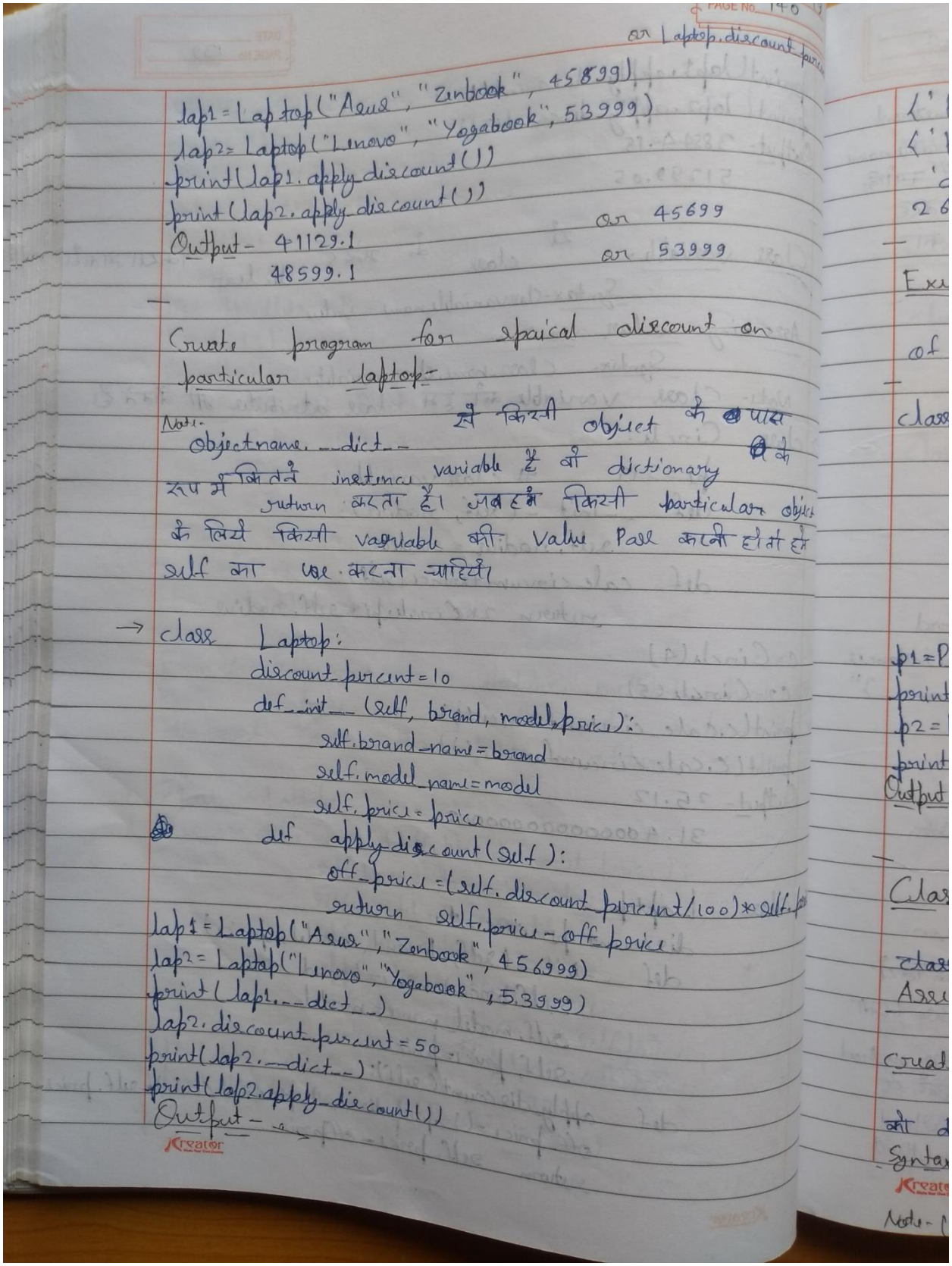


print(lap1.apply_discount(15))
print(lap2.apply_discount(5))
Output- 3884.15
51299.05

Class variable- ये class के अन्दर declear किया जाता है
Syntax- Classvariablename = value
Accessing class variable-
Syntax- Class_name.Class_variablename
Note- Class variable को हम class atributes भी बोलते हैं

→ class Circle:
    pi = 3.14 # class variable.
    def __init__(self, radius):
        self.radius = radius
    def calc_circumfrunce(self):
        return 2*Circle.pi*self.radius
c = Circle(4)
c2 = Circle(5)
print(c.calc_circumfrunce())
print(c.calc_circumfrunce())
Output- 25.12
31.400000000000002

→ class Laptop:
    discount_percent = 10
    def __init__(self, brand, model, price):
        self.brand_name = brand
        self.model_name = model
        self.price = price
    def apply_discount(self):
        off_price = (Laptop.discount_percent/100)*self.price
        return self.price - off_price



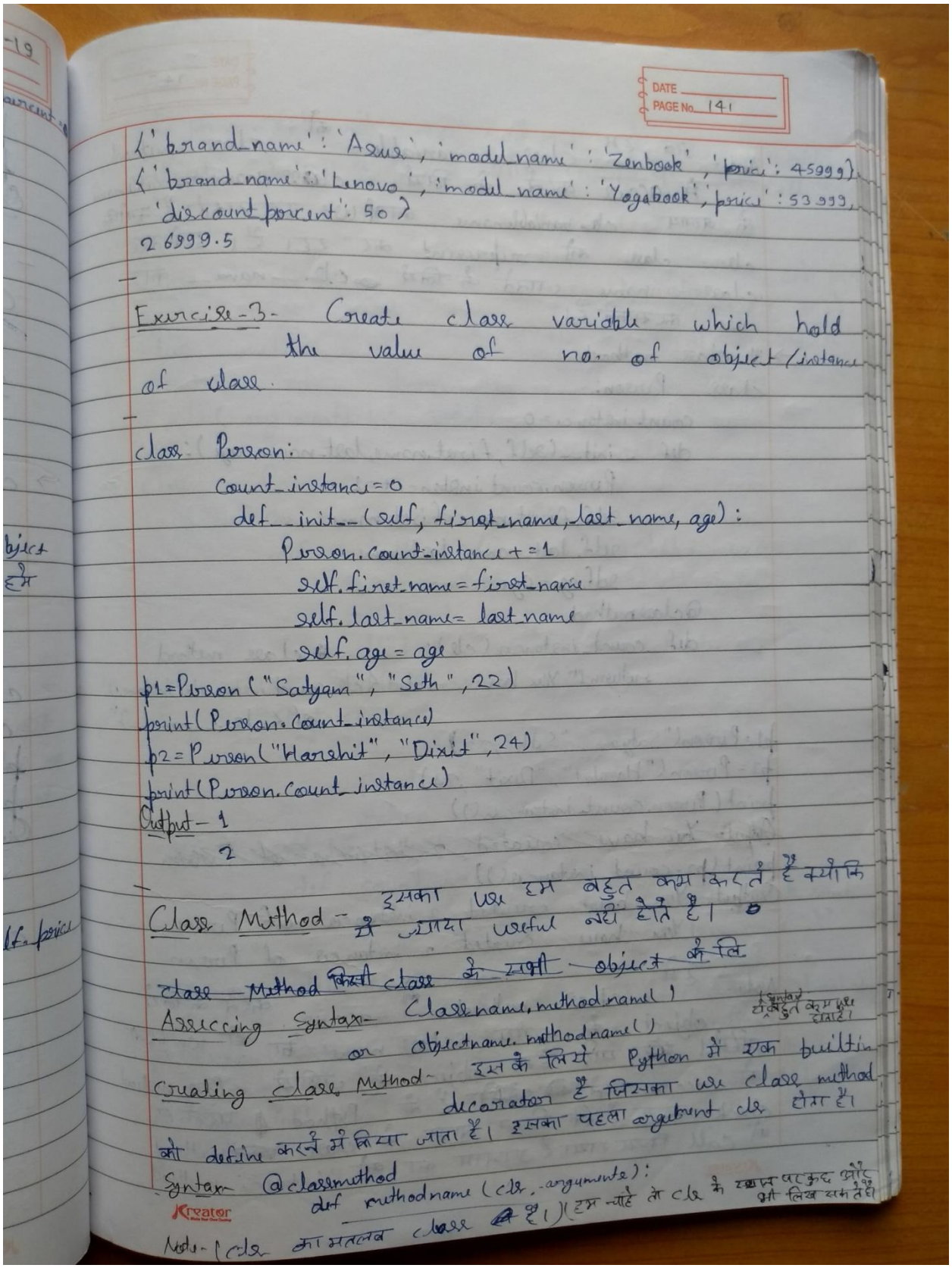
lap1 = Laptop("Asus", "Zenbook", 45899)
lap2 = Laptop("Lenovo", "Yogabook", 53999)
print(lap1.apply_discount())
print(lap2.apply_discount())
Output - 41129.1
48599.1

or 45699
or 53999

Create program for spaical discount on particular laptop -

Note - objectname.__dict__ से किसी object के पास उसके रूप में कितने instance variable हैं वो dictionary return करता है। जब हमें किसी particular object के लिये किसी variable की value pass करनी हो तो हमें self का use करना चाहिये।

→ class Laptop:
    discount_percent = 10
    def __init__(self, brand, model, price):
        self.brand_name = brand
        self.model_name = model
        self.price = price
    def apply_discount(self):
        off_price = (self.discount_percent/100) * self.price
        return self.price - off_price
lap1 = Laptop("Asus", "Zenbook", 45699)
lap2 = Laptop("Lenovo", "Yogabook", 53999)
print(lap1.__dict__)
lap2.discount_percent = 50
print(lap2.__dict__)
print(lap2.apply_discount())
Output - →



{'brand_name': 'Asus', 'modul_name': 'Zenbook', 'price': 45999},
{'brand_name': 'Lenovo', 'modul_name': 'Yogabook', 'price': 53999,
'discount_porcent': 50}
26999.5

Exercise-3- Create class variable which hold the value of no. of object/instance of class.

```
class Person:
    count_instance=0
    def __init__(self, first_name, last_name, age):
        Person.count_instance+=1
        self.first_name=first_name
        self.last_name=last_name
        self.age=age
p1=Person("Satyam", "Seth", 22)
print(Person.count_instance)
p2=Person("Harshit", "Dixit", 24)
print(Person.count_instance)
```

Output- 1
2

Class Method - इसका use हम बहुत कम करते हैं क्योंकि ये ज्यादा useful नहीं होते हैं।

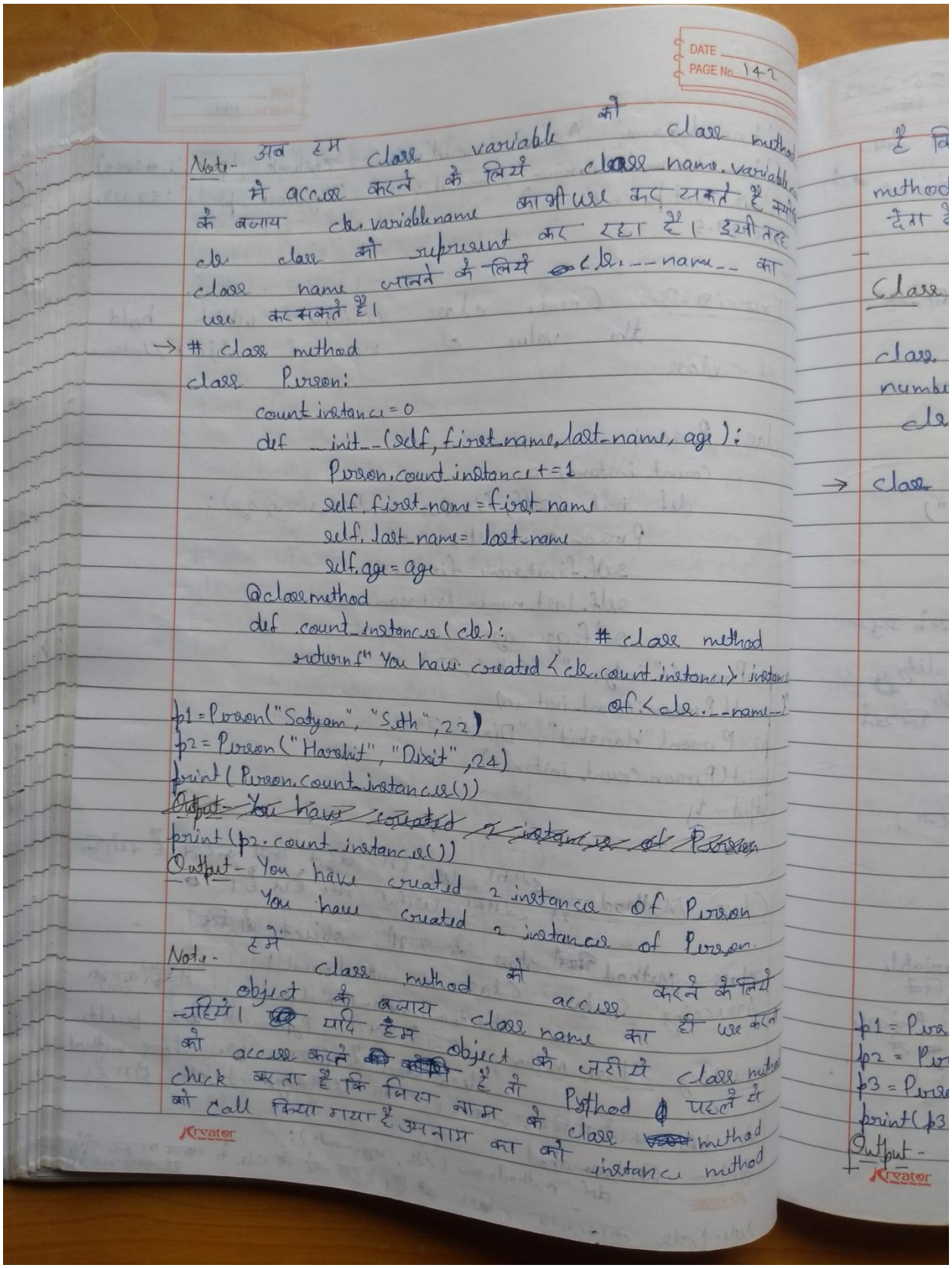
class Method किसी class के नहीं object के लिए

Accessing Syntax- Classname.method.name()
or objectname.methodname()

Creating class Method- इसके लिये Python में एक builtin decoration है जिसका use class method को define करने में किया जाता है। इसका पहला argument cls होता है।

Syntax- @classmethod
def methodname(cls, arguments):

Note- cls का मतलब class है।

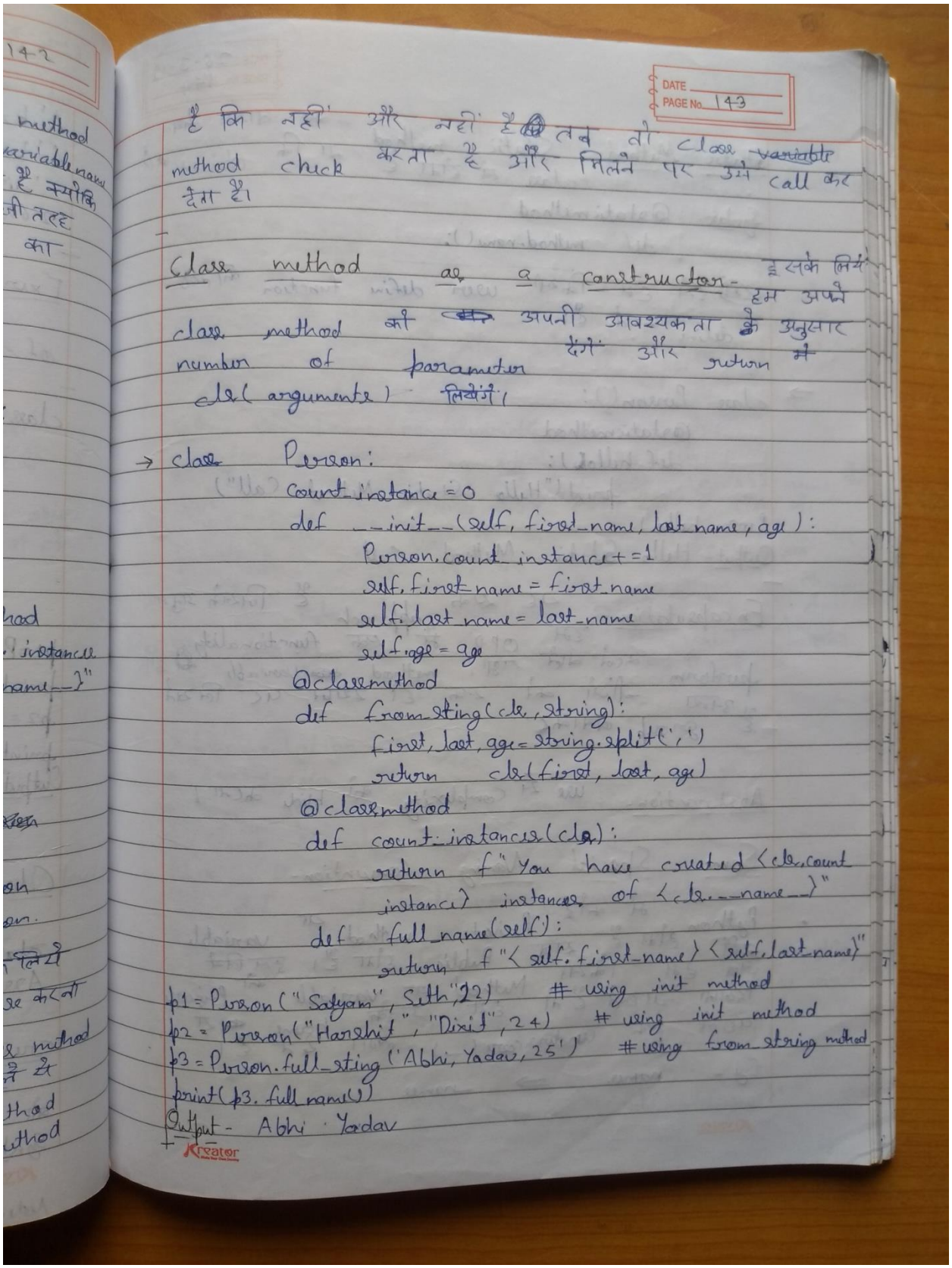


Note- अब हम class variable को class method में access करने के लिये class_name.variable_name के बजाय cls.variablename का भी use कर सकते है क्योंकि cls class को represent कर रहा है। इसी तरह class name जानने के लिये cls.__name__ का use कर सकते है।

→ # class method
class Person:
    count_instance = 0
    def __init__(self, first_name, last_name, age):
        Person.count_instance += 1
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
    @classmethod
    def count_instances(cls): # class method
        return f"You have created {cls.count_instance} instance of {cls.__name__}"

p1 = Person("Satyam", "Seth", 22)
p2 = Person("Harshit", "Dixit", 24)
print(Person.count_instances())
~~Output- You have created 2 instance of Person~~
print(p2.count_instances())
Output- You have created 2 instance of Person
You have created 2 instance of Person.

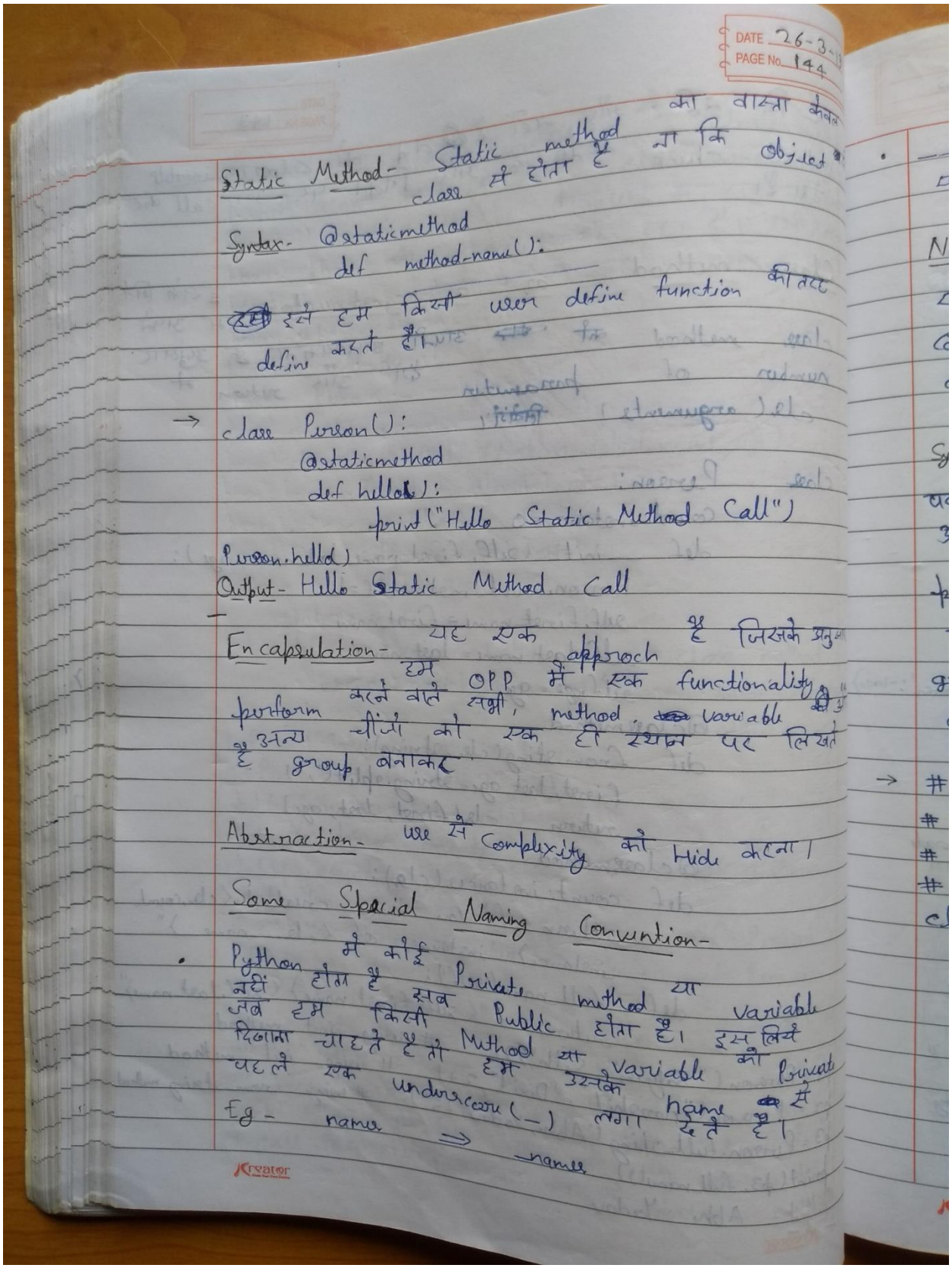
Note- हम class method को access करने के लिये object के बजाय class name का ही use करते है। यदि हम object के जरिये class method को access करते है तो Python पहले check करता है कि जिस नाम के class method को call किया गया है उसनाम का कोई instance method



है कि नहीं और नहीं है तब तो class ~~variable~~ method check करता है और मिलने पर उसे call कर देता है।

Class method as a constructor - इसके लिये हम अपने class method को ~~के~~ अपनी आवश्यकता के अनुसार number of paramter देंगे और return में cls(arguments) लिखेंगे।

→
```
class Person:
    count_instance = 0
    def __init__(self, first_name, last_name, age):
        Person.count_instance += 1
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
    @classmethod
    def from_sting(cls, string):
        first, last, age = string.split(',')
        return cls(first, last, age)
    @classmethod
    def count_instances(cls):
        return f"You have created {cls.count_instance} instances of {cls.__name__}"
    def full_name(self):
        return f"{self.first_name} {self.last_name}"
p1 = Person("Satyam", Seth, 22)      # using init method
p2 = Person("Harshit", "Dixit", 24)  # using init method
p3 = Person.full_sting('Abhi, Yadav, 25')  # using from_string method
print(p3.full_name())
```
Output - Abhi Yadav



DATE 26-3-
PAGE No. 144

Static Method- Static method का वास्ता केवल class से होता है ना कि object से

Syntax- @staticmethod
def method-name():

इसे हम किसी user define function की तरह define करते हैं।

→ class Person():
@staticmethod
def hello():
print("Hello Static Method Call")
Person.hello()
Output- Hello Static Method Call

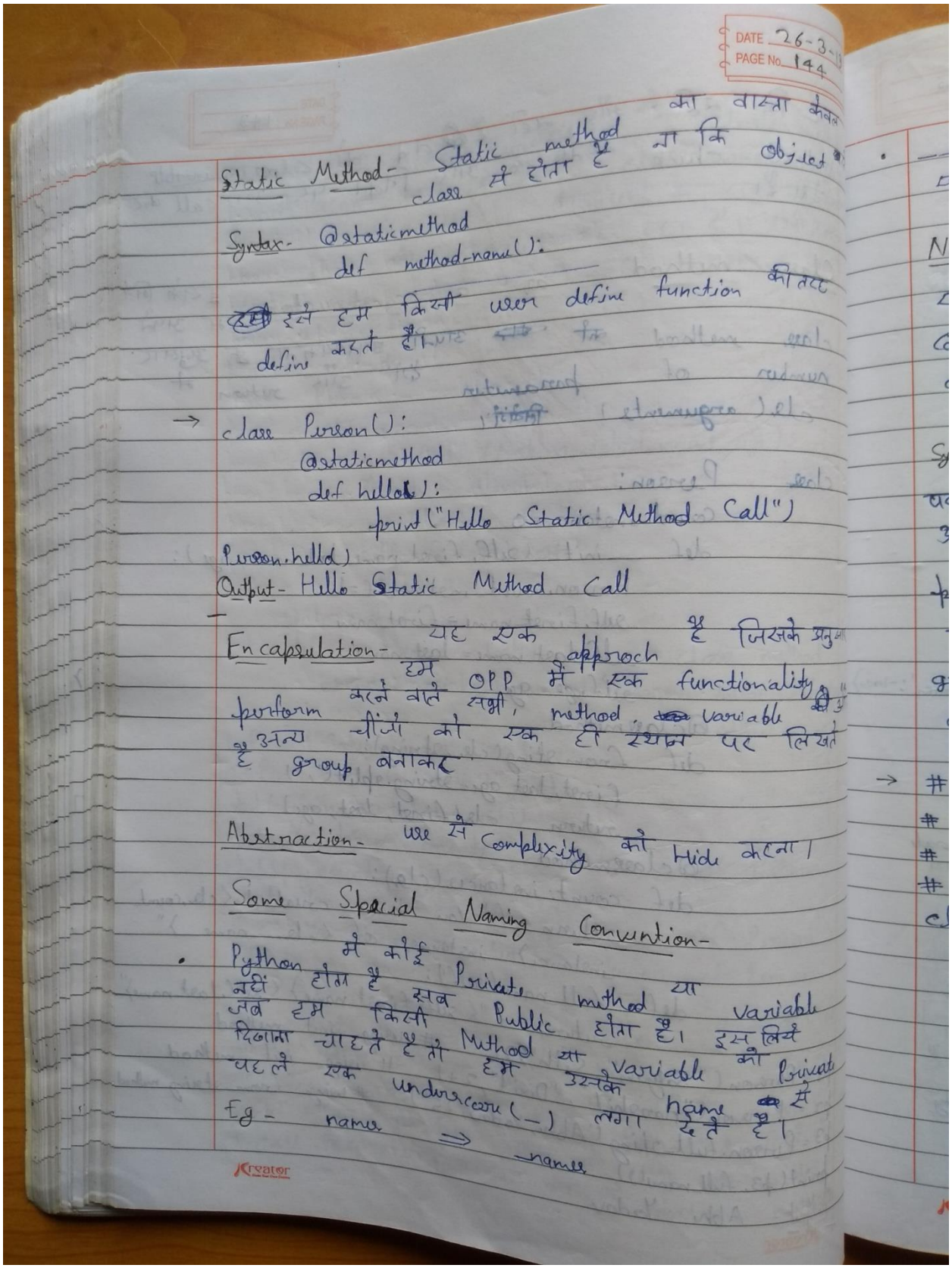
Encapsulation- यह एक approach है जिसके अनु हम OPP में एक functionality perform करने वाले सभी, method, variable और अन्य चीजों को एक ही स्थान पर लिखते हैं group बनाकर

Abstraction- use से complexity को Hide करना।

Some Spacial Naming Convention-

- Python में कोई Private method या variable नहीं होता है सब Public होता है। इस लिये जब हम किसी Method या variable को Private दिखाना चाहते हैं तो हम उसके name से पहले एक underscore (_) लगा देते हैं।

Eg- name ⇒ _name



DATE 26-3-
PAGE No. 144

Static Method- Static method का वास्ता केवल class से होता है ना कि object से

Syntax- @staticmethod
def method-name():

इसे हम किसी user define function की तरह define करते हैं।

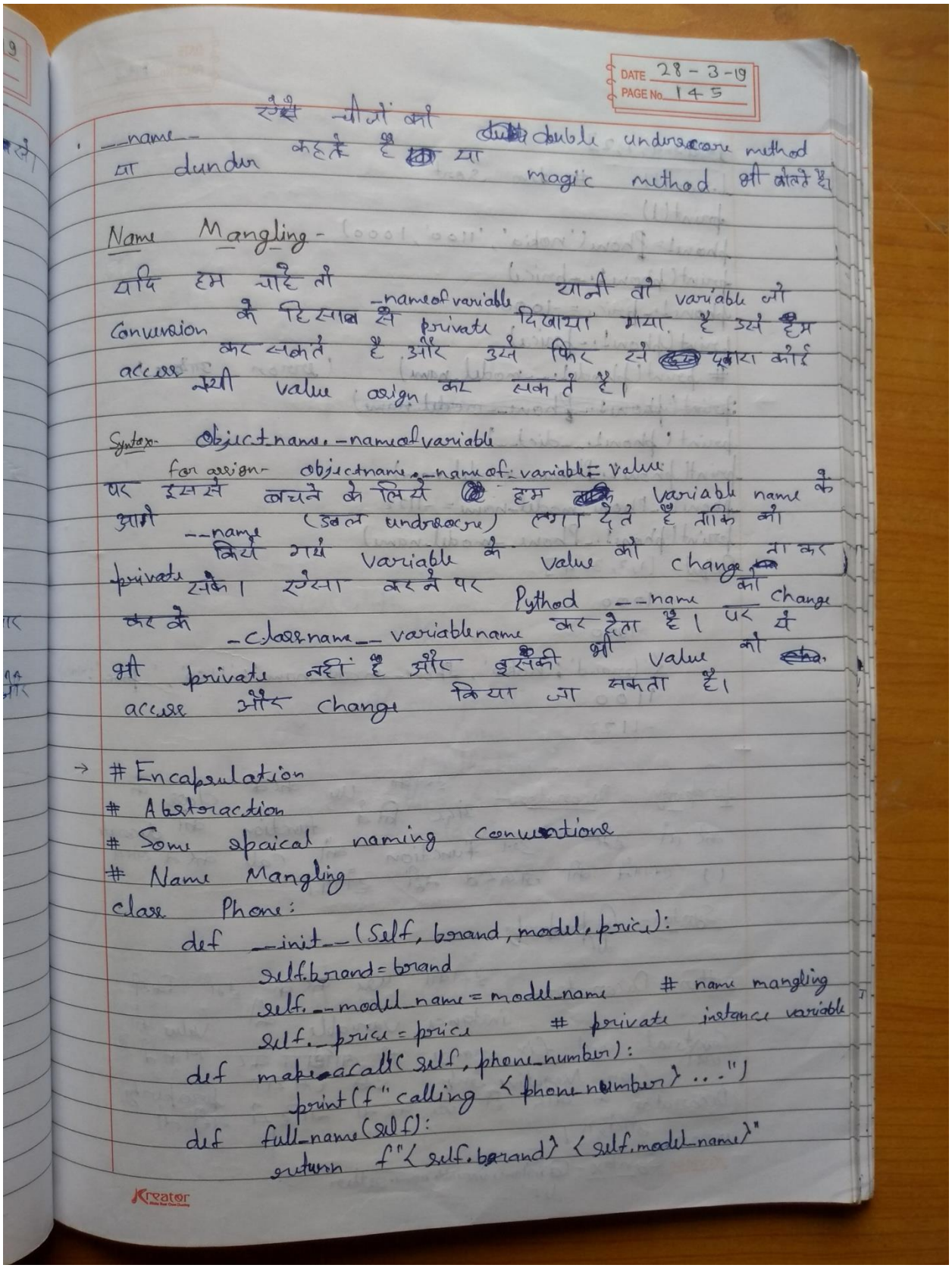
→ class Person():
@staticmethod
def hello():
print("Hello Static Method Call")
Person.hello()
Output- Hello Static Method Call

Encapsulation- यह एक approach है जिसके अनु
हम OPP में एक functionality perform करने वाले सभी, method, variable और अन्य चीजों को एक ही स्थान पर लिखते हैं group बनाकर

Abstraction- use से complexity को Hide करना।

Some Spacial Naming Convention-

- Python में कोई Private method या variable नहीं होता है सब Public होता है। इस लिये जब हम किसी Method या variable को Private दिखाना चाहते हैं तो हम उसके name से पहले एक underscore (_) लगा देते हैं।
Eg- name ⇒ _name



ऐसे चीजों को __name__ double underscore method या dunder कहते हैं या magic method भी बोलते हैं

Name Mangling -

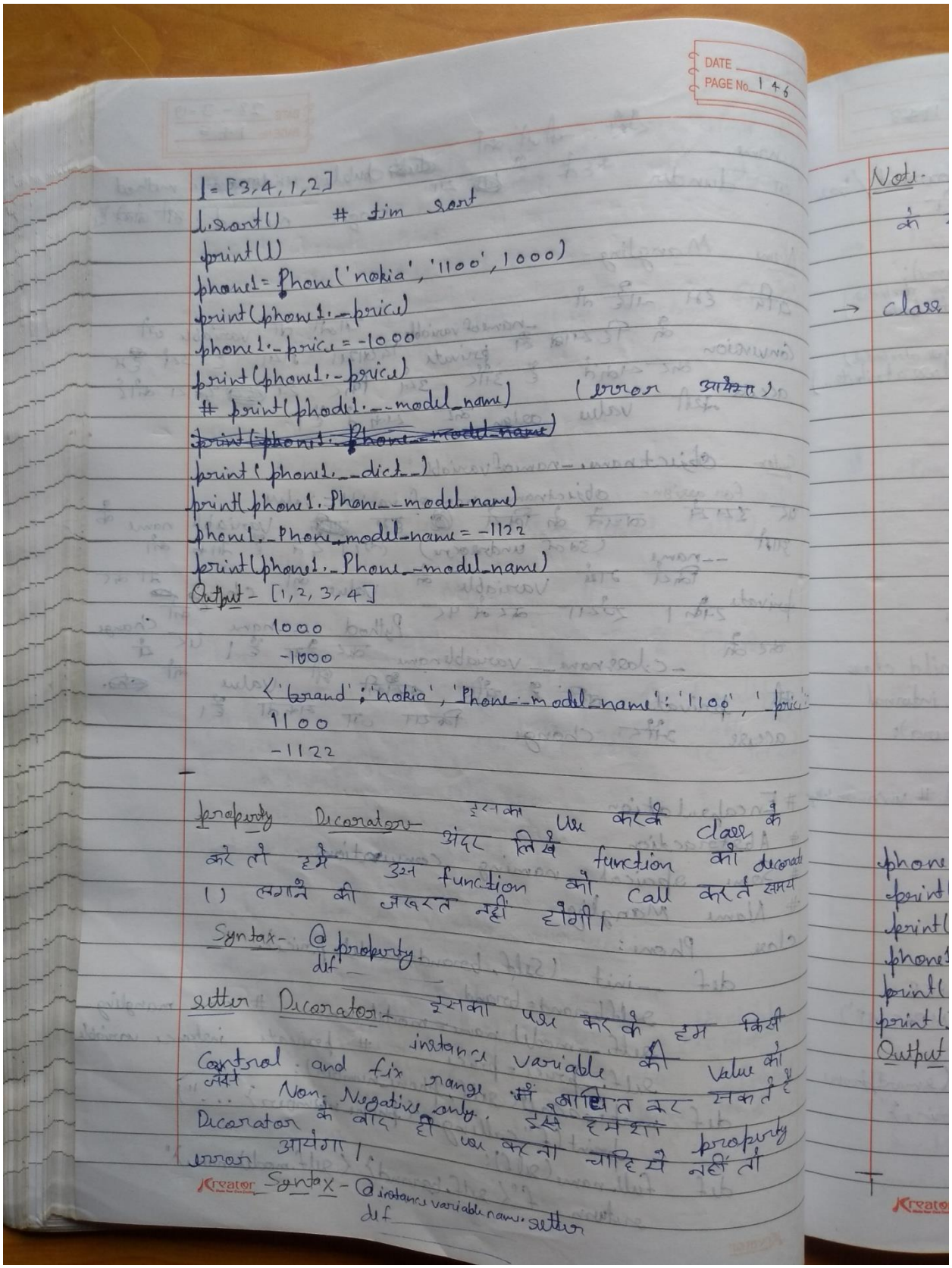
यदि हम चाहें तो Conversion के हिसाब से private दिखाया गया है उसे हम access कर सकते हैं और उसे फिर से value assign कर सकते हैं।

Syntax- Objectname._nameofvariable

for assign- objectname._nameof_variable = value

पर इससे बचने के लिये हम variable name के आगे __name (डबल underscore) लगा देते हैं ताकि जो private किये गये variable के value को change ना कर सके। ऐसा करने पर Pythod __name को change कर के _classname__variablename कर देता है। पर ये भी private नहीं है और इसकी भी value को access और change किया जा सकता है।

```
→ # Encapsulation
  # Abstraction
  # Some spaical naming convention
  # Name Mangling
  class Phone:
      def __init__(self, brand, model, price):
          self.brand = brand
          self.__model_name = model_name    # name mangling
          self._price = price               # private instance variable
      def make_a_call(self, phone_number):
          print(f"calling {phone_number} ...")
      def full_name(self):
          return f"{self.brand} {self.model_name}"
```



```
l = [3,4,1,2]
l.sort()  # tim sort
print(l)
phone1 = Phone('nokia', '1100', 1000)
print(phone1.__price)
phone1.__price = -1000
print(phone1.__price)
# print(phone1.__model_name)  (error आयेगा)
print(phone1._Phone__model_name)
print(phone1.__dict__)
print(phone1._Phone__model_name)
phone1._Phone__model_name = -1122
print(phone1._Phone__model_name)
```

Output - [1, 2, 3, 4]

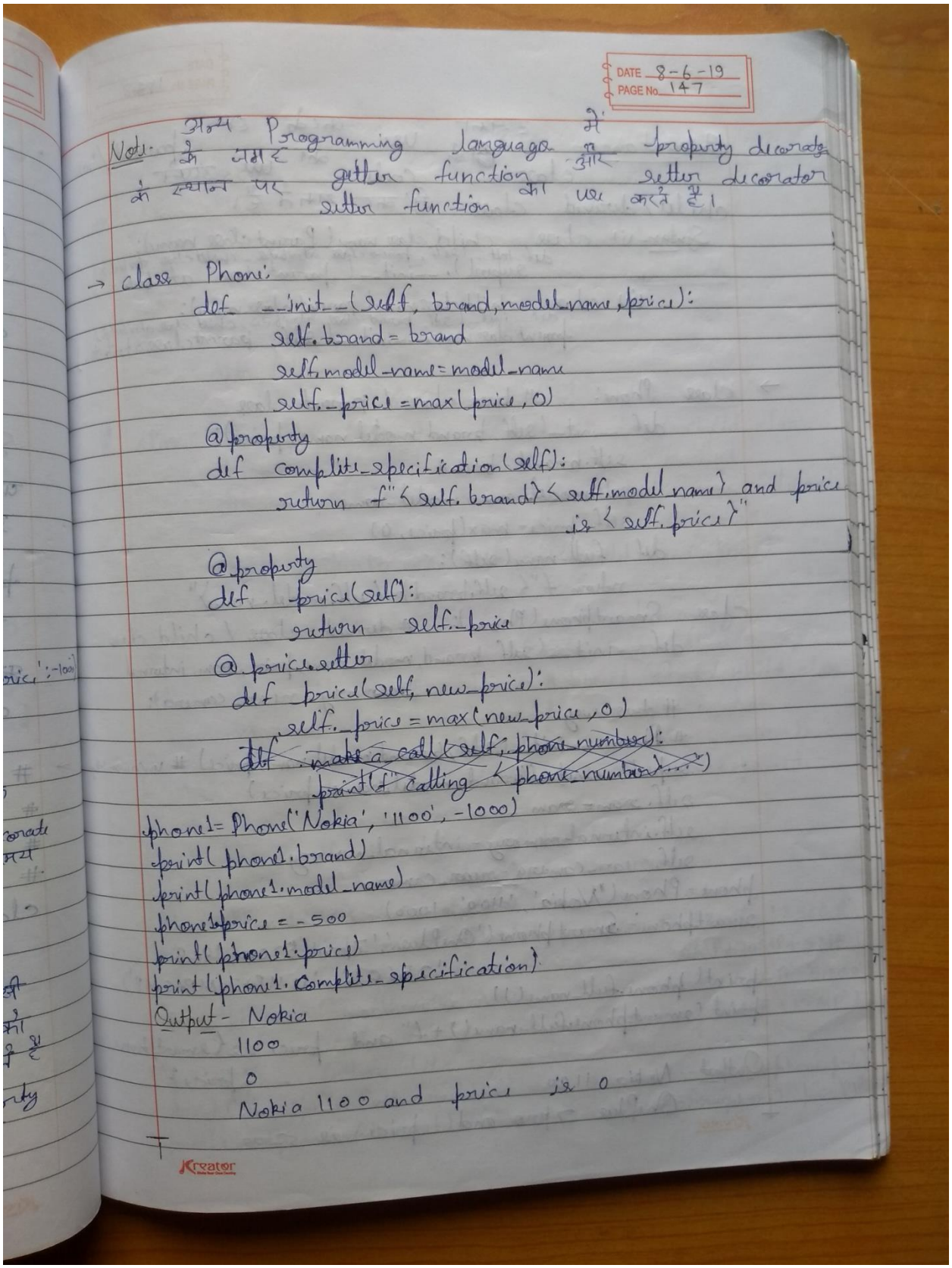
```
1000
-1000
{'brand': 'nokia', 'Phone__model_name': '1100', '__price':
1100
-1122
```

property Decorator - इसका use करके class के अंदर लिखे function को decorate करते हैं उन function को call करते समय () लगाने की जरूरत नहीं होगी।

Syntax - @property
def

setter Decorator - इसका use करके हम किसी instance variable की value को control and fix range में limited कर सकते हैं जैसे Non Negative only. इसे हमेशा property Decorator के बाद ही use करना चाहिये नहीं तो error आयेगा।

Syntax - @instance variable name.setter
def



DATE 8-6-19
PAGE No. 147

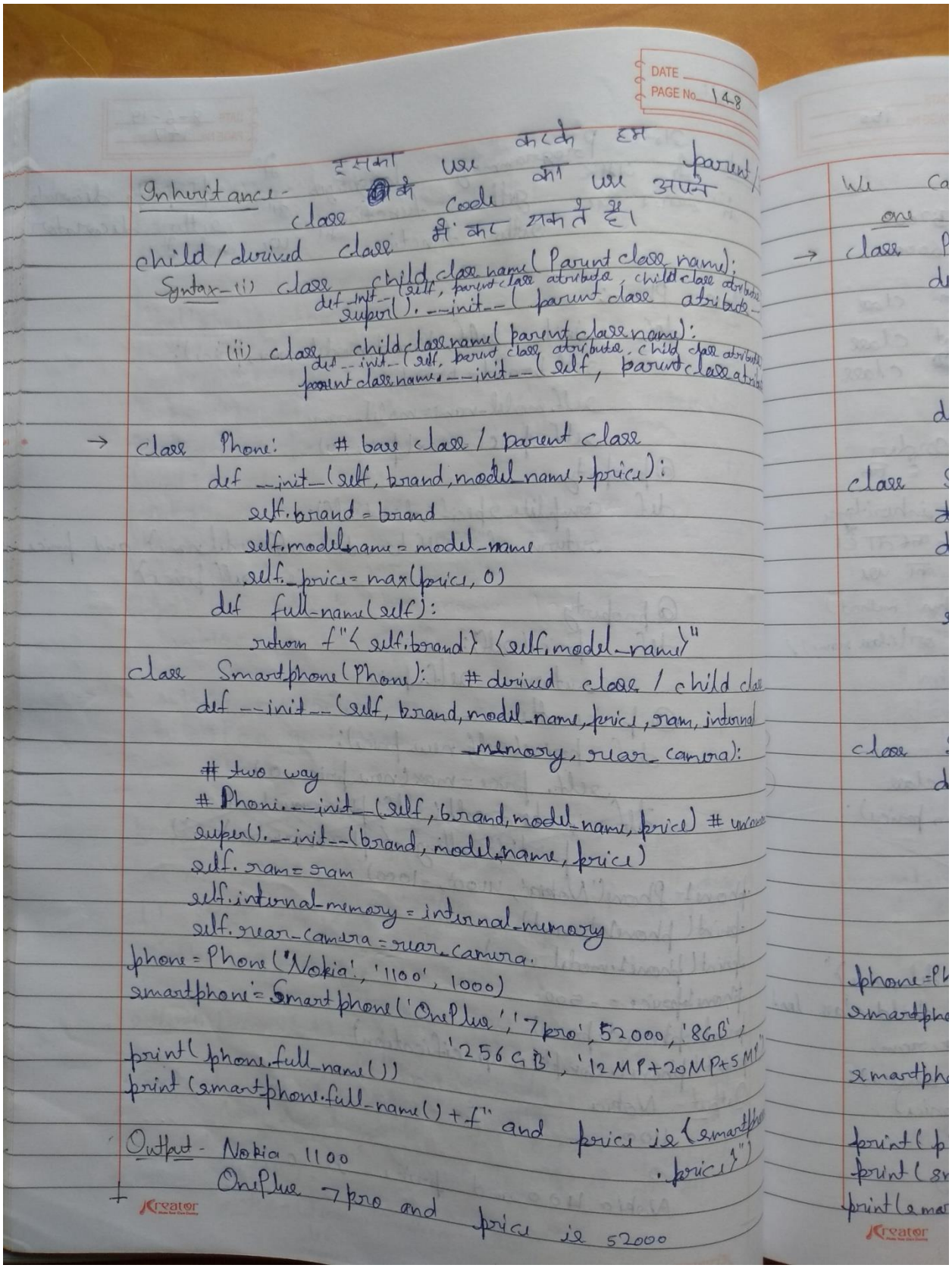
Note: अन्य Programming language में जैसे C++ और Java में property decorator के स्थान पर getter function और setter decorator setter function का use करते हैं।

→ class Phone:
    def __init__(self, brand, model_name, price):
        self.brand = brand
        self.model_name = model_name
        self._price = max(price, 0)
    @property
    def complite_specification(self):
        return f"{self.brand} {self.model_name} and price is {self.price}"

    @property
    def price(self):
        return self._price
    @price.setter
    def price(self, new_price):
        self._price = max(new_price, 0)
    def make_a_call(self, phone_number):
        print(f"calling {phone_number}....")

phone1 = Phone('Nokia', '1100', -1000)
print(phone1.brand)
print(phone1.model_name)
phone1.price = -500
print(phone1.price)
print(phone1.complite_specification)

Output - Nokia
1100
0
Nokia 1100 and price is 0



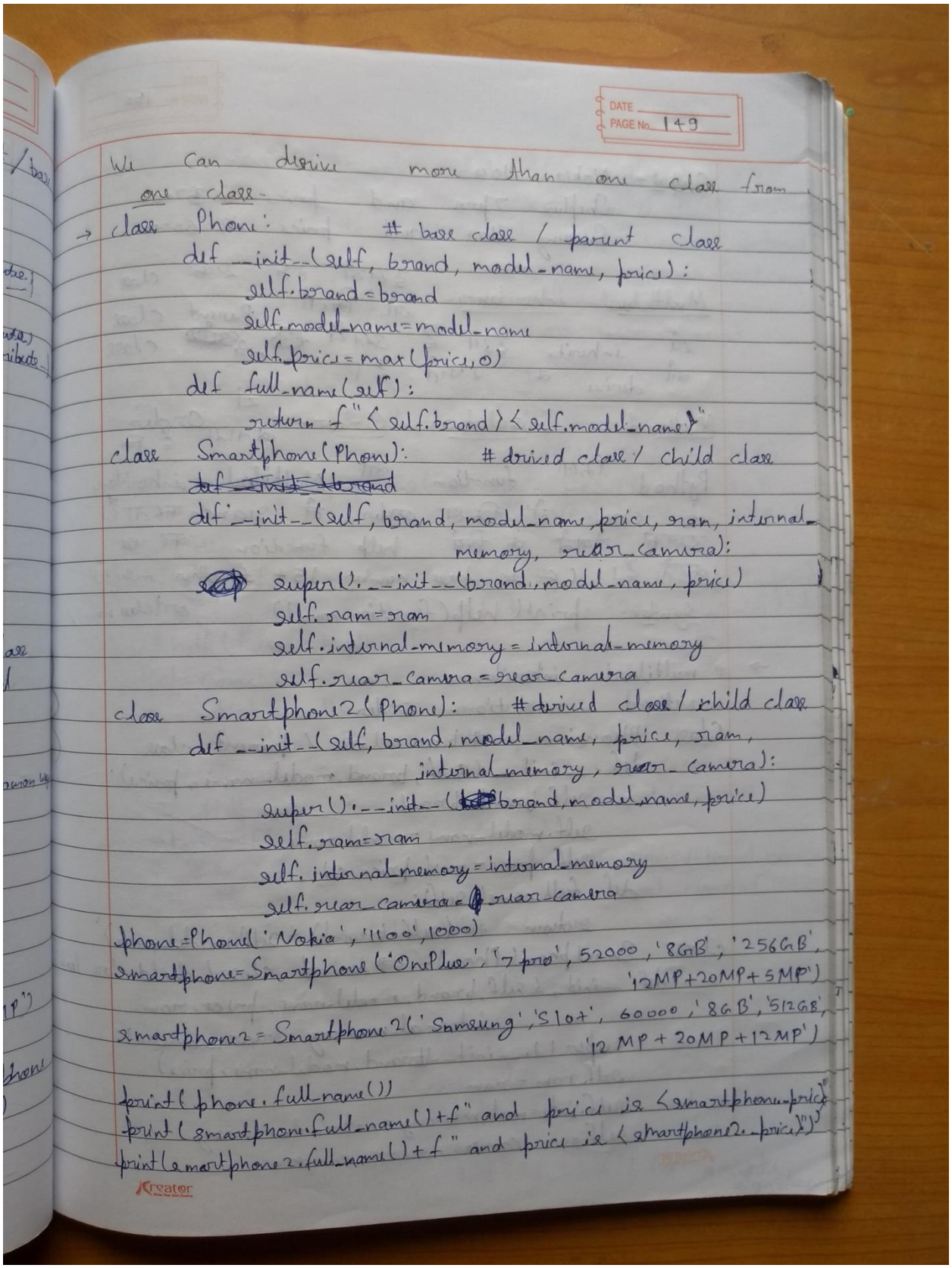
Inheritance - इसका use करके हम parent/ class के code का use अपने child/derived class में कर सकते हैं।

Syntax - (i) class child_class_name(parent_class_name):
def __init__(self, parent class attributes, child class attributes):
super().__init__(parent class attributes)

(ii) class child_class_name(parent_class_name):
def __init__(self, parent class attributes, child class attributes):
parent_class_name.__init__(self, parent class attributes)

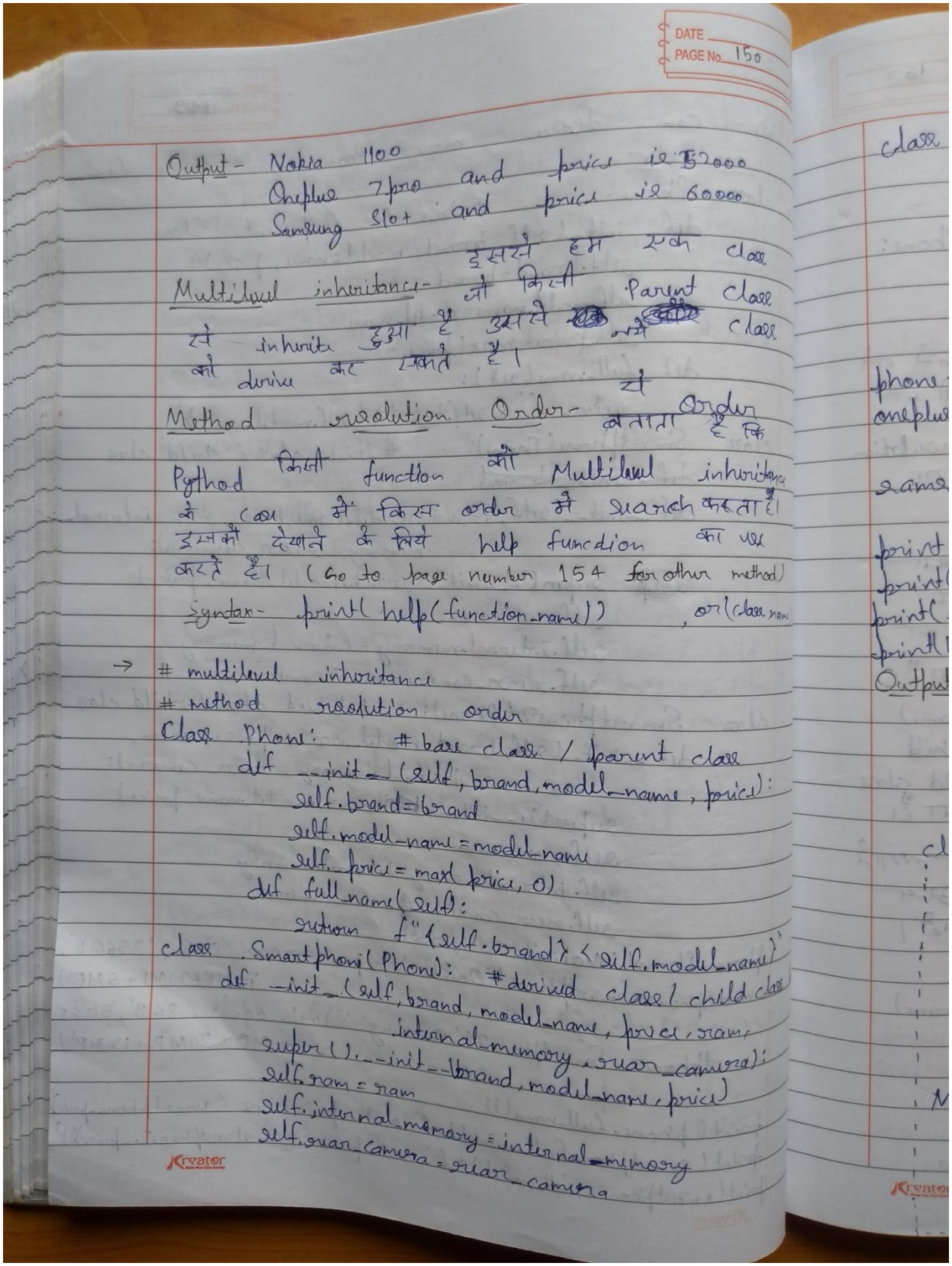
```
class Phone:    # base class / parent class
    def __init__(self, brand, model_name, price):
        self.brand = brand
        self.model_name = model_name
        self._price = max(price, 0)
    def full_name(self):
        return f"{self.brand} {self.model_name}"
class Smartphone(Phone):    # derived class / child class
    def __init__(self, brand, model_name, price, ram, internal
                 _memory, rear_camera):
        # two way
        # Phone.__init__(self, brand, model_name, price) # unbound
        super().__init__(brand, model_name, price)
        self.ram = ram
        self.internal_memory = internal_memory
        self.rear_camera = rear_camera
phone = Phone('Nokia', '1100', 1000)
smartphone = Smartphone('OnePlus', '7pro', 52000, '8GB',
                        '256GB', '12MP+20MP+5MP')
print(phone.full_name())
print(smartphone.full_name() + f" and price is {smartphone
                                                .price}")
```

Output - Nokia 1100
OnePlus 7pro and price is 52000



We can derive more than one class from one class.

→
```
class Phone:                    # base class / parent class
    def __init__(self, brand, model_name, price):
        self.brand = brand
        self.model_name = model_name
        self.price = max(price, 0)
    def full_name(self):
        return f"{self.brand} {self.model_name}"
class Smartphone(Phone):        # derived class / child class
    def __init__(brand
    def __init__(self, brand, model_name, price, ram, internal_
                                    memory, rear_camera):
        super().__init__(brand, model_name, price)
        self.ram = ram
        self.internal_memory = internal_memory
        self.rear_camera = rear_camera
class Smartphone2(Phone):       # derived class / child class
    def __init__(self, brand, model_name, price, ram,
                        internal_memory, rear_camera):
        super().__init__(brand, model_name, price)
        self.ram = ram
        self.internal_memory = internal_memory
        self.rear_camera = rear_camera
phone = Phone('Nokia', '1100', 1000)
smartphone = Smartphone('OnePlus', '7 pro', 52000, '8GB', '256GB',
                                                '12MP+20MP+5MP')
smartphone2 = Smartphone2('Samsung', 'S10+', 60000, '8GB', '512GB',
                                                '12 MP + 20MP + 12MP')

print(phone.full_name())
print(smartphone.full_name() + f" and price is {smartphone.price}")
print(smartphone2.full_name() + f" and price is {smartphone2.price}")
```



DATE
PAGE No. 150

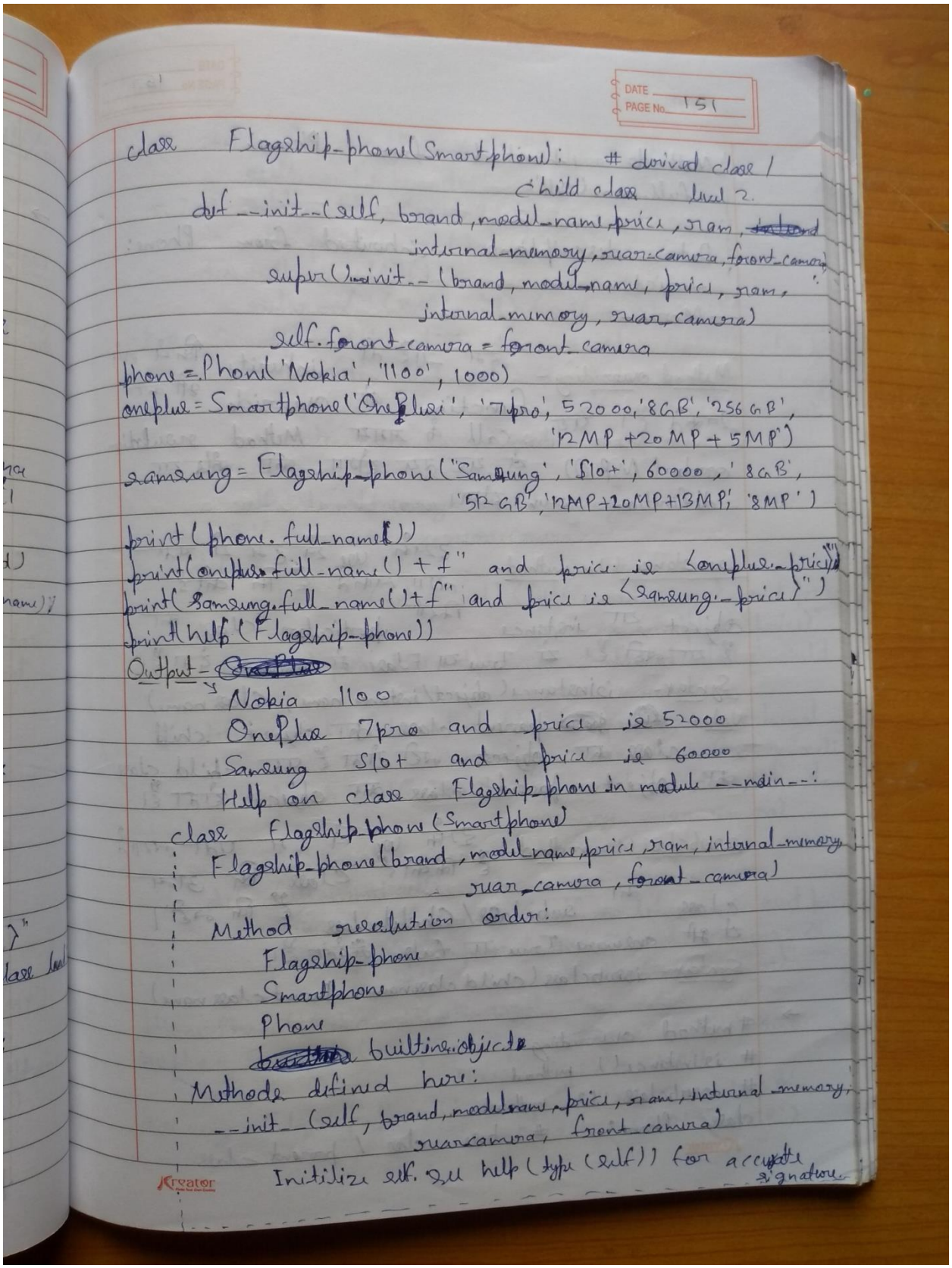
Output - Nokia 1100
Oneplus 7pro and price is ₹2000
Samsung S10+ and price is 60000

Multilevel inheritance - इसमें हम एक class से inherit हुआ है उसे किसी Parent class को drive कर सकते है।

Method resolution Order - से बताता Order है कि Python किसी function को Multilevel inheritance के case में किस order में search करता है। इनको देखने के लिये help function का use करते है। (Go to page number 154 for other method)

Syntax - print(help(function_name)) , or (class_name

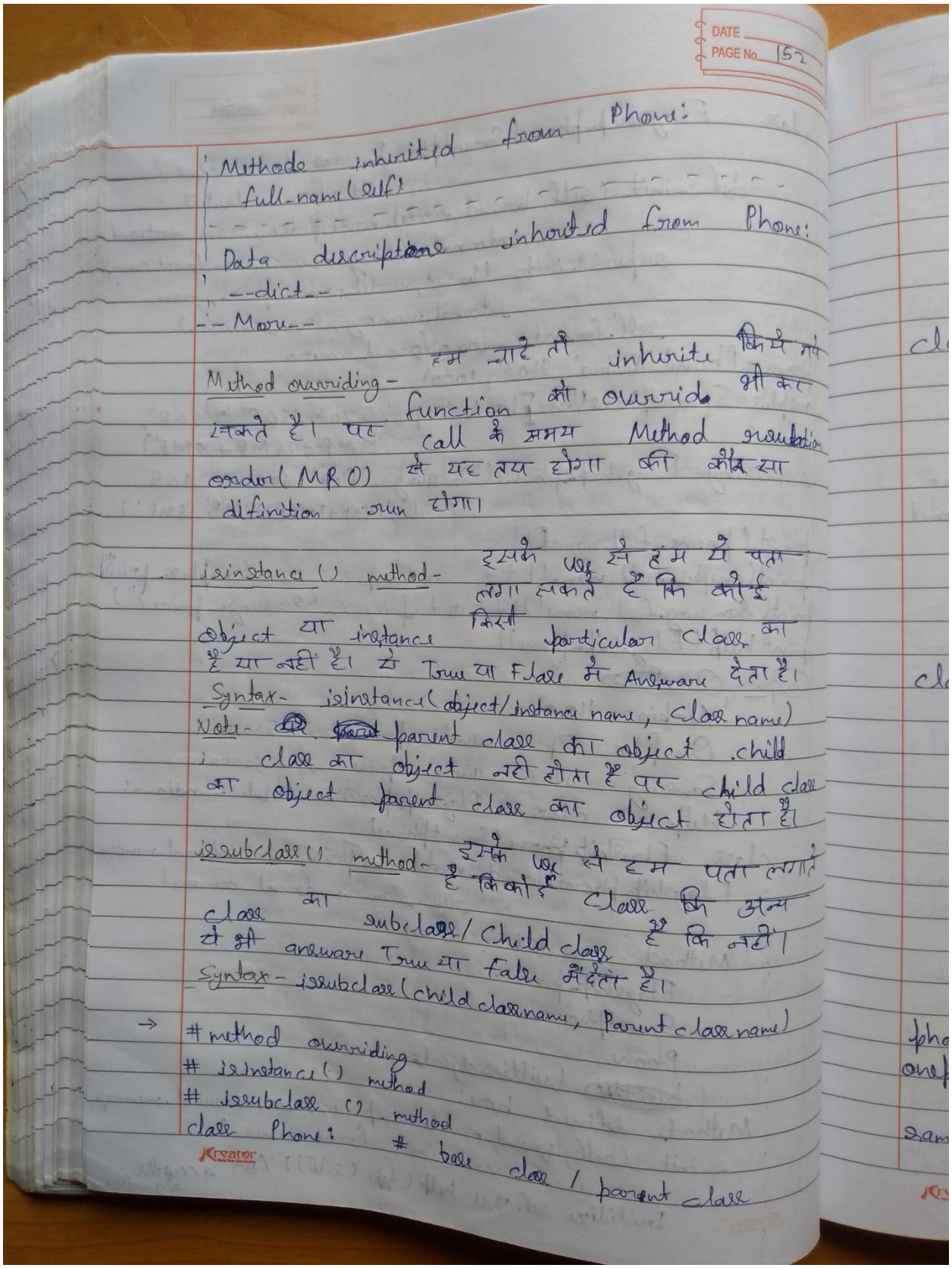
```
# multilevel inheritance
# method resolution order
Class Phone:          # base class / parent class
    def __init__(self, brand, model_name, price):
        self.brand = brand
        self.model_name = model_name
        self.price = max(price, 0)
    def full_name(self):
        return f"{self.brand} {self.model_name}"
class Smartphone(Phone):  # derived class / child class
    def __init__(self, brand, model_name, price, ram,
                 internal_memory, rear_camera):
        super().__init__(brand, model_name, price)
        self.ram = ram
        self.internal_memory = internal_memory
        self.rear_camera = rear_camera
```



```
class Flagship_phone(Smartphone): # derived class / child class level 2.
    def __init__(self, brand, model_name, price, ram, internal_memory, rear_camera, front_camera):
        super().__init__(brand, model_name, price, ram, internal_memory, rear_camera)
        self.front_camera = front_camera
phone = Phone('Nokia', '1100', 1000)
oneplus = Smartphone('OnePlus', '7pro', 52000, '8GB', '256GB', '12MP+20MP+5MP')
samsung = Flagship_phone('Samsung', 'S10+', 60000, '8GB', '512GB', '12MP+20MP+13MP', '8MP')
print(phone.full_name())
print(oneplus.full_name() + f" and price is {oneplus.price}")
print(samsung.full_name() + f" and price is {samsung.price}")
print(help(Flagship_phone))
```

Output:

```
Nokia 1100
OnePlus 7pro and price is 52000
Samsung S10+ and price is 60000
Help on class Flagship_phone in module __main__:
class Flagship_phone(Smartphone)
 |  Flagship_phone(brand, model_name, price, ram, internal_memory, rear_camera, front_camera)
 |
 |  Method resolution order:
 |      Flagship_phone
 |      Smartphone
 |      Phone
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  __init__(self, brand, model_name, price, ram, internal_memory, rear_camera, front_camera)
 |      Initialize self.  See help(type(self)) for accurate signature.
```



Methods inherited from Phone:
full_name(self)

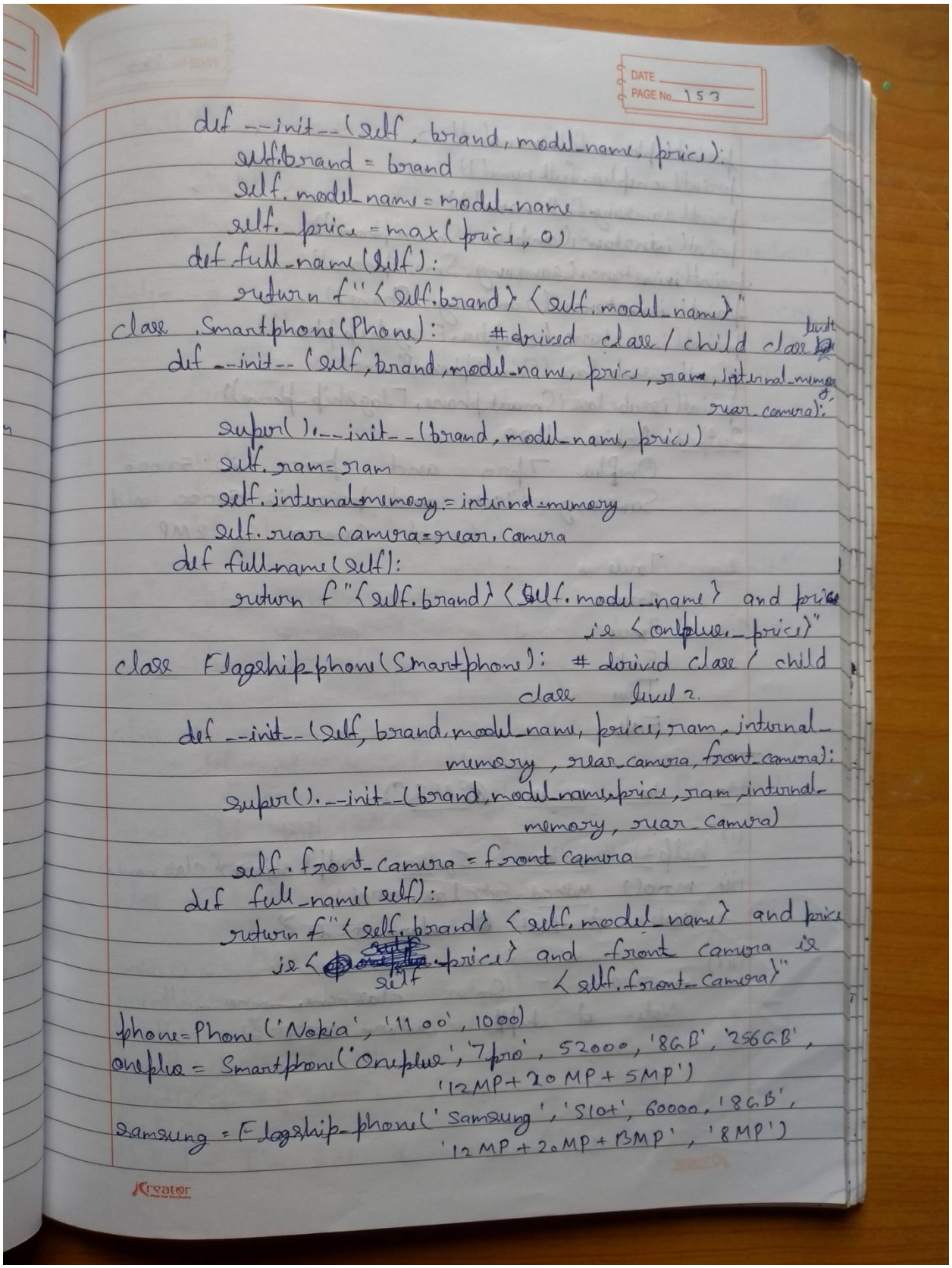
Data descriptors inherited from Phone:
__dict__
__weakref__

Method overriding - हम चाहे तो inherit किये गये भी कर सकते हैं। पर function को override call के समय Method resolution order (MRO) से यह तय होगा की कौन सा difinition run होगा।

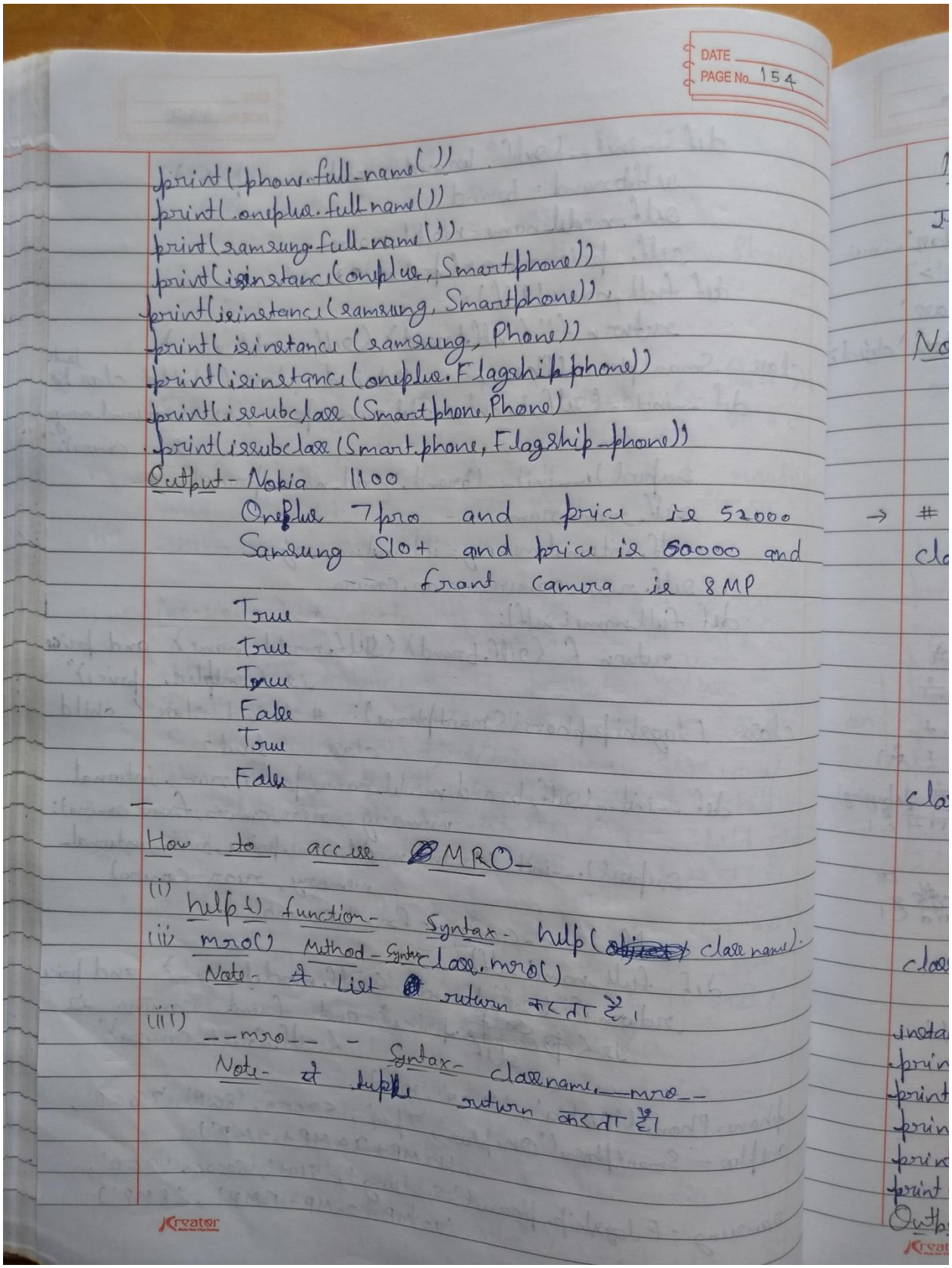
isinstance() method - इसके use से हम ये पता लगा सकते हैं कि कोई object या instance किसी particular class का है या नहीं है। ये True या False में Answare देता है।
Syntax - isinstance(object/instance name, class name)
Note - parent class का object child class का object नहीं होता है पर child class का object parent class का object होता है

issubclass() method - इसके use से हम पता लगाते हैं कि कोई class किसी अन्य class का subclass/child class है कि नहीं। ये भी answare True या False में देता है।
Syntax - issubclass(child classname, Parent class name)

→ #method overriding
isinstance() method
issubclass() method
class Phone: # base class / parent class



```
def __init__(self, brand, modul_name, price):
    self.brand = brand
    self.modul_name = modul_name
    self._price = max(price, 0)
def full_name(self):
    return f"{self.brand} {self.modul_name}"
class Smartphone(Phone):  #drived class / child class
    def __init__(self, brand, modul_name, price, ram, internal_memory,
                 rear_camera):
        super().__init__(brand, modul_name, price)
        self.ram = ram
        self.internal_memory = internal_memory
        self.rear_camera = rear_camera
    def full_name(self):
        return f"{self.brand} {self.modul_name} and price
                 is {oneplus._price}"
class Flagship_phone(Smartphone): # drived class / child
                                  class level 2.
    def __init__(self, brand, modul_name, price, ram, internal_
                 memory, rear_camera, front_camera):
        super().__init__(brand, modul_name, price, ram, internal_
                 memory, rear_camera)
        self.front_camera = front_camera
    def full_name(self):
        return f"{self.brand} {self.modul_name} and price
            is {self._price} and front camera is
                 {self.front_camera}"
phone = Phone('Nokia', '1100', 1000)
oneplus = Smartphone('Oneplus', '7pro', 52000, '8GB', '256GB',
                 '12MP+20MP+5MP')
samsung = Flagship_phone('Samsung', 'S10+', 60000, '8GB',
                 '12MP+20MP+8MP', '8MP')
```



```
print(phone.full_name())
print(oneplus.full_name())
print(samsung.full_name())
print(isinstance(oneplus, Smartphone))
print(isinstance(samsung, Smartphone))
print(isinstance(samsung, Phone))
print(isinstance(oneplus.Flagship_phone))
print(issubclass(Smartphone, Phone))
print(issubclass(Smartphone, Flagship_phone))
```

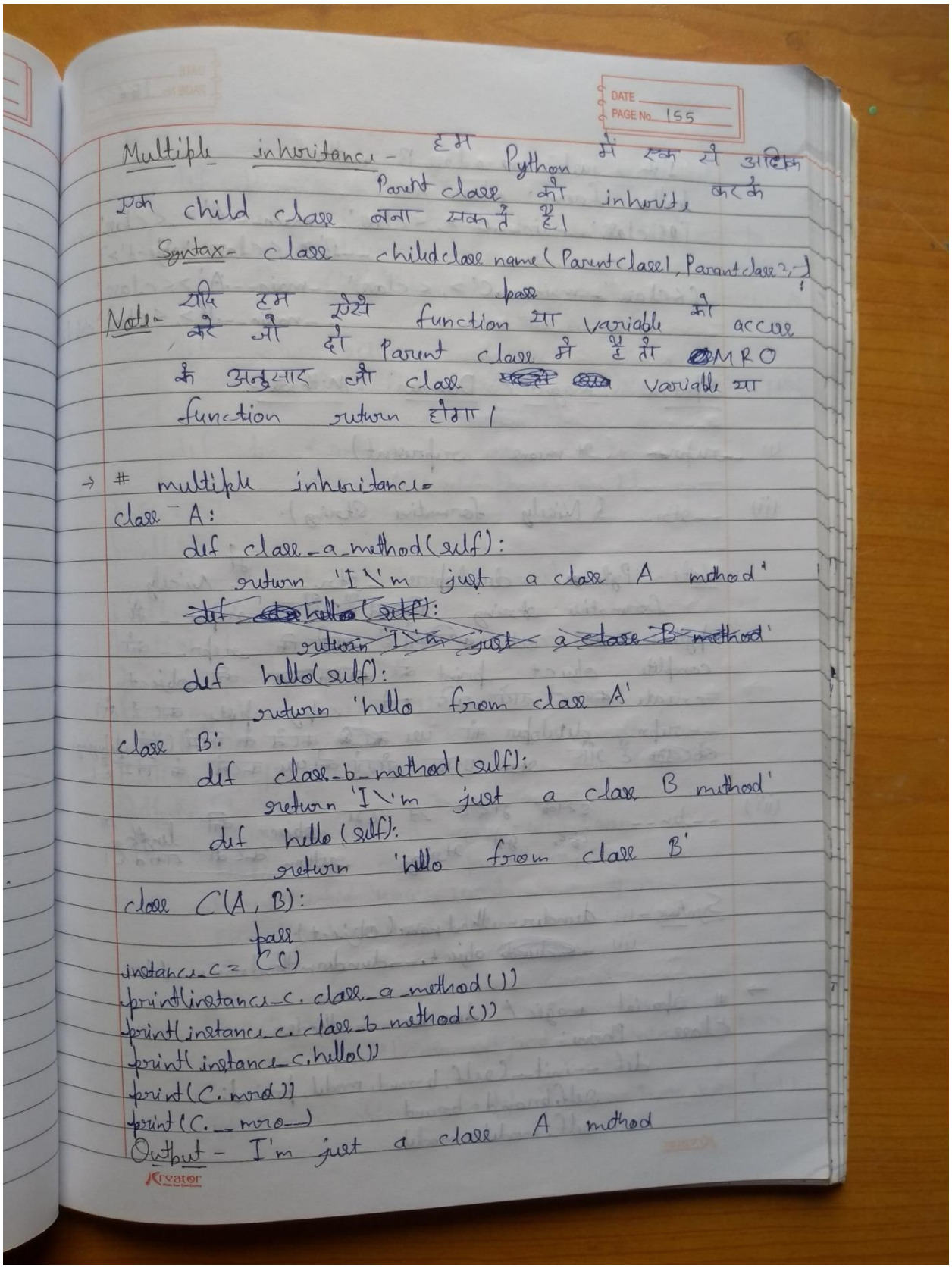
Output - Nokia 1100
OnePlus 7pro and price is 52000
Samsung S10+ and price is 60000 and front camera is 8MP
True
True
True
False
True
False

How to access MRO -

(i) help() function - Syntax - help(class name)

(ii) mro() Method - Syntax - class.mro()
Note - It list return करता है।

(iii) __mro__ - Syntax - classname.__mro__
Note - It tuple return करता है।



Multiple inheritance - हम Python में एक से अधिक Parent class को inherit करके एक child class बना सकते हैं।

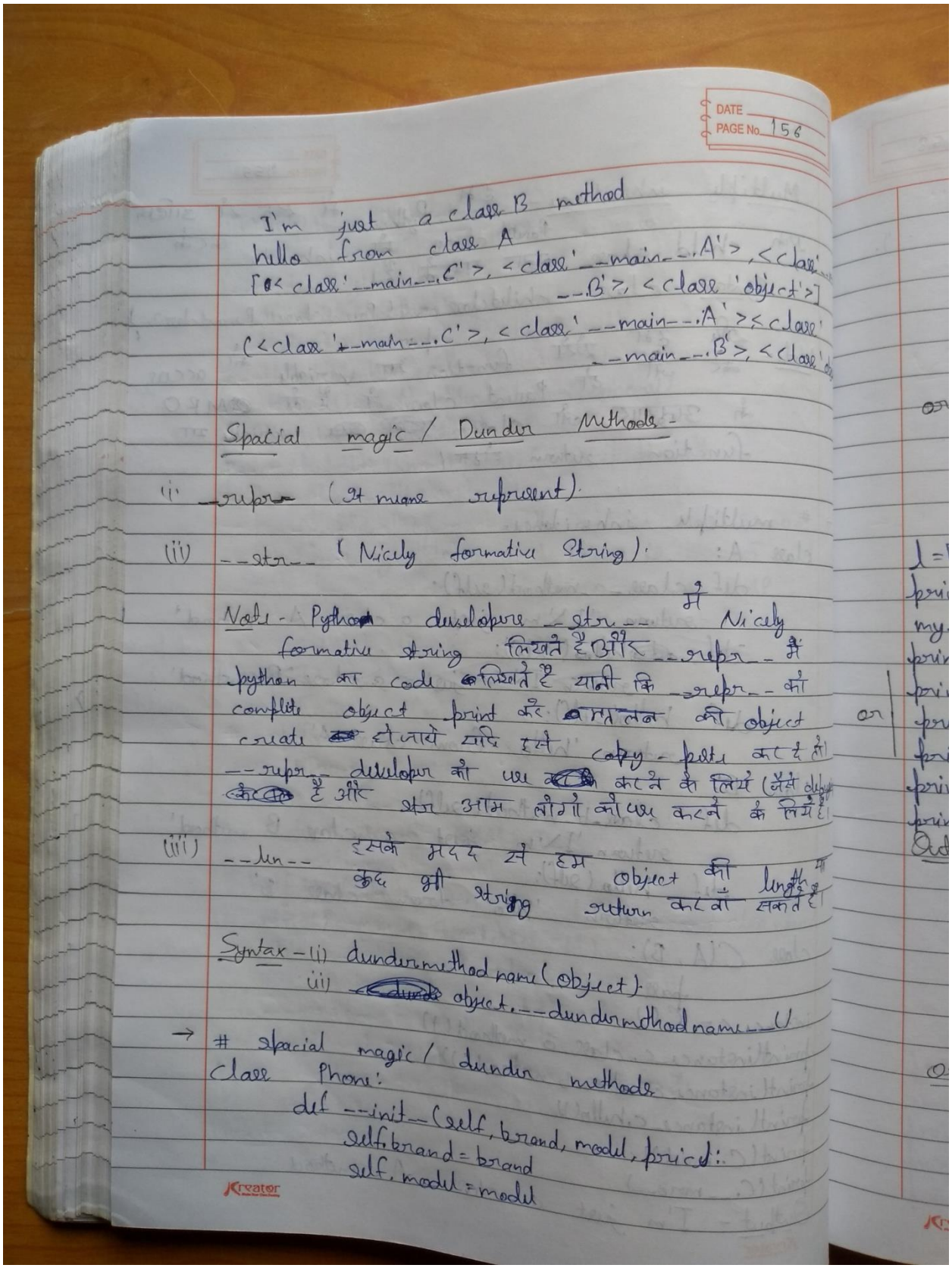
Syntax - class childclass name (Parentclass1, Parentclass2, ...):
pass

Note - यदि हम ऐसे function या variable को access करें जो दो Parent class में है तो MRO के अनुसार जो class variable या function return होगा।

→ # multiple inheritance -

```
class A:
    def class_a_method(self):
        return 'I'm just a class A method'
    def hello(self):
        return 'hello from class A'
class B:
    def class_b_method(self):
        return 'I'm just a class B method'
    def hello(self):
        return 'hello from class B'
class C(A, B):
    pass
instance_c = C()
print(instance_c.class_a_method())
print(instance_c.class_b_method())
print(instance_c.hello())
print(C.mro())
print(C.__mro__)
```

Output - I'm just a class A method



I'm just a class B method
hello from class A
[<class '__main__.C'>, <class '__main__.A'>, <class '__main__.B'>, <class 'object'>]
(<class '__main__.C'>, <class '__main__.A'>, <class '__main__.B'>, <class 'object'>)

Spacial magic / Dundar Methods -

(i) __repr__ (it means represent).

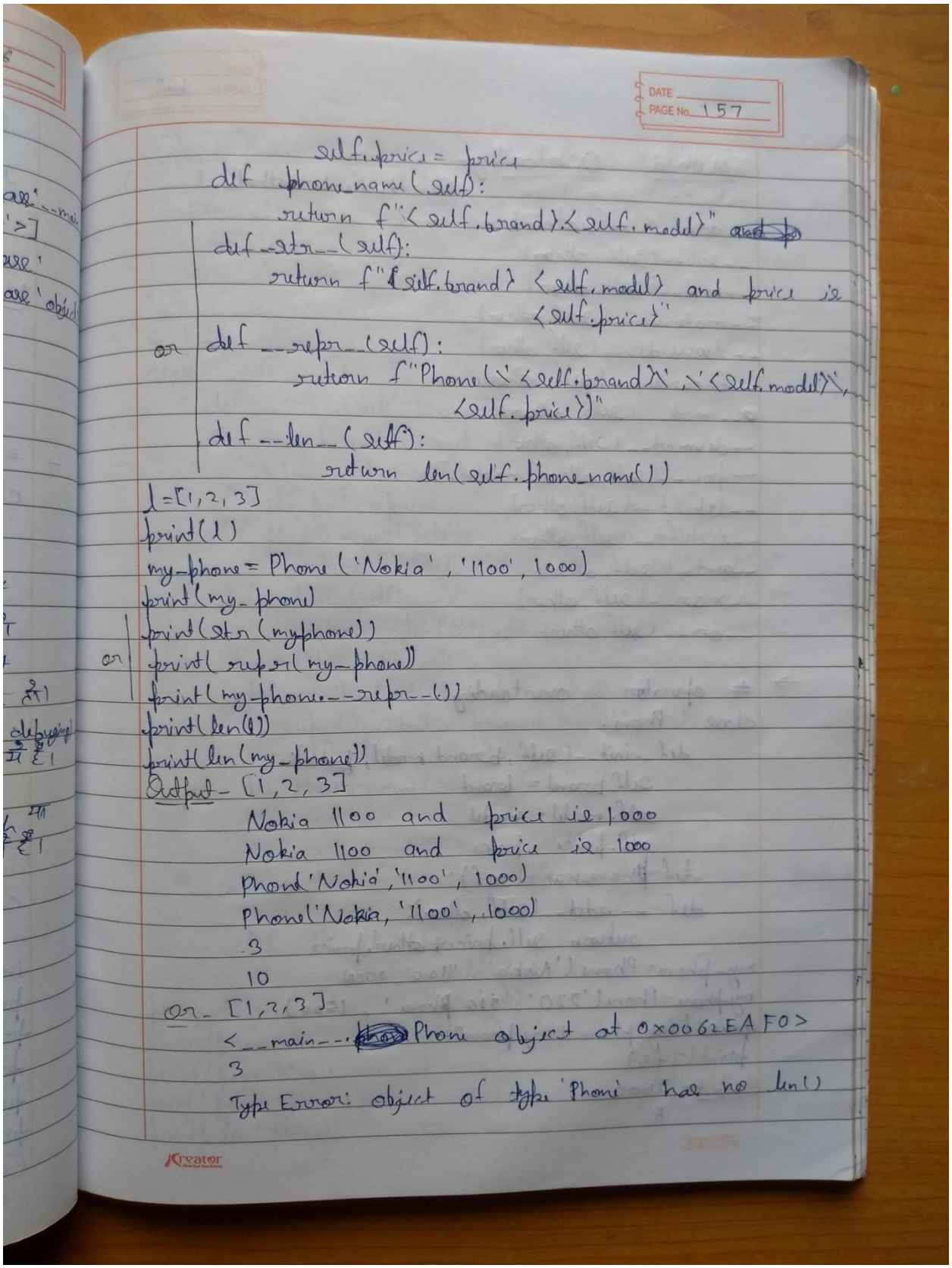
(ii) __str__ (Nicely formative String).

Note - Python developers __str__ में Nicely formative string लिखते हैं और __repr__ में python का code लिखते हैं यानी कि __repr__ को conflict object print करें तात्पर्य की object create हो जाये यदि इसे copy-paste कर दें तो __repr__ developer को use करने के लिये (जैसे debug करते हैं और str आम लोगों को use करने के लिये है।

(iii) __len__ इसके मदद से हम object की length या कुछ भी string return करवा सकते हैं।

Syntax - (i) dundermethod name (object).
(ii) object.__dundermethodname__()

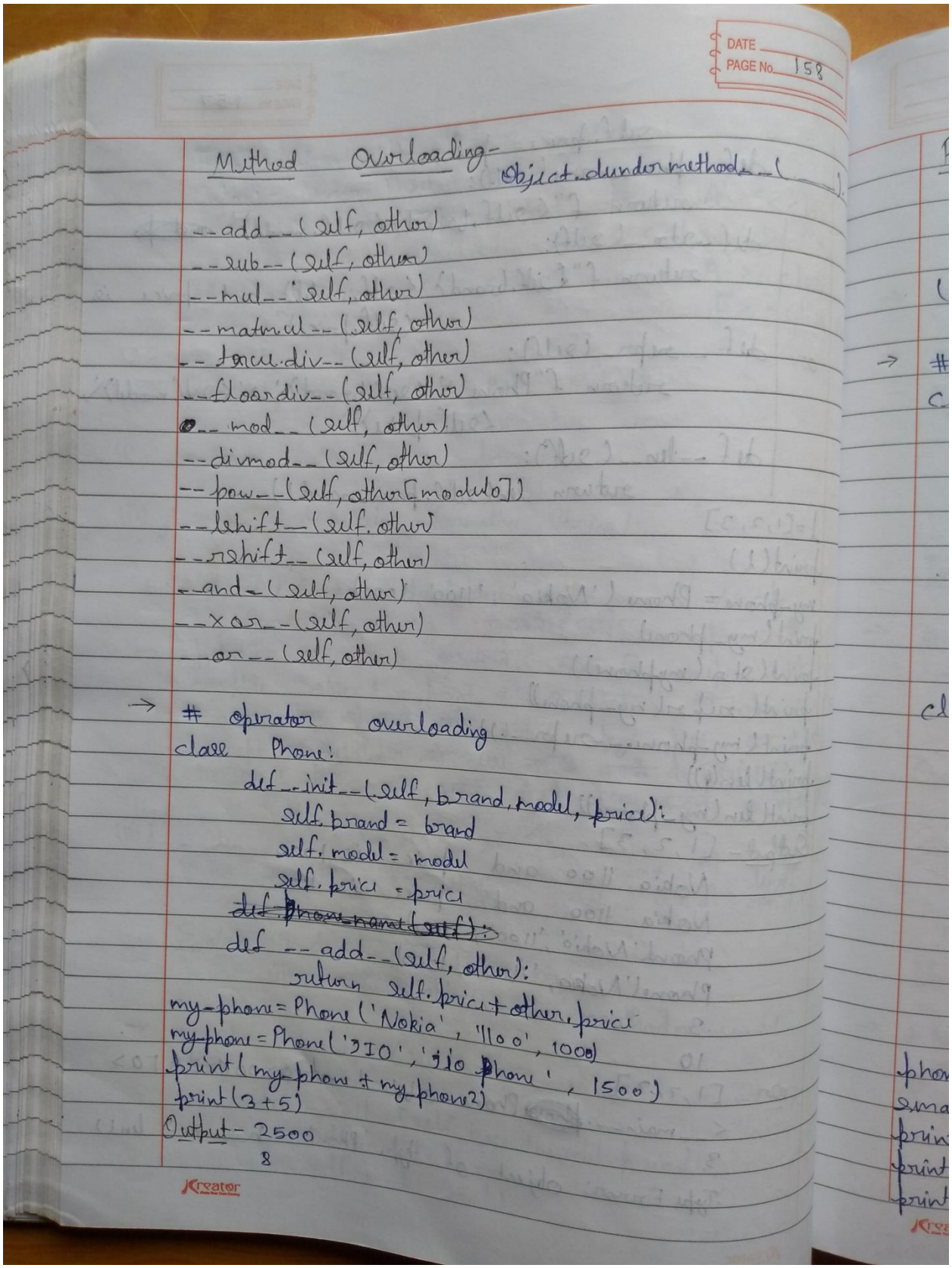
→ # spacial magic / dundar methods
Class Phone:
    def __init__(self, brand, modul, price):
        self.brand = brand
        self.modul = modul



```
        self.price = price
    def phone_name(self):
        return f"{self.brand}.{self.model}"
    def __str__(self):
        return f"{self.brand} {self.model} and price is
                                    {self.price}"
or  def __repr__(self):
        return f"Phone('{self.brand}','{self.model}',
                            {self.price})"
    def __len__(self):
        return len(self.phone_name())
l=[1,2,3]
print(l)
my_phone = Phone('Nokia', '1100', 1000)
print(my_phone)
print(str(myphone))
or  print(repr(my_phone))
print(my_phone.__repr__())
print(len(l))
print(len(my_phone))
```

Output - [1,2,3]
Nokia 1100 and price is 1000
Nokia 1100 and price is 1000
Phone('Nokia','1100',1000)
Phone('Nokia','1100',1000)
3
10

Or - [1,2,3]
<__main__.Phone object at 0x0062EAF0>
3
TypeError: object of type 'Phone' has no len()



DATE
PAGE No. 158

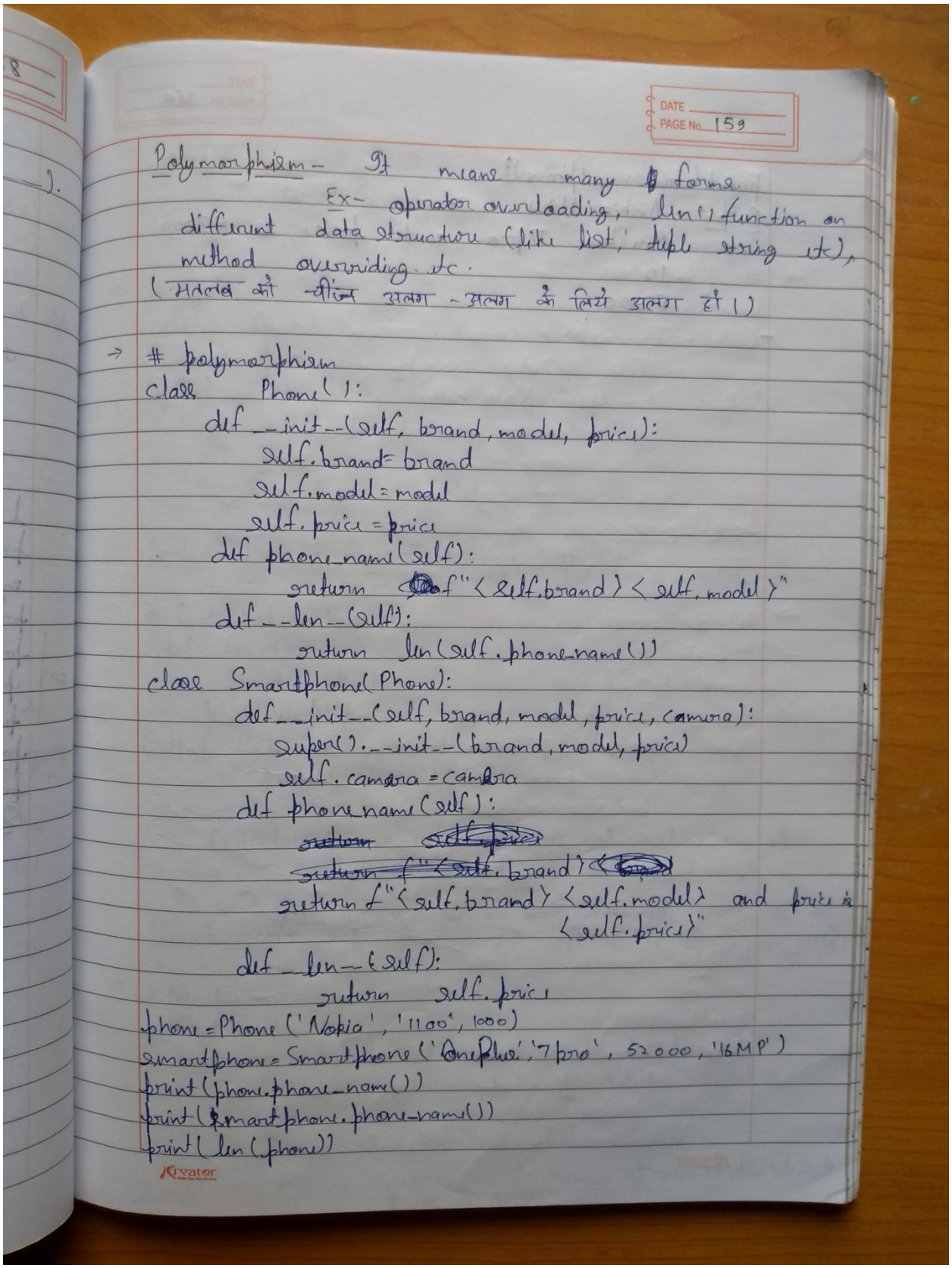
Method Overloading - Object.dunder methods __( )

__add__(self, other)
__sub__(self, other)
__mul__(self, other)
__matmul__(self, other)
__truediv__(self, other)
__floordiv__(self, other)
__mod__(self, other)
__divmod__(self, other)
__pow__(self, other[, modulo])
__lshift__(self, other)
__rshift__(self, other)
__and__(self, other)
__xor__(self, other)
__or__(self, other)

→ # operator overloading

```
class Phone:
    def __init__(self, brand, model, price):
        self.brand = brand
        self.model = model
        self.price = price
    def __add__(self, other):
        return self.price + other.price
my_phone = Phone('Nokia', '1100', 1000)
my_phone = Phone('JIO', 'Jio Phone', 1500)
print(my_phone + my_phone2)
print(3+5)
```

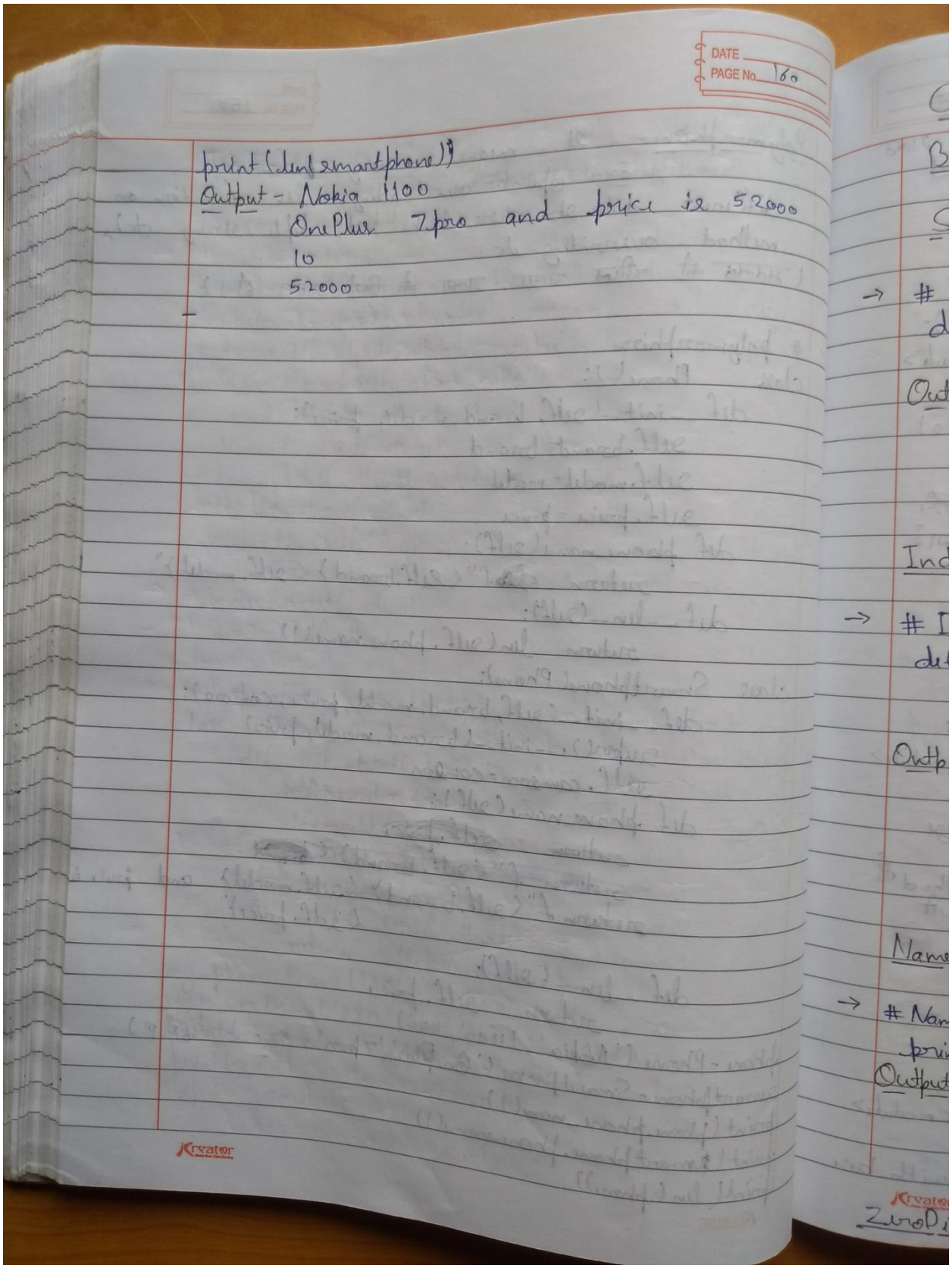
Output - 2500
8

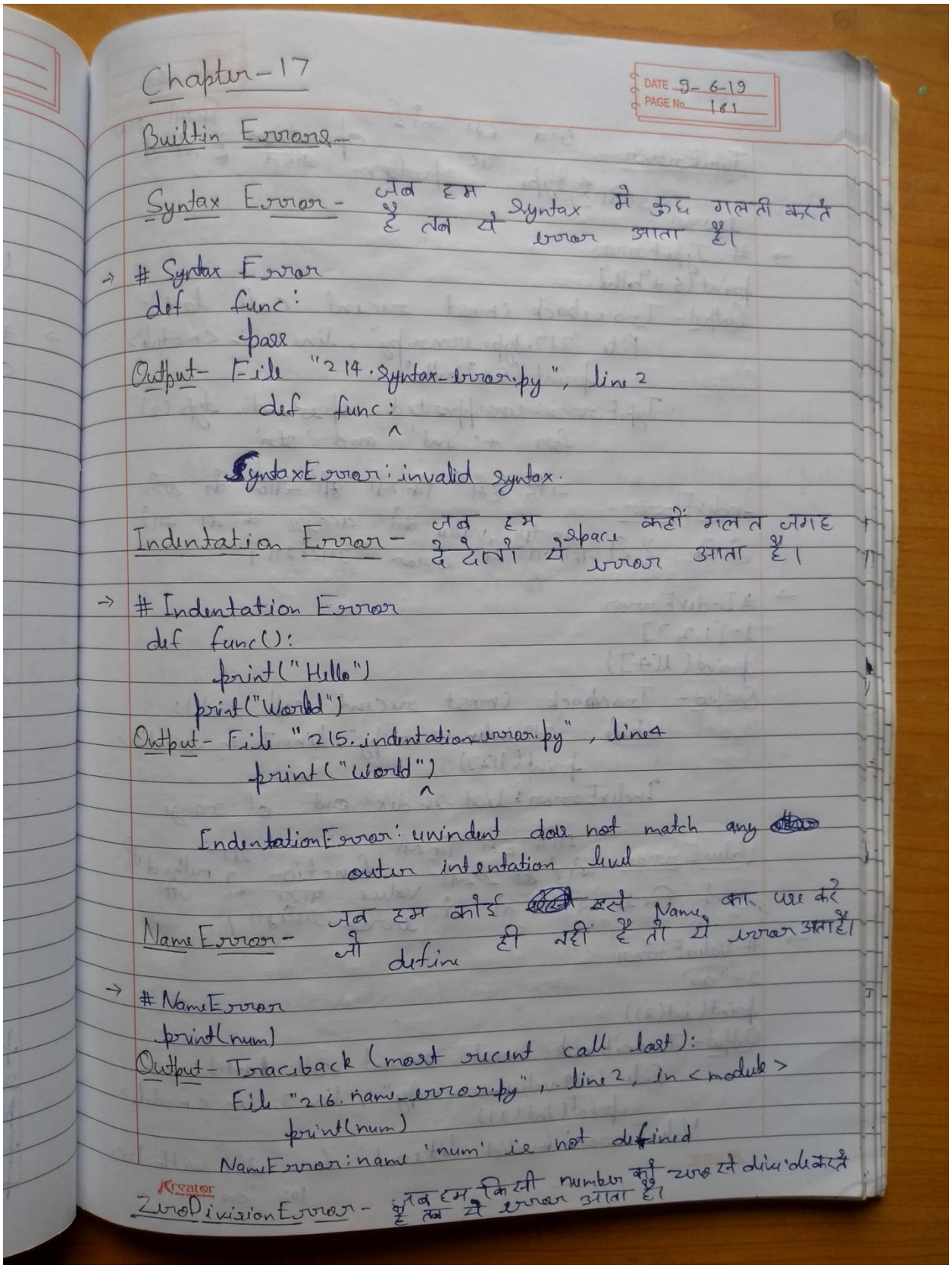


DATE
PAGE No. 159

Polymorphism - It means many forms.
Ex- operator overloading, len() function on different data structure (like list, tuple, string etc), method overriding etc.
(मतलब की चीज अलग - अलग के लिये अलग हो।)

→ # polymorphism
class Phone():
    def __init__(self, brand, model, price):
        self.brand = brand
        self.model = model
        self.price = price
    def phone_name(self):
        return f"{self.brand} {self.model}"
    def __len__(self):
        return len(self.phone_name())
class Smartphone(Phone):
    def __init__(self, brand, model, price, camera):
        super().__init__(brand, model, price)
        self.camera = camera
    def phone_name(self):
        return f"{self.brand} {self.model} and price is {self.price}"
    def __len__(self):
        return self.price
phone = Phone('Nokia', '1100', 1000)
smartphone = Smartphone('OnePlus', '7 pro', 52000, '16MP')
print(phone.phone_name())
print(smartphone.phone_name())
print(len(phone))


DATE
PAGE No. 160
print (dual smartphone)
Output - Nokia 1100
OnePlus 7pro and price is 52000
10
52000
Kreator



Chapter-17

DATE 9-6-19
PAGE No. 161

Builtin Errors-

Syntax Error- जब हम Syntax में कुछ गलती करते हैं तब ये error आता है।

→ # Syntax Error
def func:
pass
Output- File "214.Syntax_error.py", line 2
def func:
^
SyntaxError: invalid syntax.

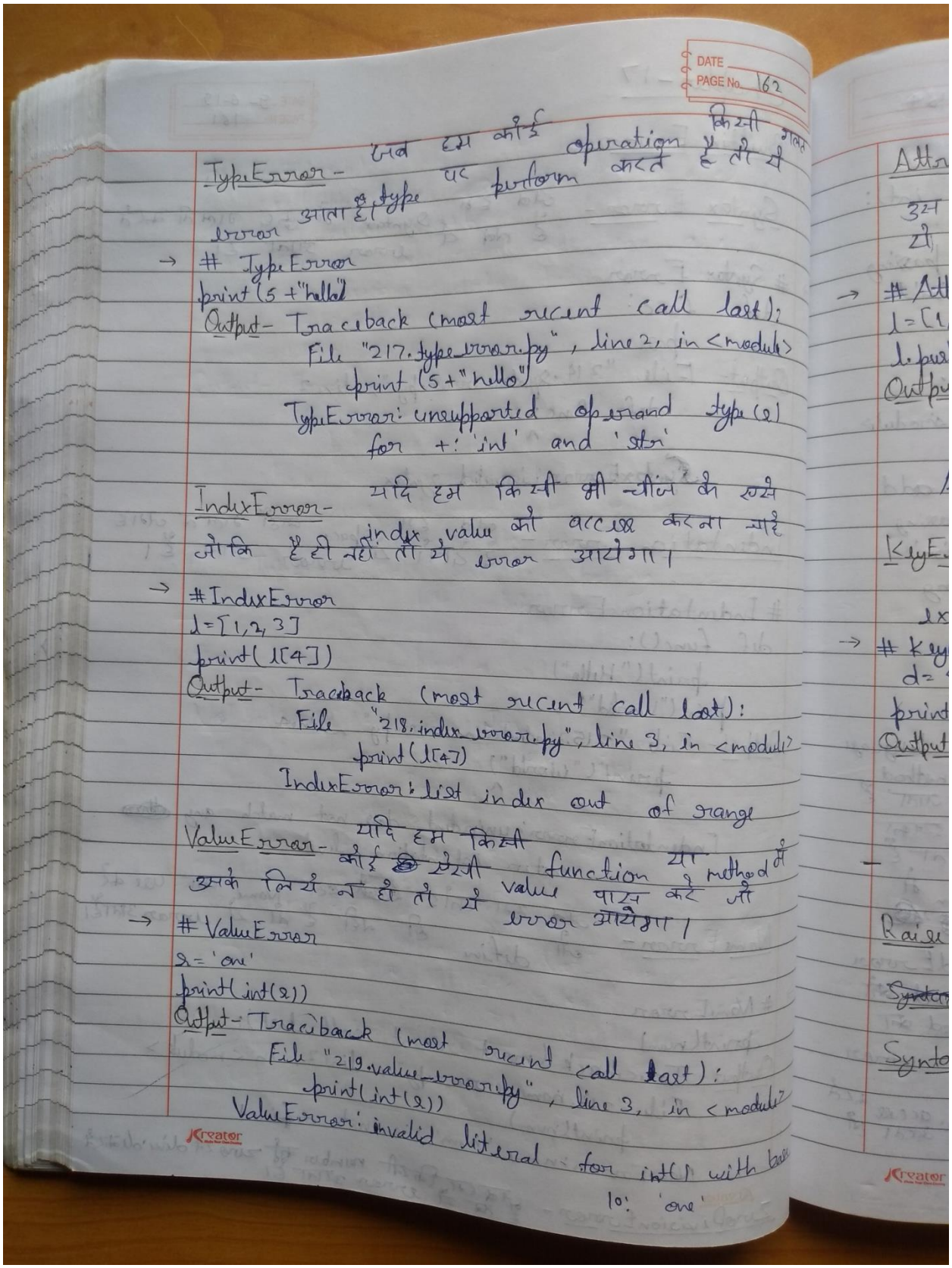
Indentation Error- जब हम space कहीं गलत जगह दे देते हैं तो ये error आता है।

→ # Indentation Error
def func():
print("Hello")
print("World")
Output- File "215.indentation_error.py", line 4
print("World")
^
IndentationError: unindent does not match any outer indentation level

Name Error- जब हम कोई ऐसे variable का use करें जो define ही नहीं है तो ये error आता है।

→ # NameError
print(num)
Output- Traceback (most recent call last):
File "216.Name_error.py", line 2, in <module>
print(num)
NameError: name 'num' is not defined

ZeroDivisionError- जब हम किसी number को zero से divide करते हैं तब ये error आता है।



TypeError - जब हम कोई operation किसी गलत type पर perform करते हैं तो ये error आता है।

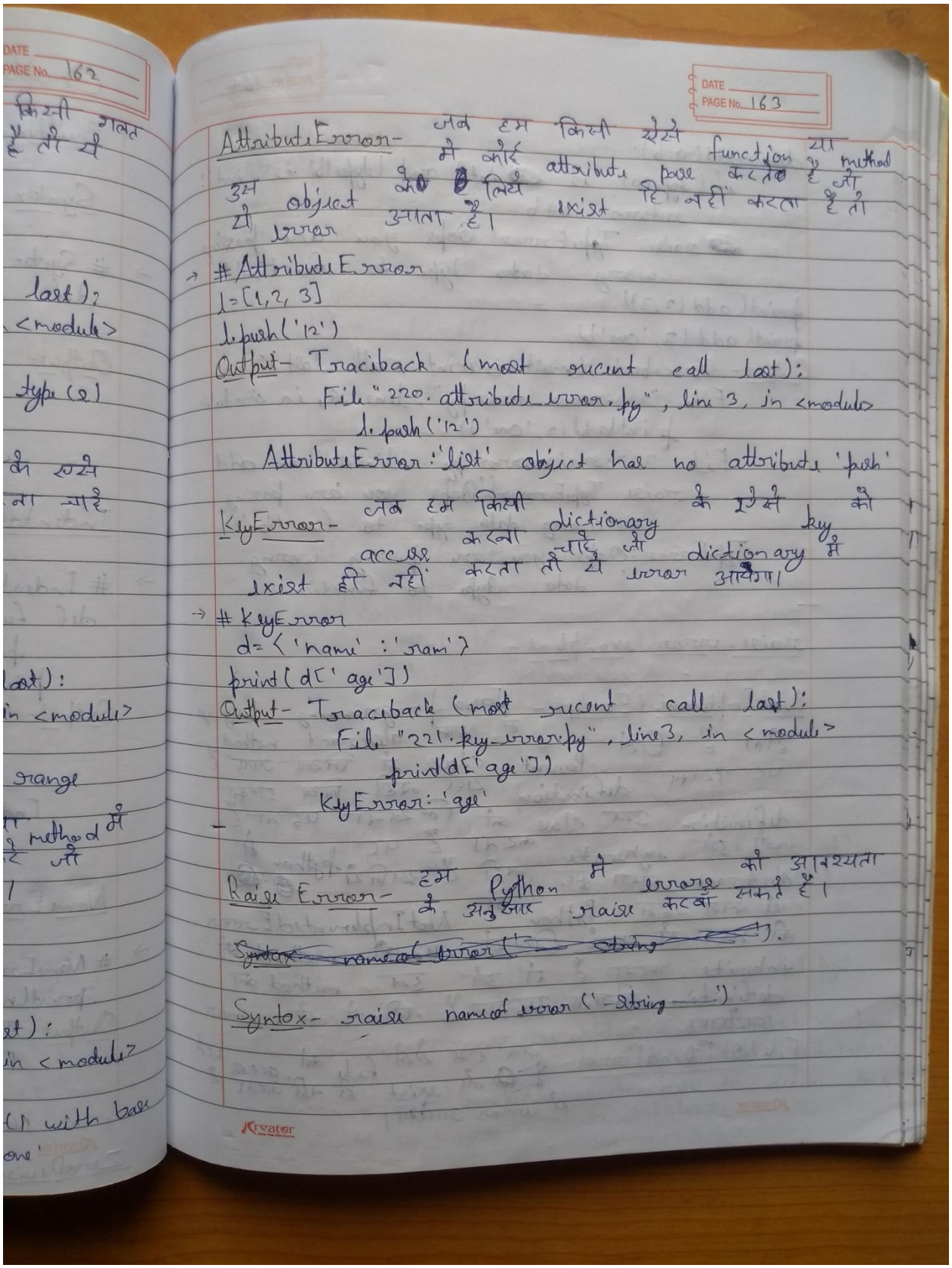
→ # TypeError
print(5 + "hello")
Output - Traceback (most recent call last):
File "217.type_error.py", line 2, in <module>
print(5 + "hello")
TypeError: unsupported operand type(s) for +: 'int' and 'str'

IndexError - यदि हम किसी भी चीज के ऐसे index value को access करना चाहें जो कि है ही नहीं तो ये error आयेगा।

→ # IndexError
l = [1, 2, 3]
print(l[4])
Output - Traceback (most recent call last):
File "218.index_error.py", line 3, in <module>
print(l[4])
IndexError: list index out of range

ValueError - यदि हम किसी function या method में कोई ऐसी value पास करें जो उसके लिये न हो तो ये error आयेगा।

→ # ValueError
x = 'one'
print(int(x))
Output - Traceback (most recent call last):
File "219.value_error.py", line 3, in <module>
print(int(x))
ValueError: invalid literal for int() with base 10: 'one'



AttributeError- जब हम किसी ऐसे function या method में कोई attribute pass करते हैं जो उस object के लिये exist ही नहीं करता है तो यह error आता है।

→ #AttributeError
l=[1,2,3]
l.push('12')
Output- Traceback (most recent call last):
File "220.attribute_error.py", line 3, in <module>
l.push('12')
AttributeError: 'list' object has no attribute 'push'

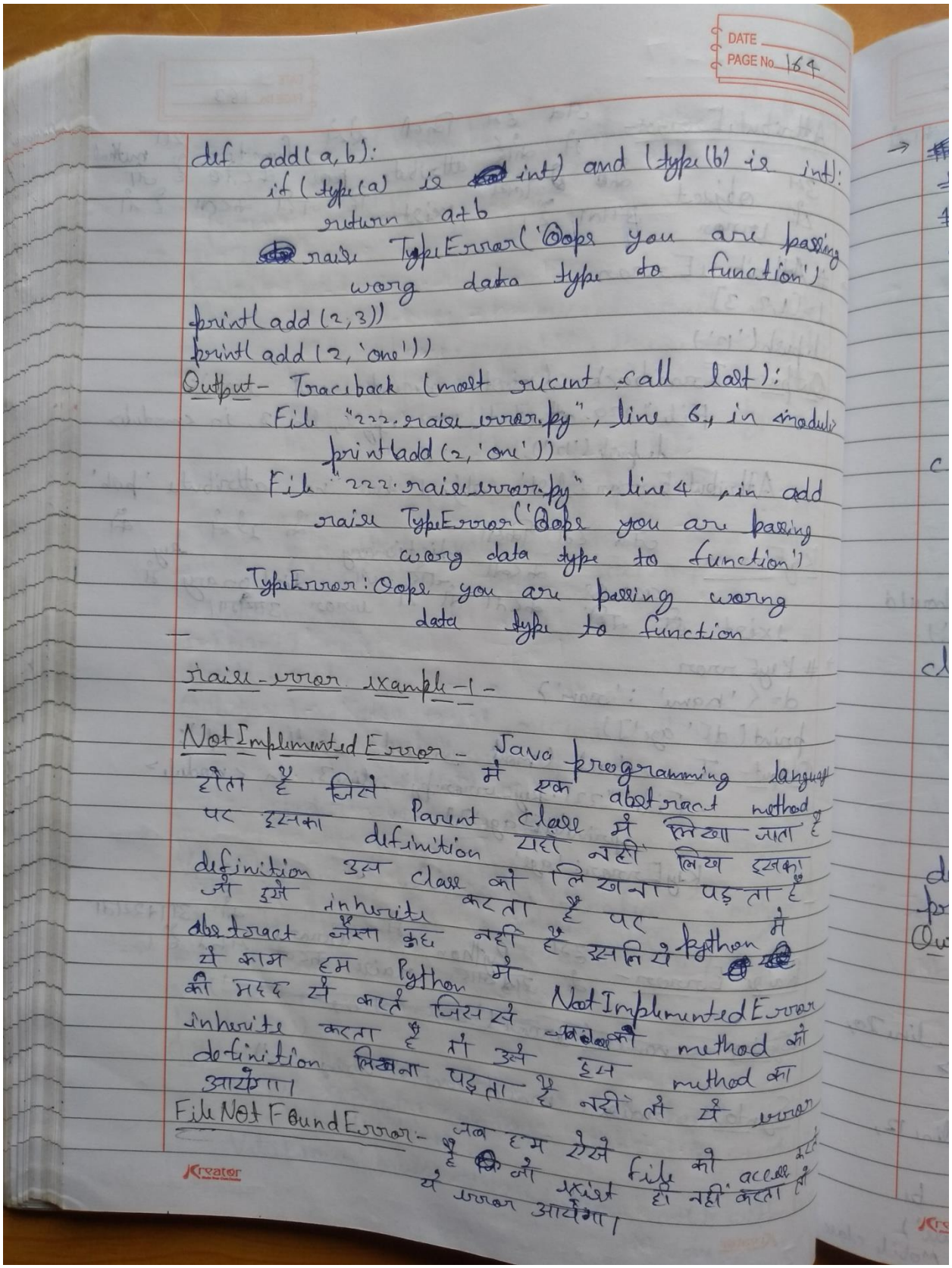
KeyError- जब हम किसी dictionary के द्वारे से key को access करना चाहें जो dictionary में exist ही नहीं करता तो यह error आयेगा।

→ #KeyError
d={'name':'ram'}
print(d['age'])
Output- Traceback (most recent call last):
File "221.key_error.py", line 3, in <module>
print(d['age'])
KeyError: 'age'

Raise Error- हम Python में errors को आवश्यता के अनुसार raise करवा सकते हैं।

~~Syntax- name of error ('string').~~

Syntax- raise name of error ('string')



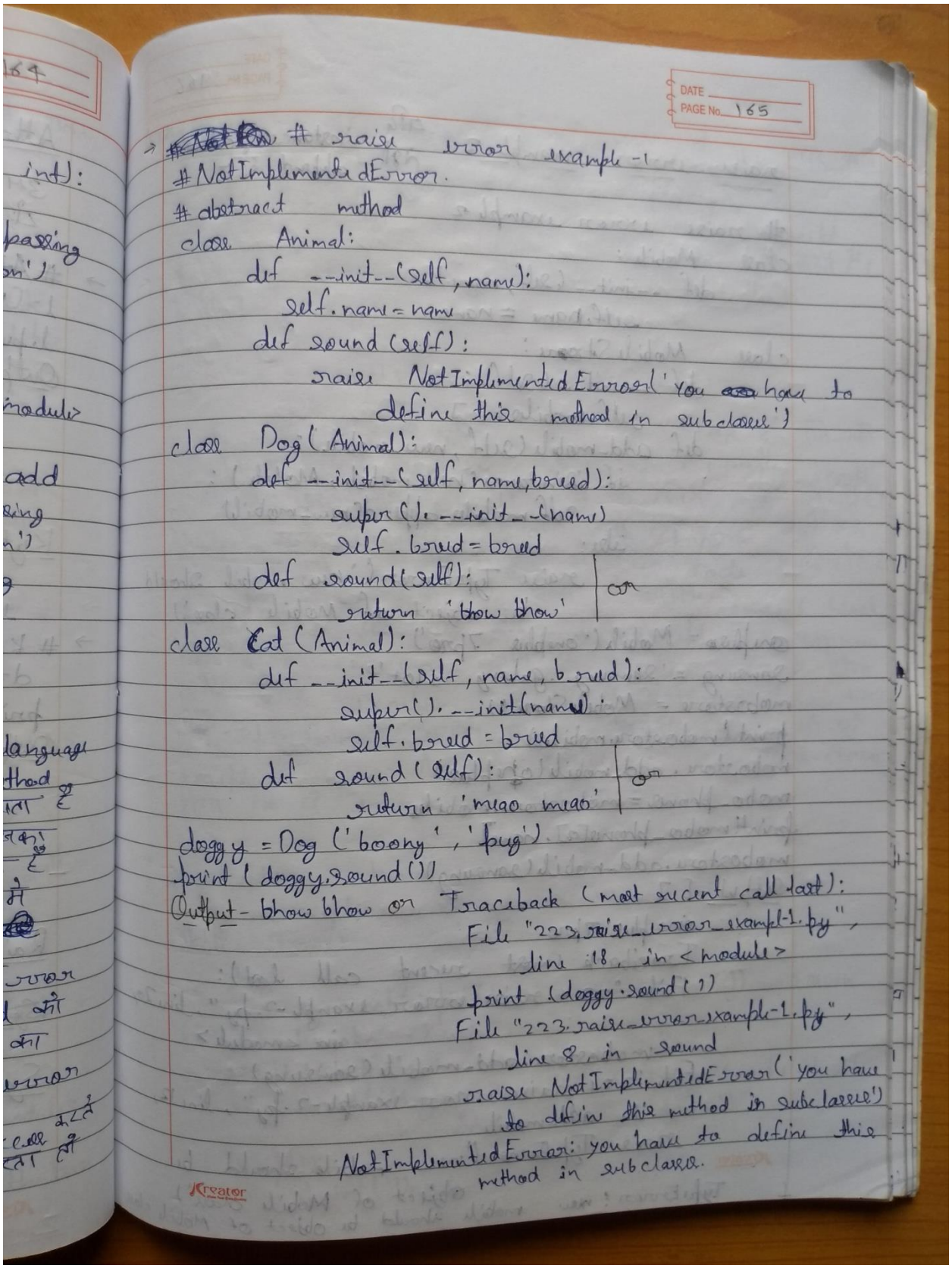
```
def add(a,b):
    if (type(a) is int) and (type(b) is int):
        return a+b
    raise TypeError('Oops you are passing
        wrong data type to function')
print(add(2,3))
print(add(2,'one'))
```

Output- Traceback (most recent call last):
File "222.raise_error.py", line 6, in <module>
print(add(2,'one'))
File "222.raise_error.py", line 4, in add
raise TypeError('Oops you are passing
wrong data type to function')
TypeError: Oops you are passing wrong
data type to function

raise-error example-1-

NotImplementedError - Java programming language में एक abstract method होता है जिसे Parent class में लिखा जाता है पर इसका definition वहाँ नहीं लिख इसका definition उस class को लिखना पड़ता है जो इसे inherit करता है पर Python में abstract जैसा कुछ नहीं है इसलिये Python में यह काम हम Python में NotImplementedError की मदद से करते जिससे कोई method को inherit करता है तो उसे इस method का definition लिखना पड़ता है नहीं तो ये error आयेगा।

FileNotFoundError - जब हम ऐसे file को access करने की कोशिश करते हैं जो exist ही नहीं करता तो ये error आयेगा।

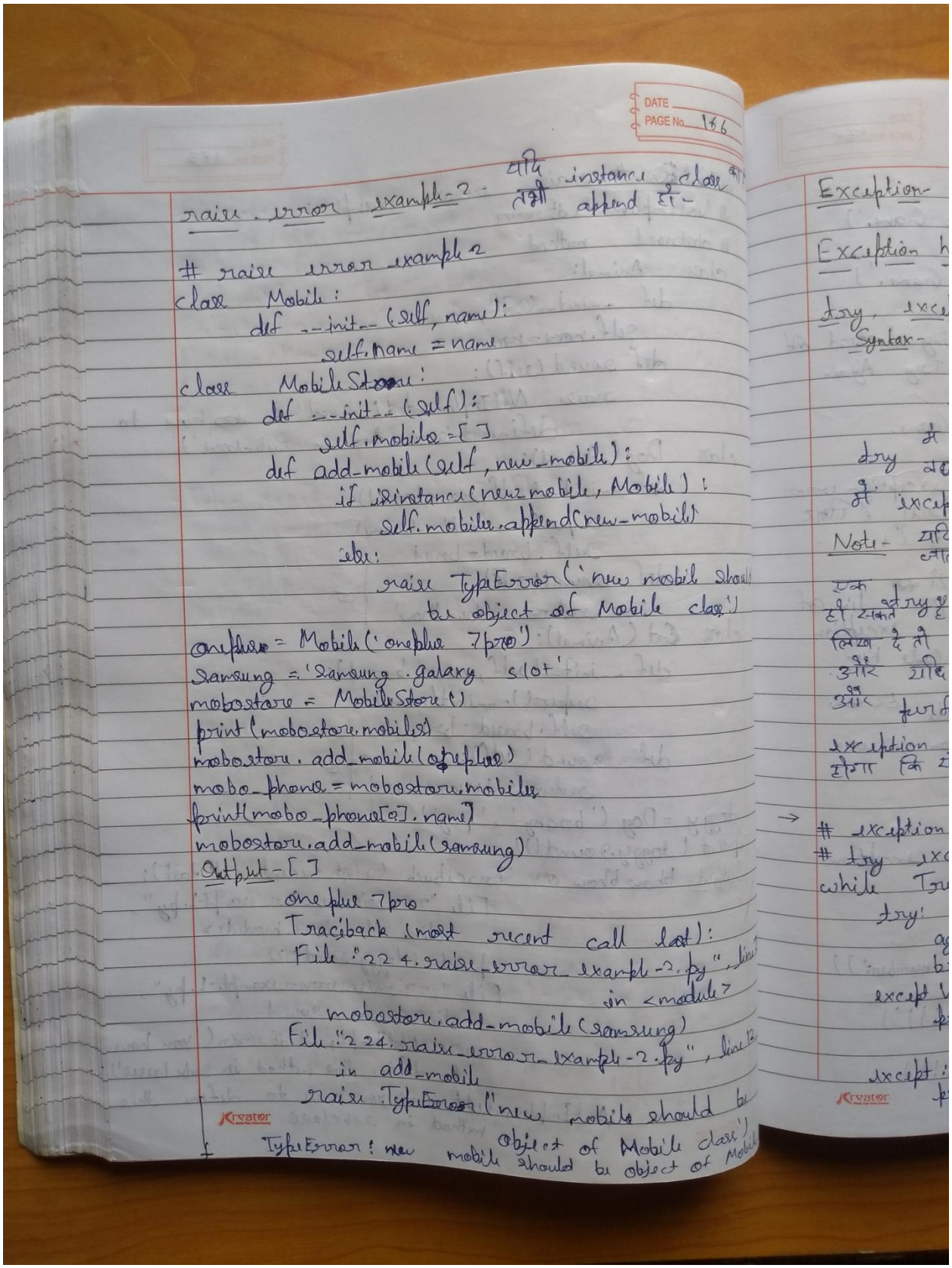


```
# raise error example -1
#NotImplementedError.
#abstract method
class Animal:
    def __init__(self, name):
        self.name = name
    def sound(self):
        raise NotImplementedError('you have to
            define this method in subclasses')
class Dog(Animal):
    def __init__(self, name, breed):
        super().__init__(name)
        self.breed = breed
    def sound(self):
        return 'bhow bhow'
class Cat(Animal):
    def __init__(self, name, breed):
        super().__init(name)
        self.breed = breed
    def sound(self):
        return 'miao miao'
doggy = Dog('boony', 'pug')
print(doggy.sound())
```

or

or

Output- bhow bhow or Traceback (most recent call last):
File "223.raise_error_example-1.py",
line 18, in <module>
print(doggy.sound())
File "223.raise_error_example-1.py",
line 8, in sound
raise NotImplementedError('you have
to define this method in subclasses')
NotImplementedError: you have to define this
method in subclasses.



raise error example-2 - यदि instance class की नहीं append है -

```
# raise error example 2
class Mobile:
    def __init__(self, name):
        self.name = name
class MobileStore:
    def __init__(self):
        self.mobiles = []
    def add_mobile(self, new_mobile):
        if isinstance(new_mobile, Mobile):
            self.mobiles.append(new_mobile)
        else:
            raise TypeError('new mobile should
                be object of Mobile class')
oneplus = Mobile('oneplus 7pro')
samsung = 'Samsung galaxy s10+'
mobostore = MobileStore()
print(mobostore.mobiles)
mobostore.add_mobile(oneplus)
mobo_phone = mobostore.mobiles
print(mobo_phone[0].name)
mobostore.add_mobile(samsung)
```

Output - []

```
oneplus 7pro
Traceback (most recent call last):
  File "224.raise_error_example-2.py", line
                                in <module>
    mobostore.add_mobile(samsung)
  File "224.raise_error_example-2.py", line 12
    in add_mobile
    raise TypeError('new mobile should be
                object of Mobile class')
TypeError: new mobile should be object of Mobile
```

Exception-

Exception h

try, exce

Syntax-

try म

में except

Note- यदि

एक try

लिखा है तो

और यदि

और फिर

exception

होगा कि

exception

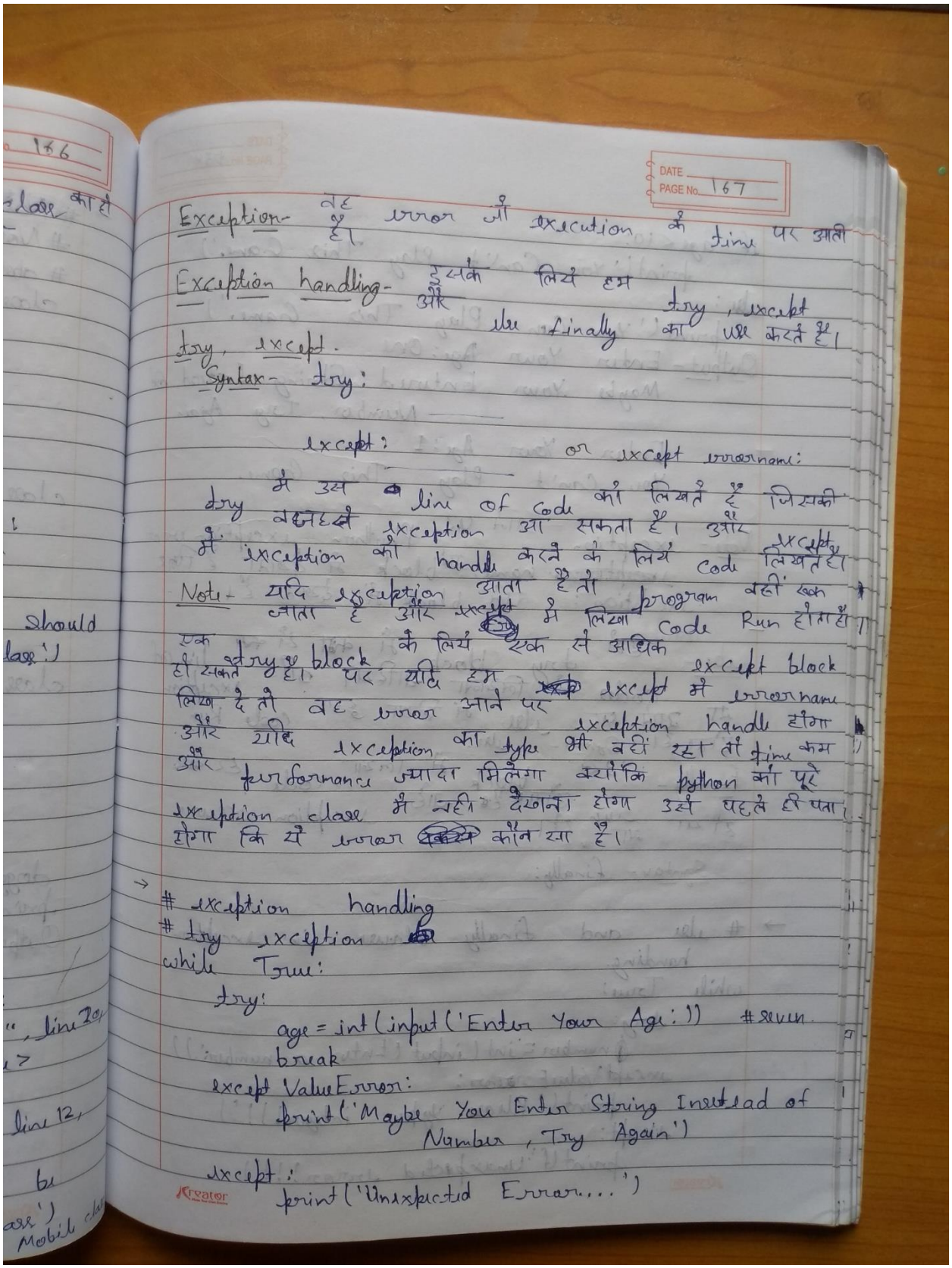
try exc

while Tru

try:

except

except



DATE
PAGE No. 167

Exception- वह error जो execution के time पर आती है।

Exception handling- इसके लिये हम try, except और use finally का use करते हैं।

try, except.

Syntax- try:

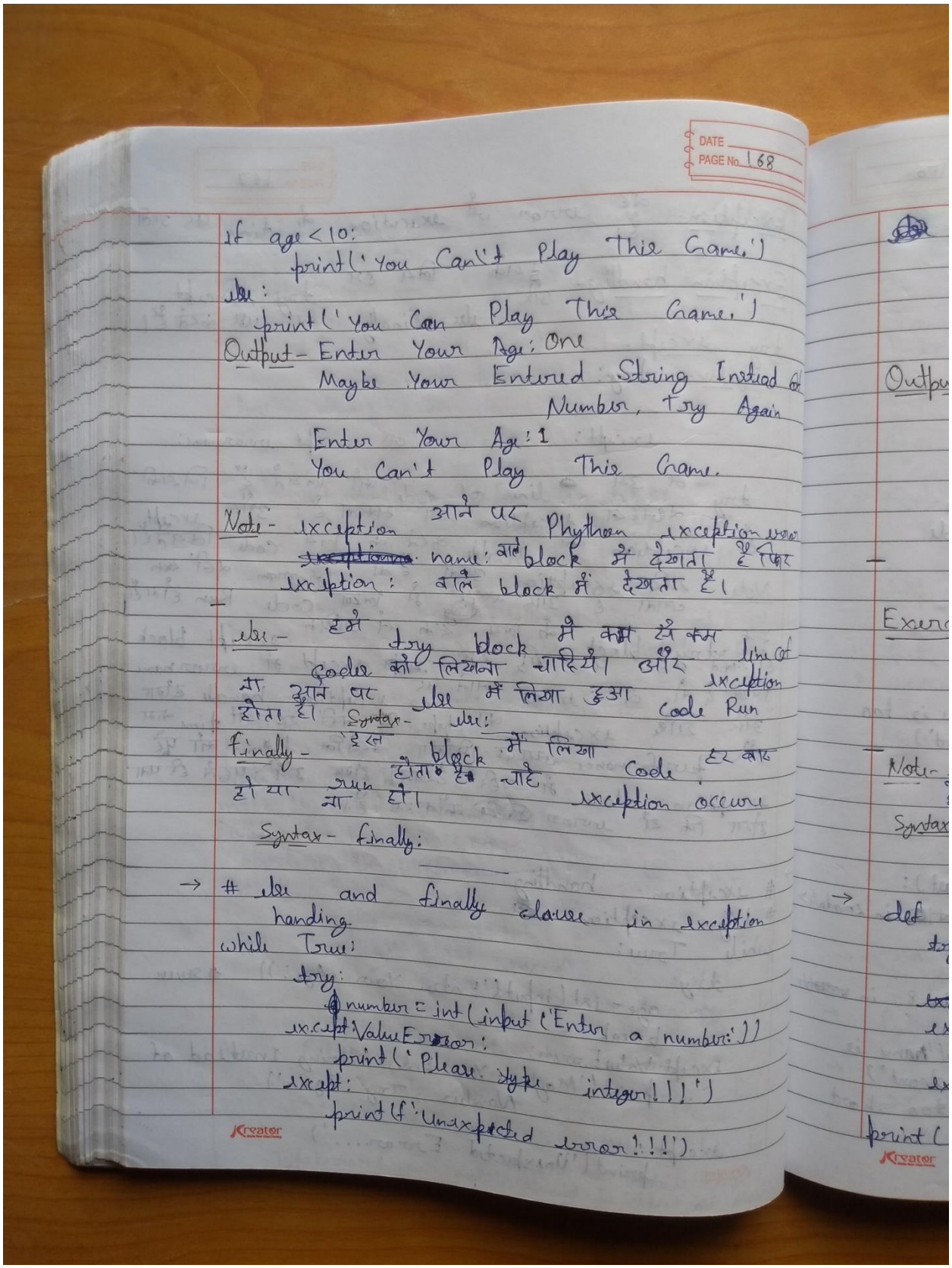
except: or except errorname:

में उस line of code को लिखते हैं जिसकी try वजह से exception आ सकता है। उसके except में exception को handle करने के लिये code लिखते हैं।

Note- यदि exception आता है तो program वहीं रुक जाता है और except में लिखा code Run होता है।

एक try के लिये एक से अधिक except block हो सकते हैं। पर यदि हम except में errorname लिख दे तो वह error आने पर exception handle होगा और यदि exception का type भी वही रहा तो time कम और performance ज्यादा मिलेगा क्योंकि python को पूरे exception class में नहीं देखना होगा उसे पहले ही पता होगा कि वो error कौन सा है।

```
# exception handling
# try exception
while True:
    try:
        age = int(input('Enter Your Age:'))  # seven
        break
    except ValueError:
        print('Maybe You Enter String Instead of
                Number, Try Again')
    except:
        print('Unexpected Error...')
```



```
if age<10:
    print('You Can't Play This Game.')
else:
    print('You Can Play This Game.')
```

Output- Enter Your Age: One
Maybe Your Entered String Instead of Number, Try Again
Enter Your Age: 1
You Can't Play This Game.

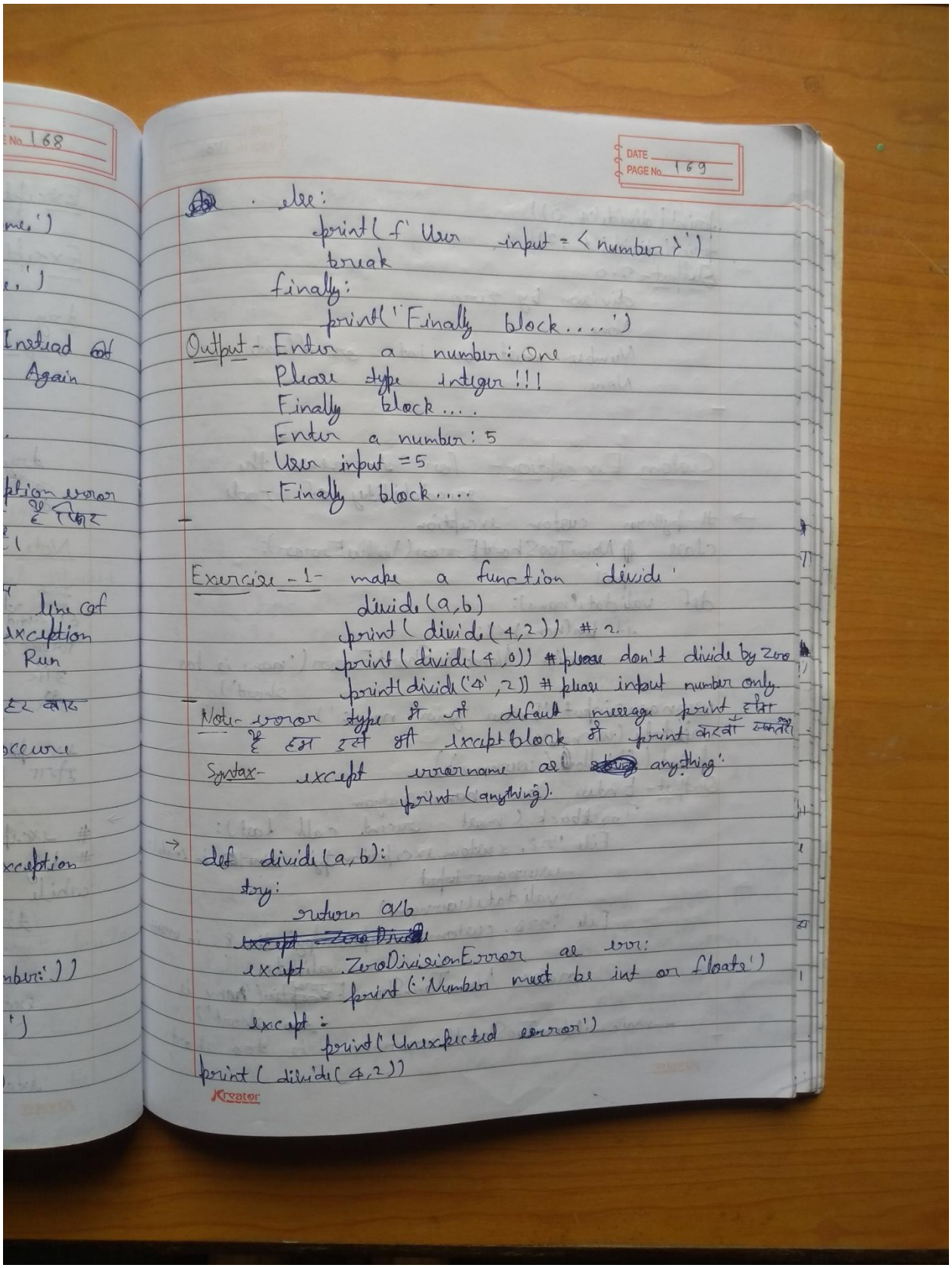
Note- exception आने पर Phython exception error name: वाले block में देखता है फिर exception : वाले block में देखता है।

Use- हमें try block में वह सभी line of code को लिखना चाहिये। और exception आने पर else में लिखा हुआ code Run होता है। Syntax- else:

finally- इस block में लिखा code हर बार होता है चाहे exception occure हो या run हो।

Syntax- finally:

```
# else and finally clause in exception handling
while True:
    try:
        number = int(input('Enter a number:'))
    except ValueError:
        print('Please type integer!!!')
    except:
        print('Unexpected error!!!')
```



```
    else:
        print(f'User input = {number}')
        break
    finally:
        print('Finally block....')
```

Output - Enter a number: One
Please type integer !!!
Finally block....
Enter a number: 5
User input = 5
Finally block....

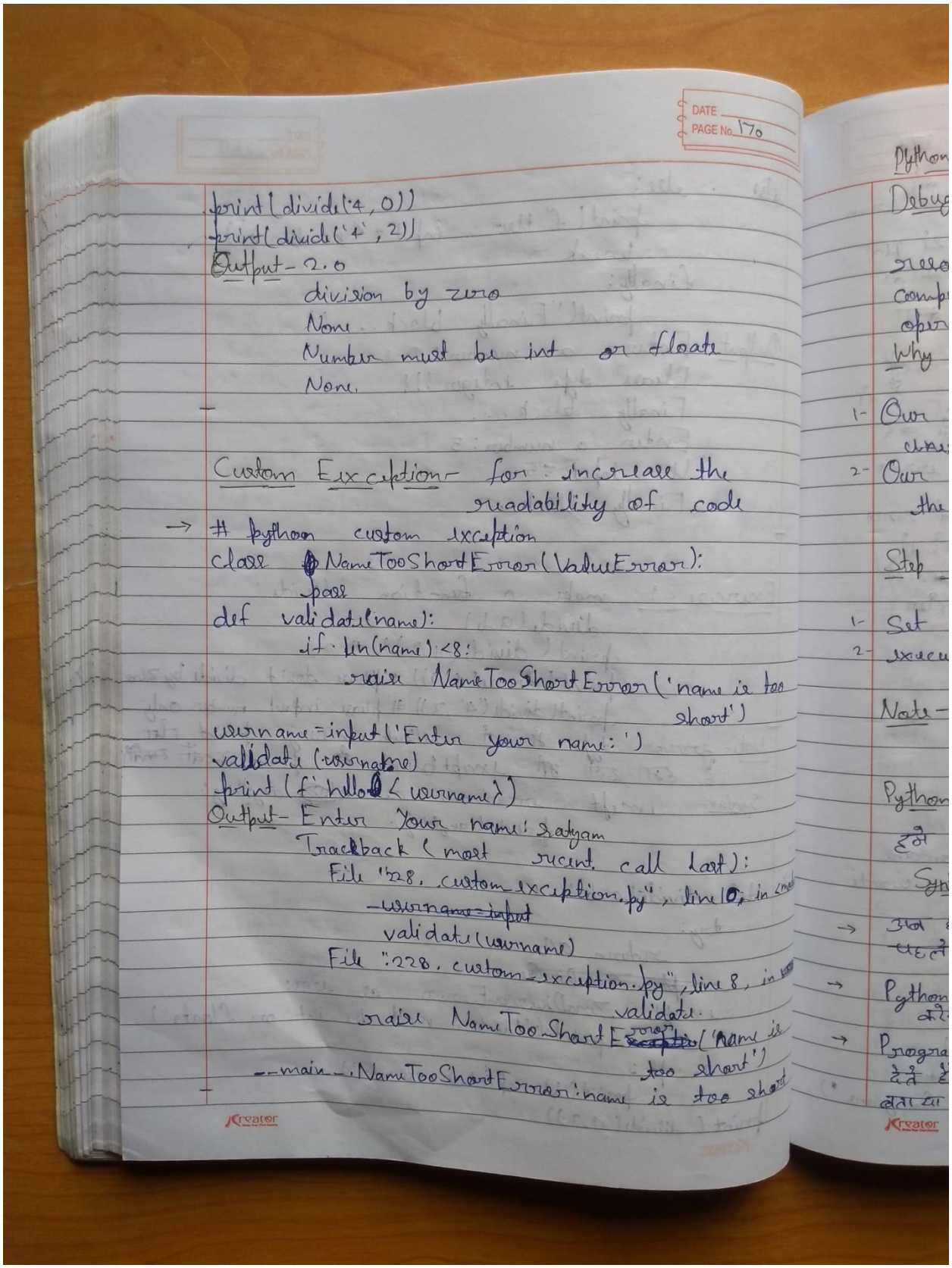
Exercise - 1- make a function 'divide'

```
divide(a,b)
print(divide(4,2)) # 2.
print(divide(4,0)) # please don't divide by zero
print(divide('4',2)) # please input number only
```

Note- error type में जो default message print होता है उस को हटा कर except block में print करवा सकते हैं

Syntax- except errorname as anything:
print(anything)

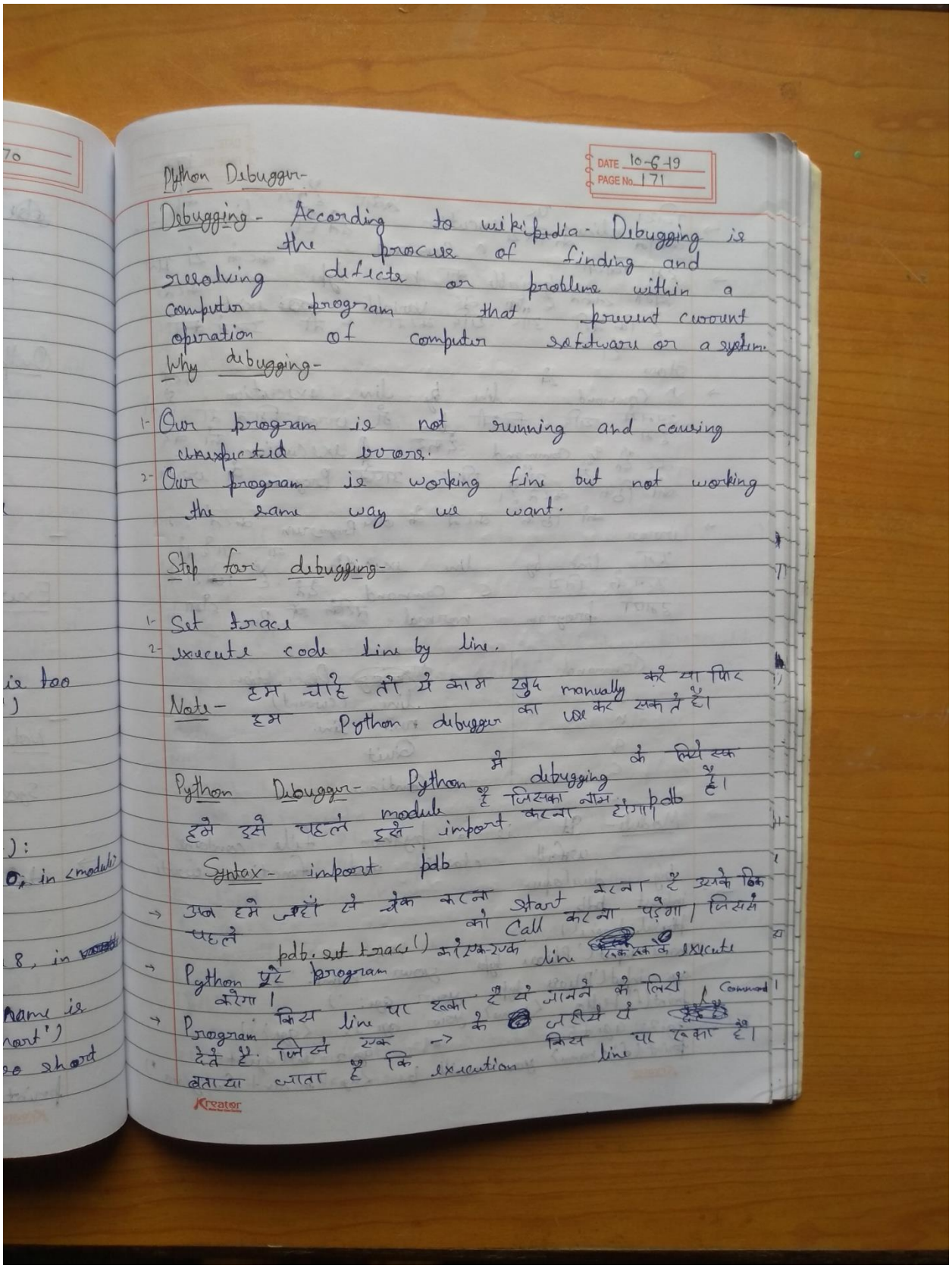
```
def divide(a,b):
    try:
        return a/b
    except ZeroDivisionError as err:
        print('Number must be int or floats')
    except:
        print('Unexpected error')
print(divide(4,2))
```



print(divide(4,0))
print(divide('4',2))
Output- 2.0
division by zero
None
Number must be int or floate
None.

Custom Exception- for increase the readability of code

→ # python custom exception
class NameTooShortError(ValueError):
pass
def validate(name):
if len(name)<8:
raise NameTooShortError('name is too short')
username=input('Enter your name:')
validate(username)
print(f'hello {username}')
Output- Enter your name: satyam
Traceback (most recent call last):
File "28. custom_exception.py", line 10, in <module>
username=input
validate(username)
File ":228. custom_exception.py", line 8, in validate
raise NameTooShortError('name is too short')
__main__.NameTooShortError: name is too short



Python Debugger-

DATE 10-6-19
PAGE No. 171

Debugging - According to wikipedia - Debugging is the process of finding and resolving defects or problems within a computer program that prevent correct operation of computer software or a system.

Why debugging-

1- Our program is not running and causing unexpected errors.
2- Our program is working fine but not working the same way we want.

Step for debugging-

1- Set trace
2- execute code line by line.

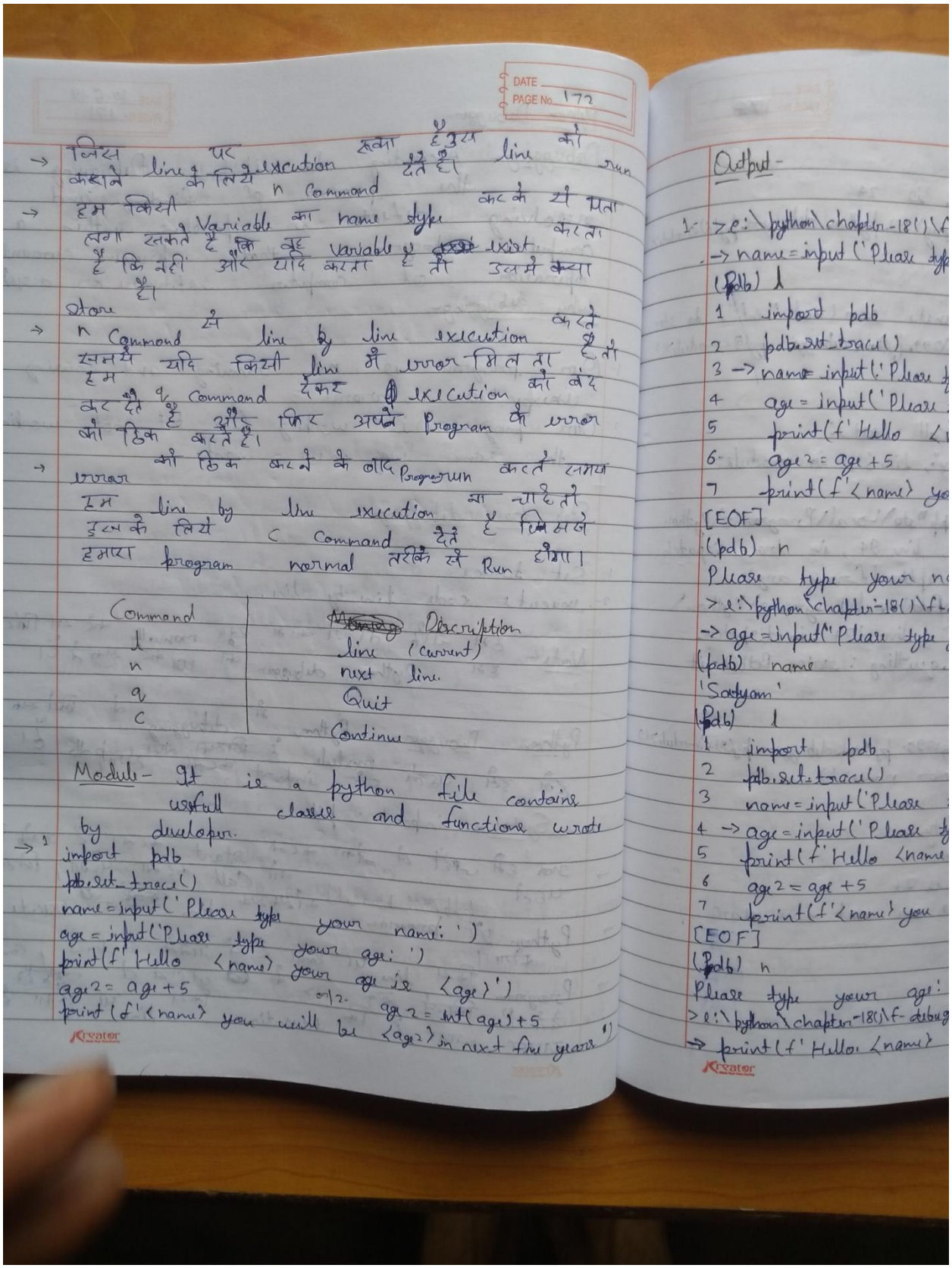
Note- हम चाहे तो ये काम खुद manually करें या फिर हम Python debugger का use कर सकते हैं।

Python Debugger- Python में debugging के लिये एक module है जिसका नाम pdb है। हमें इसे पहले इसे import करना होगा।

Syntax- import pdb

→ जब हमें कहाँ से चेक करना start करना है उसके ठीक पहले pdb.set_trace() को Call करना पड़ेगा। जिससे Python हर program का एक-एक line एक एक के execute करेगा।

→ Program जिस line पर रुका है ये जानने के लिये Command देते हैं। जिसे एक -> के द्वारा दर्शाते हैं। बताया जाता है कि execution जिस line पर रुका है।



→ किस line पर execution रुका हुआ है उसे देखने के लिये n Command देते हैं। line को run कराने

→ हम किसी Variable का name type करके ये पता लगा सकते हैं कि वह variable ~~exist~~ exist है कि नहीं और यदि करता है तो उसमें क्या है।

→ n Command से line by line execution करते समय यदि किसी line में error मिलता है तो हम c Command देकर execution को बंद कर देते हैं और फिर अपने Program के error को ठीक करते हैं।

→ error को ठीक करने के बाद Program run करते समय line by line execution ~~का~~ नहीं चाहते तो इसके लिये c Command देते हैं जिससे हमारा program normal तरीके से Run होगा।

Command	Description
l	line (current)
n	next line.
q	Quit
c	Continue

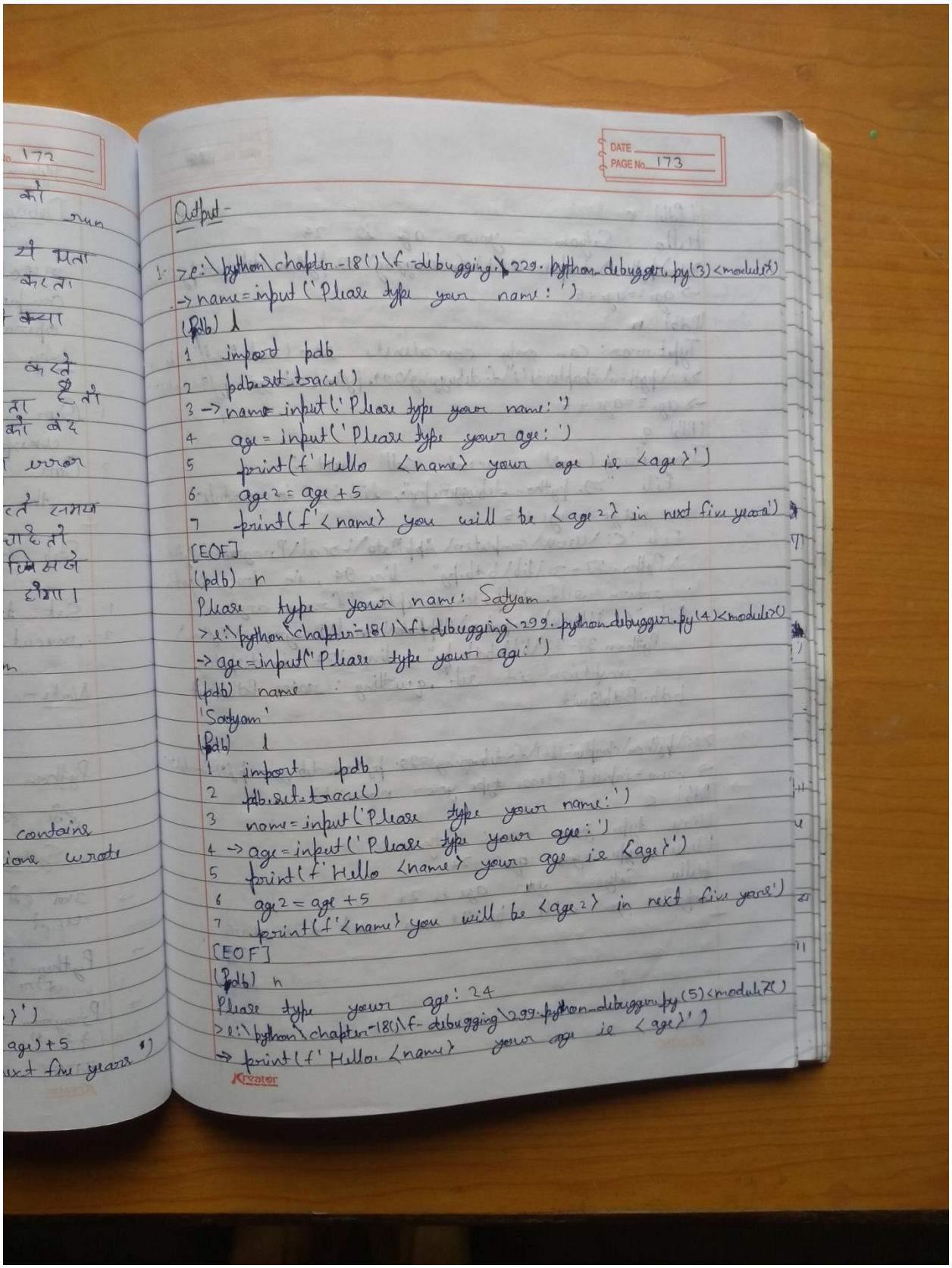
Module - It is a python file contains usefull classes and functions wrote by developer.

→ 1
```
import pdb
pdb.set_trace()
name = input('Please type your name: ')
age = input('Please type your age: ')
print(f'Hello <name> your age is <age>')
age2 = age + 5
print(f'<name> you will be <age2> in next five years')
```

2.
```
age2 = int(age) + 5
```

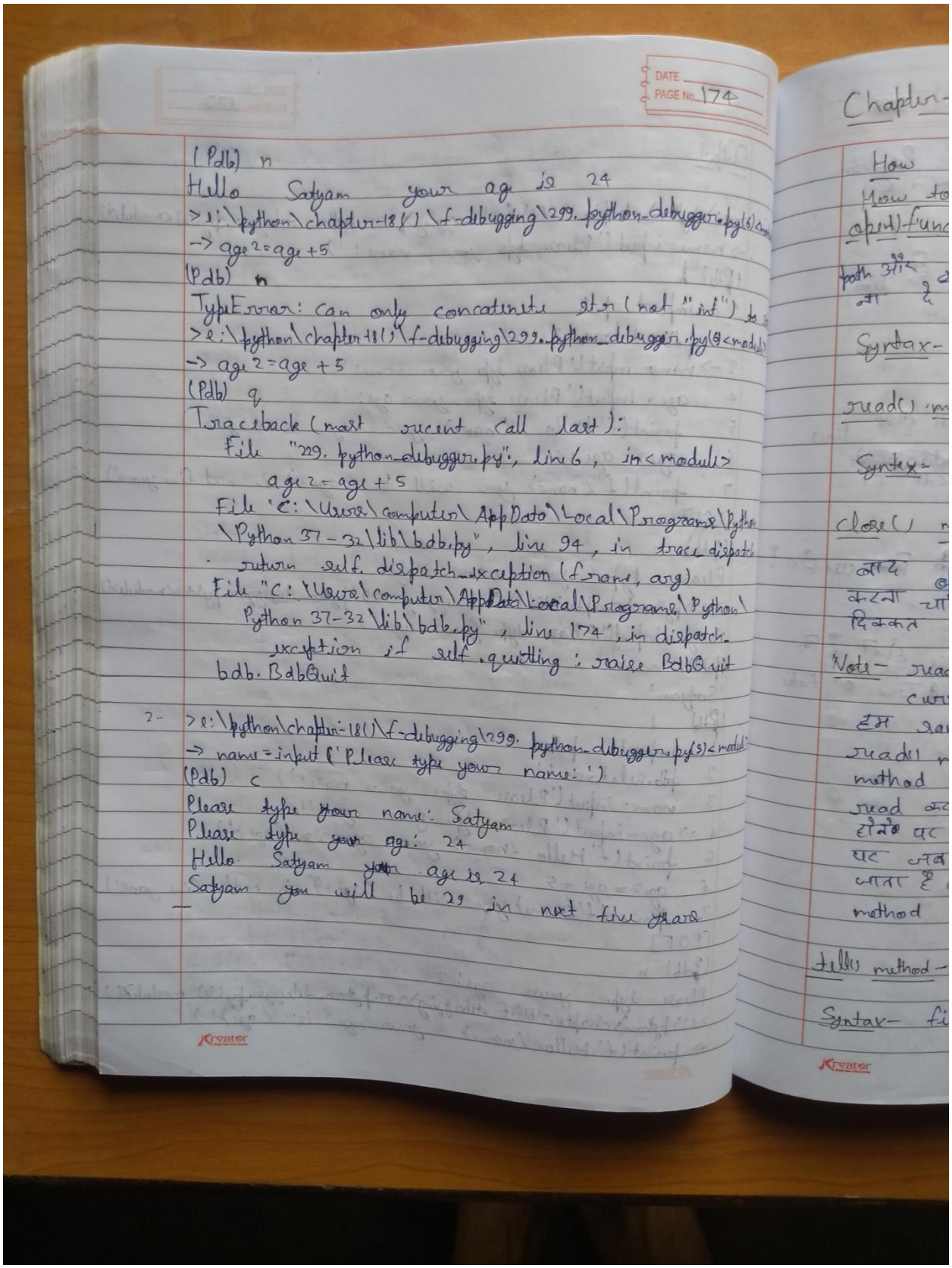
Output -

1.
```
>e:\python\chapter-18()\f
-> name = input('Please ty
(Pdb) l
1  import pdb
2  pdb.set_trace()
3 -> name = input('Please t
4  age = input('Please
5  print(f'Hello <
6  age2 = age + 5
7  print(f'<name> yo
[EOF]
(pdb) n
Please type your n
>e:\python\chapter-18()\f
-> age = input('Please type
(pdb) name
'Satyam'
(Pdb) l
1  import pdb
2  pdb.set_trace()
3  name = input('Please
4 -> age = input('Please t
5  print(f'Hello <name
6  age2 = age + 5
7  print(f'<name> you
[EOF]
(Pdb) n
Please type your age:
>e:\python\chapter-18()\f-debug
-> print(f'Hello <name>
```



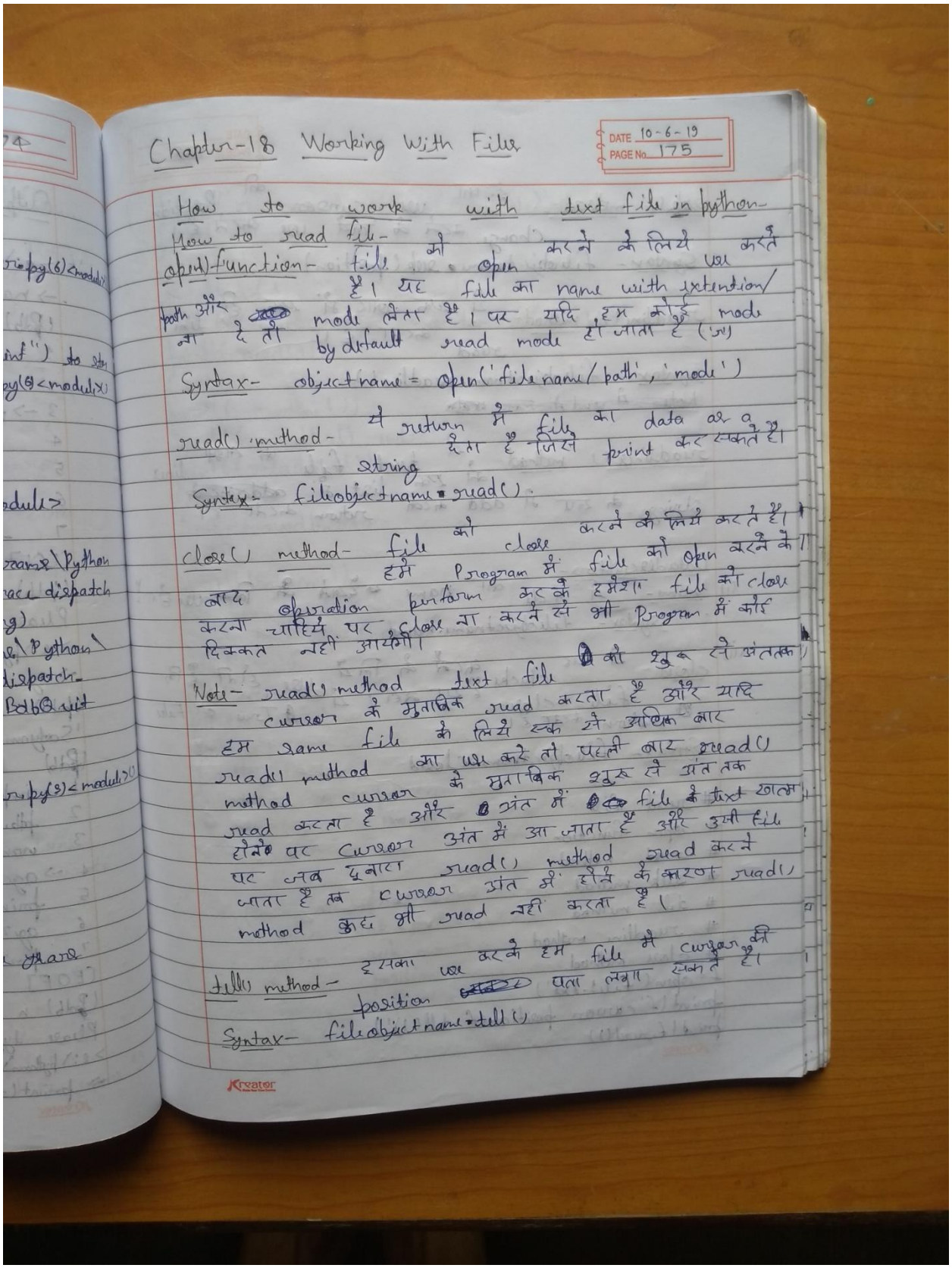
Output -

```
1. >e:\python\chapter-18()\f-debugging\299.python_debugger.py(3)<module>()
-> name=input('Please type your name:')
(Pdb) l
1  import pdb
2  pdb.set_trace()
3 -> name=input('Please type your name:')
4  age=input('Please type your age:')
5  print(f'Hello <name> your age is <age>')
6  age2=age+5
7  print(f'<name> you will be <age2> in next five years')
[EOF]
(pdb) n
Please type your name: Satyam
>e:\python\chapter-18()\f-debugging\299.python_debugger.py(4)<module>()
-> age=input('Please type your age:')
(pdb) name
'Satyam'
(Pdb) l
1  import pdb
2  pdb.set_trace()
3  name=input('Please type your name:')
4 -> age=input('Please type your age:')
5  print(f'Hello <name> your age is <age>')
6  age2=age+5
7  print(f'<name> you will be <age2> in next five years')
[EOF]
(Pdb) n
Please type your age: 24
>e:\python\chapter-18()\f-debugging\299.python_debugger.py(5)<module>()
-> print(f'Hello <name> your age is <age>')
```



(Pdb) n
Hello Satyam your age is 24
> i:\python\chapter-18()\f-debugging\299.python_debugger.py(5)<module>
-> age2 = age + 5
(Pdb) n
TypeError: can only concatenate str (not "int") to str
> i:\python\chapter-18()\f-debugging\299.python_debugger.py(5)<module>
-> age2 = age + 5
(Pdb) q
Traceback (most recent call last):
File "299.python_debugger.py", line 6, in <module>
age2 = age + 5
File "C:\Users\computer\AppData\Local\Programs\Python\Python37-32\lib\bdb.py", line 94, in trace_dispatch
return self.dispatch_exception(frame, arg)
File "C:\Users\computer\AppData\Local\Programs\Python\Python37-32\lib\bdb.py", line 174, in dispatch_exception
if self.quitting: raise BdbQuit
bdb.BdbQuit

2- > i:\python\chapter-18()\f-debugging\299.python_debugger.py(3)<module>
-> name = input("Please type your name: ")
(Pdb) c
Please type your name: Satyam
Please type your age: 24
Hello Satyam your age is 24
Satyam you will be 29 in next five years



Chapter-18 Working With Files

DATE 10-6-19
PAGE No. 175

How to work with text file in python-

How to read file-

open()-function- file को open करने के लिये use करते हैं। यह file का name with extension/path और mode लेता है। पर यदि हम कोई mode न दे तो by default read mode हो जाता है (w)

Syntax- objectname = open('filename/path', 'mode')

read() method- यह return में file का data as a string देता है जिसे print कर सकते हैं।

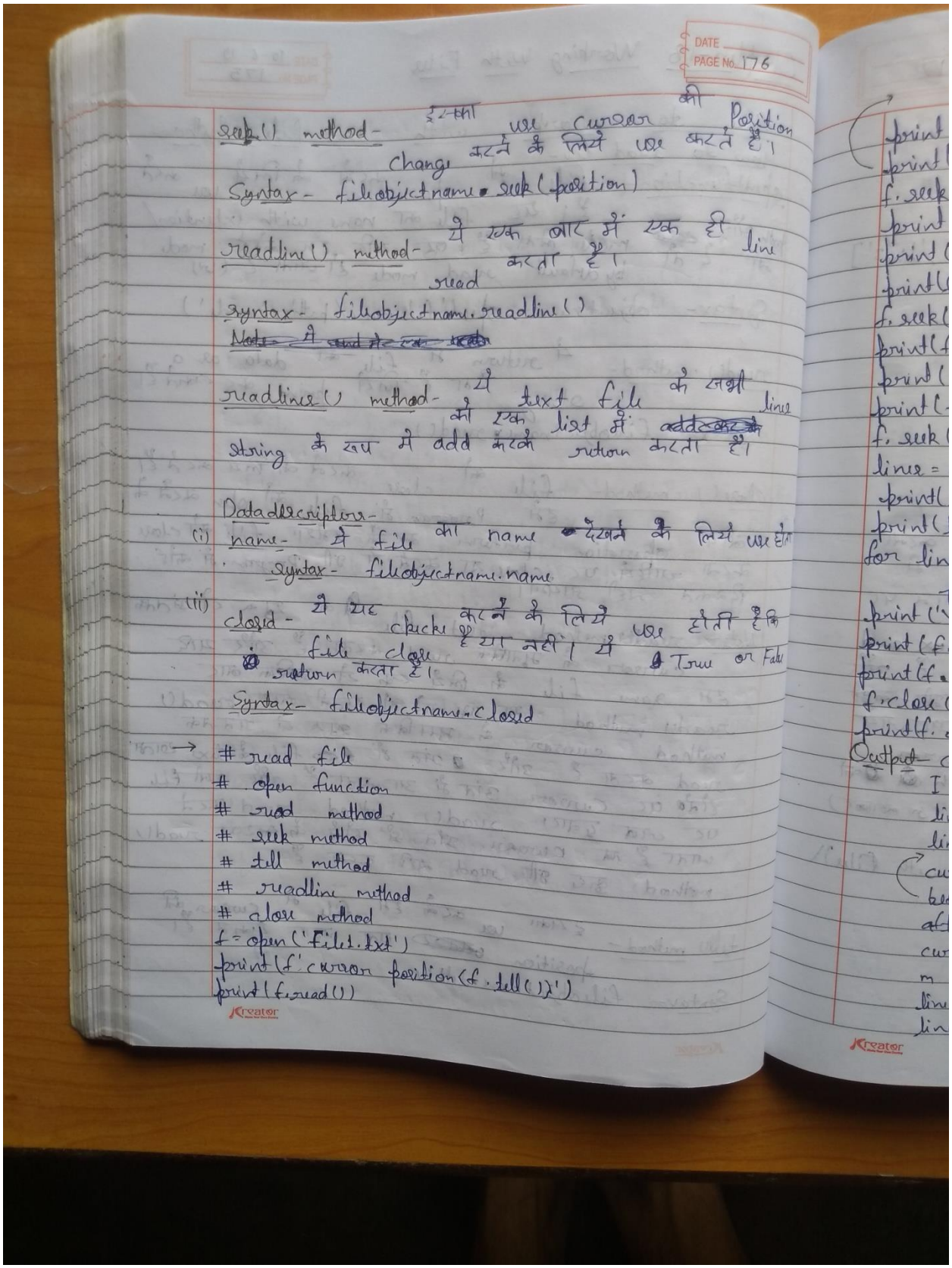
Syntax- fileobjectname.read()

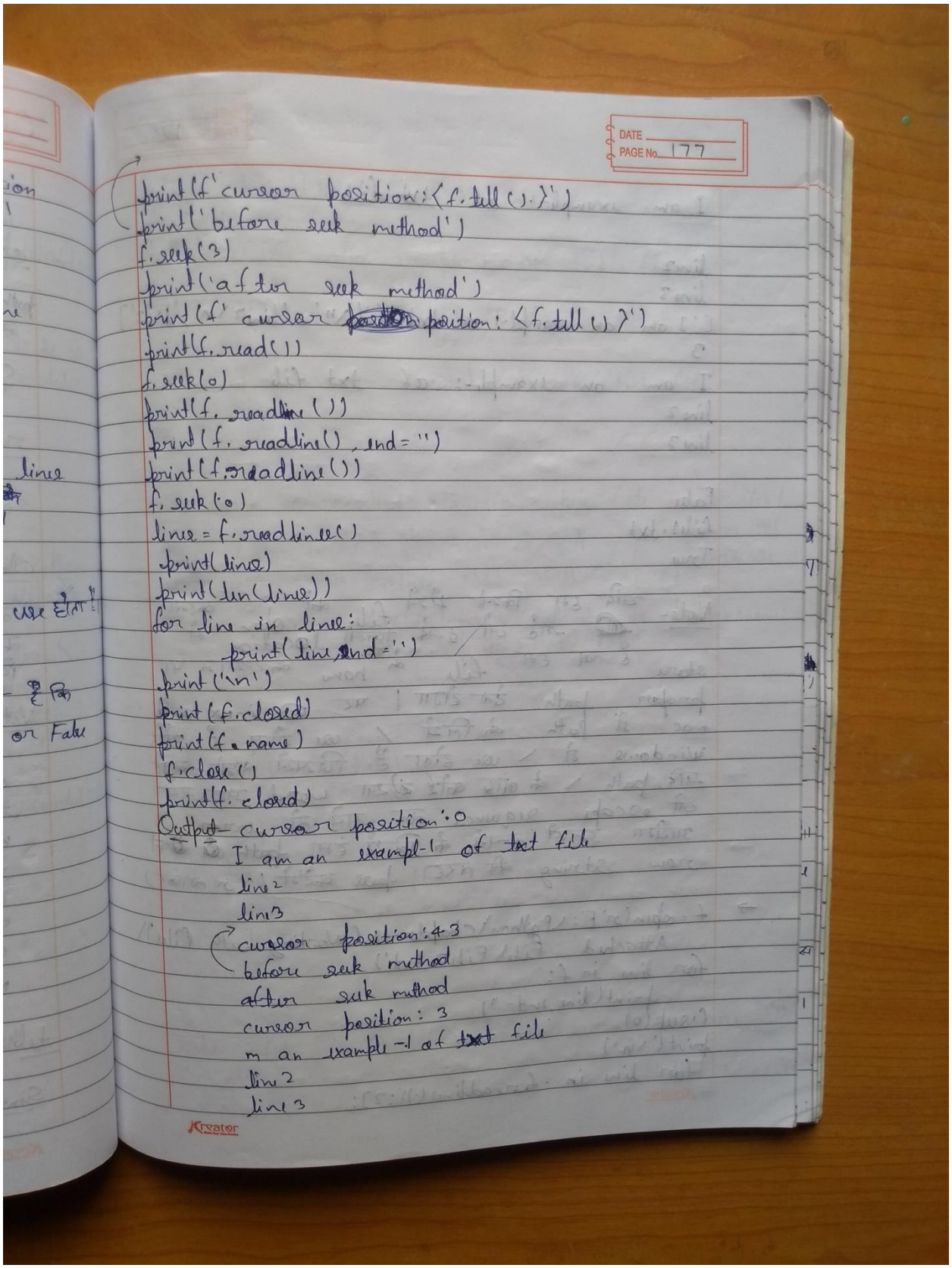
close() method- file को close करने के लिये करते हैं। हमें Program में file को open करने के बाद operation perform करके हमेशा file को close करना चाहिये पर close न करने से भी Program में कोई दिक्कत नहीं आयेगी।

Note- read() method text file को शुरू से अंत तक cursor के मुताबिक read करता है और यदि हम same file के लिये एक से अधिक बार read() method का use करे तो पहली बार read() method cursor के मुताबिक शुरू से अंत तक read करता है और अंत में file के text खत्म होने पर cursor अंत में आ जाता है और उसी file पर जब दुबारा read() method use करने जाता है तब cursor अंत में होने के कारण read() method कुछ भी read नहीं करता है।

tell() method- इसका use करके हम file में cursor की position पता लगा सकते हैं।

Syntax- fileobjectname.tell()

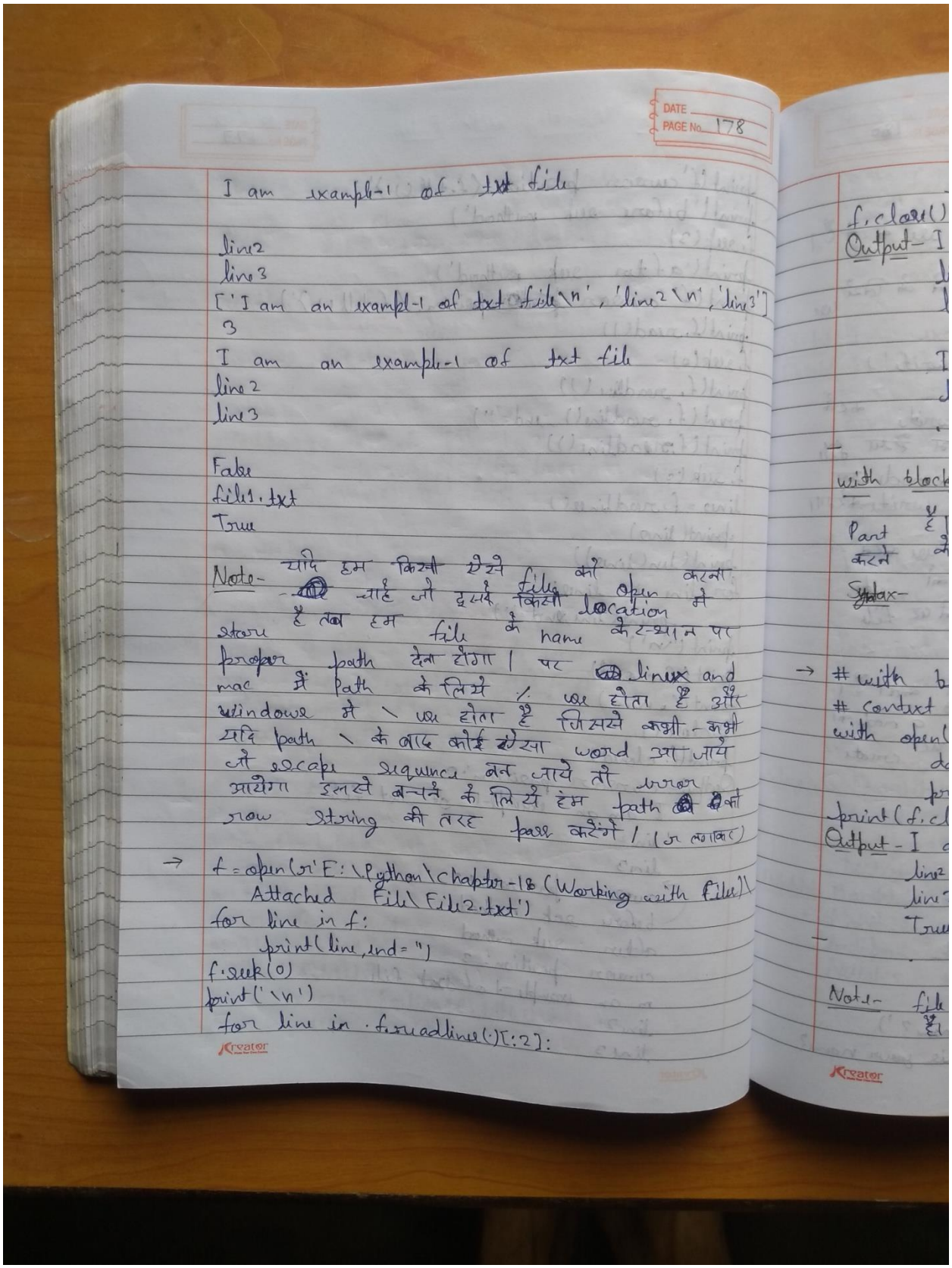

DATE
PAGE No. 176
seek() method- use cursor की Position change करने के लिये use करते हैं।
Syntax- fileobjectname.seek(position)
readline() method- यह एक बार में एक ही line read करता है।
Syntax- fileobjectname.readline()
readlines() method- यह text file के सभी lines को एक list में string के रूप में add करके return करता है।
Dataattributes-
(i) name- यह file का name देखने के लिये use होता
Syntax- fileobjectname.name
(ii) closed- यह check करने के लिये use होती है कि file close है या नहीं। यह True or False return करता है।
Syntax- fileobjectname.closed
# read file
# open function
# read method
# seek method
# tell method
# readline method
# close method
f = open('File1.txt')
print(f'cursor position(f.tell())')
print(f.read())
Output-



```
print(f'cursor position:{f.tell()}')
print('before seek method')
f.seek(3)
print('after seek method')
print(f'cursor position: {f.tell()}')
print(f.read())
f.seek(0)
print(f.readline())
print(f.readline(), end='')
print(f.readline())
f.seek(0)
line = f.readlines()
print(line)
print(len(line))
for line in line:
    print(line, end='')
print('\n')
print(f.closed)
print(f.name)
f.close()
print(f.closed)
```

Output

```
cursor position:0
I am an example-1 of text file
line2
line3
cursor position:43
before seek method
after seek method
cursor position: 3
m an example-1 of text file
line2
line3
```



I am example-1 of txt file

line2
line3
['I am an example-1 of txt file\n', 'line2\n', 'line3']
3
I am an example-1 of txt file
line2
line3

False
file1.txt
True

Note- यदि हम किसी ऐसे file को open करना चाहें जो हमारे किसी location में है तब हम file के name के साथ पर proper path देना होगा। पर linux and mac में path के लिये / use होता है और windows में \ use होता है जिससे कभी-कभी यदि path \ के बाद कोई ऐसा word आ जाये जो escape sequence बन जाये तो error आयेगा इससे बचने के लिये हम path को raw string की तरह pass करेंगे। (r लगाकर)

→ f = open(r'E:\Python\Chapter-18 (Working with File)\Attached File\File2.txt')
for line in f:
    print(line, end='')
f.seek(0)
print('\n')
for line in f.readlines()[:2]:

f.close()
Output- I

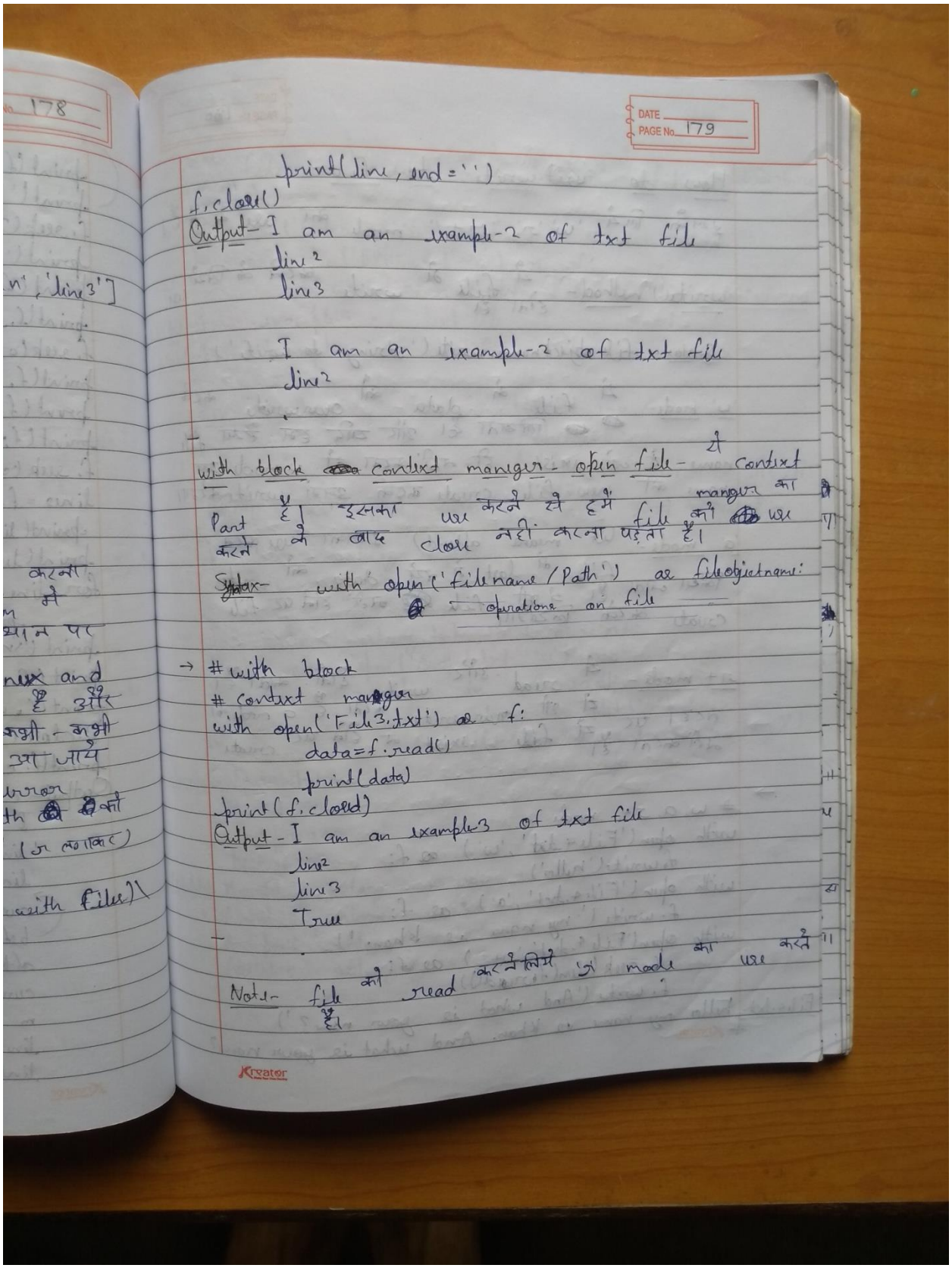
with block

Part
करने

Syntax-

→ # with b
context
with open(
print(f.cl
Output - I
line2
line3
True

Note- file



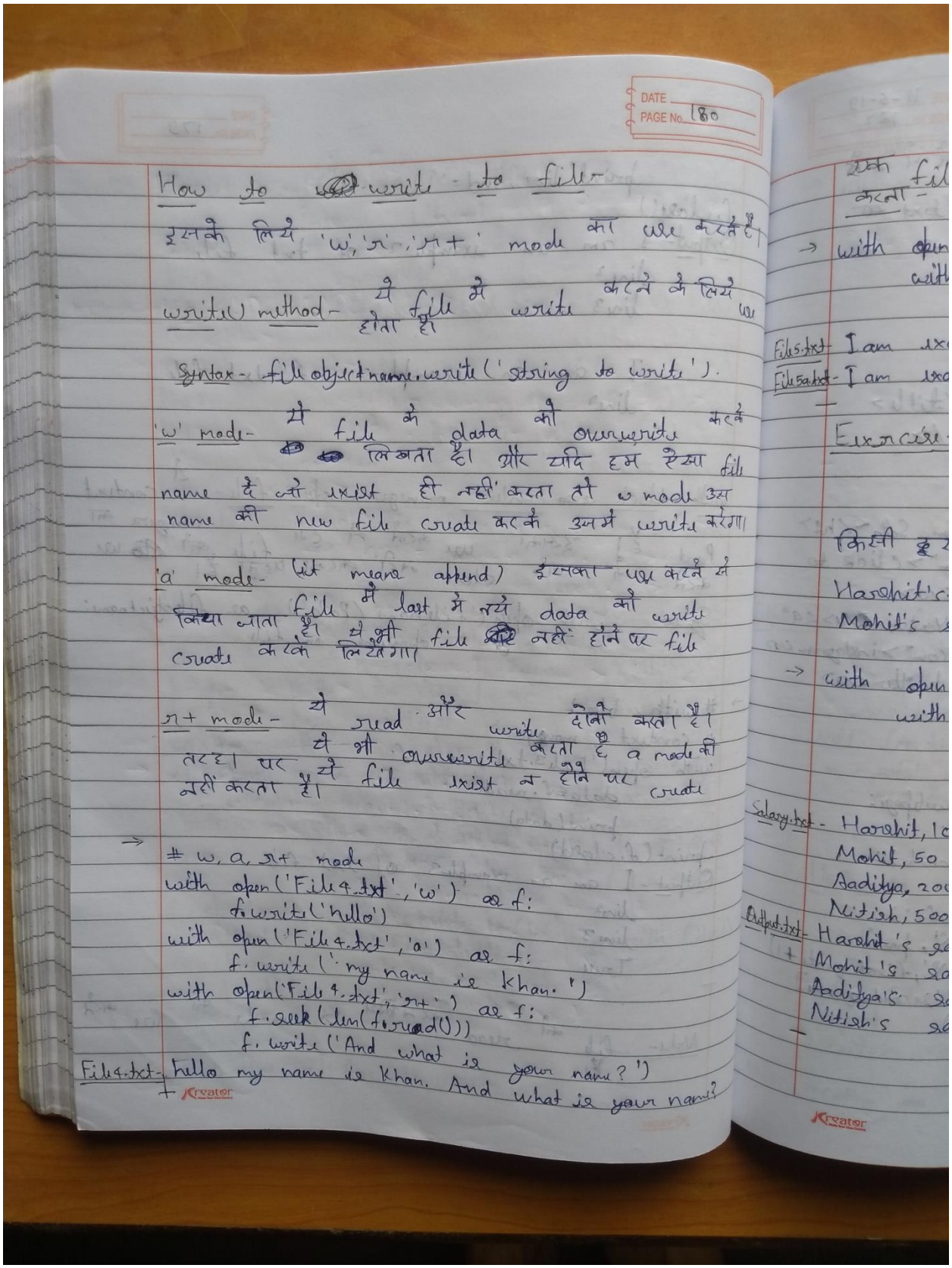
print(line, end='')
f.close()
Output- I am an example-2 of txt file
line2
line3

I am an example-2 of txt file
line2

with block context manager - open file - ये context Part है। इसका use करने से हमें file को use करने के बाद close नहीं करना पड़ता है।
Syntax- with open('filename/Path') as fileobjectname:
operations on file

→ # with block
context manager
with open('File3.txt') as f:
data=f.read()
print(data)
print(f.closed)
Output- I am an example-3 of txt file
line2
line3
True

Note:- file को read करने लिये 'r' mode का use करते हैं।



DATE
PAGE No. 180

How to write to file-

इसके लिये 'w', 'x', 'x+' mode का use करते हैं।

write() method- ये file में write करने के लिये use होता है।

Syntax- file object name.write('string to write').

'w' mode- ये file के data को overwrite करके लिखता है। और यदि हम ऐसा file name दें जो exist ही नहीं करता तो w mode उस name की new file create करके उसमें write करेगा।

'a' mode- (it means append) इसका use करने से file में last में नये data को write किया जाता है। ये भी file नहीं होने पर file create करके लिखेगा।

x+ mode- ये read और write दोनों करता है। ये भी overwrite करता है a mode की तरह। पर ये file exist न होने पर create नहीं करता है।

→
```
# w, a, x+ mode
with open('File4.txt', 'w') as f:
    f.write('hello')
with open('File4.txt', 'a') as f:
    f.write('my name is Khan.')
with open('File4.txt', 'x+') as f:
    f.seek(len(f.read()))
    f.write('And what is your name?')
```
File4.txt- hello my name is Khan. And what is your name?

→ with open

File5.txt- I am

File6.txt- I am

Exercise

किसी

Harshit's

Mohit's

→ with open

Salary.txt- Harshit, 1

Mohit, 50

Aaditya, 20

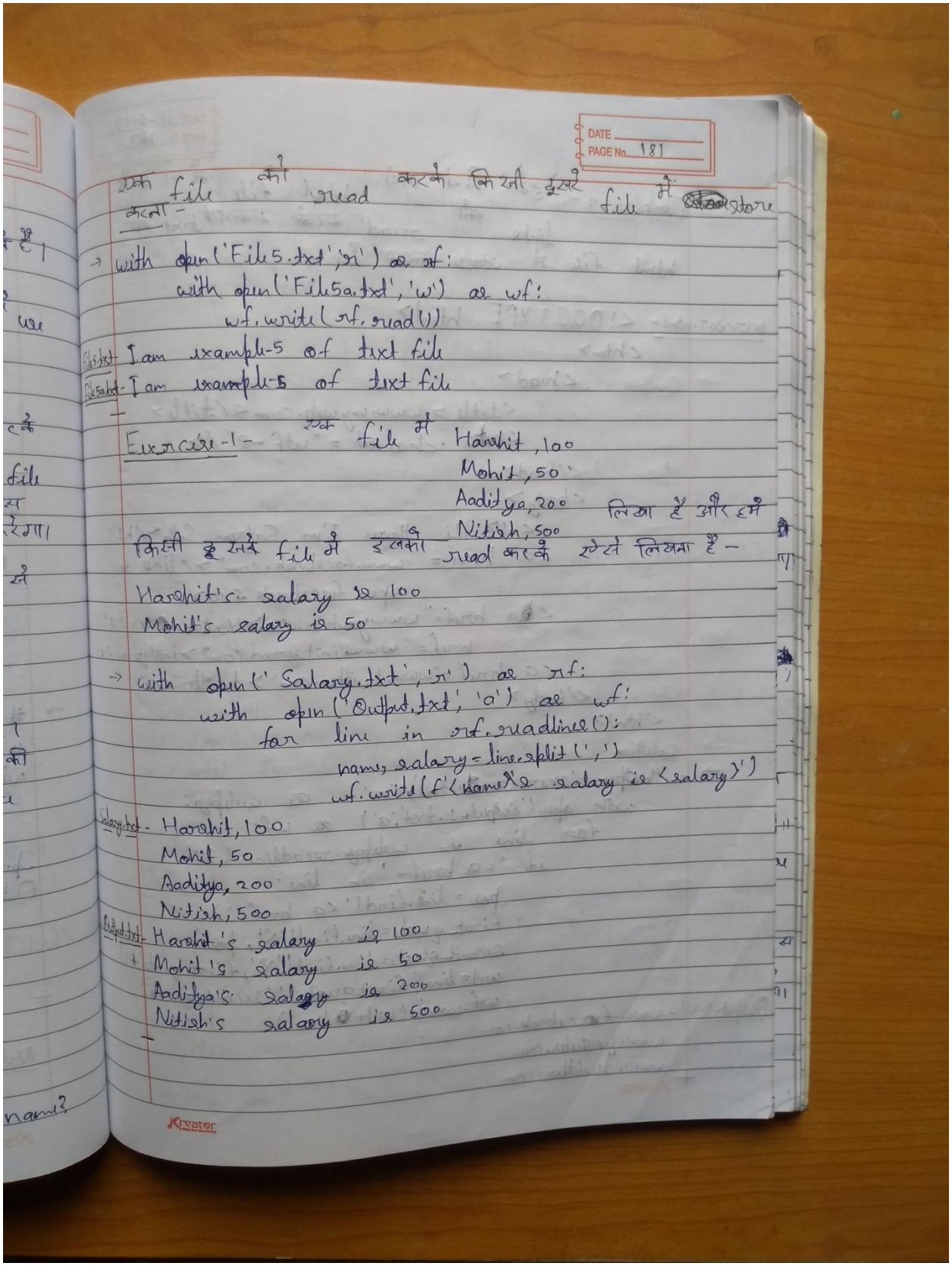
Nitish, 50

Output.txt- Harshit's

Mohit's

Aaditya's

Nitish's



DATE
PAGE No. 181

एक file को read करके किसी दूसरे file में store करना -

→ with open('File5.txt','r') as rf:
    with open('File5a.txt','w') as wf:
        wf.write(rf.read())

File5.txt - I am example-5 of text file

File5a.txt - I am example-5 of text file

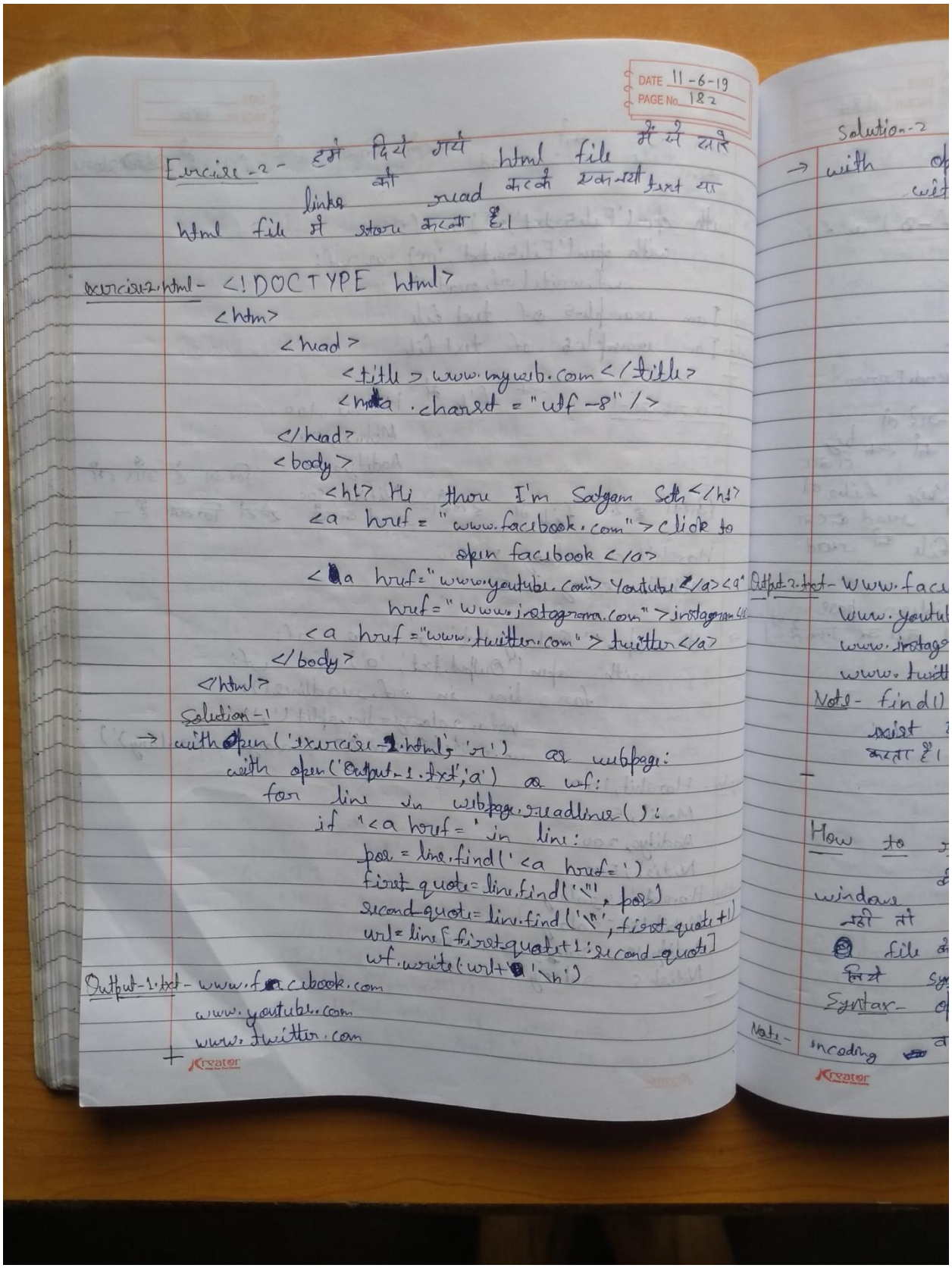
Exercise-1 - एक file में Harshit, 100
Mohit, 50
Aaditya, 200
Nitish, 500
लिखा है और हमें किसी दूसरे file में इनको read करके ऐसे लिखना है -

Harshit's salary is 100
Mohit's salary is 50

→ with open('Salary.txt','r') as rf:
    with open('Output.txt','a') as wf:
        for line in rf.readlines():
            name, salary = line.split(',')
            wf.write(f'{name}'s salary is {salary}')

Salary.txt - Harshit, 100
Mohit, 50
Aaditya, 200
Nitish, 500

Output.txt - Harshit's salary is 100
Mohit's salary is 50
Aaditya's salary is 200
Nitish's salary is 500



DATE 11-6-19
PAGE No. 182

Exercise-2 - हमें दिये गये links की html file में से सारे read करके एक नयी text या html file में store करना है।

exercise2.html -
```
<!DOCTYPE html>
<html>
    <head>
        <title> www.myweb.com </title>
        <meta charset = "utf-8" />
    </head>
    <body>
        <h1> Hi there I'm Satyam Seth </h1>
        <a href = "www.facebook.com"> click to
                open facebook </a>
        <a href = "www.youtube.com"> Youtube </a> <a
            href = "www.instagram.com"> instagram </a>
        <a href = "www.twitter.com"> twitter </a>
    </body>
</html>
```

Solution-1

→
```
with open('exercise-2.html', 'r') as webpage:
    with open('Output-1.txt', 'a') as wf:
        for line in webpage.readlines():
            if '<a href=' in line:
                pos = line.find('<a href=')
                first_quote = line.find('"', pos)
                second_quote = line.find('"', first_quote+1)
                url = line[first_quote+1:second_quote]
                wf.write(url+'\n')
```

Output-1.txt -
```
www.facebook.com
www.youtube.com
www.twitter.com
```

Solution-2

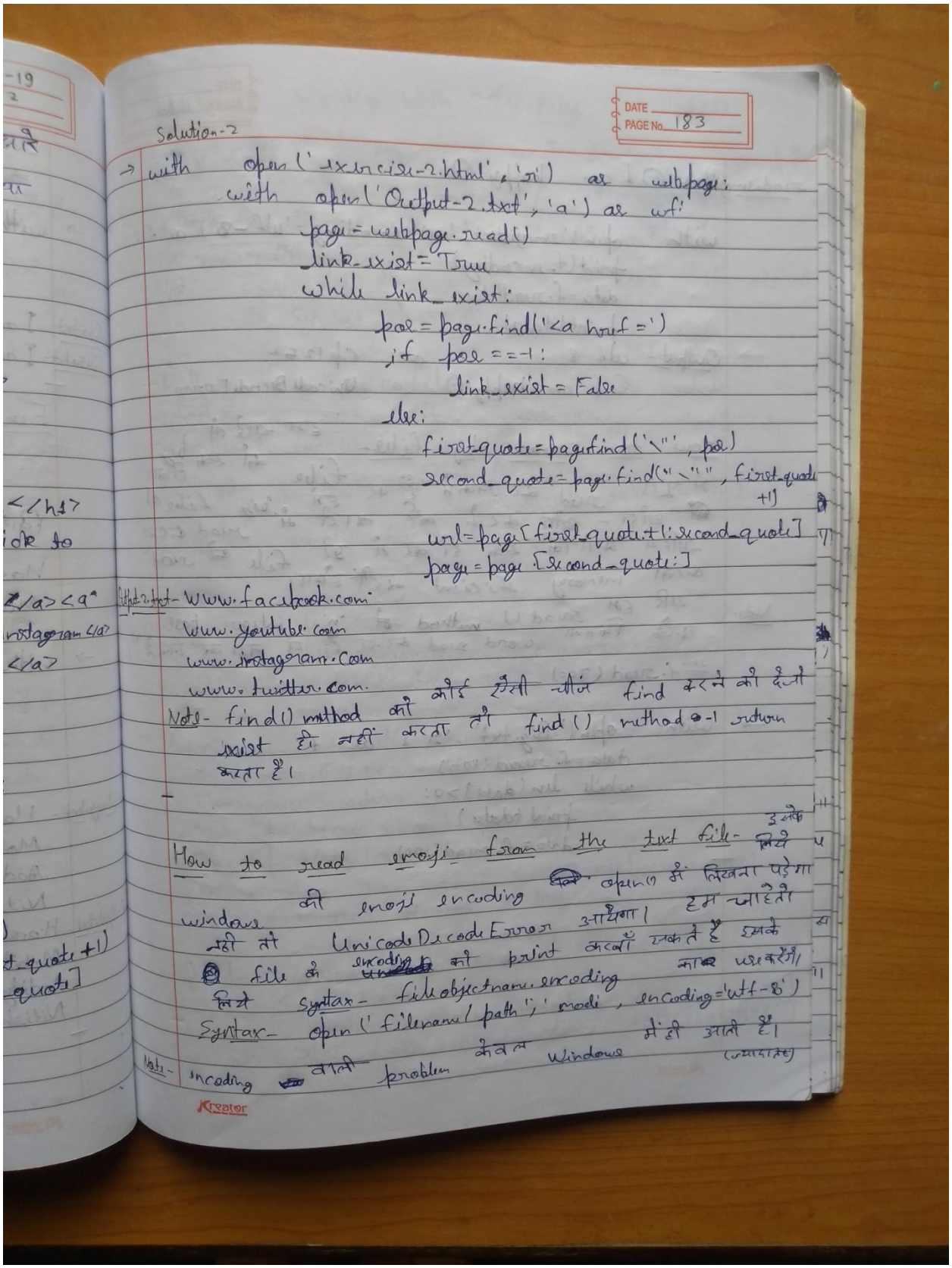
→ with op
wit

Output-2.txt - www.facu
www.youtu
www.instagr
www.twitt

Note - find()
exist
करता है।

How to
windows
नहीं तो
file
लिखने
Syntax - o

Note - encoding



Solution-2

```
with open('exercise-2.html','r') as webpage:
    with open('Output-2.txt','a') as wf:
        page = webpage.read()
        link_exist = True
        while link_exist:
            pos = page.find('<a href =')
            if pos == -1:
                link_exist = False
            else:
                first_quote = page.find('"', pos)
                second_quote = page.find('"', first_quote+1)
                url = page[first_quote+1:second_quote]
                page = page[second_quote:]
```

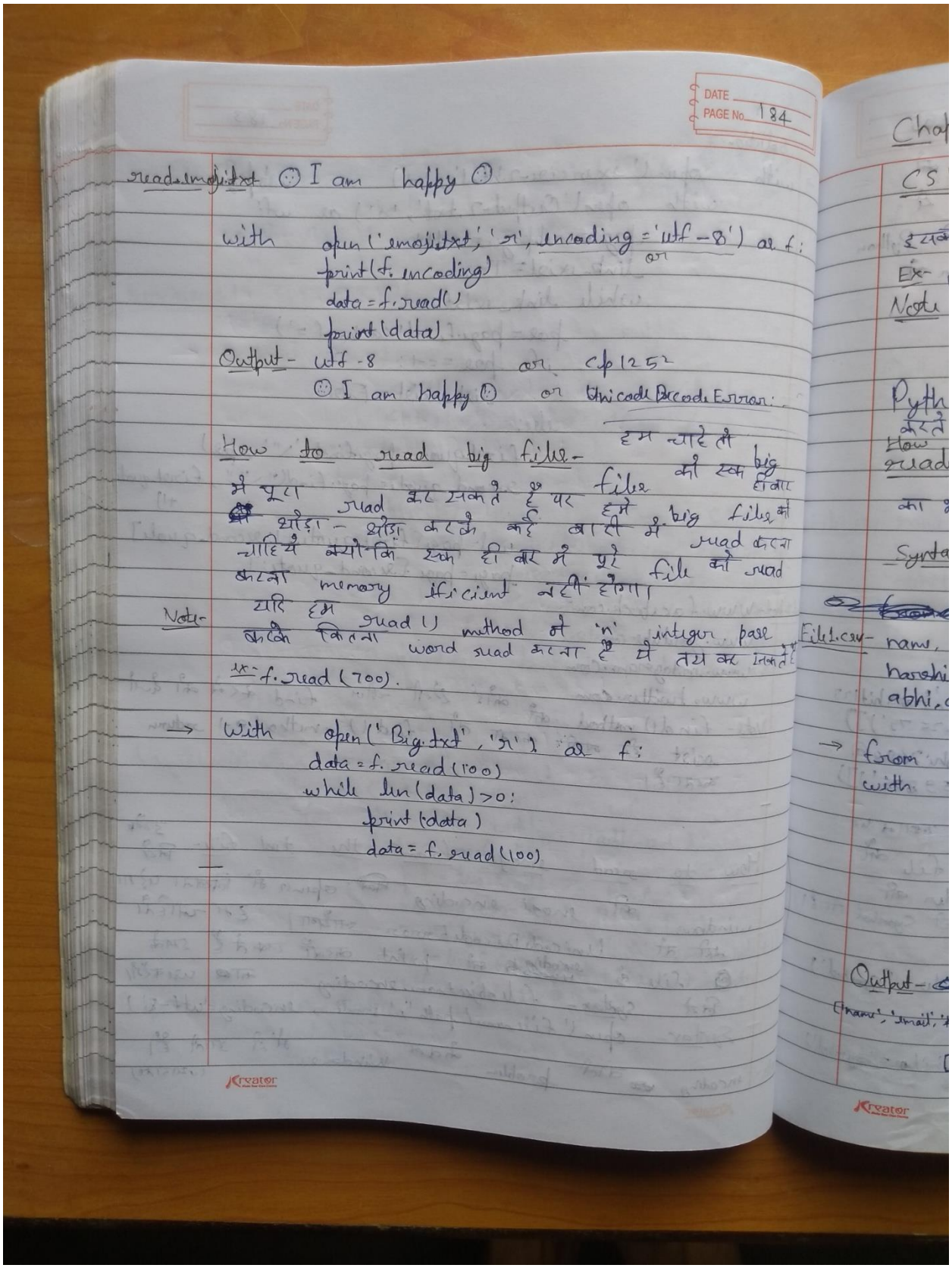
Output-2.txt - www.facebook.com
www.youtube.com
www.instagram.com
www.twitter.com.

Note- find() method को कोई ऐसी चीज find करने को देते exist ही नहीं करता तो find() method -1 return करता है।

How to read emoji from the text file- इसके लिये windows की emoji encoding चाहिये open() में लिखना पड़ेगा नहीं तो UnicodeDecodeError आयेगा। हम चाहें तो file के encoding को print करवा सकते हैं इसके लिये syntax- fileobjectname.encoding

Syntax- open('filename/path','mode',encoding='utf-8')

Note- encoding वाली problem केवल Windows में ही आती है। (ज्यादातर)



DATE
PAGE No. 184

read_emoji.txt ☺ I am happy ☺

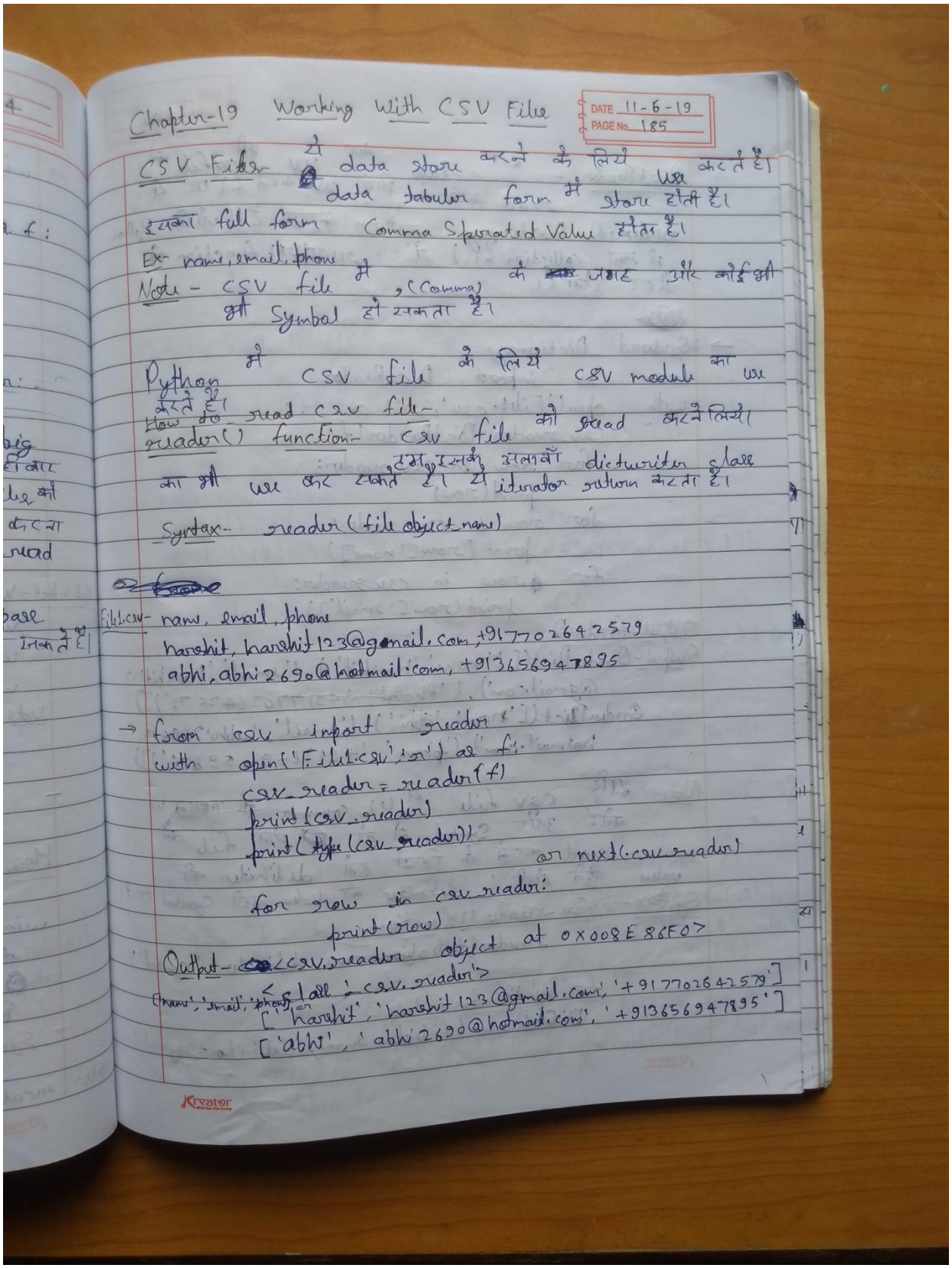
```
with open('emoji.txt', 'r', encoding='utf-8') as f:
    print(f.encoding)
    data = f.read()
    print(data)
```

Output - utf-8 or cp1252
☺ I am happy ☺ or UnicodeDecodeError:

How to read big file - हम चाहे तो किसी भी big file को एक बार में पूरा read कर सकते है पर हमें बड़ी - बड़ी file को छोटे - छोटे करके कई बार में read करना चाहिये क्योंकि एक ही बार में पूरे file को read करना memory efficient नहीं होगा।

Note - यदि हम read() method में 'n' integer pass करके कितना word read करना है ये तय कर सकते है - f.read(700).

→
```
with open('Big.txt', 'r') as f:
    data = f.read(100)
    while len(data)>0:
        print(data)
        data = f.read(100)
```



Chapter-19 Working With CSV File

DATE 11-6-19
PAGE No. 185

CSV File - data store करने के लिये use करते है।
data tabular form में store होती है।
इसका full form Comma Separated Value होता है।
Ex- name, email, phone
Note - CSV file में , (Comma) की जगह और कोई भी Symbol हो सकता है।

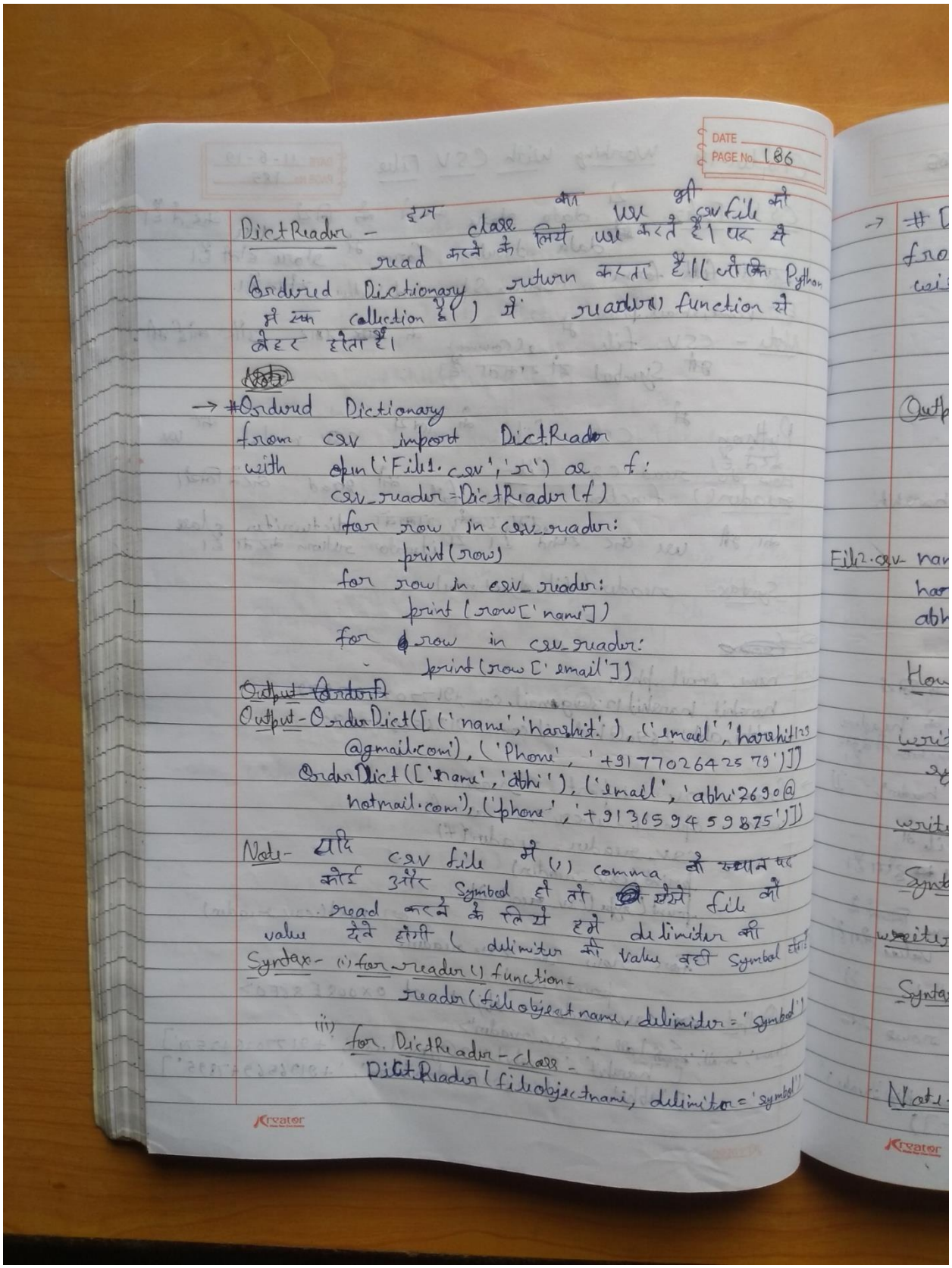
Python में CSV file के लिये csv module का use करते है।
How to read csv file-
reader() function- csv file को read करने लिये।
हम इसके अलावा dictwriter class का भी use कर सकते है। ये iterator return करता है।

Syntax- reader(file object name)

file1.csv- name, email, phone
harshit, harshit123@gmail.com, +917702642579
abhi, abhi2690@hotmail.com, +9136569478 95

→ from csv import reader
with open('File1.csv','r') as f:
    csv_reader = reader(f)
    print(csv_reader)
    print(type(csv_reader))
    or next(csv_reader)
    for row in csv_reader:
        print(row)

Output- <csv.reader object at 0x008E86F0>
['name', 'email', 'phone'] or all : csv.reader>
['harshit', 'harshit123@gmail.com', '+917702642579']
['abhi', 'abhi2690@hotmail.com', '+9136569478 95']



DictReader - इस class का use भी csv file को read करने के लिये use करते हैं। पर ये Ordered Dictionary return करता है। (as Python में इस collection को) में readers() function से बेहतर होता है।

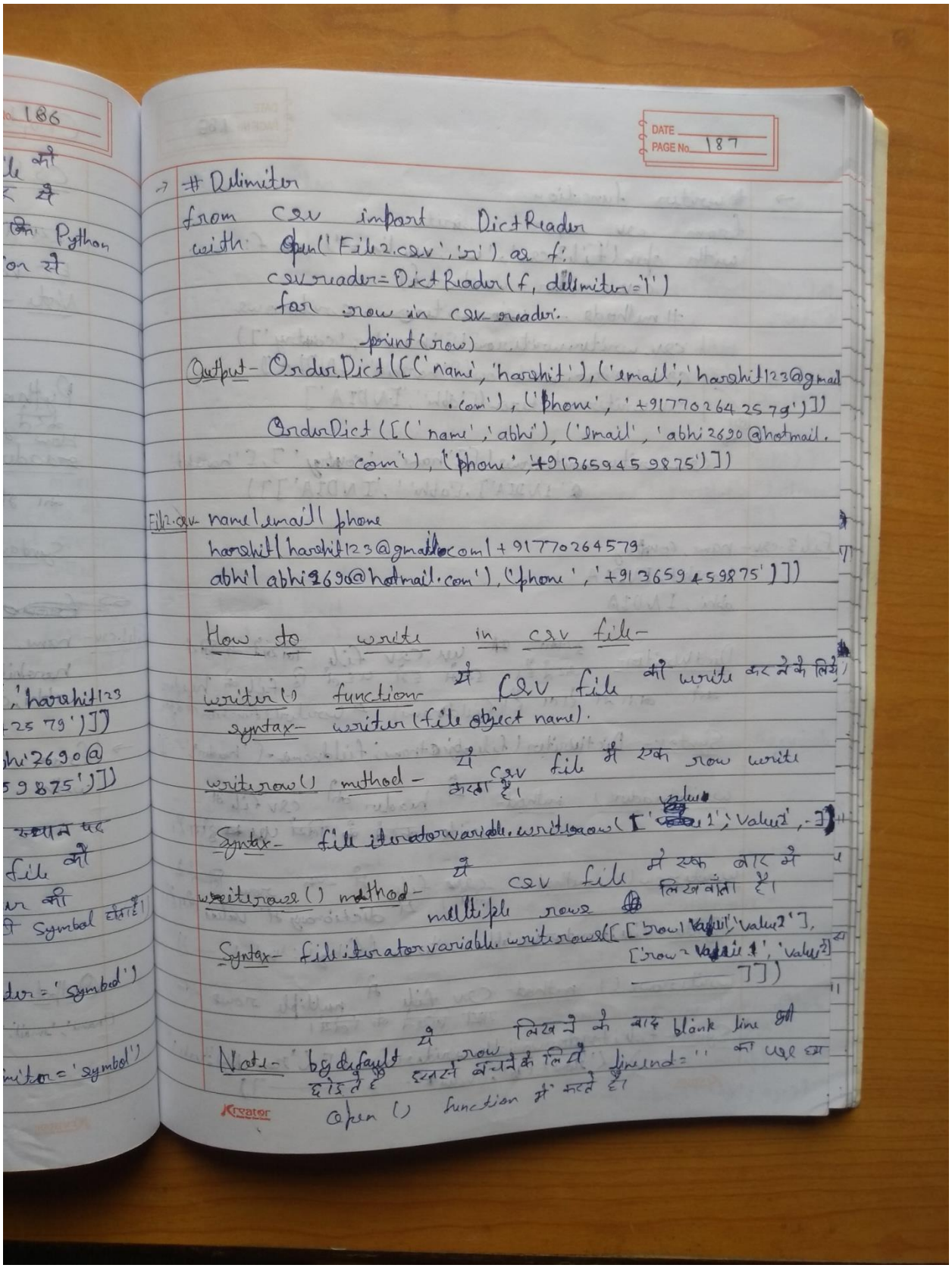
→ #Ordered Dictionary
from csv import DictReader
with open('File1.csv','r') as f:
    csv_reader = DictReader(f)
    for row in csv_reader:
        print(row)
    for row in csv_reader:
        print(row['name'])
    for row in csv_reader:
        print(row['email'])

Output - OrderDict([('name','harshit'),('email','harshit129@gmail.com'),('Phone','+917702642579')])
OrderDict([('name','abhi'),('email','abhi2690@hotmail.com'),('phone','+9136594598 75')])

Note- यदि csv file में (,) comma की जगह पर कोई और Symbol है तो ऐसे file को read करने के लिये हमें delimiter की value देनी होगी (delimiter की value वही Symbol होगा

Syntax- (i) for reader() function-
reader(file object name, delimiter = 'symbol')

(ii) for DictReader - class -
DictReader(file object name, delimiter = 'symbol')



Delimiter

```
from csv import DictReader
with open('File2.csv','r') as f:
    csv_reader = DictReader(f, delimiter='|')
    for row in csv_reader:
        print(row)
```

Output - OrderDict([('name','harshit'),('email','harshit123@gmail.com'),('phone','+917702642579')])
OrderDict([('name','abhi'),('email','abhi2690@hotmail.com'),('phone','+9136594 59875')])

File2.csv - name|email|phone
harshit|harshit123@gmail.com|+917702645 79
abhi|abhi2690@hotmail.com'),('phone','+9136594 59875')])

How to write in csv file -

writer() function से csv file को write करने के लिये।

Syntax - writer(file object name).

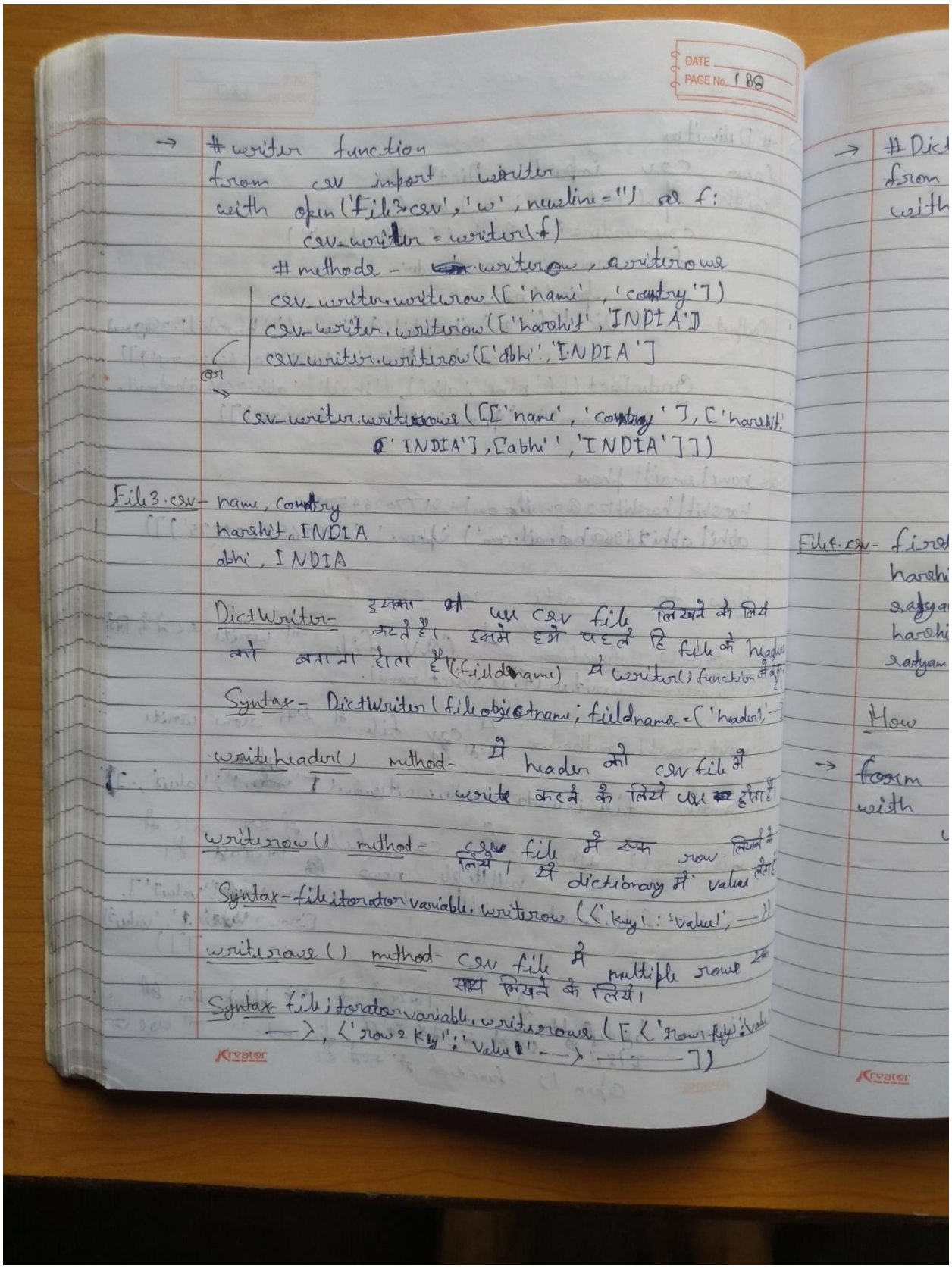
writerow() method - से csv file में एक row write करता है।

Syntax - file iterator variable.writerow(['value1','value2',...])

writerows() method - से csv file में एक बार में multiple rows लिख सकते हैं।

Syntax - file iterator variable.writerows([['row1 value1','value2'], ['row2 value1','value2']])

Note - by default से row लिखने के बाद blank line आती है इसे avoid करने के लिये lineend='' का use open() function में करते हैं।



# writer function
from csv import writer
with open('file3.csv', 'w', newline='') as f:
    csv_writer = writer(f)
    # methods - writerow, writerows
    csv_writer.writerow(['name', 'country'])
    csv_writer.writerow(['harshit', 'INDIA'])
    csv_writer.writerow(['abhi', 'INDIA'])
or
csv_writer.writerows([['name', 'country'], ['harshit', 'INDIA'], ['abhi', 'INDIA']])

file3.csv - name, country
harshit, INDIA
abhi, INDIA

DictWriter - इसका use csv file लिखने के लिये करते है। इसमे हमे पहले ही file के header को बताना होता है (fieldname) से writer() function से अलग।

Syntax - DictWriter(file object name, fieldnames=['header1', ...])

writeheader() method - से header को csv file में write करने के लिये use होता है।

writerow() method - csv file में एक row लिखने के लिये, से dictionary में value लेते है।

Syntax - file iterator variable.writerow({'key1': 'value1', ...})

writerows() method - csv file में multiple rows एक साथ लिखने के लिये।

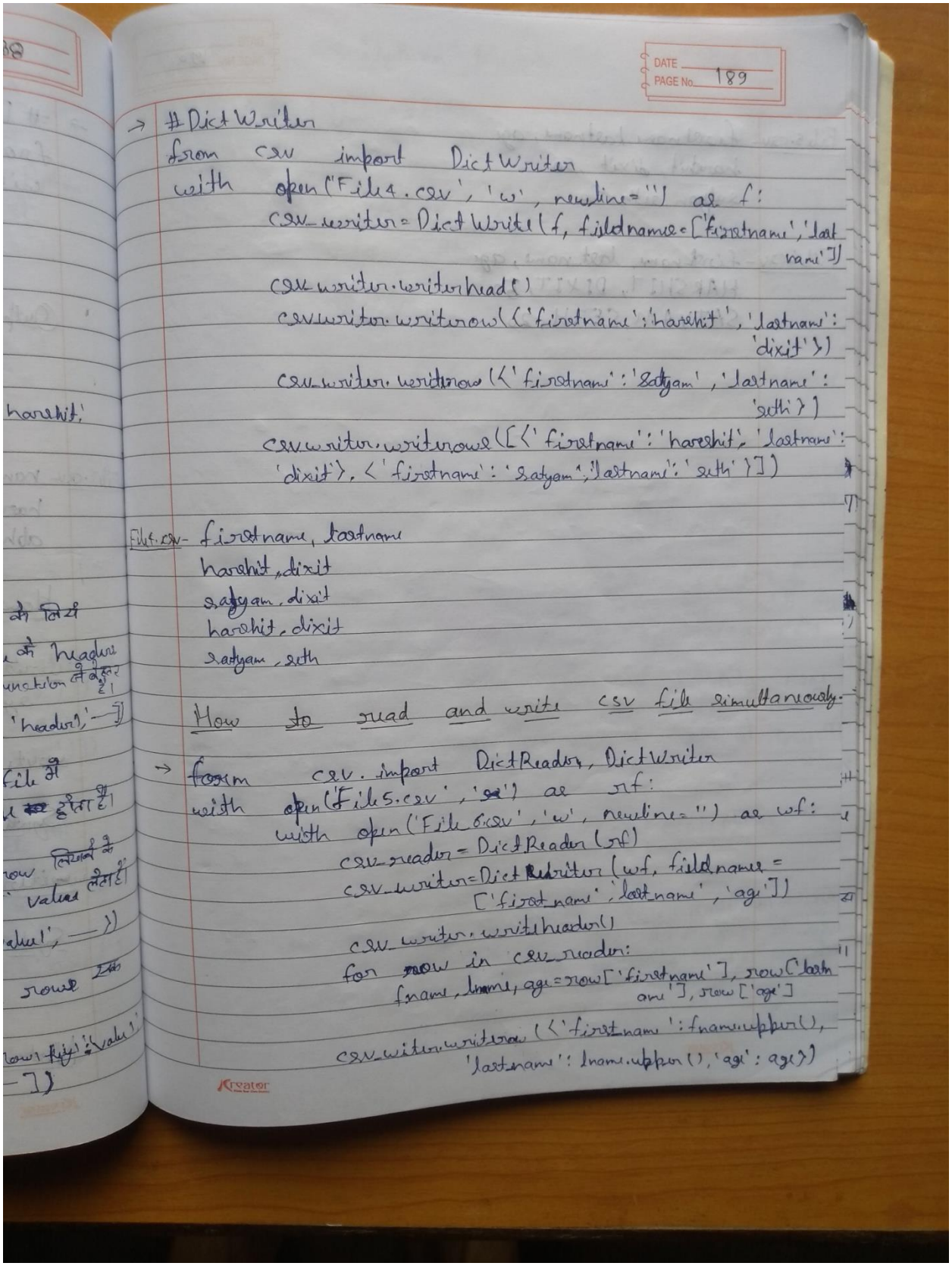
Syntax - file iterator variable.writerows([{'row1 key1': 'value1', ...}, {'row2 key1': 'value1', ...}, ...])

Dict
from
with

file4.csv - first
harshi
satya
harshi
satyam

How

from
with



→ # DictWriter

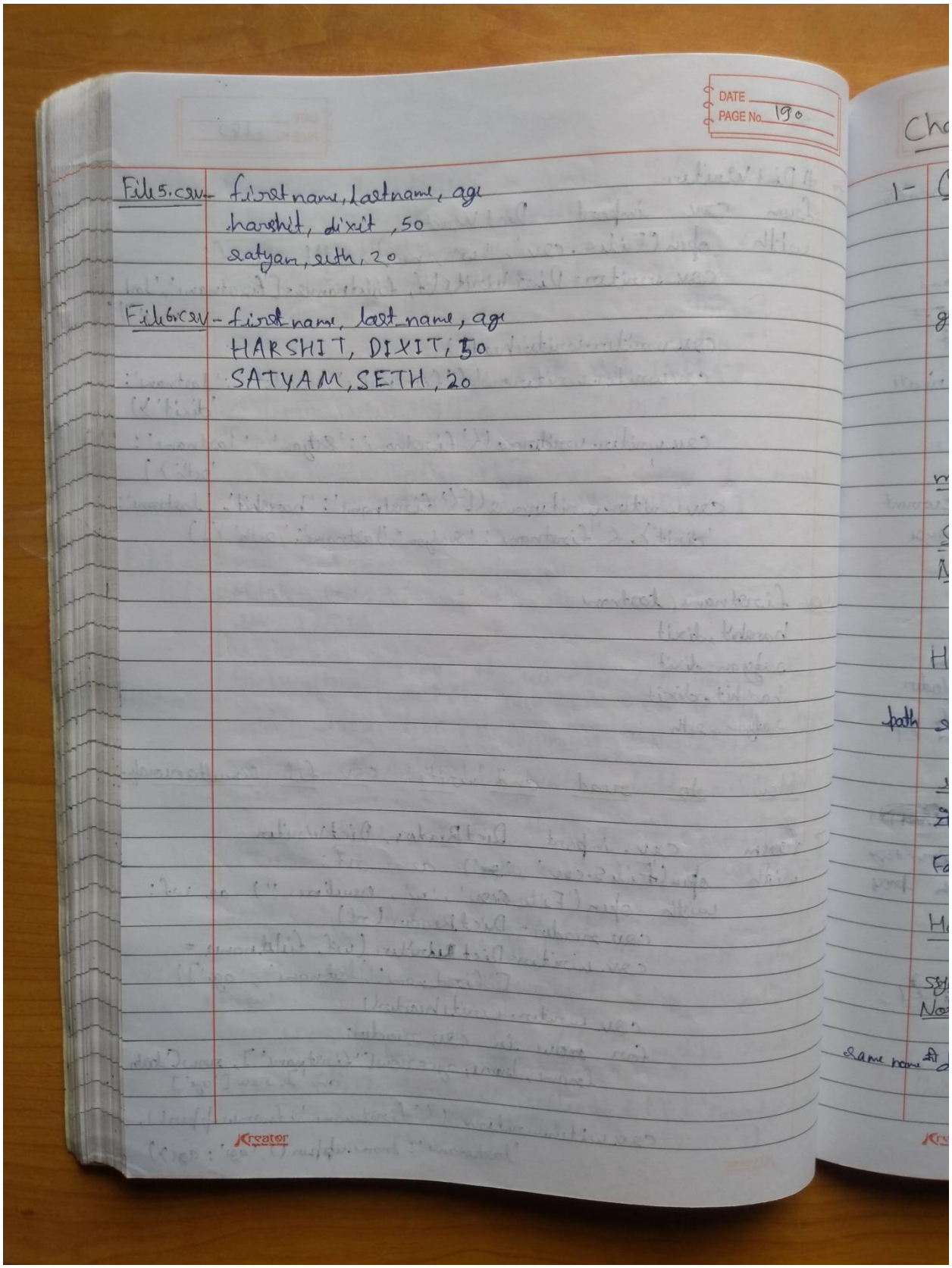
```
from csv import DictWriter
with open('File4.csv', 'w', newline='') as f:
    csv_writer = DictWriter(f, fieldnames=['firstname', 'lastname'])

    csv_writer.writeheader()
    csv_writer.writerow({'firstname': 'harshit', 'lastname': 'dixit'})
    csv_writer.writerow({'firstname': 'satyam', 'lastname': 'seth'})
    csv_writer.writerows([{'firstname': 'harshit', 'lastname': 'dixit'}, {'firstname': 'satyam', 'lastname': 'seth'}])
```

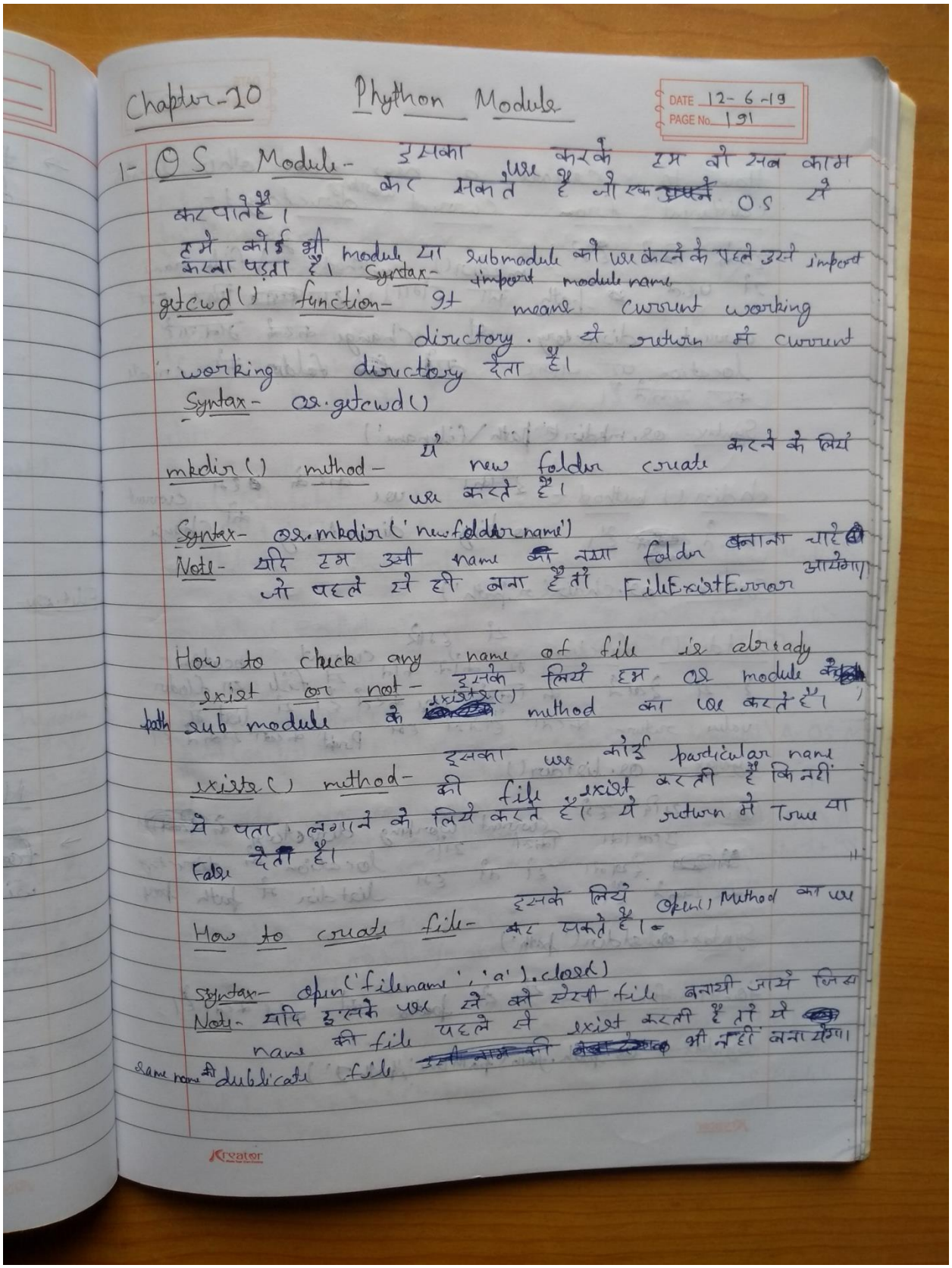
File1.csv - firstname, lastname
harshit, dixit
satyam, dixit
harshit, dixit
satyam, seth

How to read and write csv file simultaneously:

→

```
from csv import DictReader, DictWriter
with open('File5.csv', 'r') as rf:
    with open('File6.csv', 'w', newline='') as wf:
        csv_reader = DictReader(rf)
        csv_writer = DictWriter(wf, fieldnames=['first_name', 'last_name', 'age'])
        csv_writer.writeheader()
        for row in csv_reader:
            fname, lname, age = row['firstname'], row['lastname'], row['age']
            csv_writer.writerow({'firstname': fname.upper(), 'lastname': lname.upper(), 'age': age})
```


DATE
PAGE No. 190
File5.csv - first name, Last name, age
harshit, dixit, 50
satyam, seth, 20
File6.csv - first name, last name, age
HARSHIT, DIXIT, 50
SATYAM, SETH, 20
Kreator



Chapter-10 Python Module

DATE 12-6-19
PAGE No. 191

1- OS Module- इसका use करके हम वो सब काम कर सकते है जो एक user OS से कर पाते है।

हमे कोई भी module या submodule को use करने के पहले उसे import करना पड़ता है। Syntax- import module name

getcwd() function- इसे means current working directory. ये return में current working directory देता है।

Syntax- os.getcwd()

mkdir() method- ये new folder create करने के लिये use करते है।

Syntax- os.mkdir('newfolder_name')

Note- यदि हम उसी name का नया folder बनाना चाहेंगे जो पहले से ही बना है तो FileExistError आयेगा।

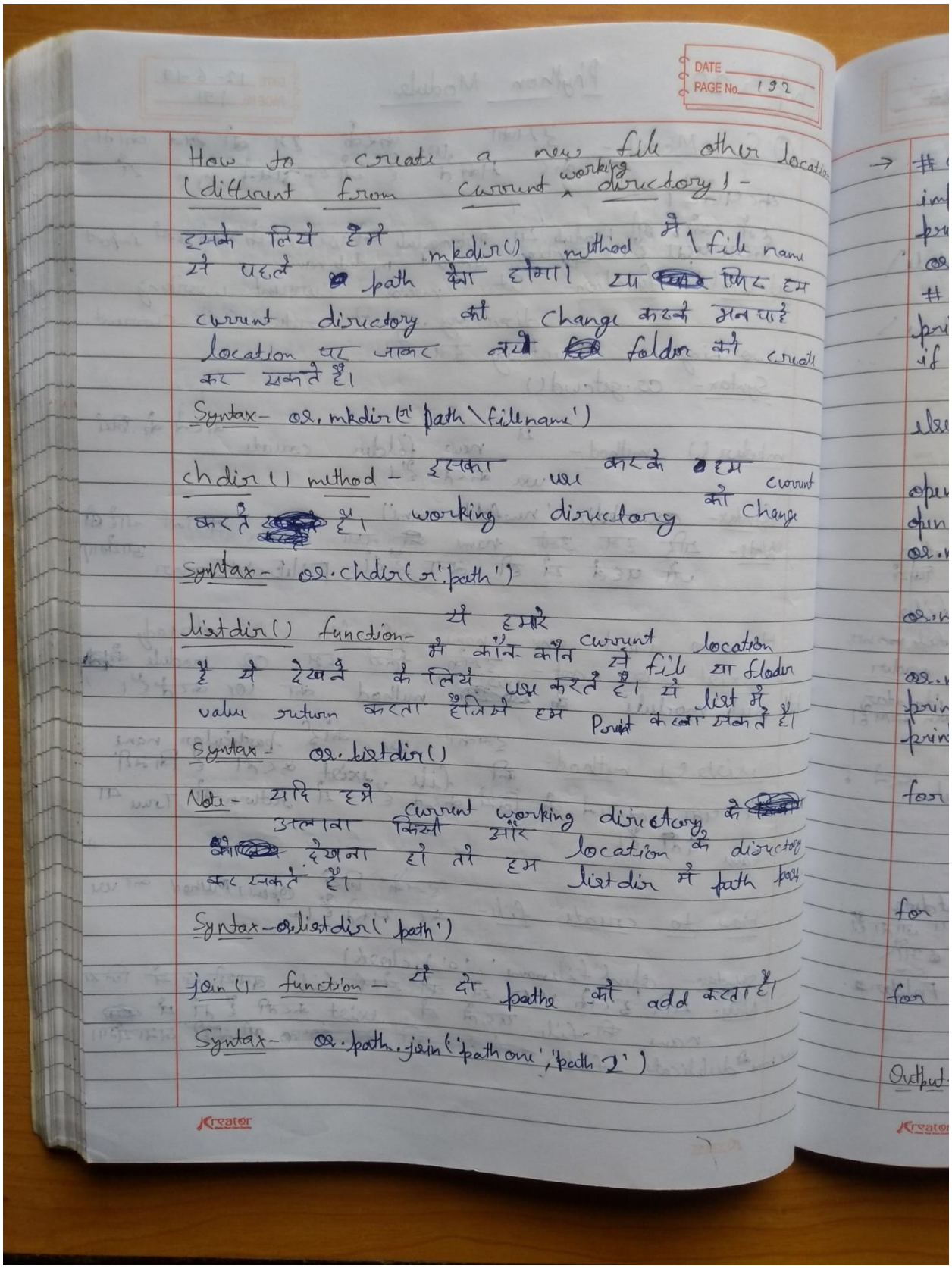
How to check any name of file is already exist or not- इसके लिये हम os module के path sub module के exists() method का use करते है।

exists() method- इसका use कोई particular name की file exist करती है कि नहीं ये पता लगाने के लिये करते है। ये return में True या False देता है।

How to create file- इसके लिये open() Method का use कर सकते है।-

Syntax- open('filename','a').close()

Note- यदि इसके use से वो ऐसी file बनायी जाये जिस name की file पहले से exist करती है तो ये same name की dublicate file भी नहीं बना पायेगा।



How to create a new file other location (different from current working directory) -

इसके लिये हमें mkdir() method में file name से पहले path देना होगा। या फिर हम current directory को change करके अन्य location पर जाकर नये folder को create कर सकते हैं।

Syntax- os.mkdir('path\filename')

chdir() method - इसका use करके हम current working directory को change करते हैं।

Syntax- os.chdir('path')

listdir() function- यह हमारे current location में कौन-कौन से file या folder हैं यह देखने के लिये use करते हैं। यह list में value return करता है जिसे हम Print करा सकते हैं

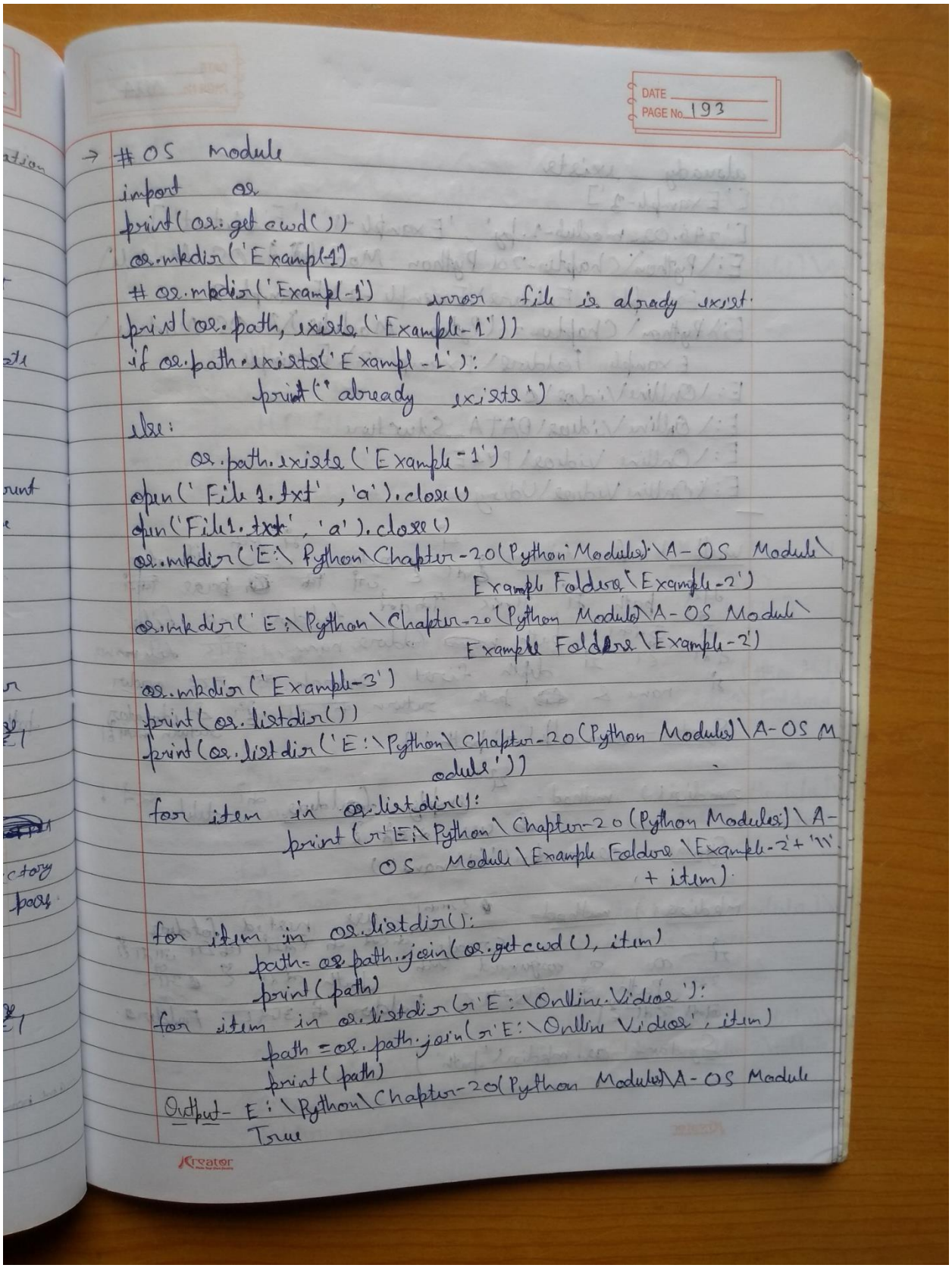
Syntax- os.listdir()

Note- यदि हमें current working directory के अलावा किसी और location के directory को देखना हो तो हम listdir में path pass कर सकते हैं

Syntax- os.listdir('path')

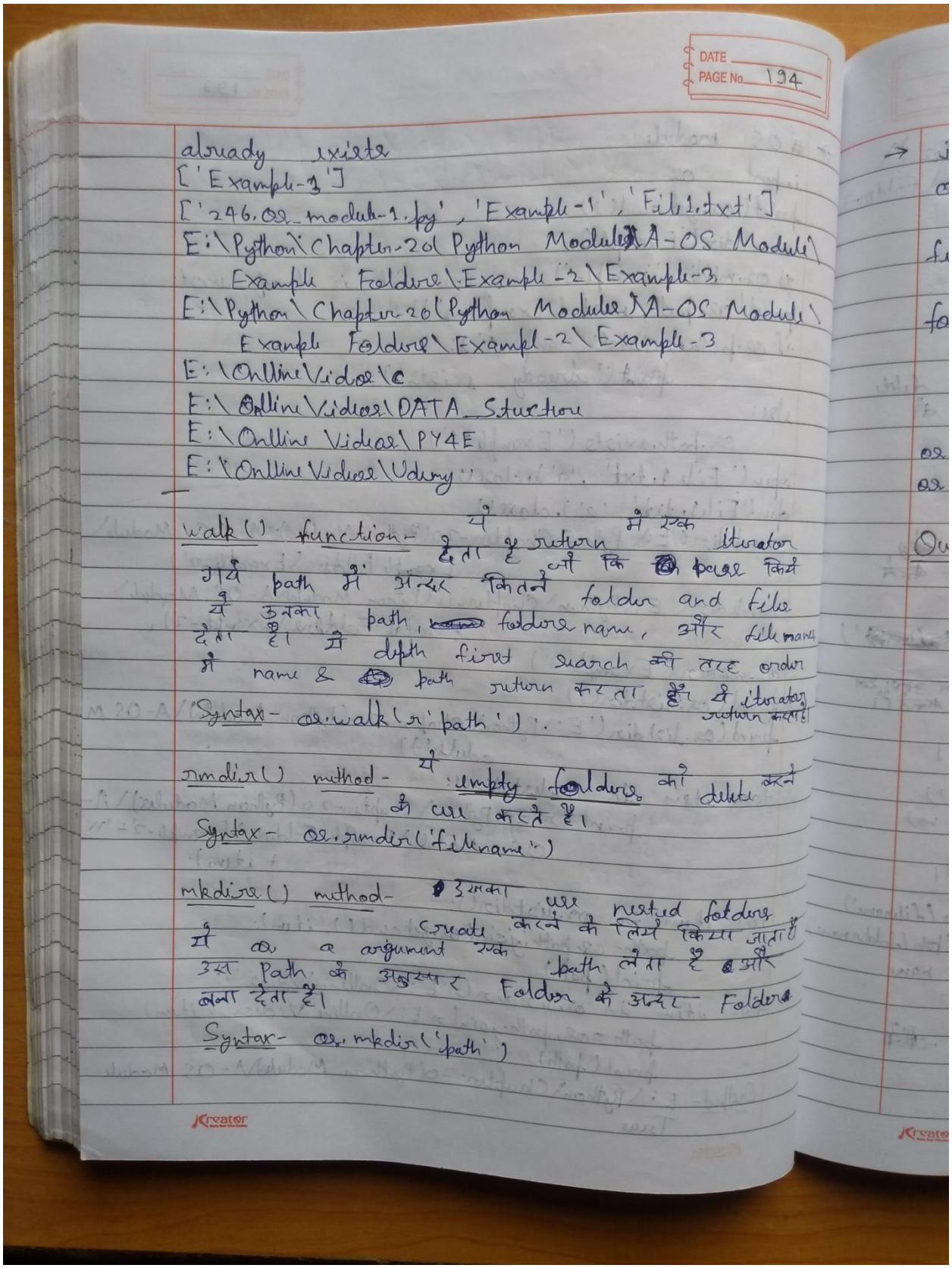
join() function - यह दो paths को add करता है

Syntax- os.path.join('path one', 'path 2')



```
# OS Module
import os
print(os.getcwd())
os.mkdir('Example-1')
# os.mkdir('Example-1')   error file is already exist.
print(os.path.isdir('Example-1'))
if os.path.isdir('Example-1'):
        print("already exists")
else:
    os.path.isdir('Example-1')
open('File1.txt','a').close()
open('File1.txt','a').close()
os.mkdir('E:\Python\Chapter-20(Python Modules)\A-OS Module\
                    Example Folders\Example-2')
os.mkdir('E:\Python\Chapter-20(Python Modules)\A-OS Module\
                    Example Folders\Example-2')
os.mkdir('Example-3')
print(os.listdir())
print(os.listdir('E:\Python\Chapter-20(Python Modules)\A-OS M
                    odule'))

for item in os.listdir():
        print(r'E:\Python\Chapter-20(Python Modules)\A-
            OS Module\Example Folders\Example-2'+'\\'
                + item)

for item in os.listdir():
    path=os.path.join(os.getcwd(),item)
    print(path)
for item in os.listdir(r'E:\Online Videos'):
    path=os.path.join(r'E:\Online Videos',item)
    print(path)
```

Output- E:\Python\Chapter-20(Python Modules)\A-OS Module
True



already existe
['Example-3']
['246.os_module-1.py', 'Example-1', 'File1.txt']
E:\Python\Chapter-20 Python Modules\A-OS Module\Example Folders\Example-2\Example-3
E:\Python\Chapter-20\Python Modules\A-OS Module\Example Folders\Example-2\Example-3
E:\OnlineVideos\C
E:\OnlineVideos\DATA_Structure
E:\Online Videos\PY4E
E:\OnlineVideos\Udemy

walk() function- यह return में एक iterator देता है जो कि pass किये गये path में अन्दर कितने folder and file है उनका path, folders name, और file name देता है। यह depth first search की तरह order में name & path return करता है। यह iterator return करता है।

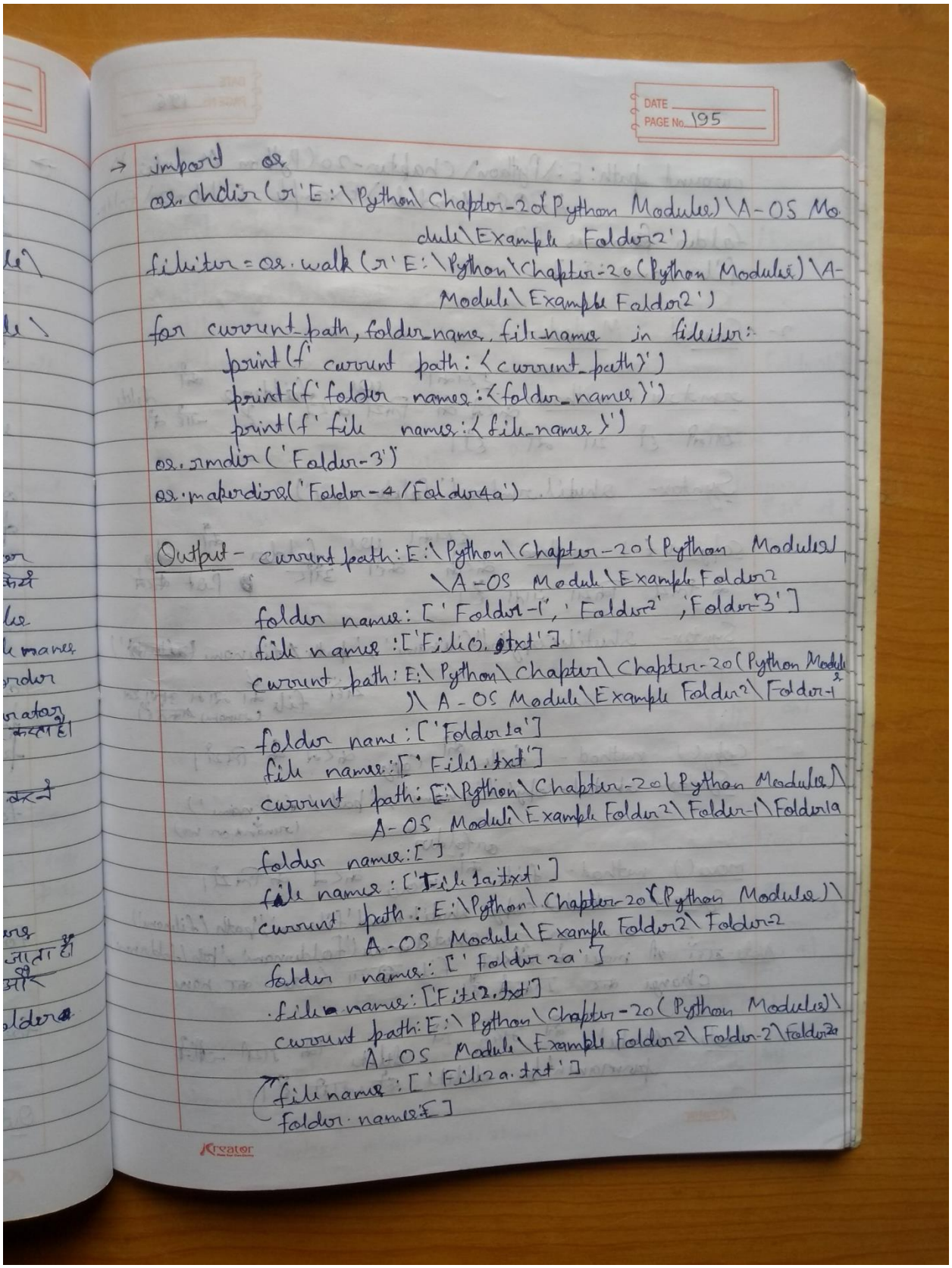
Syntax- os.walk('path').

rmdir() method- यह empty folders को delete करने के use करते है।

Syntax- os.rmdir('filename')

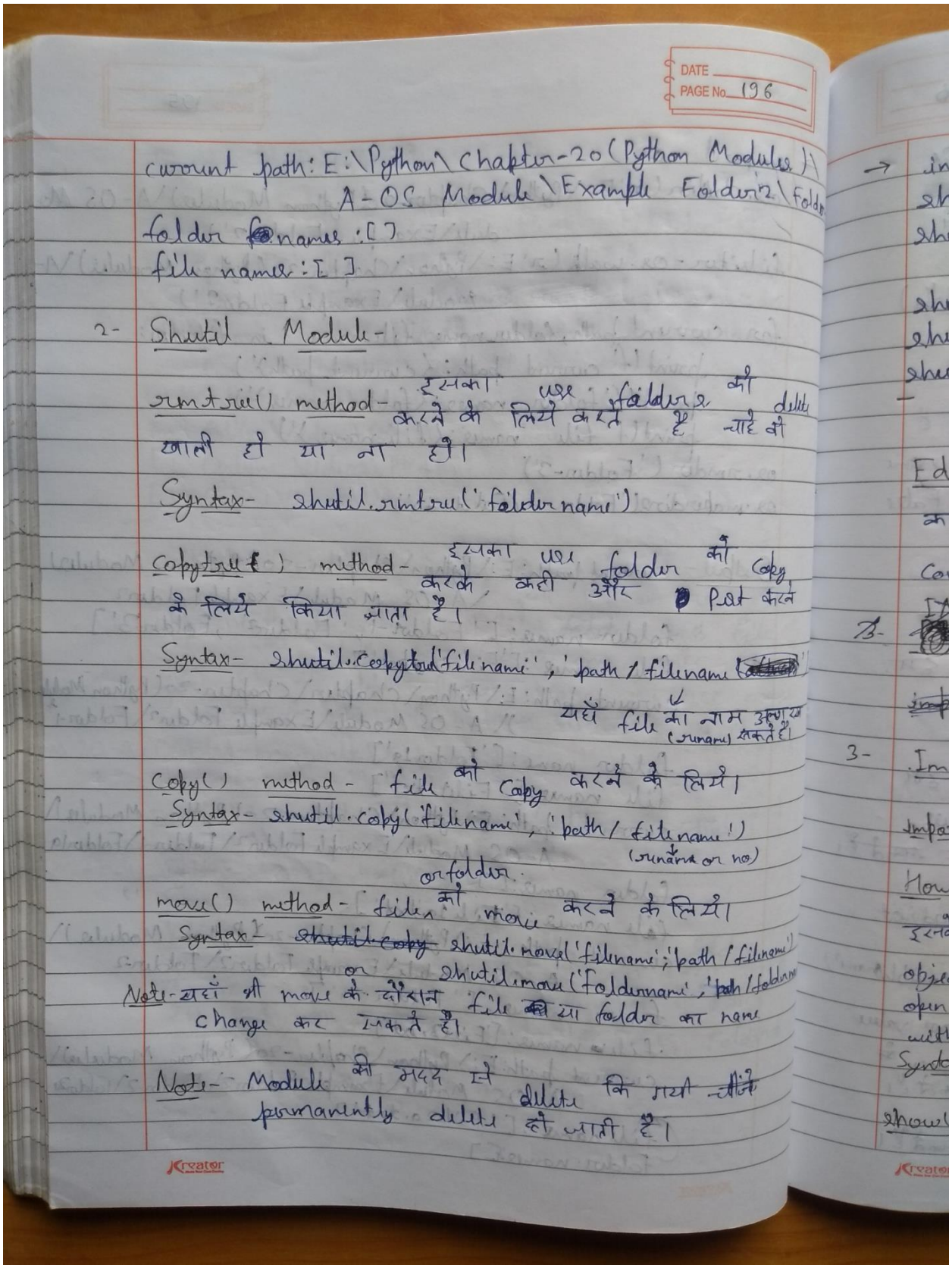
mkdirs() method- इसका use nested folders create करने के लिये किया जाता है यह os a argument एक path लेता है और उस Path के अनुसार Folder के अन्दर Folders बना देता है।

Syntax- os.mkdirs('path')



→ import os
os.chdir(r'E:\Python\Chapter-20(Python Modules)\A-OS Module\Example Folder2')
fileiter = os.walk(r'E:\Python\Chapter-20(Python Modules)\A-Module\Example Folder2')
for current_path, folder_name, file_name in fileiter:
    print(f'current path: {current_path}')
    print(f'folder names: {folder_name}')
    print(f'file names: {file_name}')
os.rmdir('Folder-3')
os.makedirs('Folder-4/Folder4a')

Output - current path: E:\Python\Chapter-20(Python Modules)\A-OS Module\Example Folder2
folder names: ['Folder-1', 'Folder2', 'Folder-3']
file names: ['File0.txt']
current path: E:\Python\Chapter\Chapter-20(Python Modules)\A-OS Module\Example Folder2\Folder-1
folder name: ['Folder1a']
file names: ['File1.txt']
current path: E:\Python\Chapter-20(Python Modules)\A-OS Module\Example Folder2\Folder-1\Folder1a
folder names: []
file names: ['File1a.txt']
current path: E:\Python\Chapter-20(Python Modules)\A-OS Module\Example Folder2\Folder-2
folder names: ['Folder-2a']
file names: ['File2.txt']
current path: E:\Python\Chapter-20(Python Modules)\A-OS Module\Example Folder2\Folder-2\Folder2a
file names: ['File2a.txt']
folder names: []



DATE
PAGE No. 196

current path: E:\Python\Chapter-20(Python Modules)\A-OS Module\Example Folder2\Folder
folder names: []
file names: []

2- Shutil Module-

rmtree() method- इसका use folder को delete करने के लिये करते है चाहे वो खाली हो या ना हो।

Syntax- shutil.rmtree('foldername')

copytree() method- इसका use folder को copy करके कही और Past करने के लिये किया जाता है।

Syntax- shutil.copytree('filename', 'path/filename')
यहाँ file का नाम अलग (rename) कर सकते है।

copy() method- file को copy करने के लिये।
Syntax- shutil.copy('filename', 'path/filename')
(rename or no)

move() method- file या folder को move करने के लिये।
Syntax- shutil.move('filename', 'path/filename')
or
shutil.move('foldername', 'path/foldername')

Note- यहाँ भी move के दौरान file या folder का name change कर सकते है।

Note- Module की मदद से delete कि गयी चीजें permanently delete हो जाती है।

PAGE No. 197

```
import shutil
shutil.rmtree('Example Folder1')
shutil.copytree('Example Folder2', 'Example Folder2/Example Folder2a')
shutil.copy('File1.txt', 'Example Folder2/File1a.txt')
shutil.move('File2.txt', 'Example Folder2/File2.txt')
shutil.move('Example Folder3', 'Example Folder2/Example Folder2b')
```

Edit IMAGES using Python- इसके लिये हम Pillow library का use करते है। इसे install करने के लिये हम ये command देते है- pip install Pillow

~~1- IMAGE Module- यह Pillow library का एक module है।~~

~~import Syntax- import PIL~~

3- **Image Module-** ये Python की library Pillow का एक module है।

Import Syntax- from PIL import image

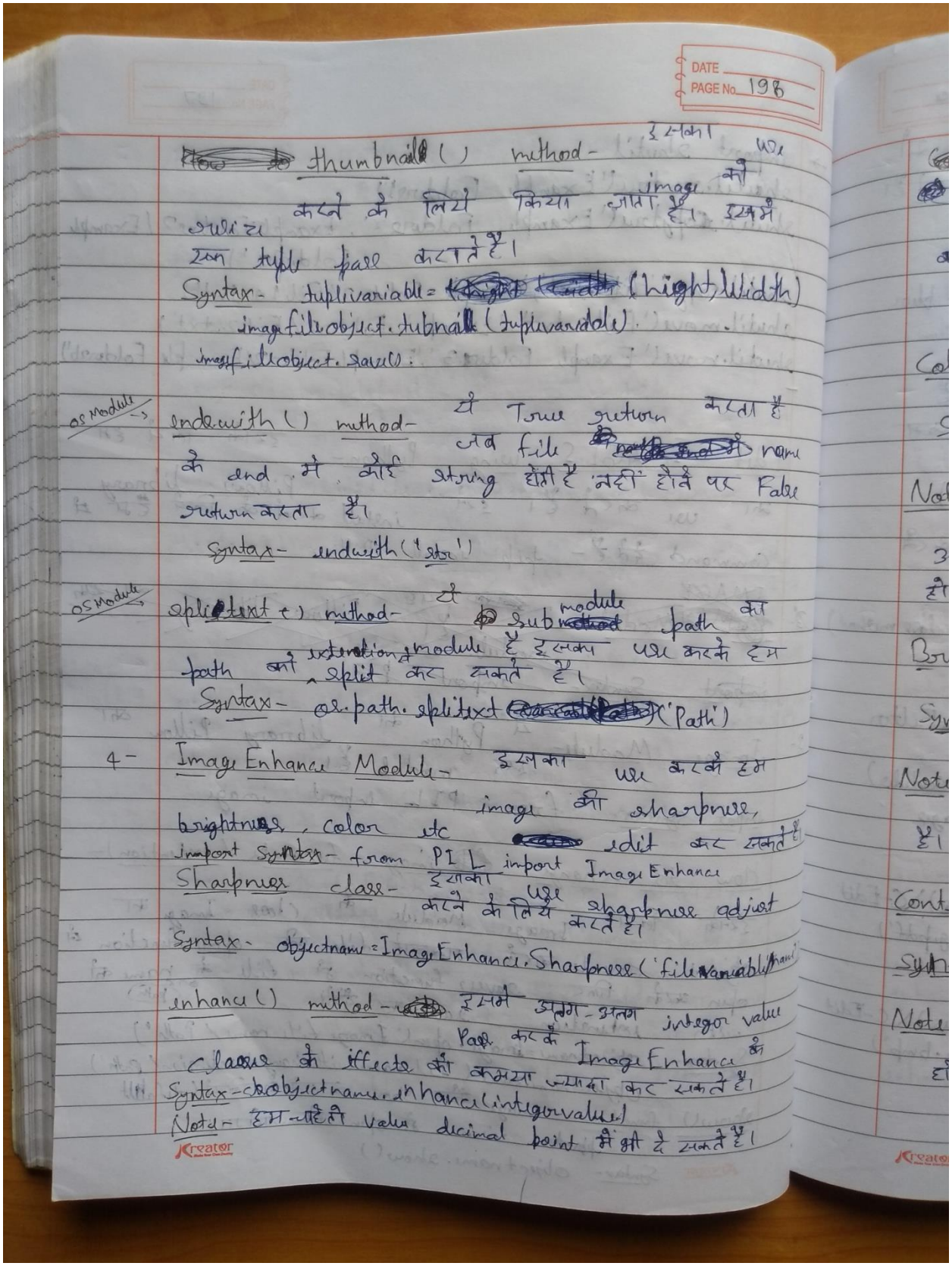
How to change image formate (extention)-

इसके लिये Image Module के class Image का object create करके मनचाहे file की open() function से open करके फिर save() function से file के name को with extention (path) save कर देंगे।

Syntax- objectname = Image.open('Image file name/ Path')
objectname.save('filename with changed extension/ Path')

show() function- image को show कराने के लिये (default photo viewer में open हो जायेगा।

Syntax- objectname.show()



thumbnail() method- इसका use करके हम image को resize करने के लिये किया जाता है। इसमें tuple pass करवाते हैं।

Syntax- tuplevariable = (hight, width)
imagefileobject.thumbnail(tuplevariable)
imagefileobject.save()

os module → endswith() method- यह True return करता है जब file name के end में कोई string होती है नहीं होने पर False return करता है।

Syntax- endswith('str')

os module → splitext() method- यह sub module है इसका use करके हम path को extension, path को split कर सकते हैं।

Syntax- os.path.splitext('Path')

4- Image Enhance Module- इसका use करके हम image की sharpness, brightness, color etc edit कर सकते हैं।

import Syntax- from PIL import ImageEnhance

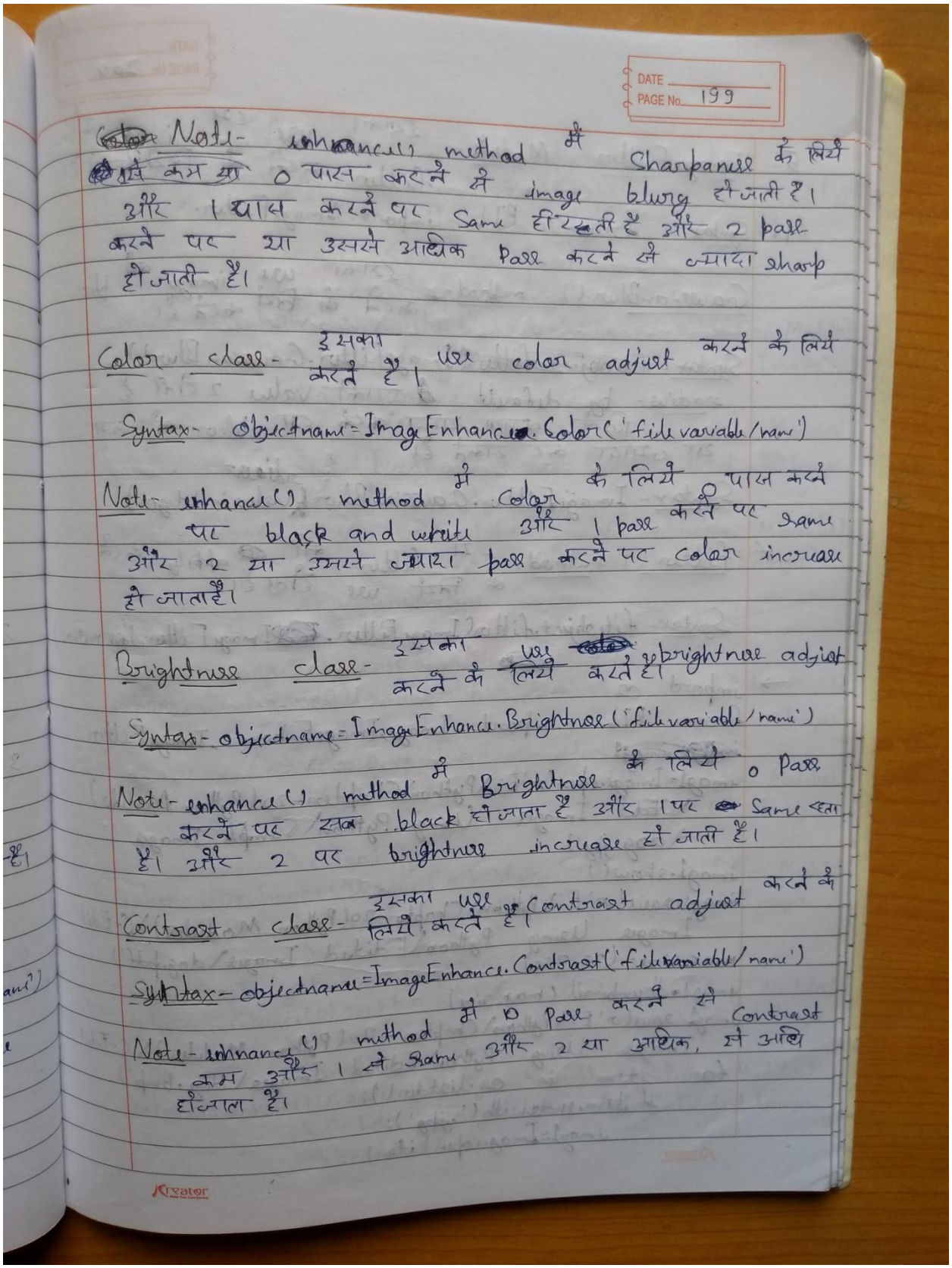
Sharpness class- इसका use करने के लिये करते हैं।

Syntax- objectname = ImageEnhance.Sharpness('filevariable')

enhance() method- इसमें अलग-अलग integer value pass करके ImageEnhance के classes के effects को कमया ज्यादा कर सकते हैं।

Syntax- objectname.enhance(integervalue)

Note- इसमें value decimal point में भी दे सकते हैं।



DATE
PAGE No. 199

Note- enhance() method में Sharpness के लिये 0 से कम या 0 पास करने से image blurry हो जाती है। और 1 पास करने पर same रहती है और 2 pass करने पर या उससे अधिक Pass करने से ज्यादा sharp हो जाती है।

Color class- इसका use color adjust करने के लिये करते है।

Syntax- objectname = ImageEnhance.Color('file variable/name')

Note- enhance() method में Color के लिये 0 पास करने पर black and white और 1 pass करने पर same और 2 या उससे ज्यादा pass करने पर color increase हो जाता है।

Brightness class- इसका use brightness adjust करने के लिये करते है।

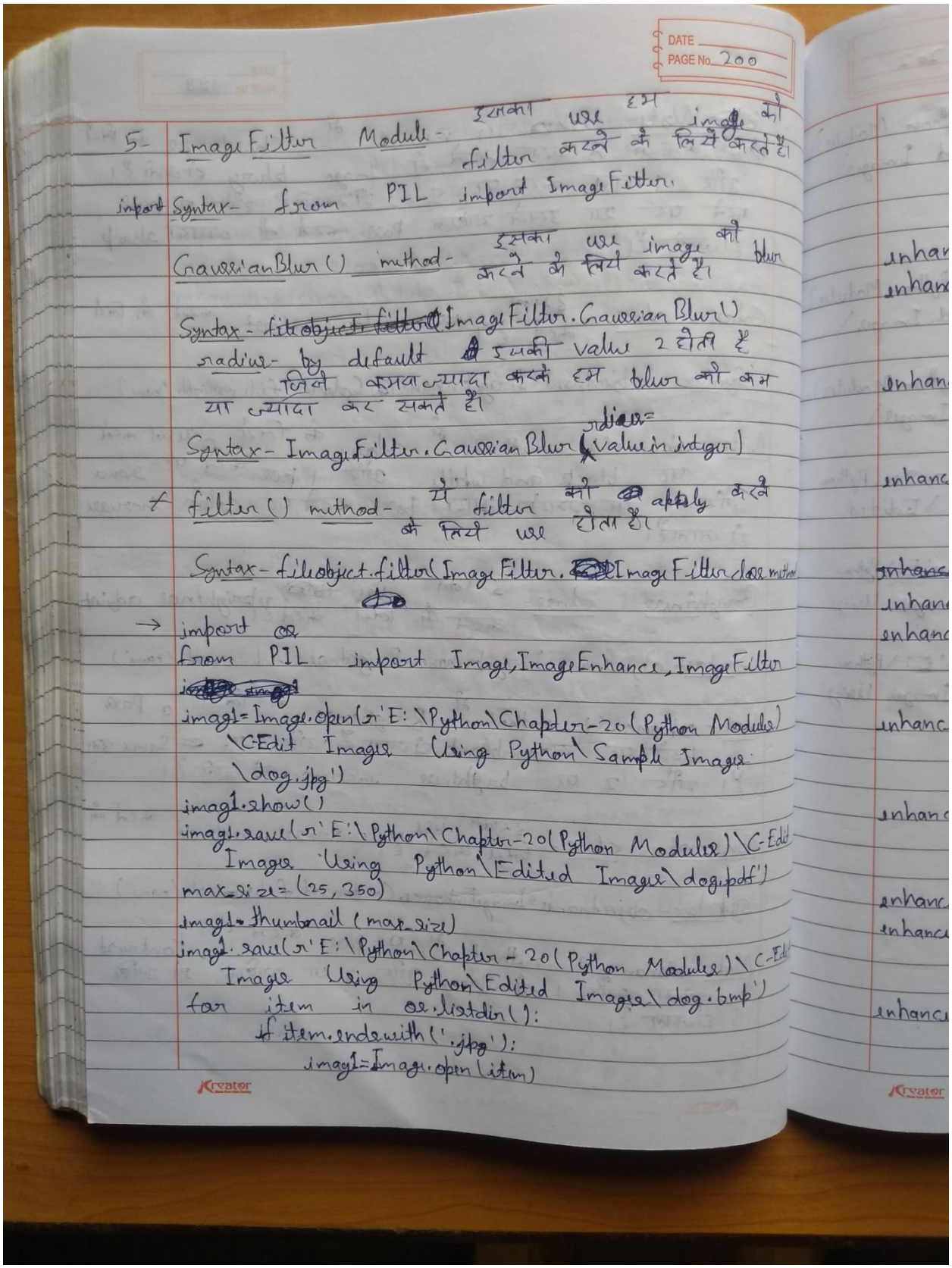
Syntax- objectname = ImageEnhance.Brightness('file variable/name')

Note- enhance() method में Brightness के लिये 0 Pass करने पर सब black हो जाता है और 1 पर same रहता है। और 2 पर brightness increase हो जाती है।

Contrast class- इसका use Contrast adjust करने के लिये करते है।

Syntax- objectname = ImageEnhance.Contrast('file variable/name')

Note- enhance() method में 0 Pass करने से Contrast कम और 1 से same और 2 या अधिक, से अधिक हो जाता है।



5- ImageFilter Module - इसका use हम image को filter करने के लिये करते है।

import Syntax - from PIL import ImageFilter.

GaussianBlur() method - इसका use image को blur करने के लिये करते है।

Syntax - ~~fileobject=filter~~ ImageFilter.GaussianBlur()

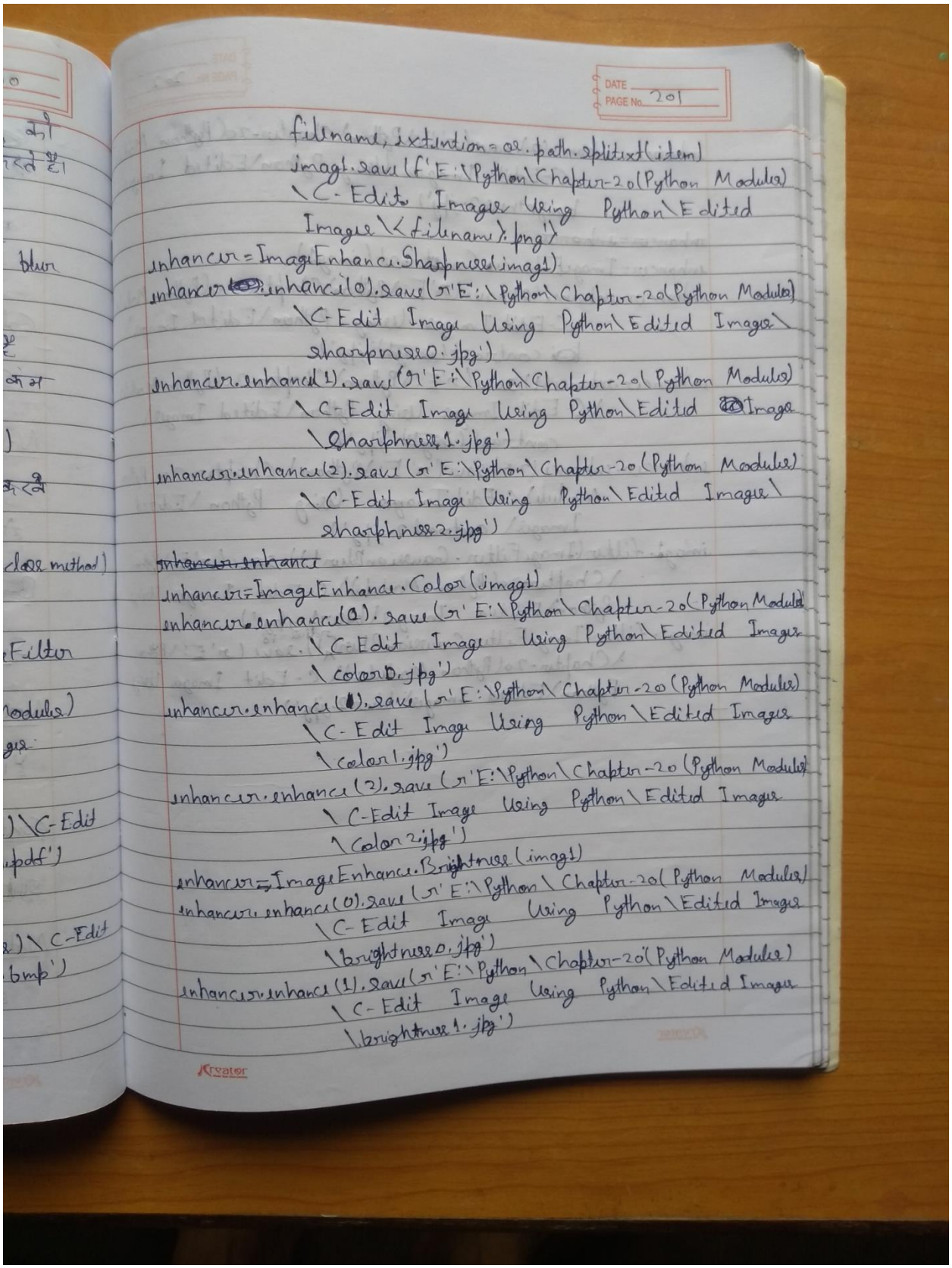
radius - by default इसकी value 2 होती है जिसे कमज्यादा करके हम blur को कम या ज्यादा कर सकते है।

Syntax - ImageFilter.GaussianBlur(radius=value in integer)

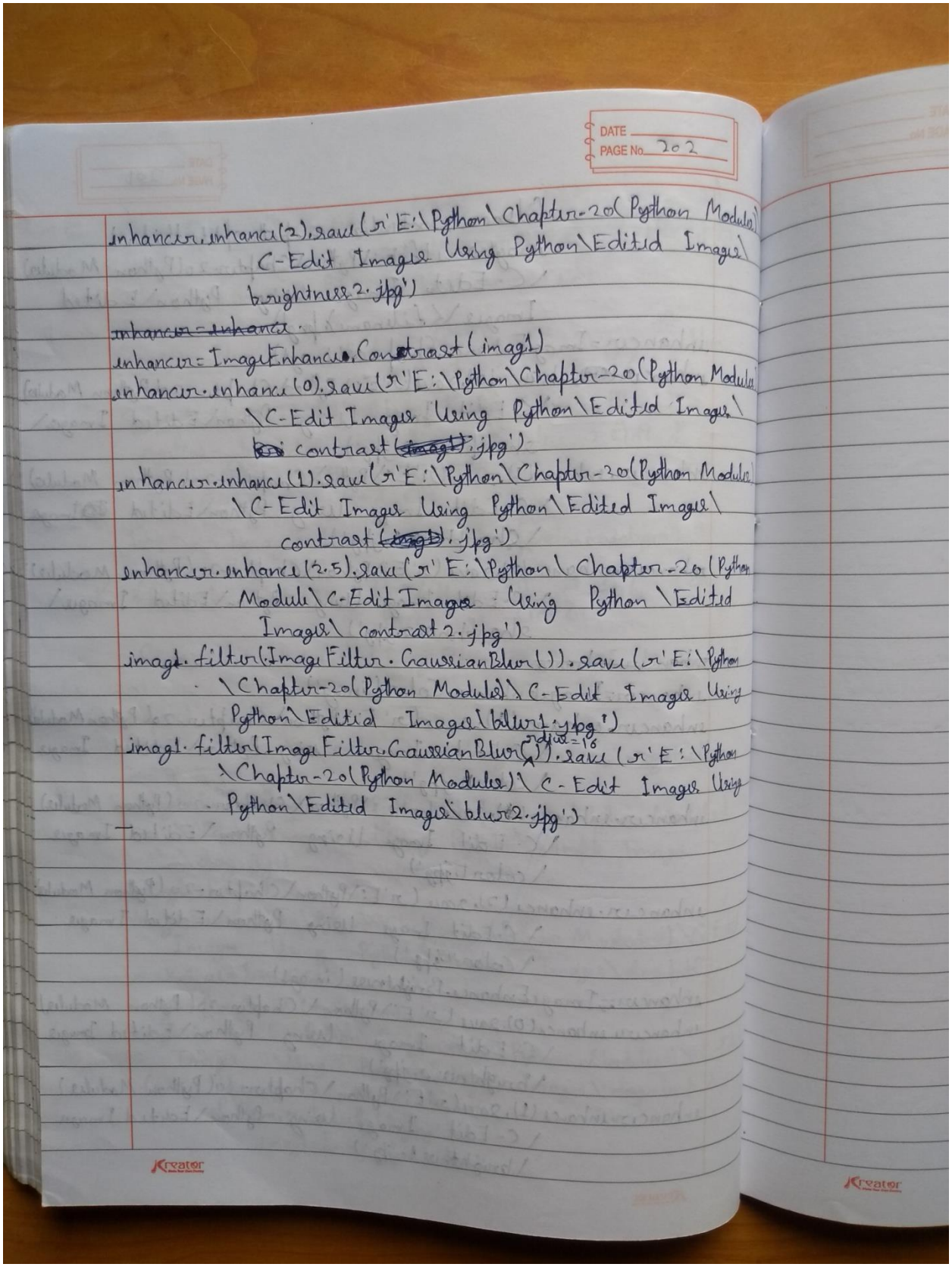
filter() method - ये filter को apply करने के लिये use होता है।

Syntax - fileobject.filter(ImageFilter. ImageFilter class method)

→ import os
from PIL import Image, ImageEnhance, ImageFilter
~~import image~~
img1=Image.open(r'E:\Python\Chapter-20(Python Modules)\CEdit Images Using Python\Sample Images\dog.jpg')
img1.show()
img1.save(r'E:\Python\Chapter-20(Python Modules)\C-Edit Images Using Python\Edited Images\dog.pdf')
max_size=(25,350)
img1.thumbnail(max_size)
img1.save(r'E:\Python\Chapter-20(Python Modules)\C-Edit Images Using Python\Edited Images\dog.bmp')
for item in os.listdir():
    if item.endswith('.jpg'):
        img1=Image.open(item)



```
filename, extention = os.path.splitext(item)
imag1.save(f'E:\Python\Chapter-20(Python Modules)
\C-Edit Images Using Python\Edited
Images\{filename}.png')
enhancer = ImageEnhance.Sharpness(imag1)
enhancer.enhance(0).save(r'E:\Python\Chapter-20(Python Modules)
\C-Edit Image Using Python\Edited Images\
sharpness0.jpg')
enhancer.enhance(1).save(r'E:\Python\Chapter-20(Python Modules)
\C-Edit Image Using Python\Edited Image
\Sharpness1.jpg')
enhancer.enhance(2).save(r'E:\Python\Chapter-20(Python Modules)
\C-Edit Image Using Python\Edited Images\
sharpness2.jpg')
enhancer=ImageEnhance.Color(imag1)
enhancer.enhance(0).save(r'E:\Python\Chapter-20(Python Modules
\C-Edit Image Using Python\Edited Images
\color0.jpg')
enhancer.enhance(1).save(r'E:\Python\Chapter-20(Python Modules)
\C-Edit Image Using Python\Edited Images
\color1.jpg')
enhancer.enhance(2).save(r'E:\Python\Chapter-20(Python Modules)
\C-Edit Image Using Python\Edited Images
\color2.jpg')
enhancer=ImageEnhance.Brightness(imag1)
enhancer.enhance(0).save(r'E:\Python\Chapter-20(Python Modules)
\C-Edit Image Using Python\Edited Images
\brightness0.jpg')
enhancer.enhance(1).save(r'E:\Python\Chapter-20(Python Modules)
\C-Edit Image Using Python\Edited Images
\brightness1.jpg')
```



```
enhancer.enhance(2).save(r'E:\Python\Chapter-20(Python Modules)
        C-Edit Images Using Python\Edited Images\
        brightness2.jpg')
enhancer=ImageEnhance.Contrast(img1)
enhancer.enhance(0).save(r'E:\Python\Chapter-20(Python Modules
        \C-Edit Images Using Python\Edited Images\
        contrast0.jpg')
enhancer.enhance(1).save(r'E:\Python\Chapter-20(Python Modules
        \C-Edit Images Using Python\Edited Images\
        contrast1.jpg')
enhancer.enhance(2.5).save(r'E:\Python\Chapter-20(Python
        Modules\C-Edit Images Using Python\Edited
        Images\contrast2.jpg')
img1.filter(ImageFilter.GaussianBlur()).save(r'E:\Python
        \Chapter-20(Python Modules)\C-Edit Images Using
        Python\Edited Images\blur1.jpg')
img1.filter(ImageFilter.GaussianBlur(radius=10)).save(r'E:\Python
        \Chapter-20(Python Modules)\C-Edit Images Using
        Python\Edited Images\blur2.jpg')
```