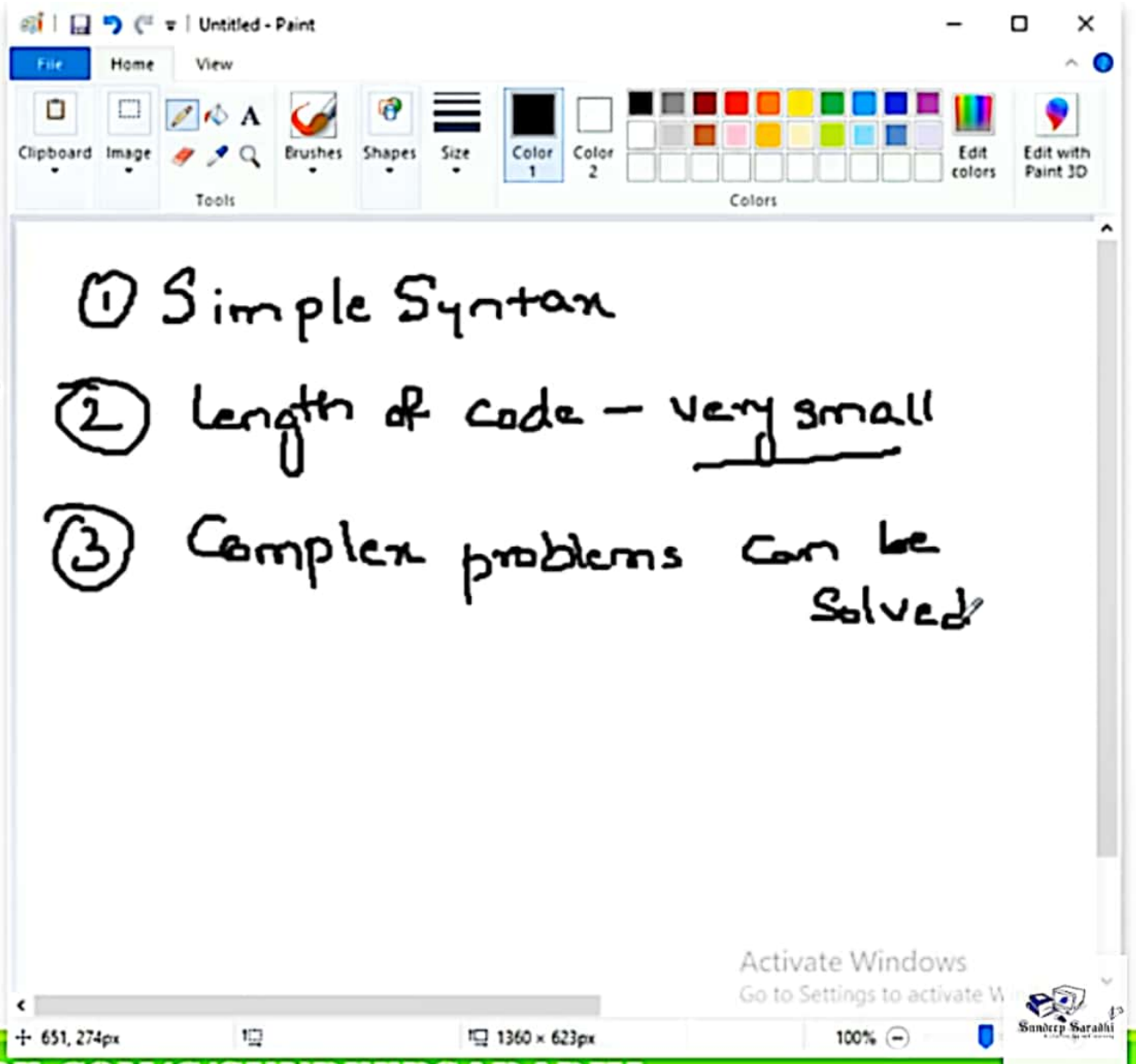
Contents

Introduction to Python

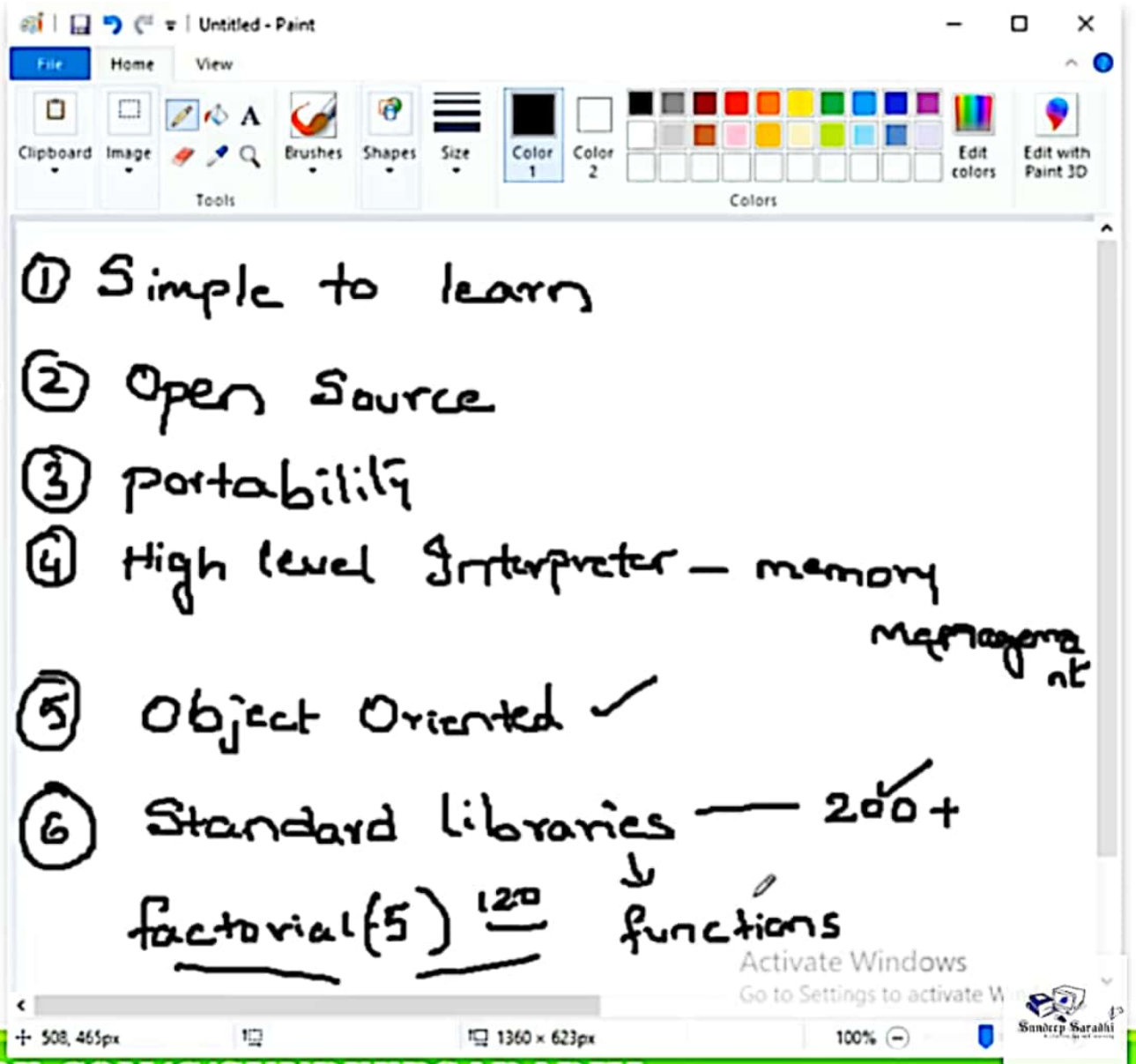
- ❑ Why Python
- ❑ Differences between C, Java & Python
- ❑ Features of Python
- ❑ History of Python

- ❑ Why Python
- ❑ Differences between C,
- ❑ Features of Python
- ❑ History of Python

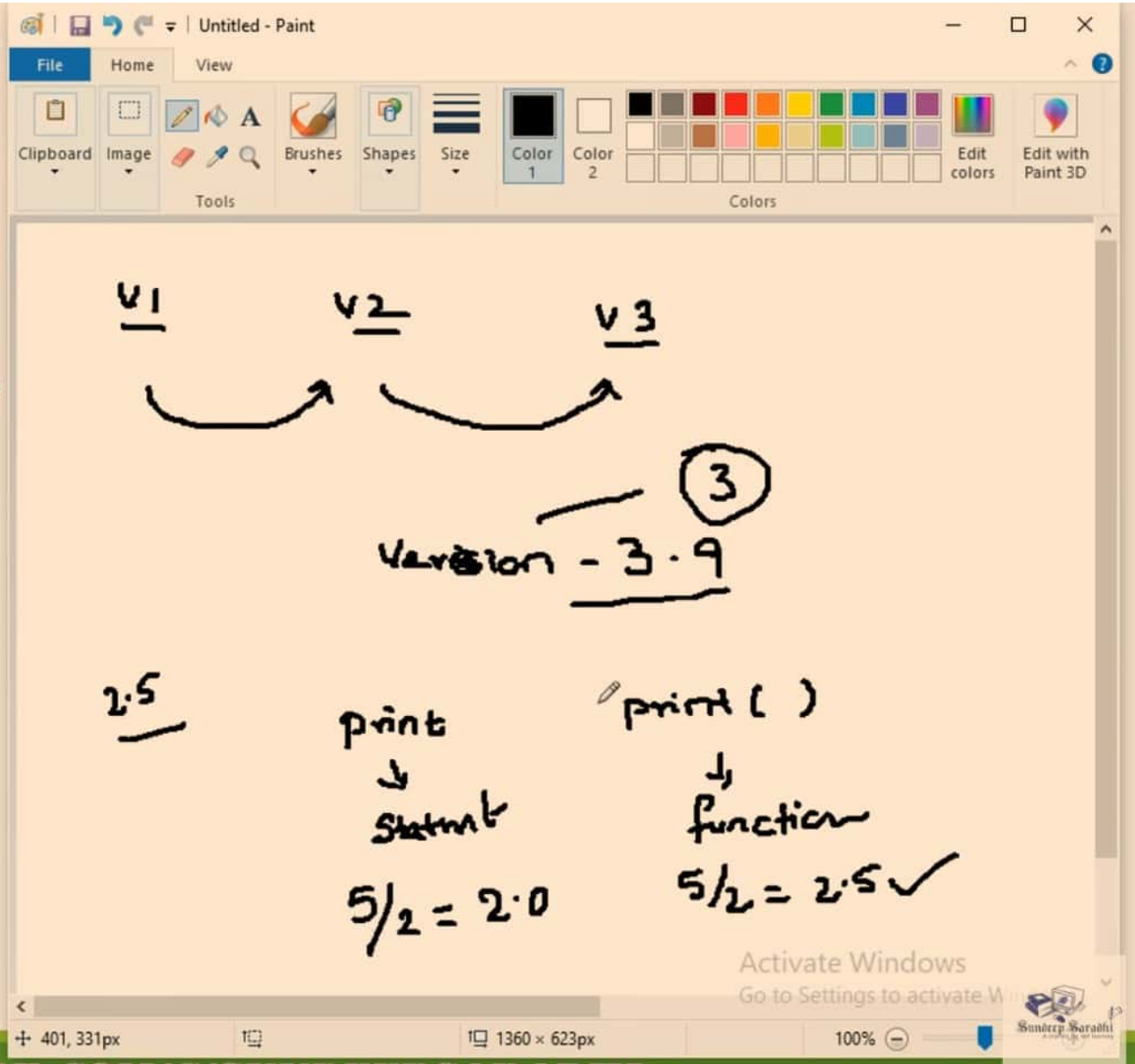


	C	C++	JAVA	PYTHON
Language Type	Procedure Oriented Language	Object Oriented Language	Object Oriented Language	Both Procedural & Object Oriented Language
Building Block	Function Driven	Object Driven	Both Object and Class Driven	Function, Object and Class Driven
Extension	.c	.cpp	.java	.py
Platform	Dependent	Independent	Independent	Independent
Comment Style	/* */	// Single Line /* */ Multi-Line	// Single Line /* */ Multi-Line	# Single Line """ """ Multi Line
Translation Type	Compiled	Compiled	Compiled & Interpreted	Interpreted
Database Connectivity	not supported	not supported	Supported	Supported
Representing Block of Statements	{ }	{ }	{ }	Indentation
Declaring Variables	Required	Required	Required	Not Required
Applications	Compilers , Interpreters , Embedded Programming etc.	Simple Desktop Applications, Embedded Systems etc.	Desktop GUI, Mobile, Web, Gaming etc.	Desktop, Web, Gaming, Network Programming etc.
IDE	Turbo C, Code Blocks	Turbo C++, Code Blocks	Eclipse , NetBeans etc	PyCharm , PyDev, IDLE, Jupyter Notebook, Spyder etc

- ❑ Why Python
- ❑ Differences between C,
- ❑ Features of Python
- ❑ History of Python



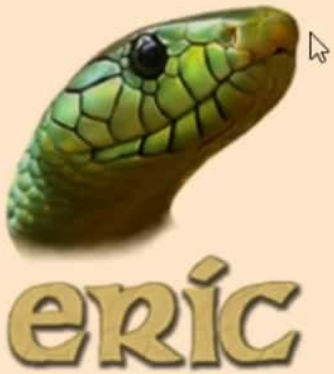
- Why Python
- Differences between C,
- Features of Python
- History of Python

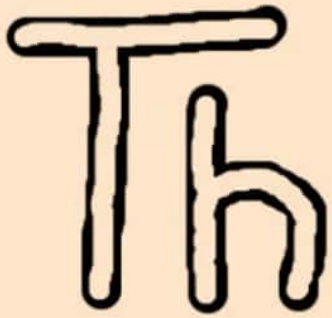


Applications of Python

- Web & Internet Development
- Desktop GUI Applications
- Artificial Intelligence
- Machine Learning
- Image Processing Applications
- Business Applications
- Games and 3D Graphics
- Network Programming
- Database Access

Integrated Development Environment
IDEs

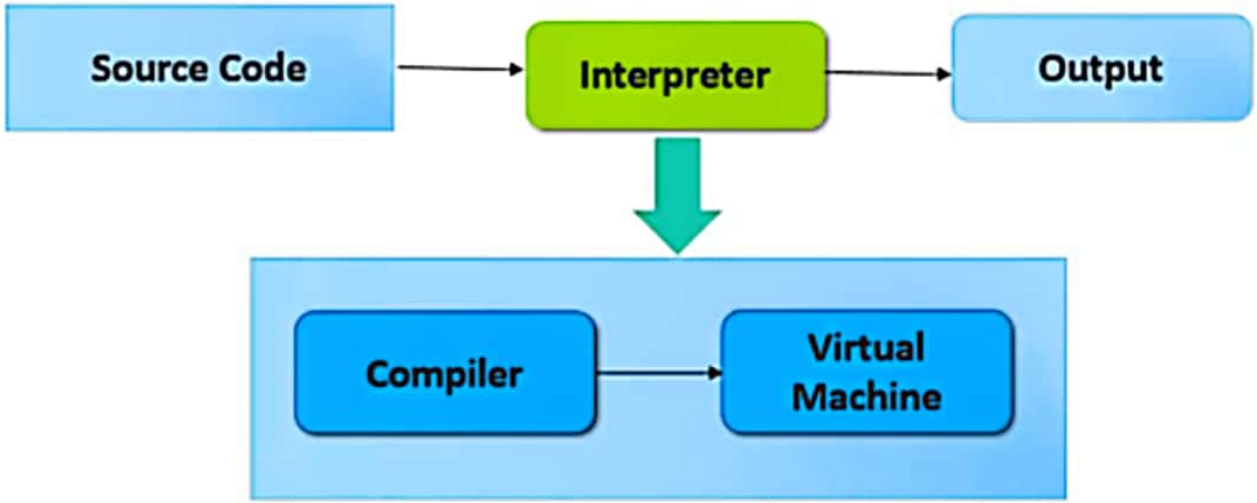




Introduction to Python Interpreter

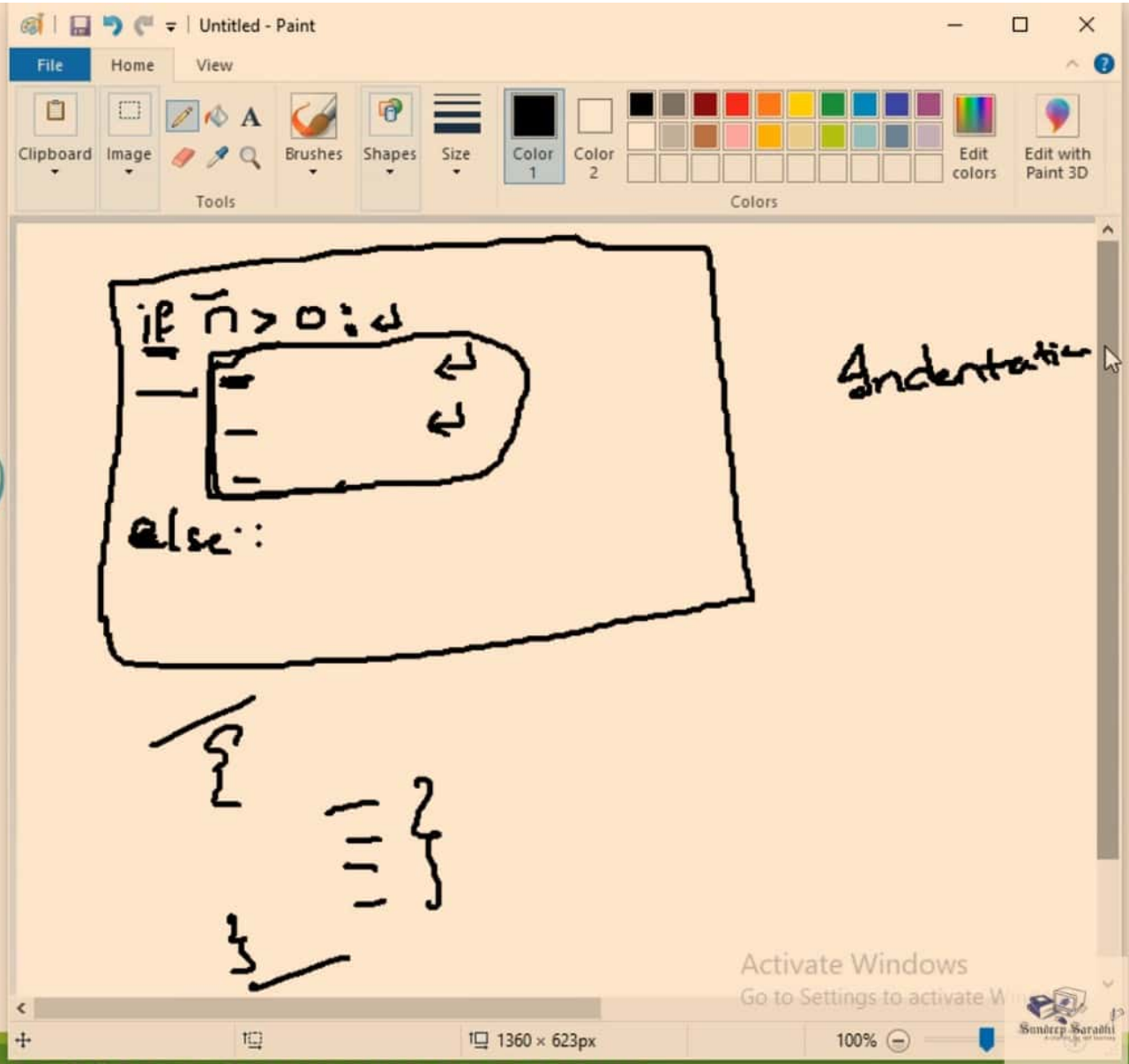
COMPILER

INTERPRETER

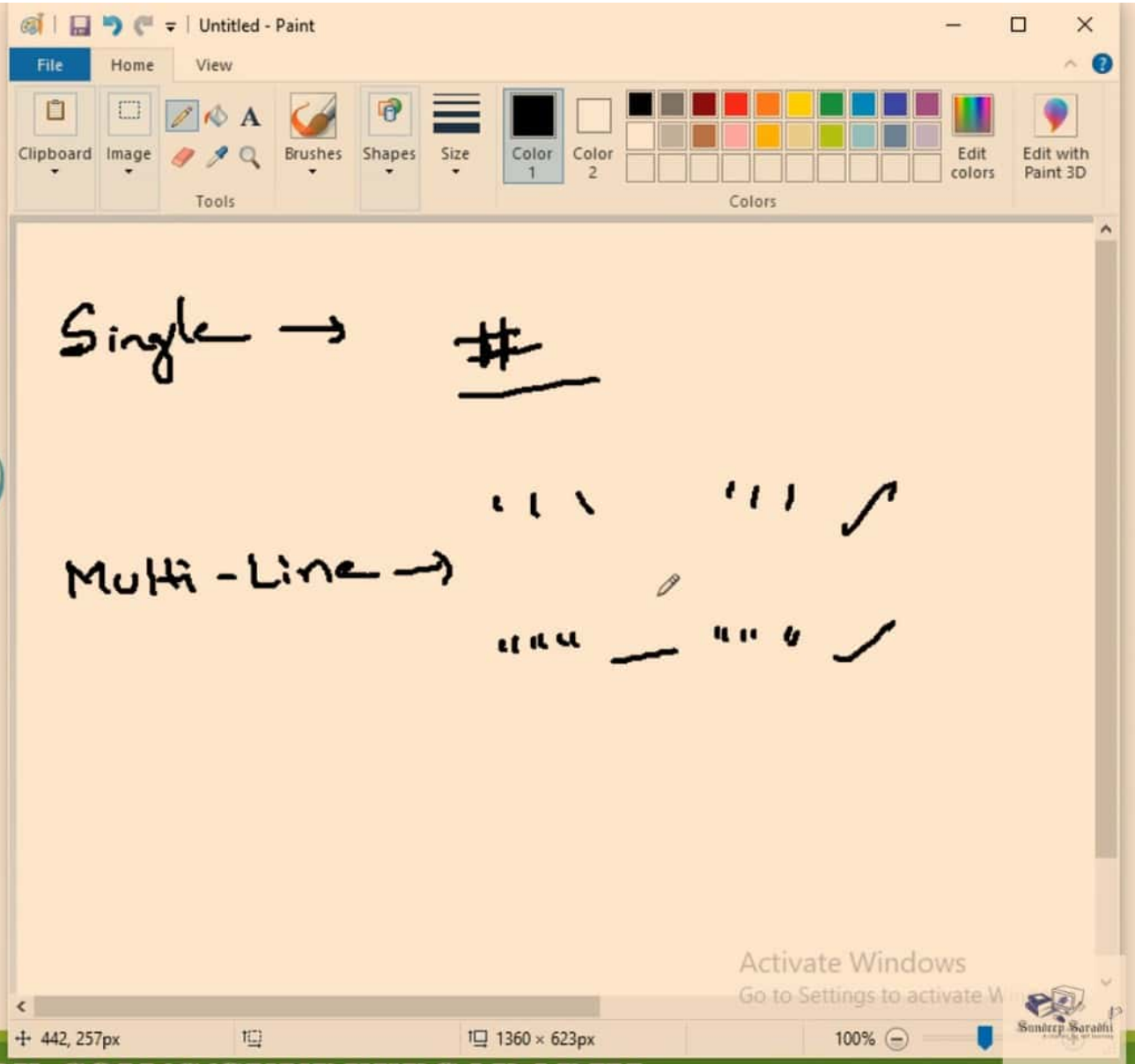


Indentation and Comments

Indentatio



Indentatio



Home Page - Select or create a n | Untitled11 - Jupyter Notebook

localhost:8888/notebooks/Untitled11.ipynb?kernel_name=python3

jupyter Untitled11 Last Checkpoint: 4 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [2]:
```
if 10>0:
    #This is a comment
    print("Positive Number")
    #print("10 is greater")
else:
    print("10 is smaller")
```

```
Positive Number
```

In [ ]:

In [ ]:

- Python requires indentation as a part of syntax.
- Indentation signifies the start and end of block of code.
- Programs will not run without correct indentation.


```
if n > 0 :
    print("Positive Number")
else :
    print("Negative Number")
```

```
if n > 0
{
    print("Positive Number")
}
else
{
    print("Negative Number")
}
```

Single Line Comment :

as prefix to the line

Multi Line Comment :

“ “ “ Comment “ “ “

or

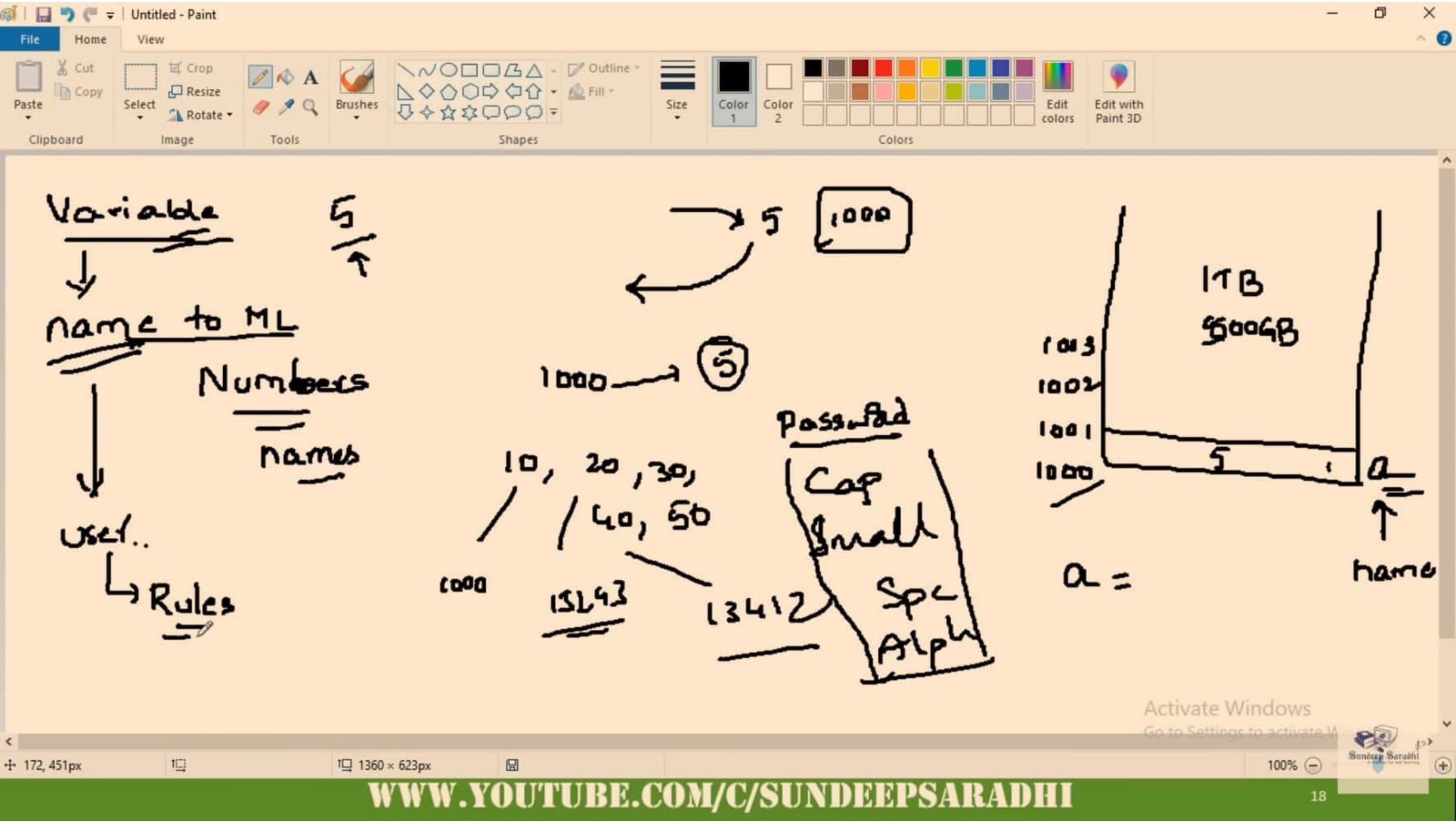
‘ ‘ ‘ Comment ‘ ‘ ‘

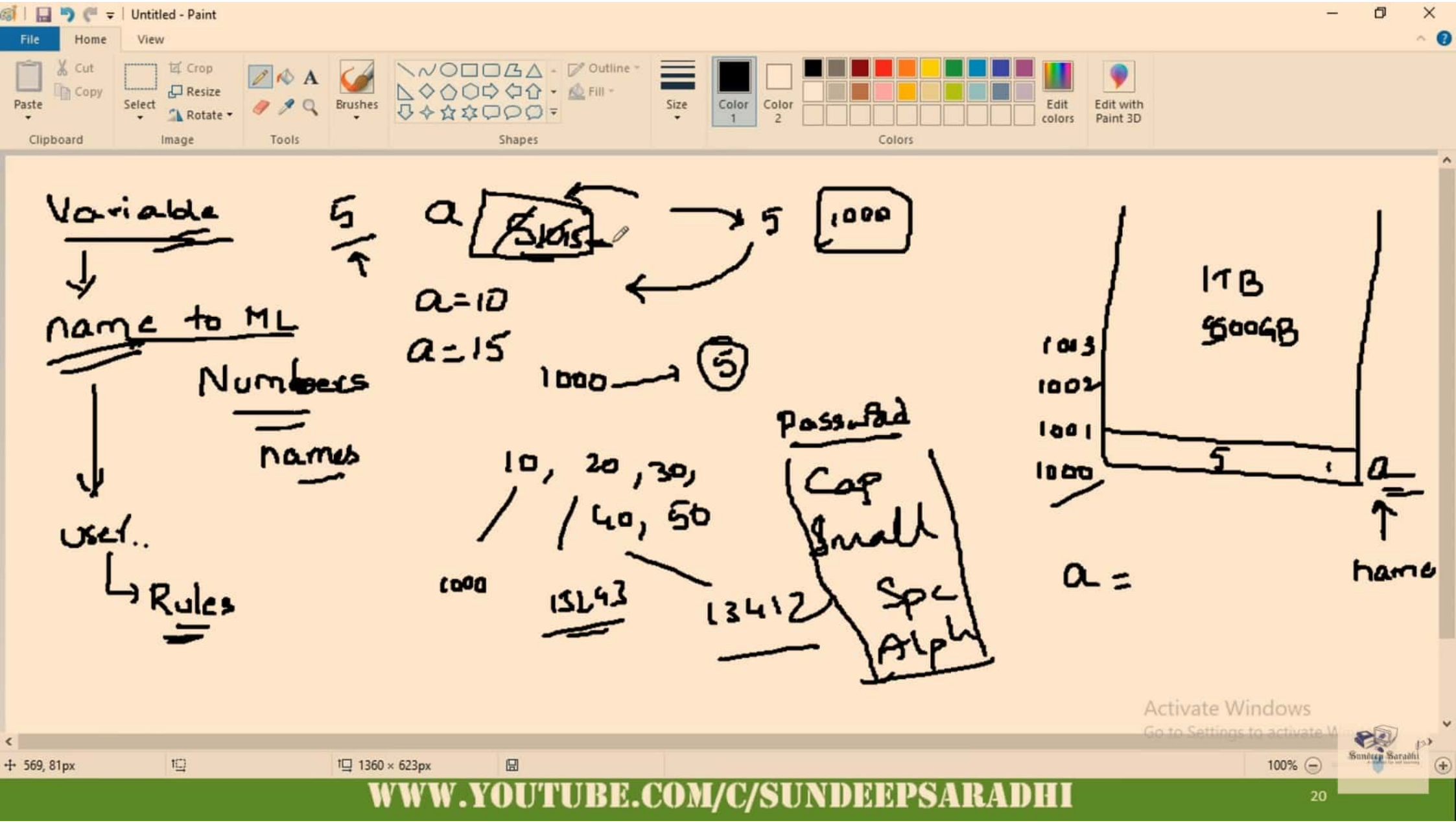
Keywords

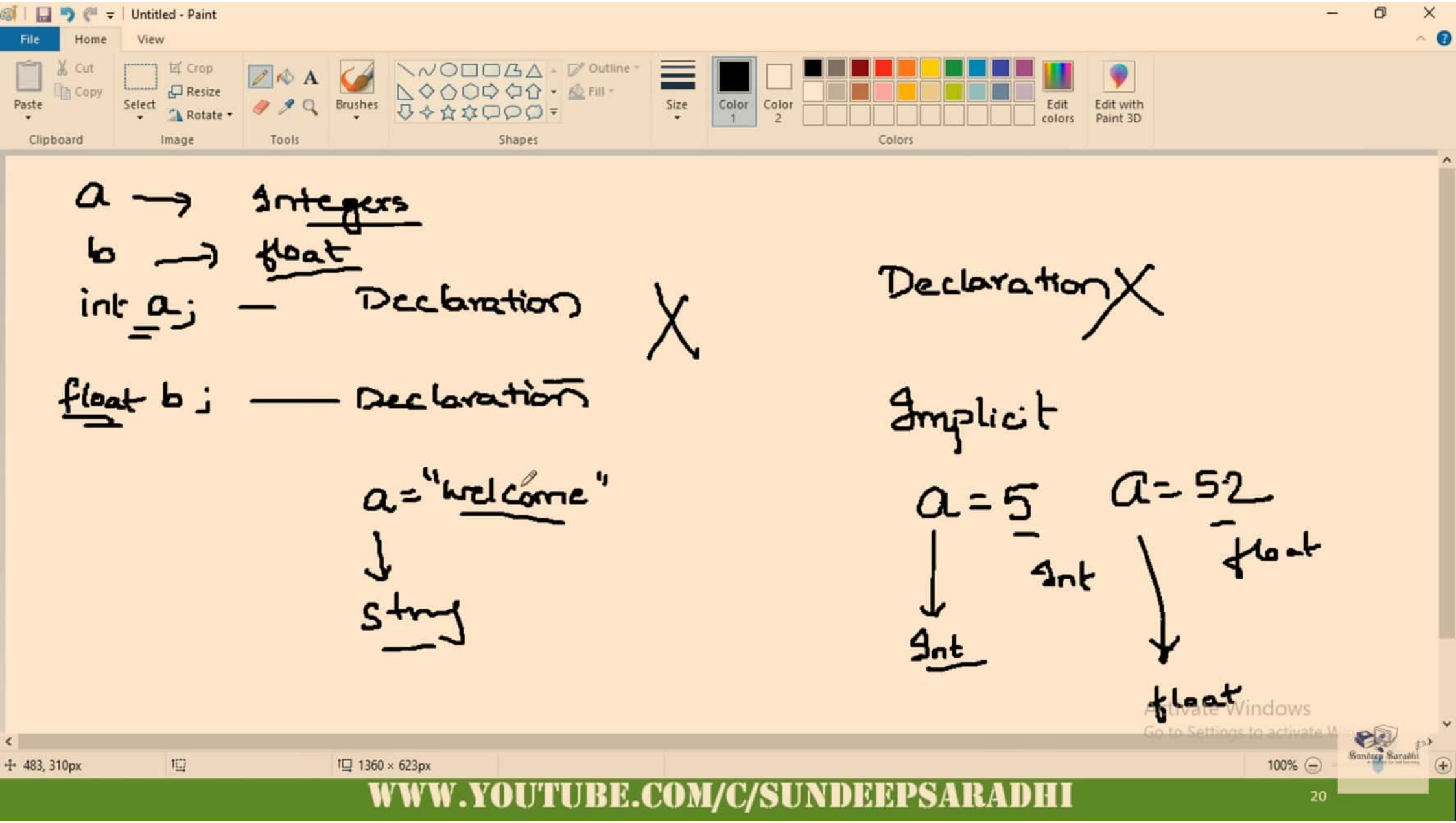
➢ **Keywords** are the reserved words which have predefined meaning and functionality.

False	class	finally	is
return	None	continue	for
lambda	try	True	def
from	global	nonlocal	while
And	del	not	with
as	elif	if	or
yield	assert	else	import
pass	break	except	in
raise			

Variables - Identifiers







Variable Name

- A Variable is defined as an alternate name for Memory Location and can have a short name or a more descriptive name.

Rules for Python Variables

- A Variable name must start with a letter or the underscore character
- A Variable name cannot start with a number
- A Variable name can only have alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive. (name, Name and NAME are three different variables.
- Variable name should not match with

Built-In Types

- **Numeric**
  - Integer - Whole Numbers Ex: 3, 400, 500
  - Complex Number – Numbers with real and imaginary values , Ex: a+bj
  - Float - Numbers with decimal point , Ex: 3.14, 4.15
- **Dictionary** - Unordered Key: Value pairs { key1 : value1, key2 : value2…. }
- **Boolean** - True or False
- **Set** - Unordered Collection of unique object , Ex: {10, 20, 30}
- **Sequence Type**
  - Strings - Ordered sequence of characters , Ex: “python”, “sundeep”
  - List - Ordered sequence of object , Ex: [10 , 20 , 30]
  - Tuple - Ordered immutable sequence of objects , Ex: (10, 20, 30)

Assigning Values to Variables

1) Basic form: This form is the most commonly used form for assignment.

Example: >>> str=" Hello"

>>> print(str)

Hello

2) Tuple Assignment:

```
>>> x,y=(50,100)
>>> print('x=',x)
x= 50
>>> print("y=",y)
y= 100
```

3) List Assignment: This works in the same way as the tuple assignment.

```
>>> x,y=[3,5]
>>> print('x=',x,"y=",y)
x= 3  y= 5
```

4) Sequence Assignment: Any sequence of names can be assigned to any sequence of values, and Python assigns the items one at a time by position.

```
>>> a,b,c='hai'
>>> a
'h'
>>> b
'a'
>>> c
'i'
```

5) Extended Sequence unpacking: It allows us to be more flexible in how we select portions of a sequence to assign.

```
>>> p, *q="Hello"
```

Here, p is matched with the first character in the string on the right and q with the rest. The starred name (*q) is assigned a list, which collects all items in the sequence not assigned to other names.

```
>>> p
'H'
>>> q
['e', 'l', 'l', 'o']
```

6) Multiple target assignment: In this form, Python assigns a reference to the same object (the object which is rightmost) to all the target on the left.

```
>>> a=b=76
>>> print(a)
76
>>> print(b)
76
```

7) Augmented assignment: The augmented assignment is a shorthand assignment that combines an expression and an assignment.

```
>>> x=8
>>> x+=1
>>> print(x)
9
```

There are several other augmented assignment forms: -=, **=, &=, etc.

Home Page - Select or create a n | Untitled11 - Jupyter Notebook

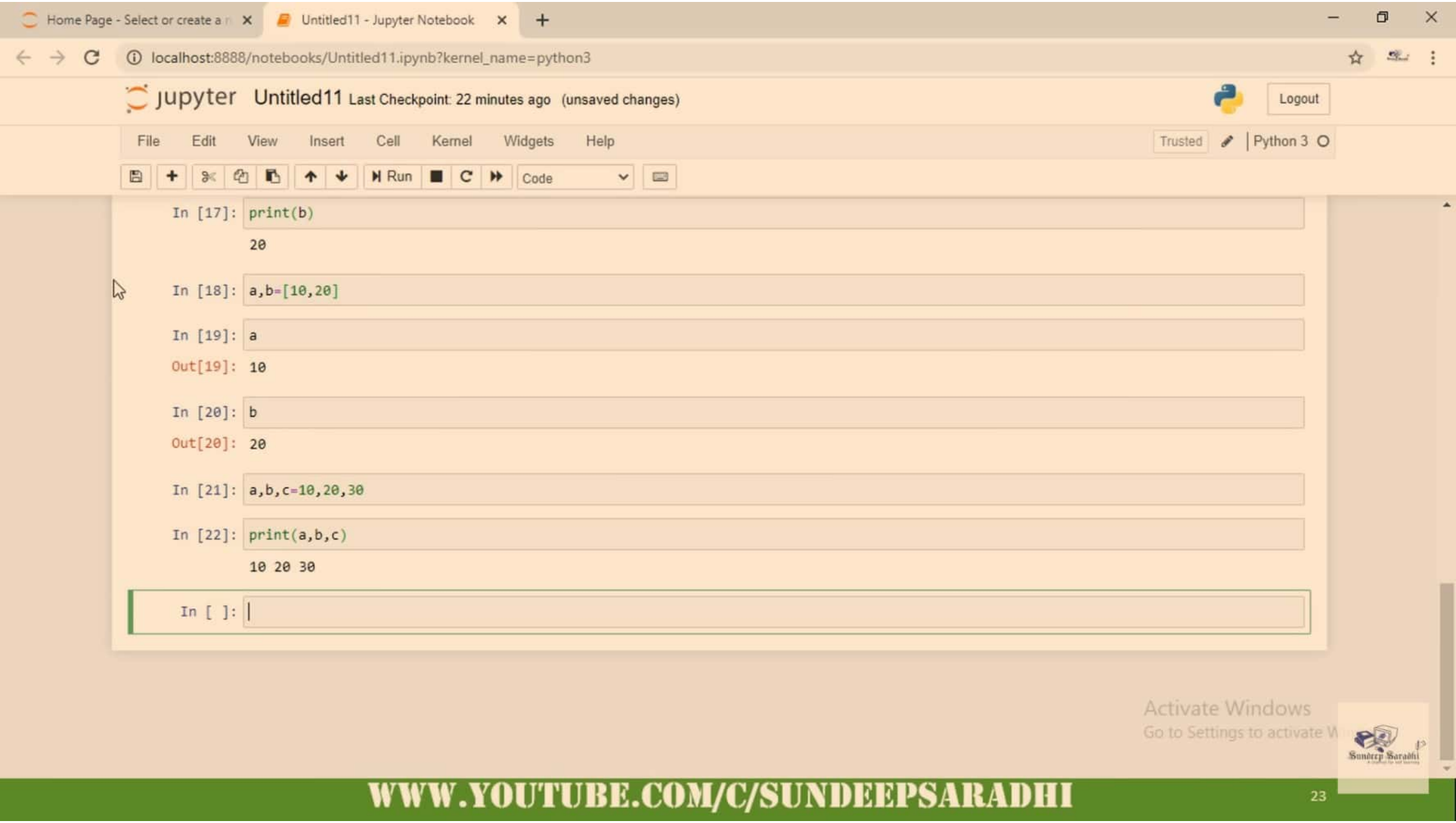
localhost:8888/notebooks/Untitled11.ipynb?kernel_name=python3

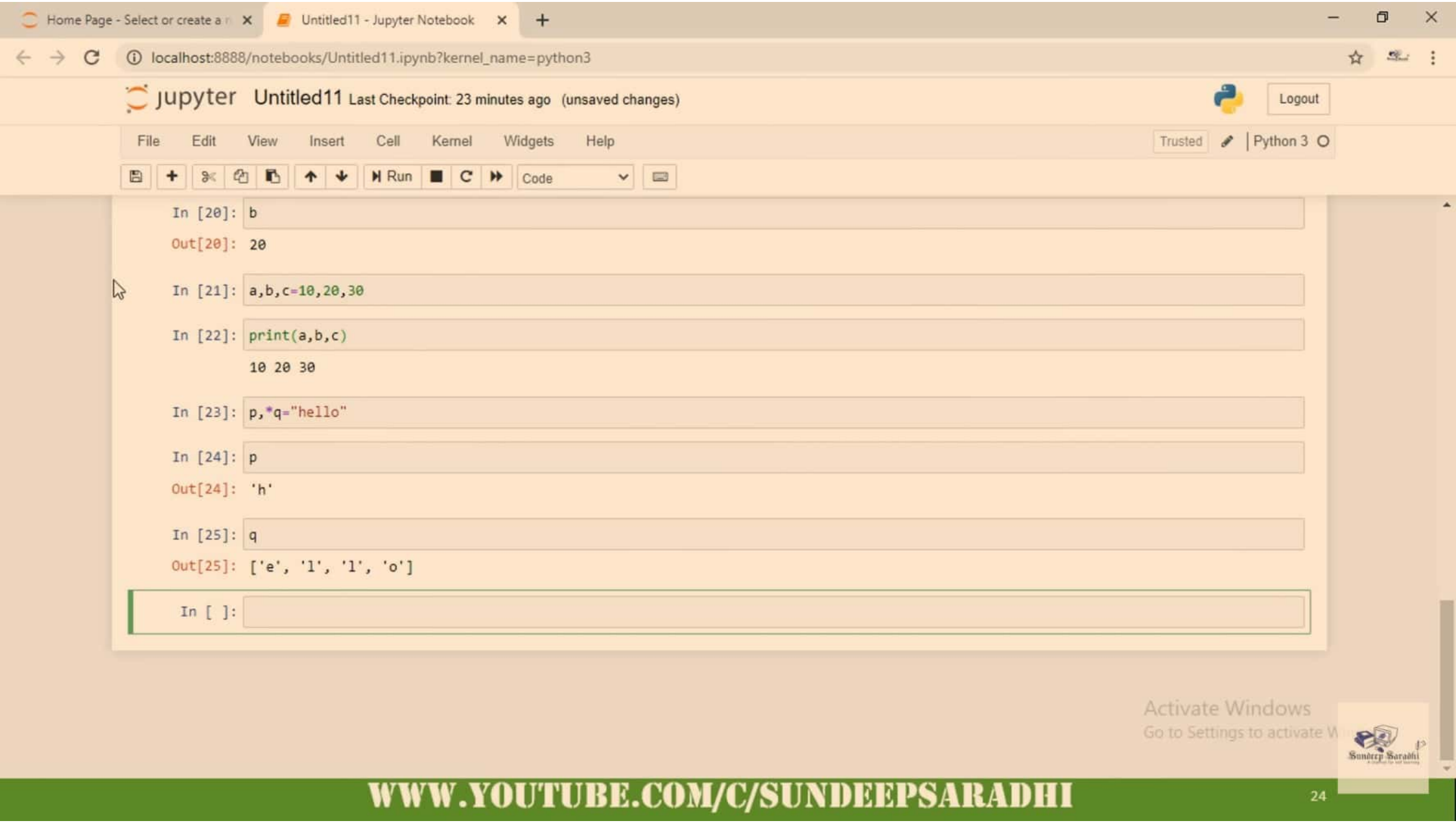
jupyter Untitled11 Last Checkpoint: 20 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
Out[8]: '10'

In [9]: a=5

In [10]: type(a)
Out[10]: int

In [11]: a=5.4

In [12]: type(a)
Out[12]: float

In [13]: a="welcome"

In [14]: type(a)
Out[14]: str

In [ ]:
```





Home Page - Select or create a n | Untitled11 - Jupyter Notebook

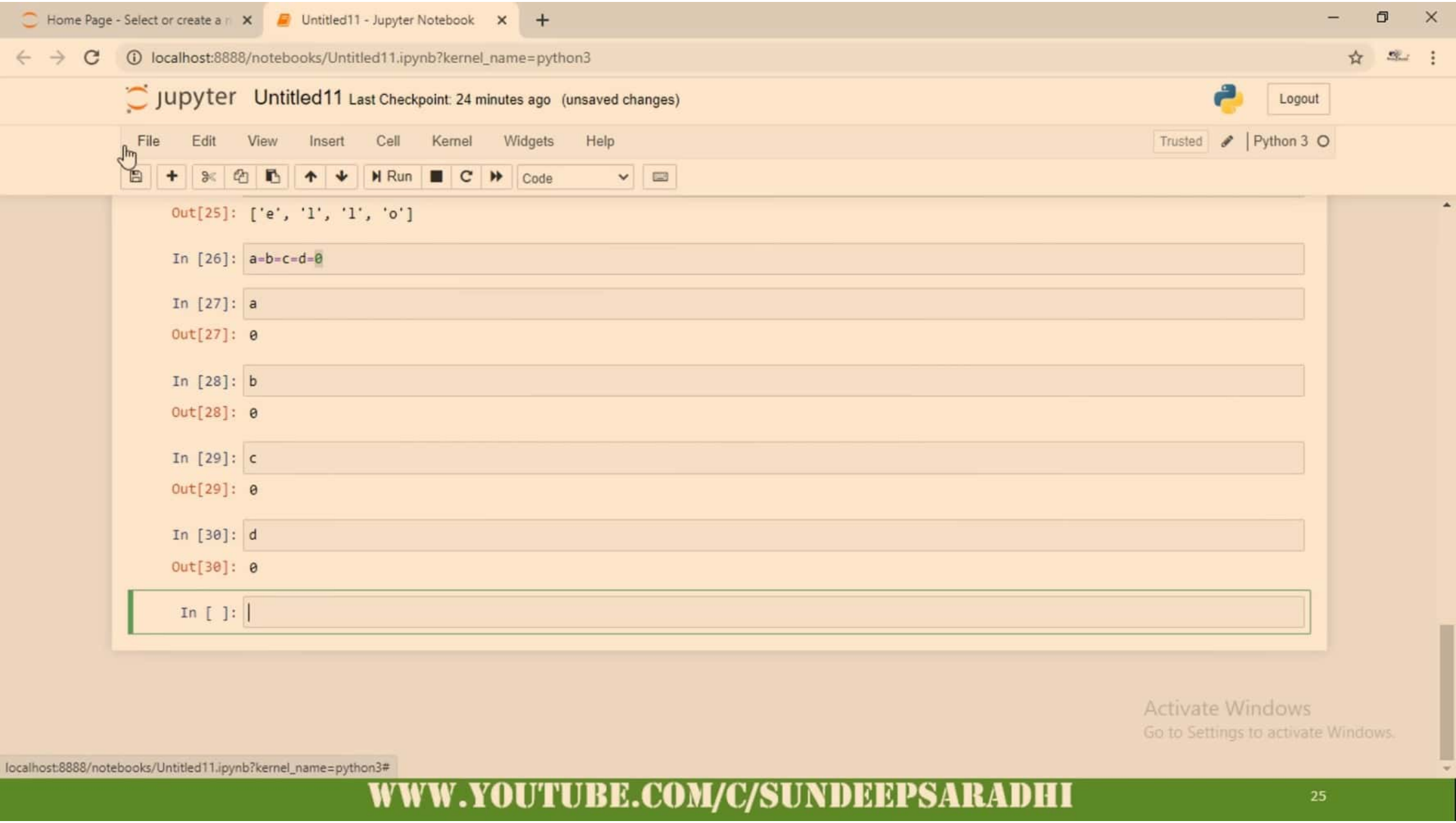
localhost:8888/notebooks/Untitled11.ipynb?kernel_name=python3

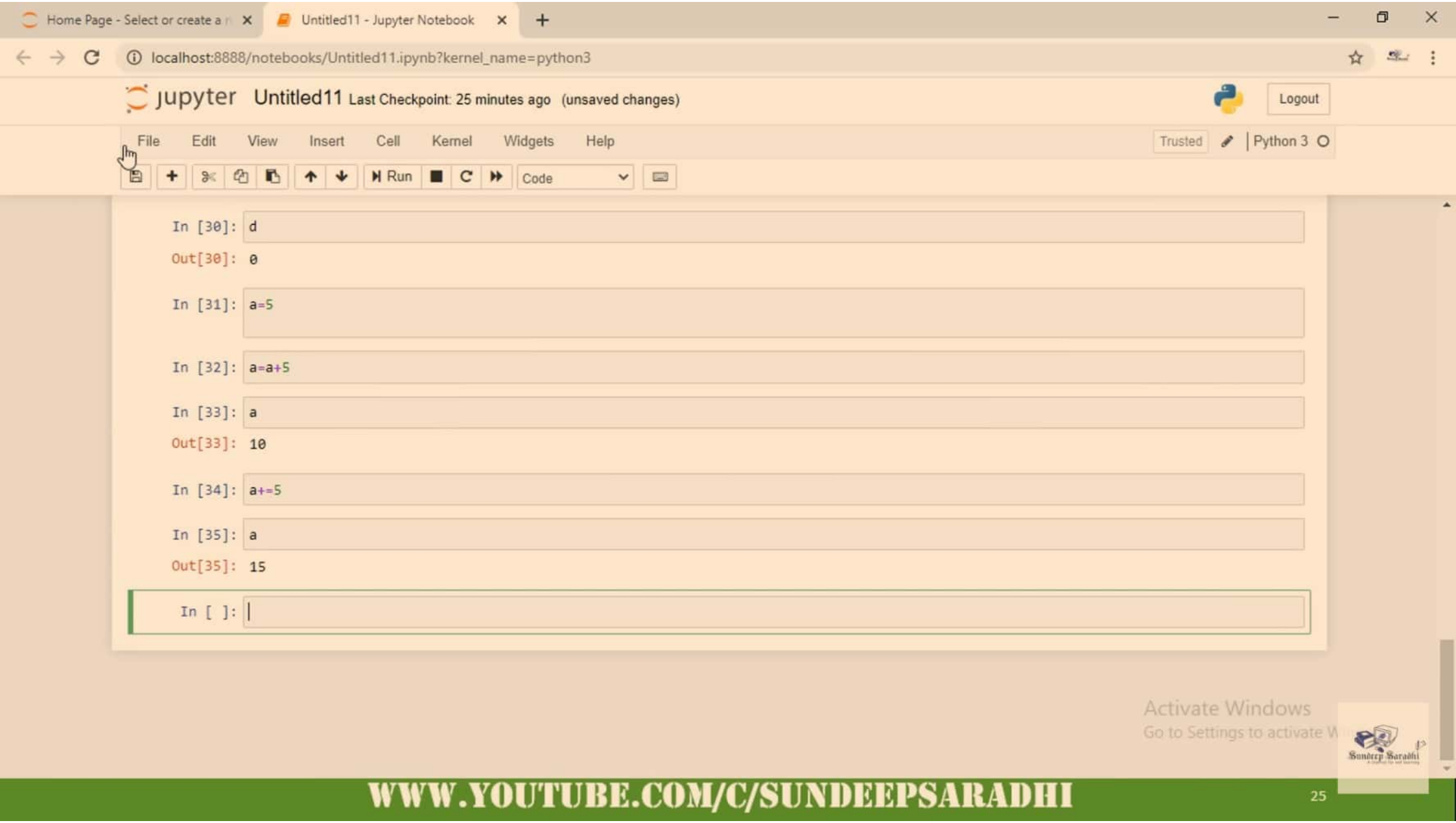
jupyter Untitled11 Last Checkpoint: 23 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
In [20]: b
Out[20]: 20

In [21]: a,b,c=10,20,30

In [22]: print(a,b,c)
         10 20 30

In [23]: p,*q="hello"

In [24]: p
Out[24]: 'h'

In [25]: q
Out[25]: ['e', 'l', 'l', 'o']

In [ ]:
```





Input and Output Functions

Input ( )

- The input( ) function is used to read data from keyboard.
- It reads data into string format.
- Programmer have to convert data into specific format before using them.
- To convert use the int, float and str functions
  - Int(2.3) returns 2, int("123") returns 123
  - Float(6) returns 6.0, float("3.14") returns 3.14
  - Str(123) returns "123" , str(4.5) returns "4.5"
- If the conversion isn't possible, there will be an error.

  Example : int("abc"), int("two")

Example-1 : READING SINGLE INPUT WITHOUT REQUEST MESSAGE

```
a=input()
```

Input: 10

Example-2 : READING SINGLE INPUT WITH REQUEST MESSAGE

```
a=input("Enter the value of a:")
```

Input: Enter the value of a:

10

Example-3 : READING MULTIPLE INPUTS IN A SINGLE LINE USING split( )

```
a=input( ).split(" ")
```

Input : 10 20 30 40 50

print ( )

The print( ) function prints the specified text or value to the screen.

Example-1: print("Hello World")

Output : Hello World

Example-2: ADD A NEW LINE OR VERTICAL SPACE BETWEEN TWO OUTPUTS

```
print("Hello!")
print("Welcome to Python Programming")
```

Output: Hello!

Welcome to Python Programming

Example-3: OPTION KEYWORD ARGUMENT “sep=“

```
print(“Welcome”,”To”,”Python”,”Programming”)
```

Output: Welcome To Python Programming

```
print (“Welcome”,”To”,”Python”,”Programming”,sep=“ \n “)
```

Output : Welcome

To

Python

Programming

```
print (“Welcome”,”To”,”Python”,”Programming”,sep=“ , “)
```

Output : Welcome,TO,Python,Programming

Example-4 : USING KEYWORD ARGUMENT “end=“

- “end= “ is a string appended after the last value, defaults to a new line.
- It allows the programmer to define a custom ending character for each print call other than the default newline or \n.

```
print(“Hello!”,end=“ “)

print(“Welcome to python programming”)
```

Output: Hello! Welcome to python programming

Example-5: USING FORMAT SPECIFIERS

```
a=10

b=2.5

print(“%d is an integer and %f is a float.”%(a,b))
```

Output: 10 is an integer and 2.5 is a float.

Example-6: WITHOUT USING FORMAT SPECIFIERS

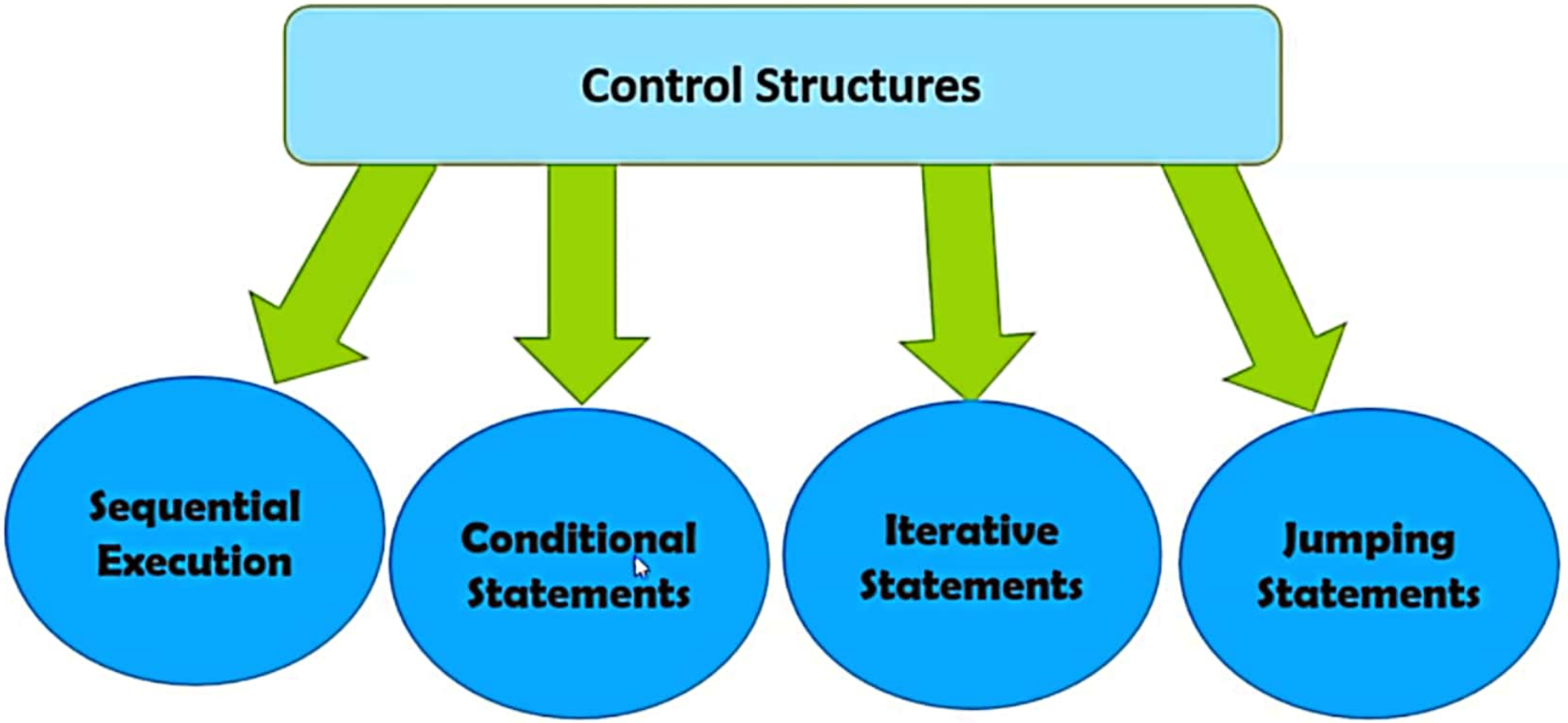
```
a=10

b=2.5

print(a,"is an integer and",b,"is a float.")
```

Output : 10 is an integer and 2.5 is a float.

Operators

Arithmetic Operators	-	+ , - , * , / , // , % , **
Relational Operators	-	< , > , <= , >= , == , !=
Assignment Operators	-	= , += , -= , *= , /= , %= , //= etc.,
Bitwise Operators	-	& , | , ^ , ~ , << , >>
Logical Operators	-	and , or , not
Membership Operators	-	in , not in
Identity Operators	-	is , is not

Control Structures



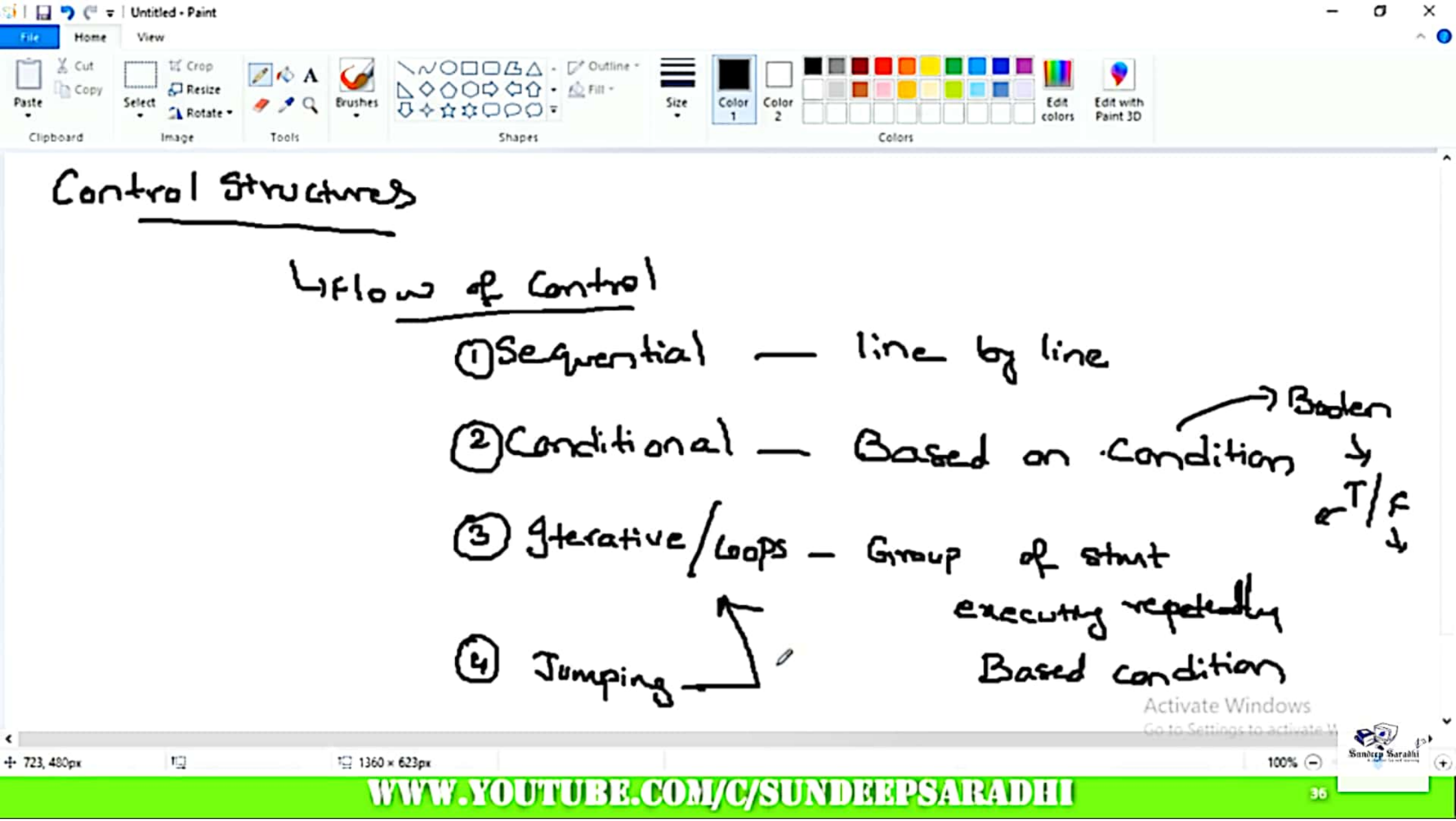
if , if else ,
if elif else , nested if

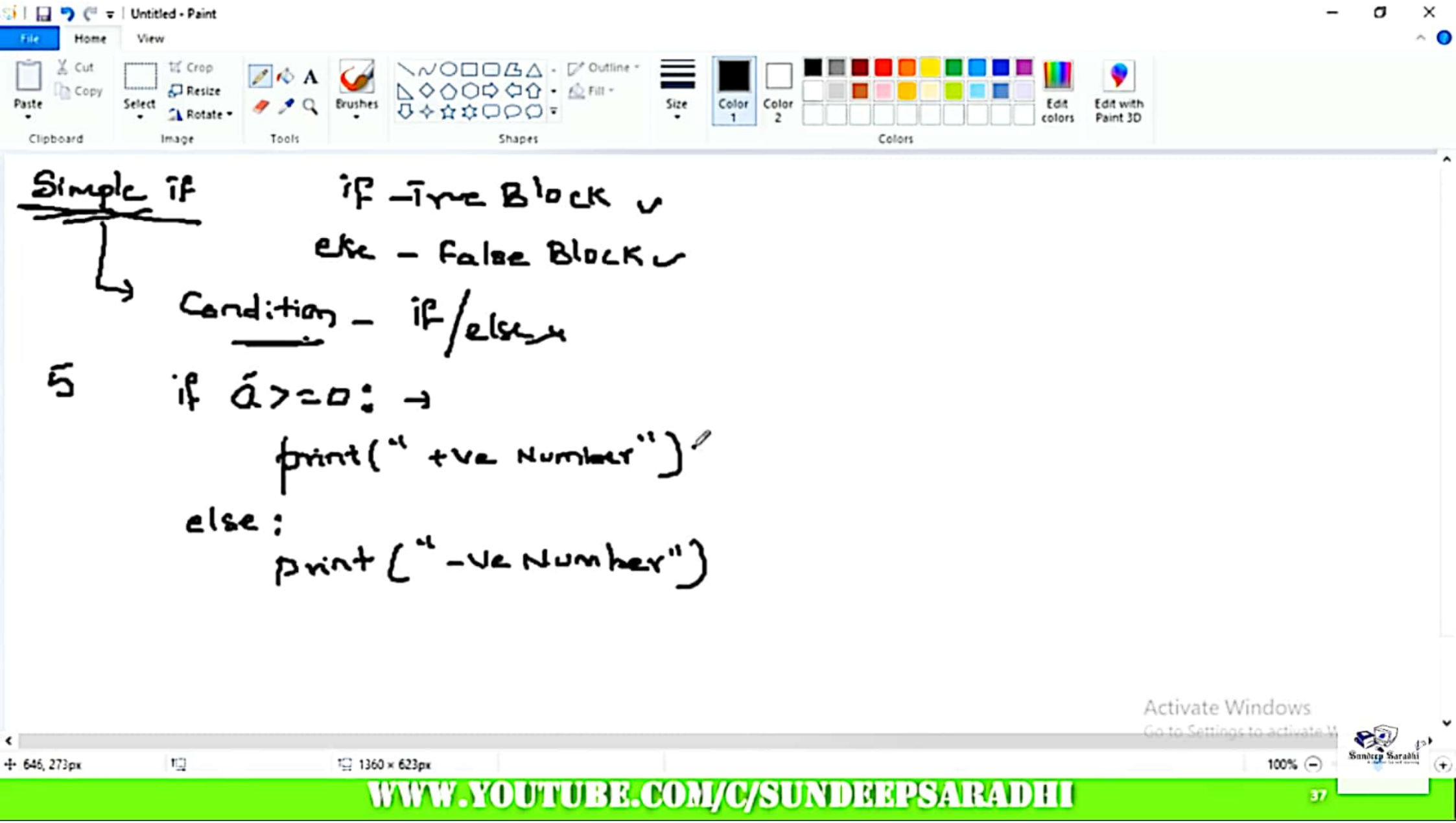
While , for

break , continue , pass

Control Structures

- Conditional Statements
  - if
  - if – else
  - Nested If – else
- Jumping Statements
  - continue
  - break
  - pass
- Python Loops
  - while
  - for
  - Nested loops





In [7]:
```
a=int(input("Enter a:"))
b=int(input("Enter b:"))
if a>b:
    print("a is greater than b")
elif a<b:
    print("a is smaller than b")
else:
    print("Both are equal")
```

```
Enter a:5
Enter b:5
Both are equal
```

Home Page - Select or create a n | Untitled12 - Jupyter Notebook

localhost:8888/notebooks/Untitled12.ipynb?kernel_name=python3

jupyter Untitled12 Last Checkpoint: 8 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

```
Enter a:3
Enter b:5
Both are equal
```

In [10]:
```
a=int(input("Enter a:"))
b=int(input("Enter b:"))
c=int(input("Enter c:"))
if a>b:
    if a>c:
        print("a is Big")
    else:
        print("c is Big")
else:
    if b>c:
        print("b is Big")
    else:
        print("c is Big")
```

```
Enter a:4
Enter b:5
Enter c:3
b is Big
```

In [ ]:

Untitled - Paint

File Home View

Paste Cut Copy | Select Crop Resize Rotate | Brushes | Shapes Outline Fill | Size | Color 1 Color 2 | Edit colors | Edit with Paint 3D

Clipboard Image Tools Shapes Colors

Iterative

Iteration

Continously executed

Condition — True

5 iterations

for loop

while loop

Fake — stop

1. loop variable initialize
2. condition → Loop Terminate
3. loop variable update

while condition :

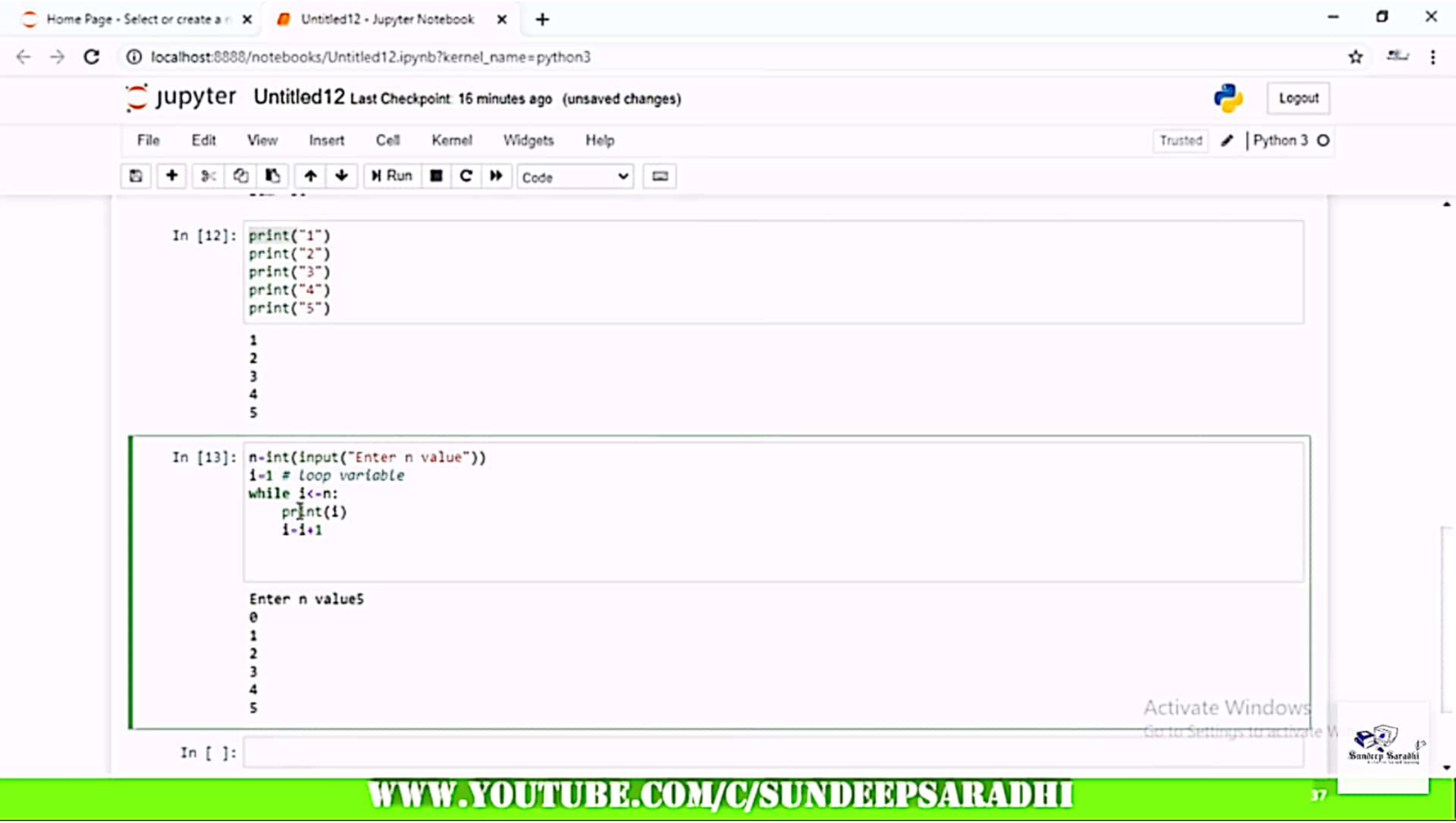
Activate Windows

943, 388px 1360 × 623px 100%

WWW.YOUTUBE.COM/C/SUNDEEPSARADHI

37

Scanned with CamScanner



Home Page - Select or create a | Untitled12 - Jupyter Notebook

localhost:8888/notebooks/Untitled12.ipynb?kernel_name=python3

jupyter Untitled12 Last Checkpoint: 17 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

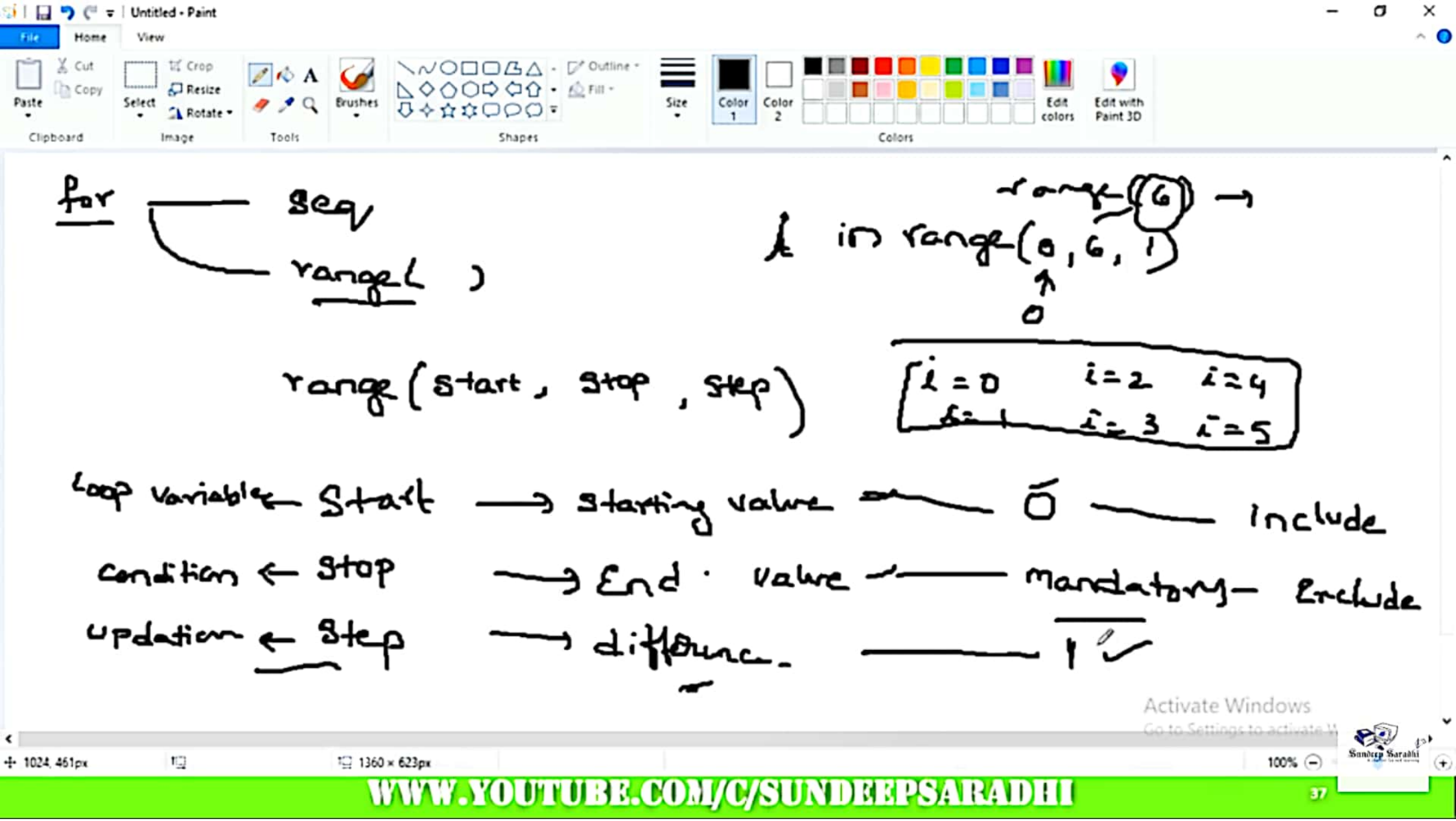
In [12]:
```
print("1")
print("2")
print("3")
print("4")
print("5")
```

```
1
2
3
4
5
```

In [16]:
```
n=int(input("Enter n value"))
i=1 # loop variable
while i<=n:
    print(i,end=" ")
    i=i+1
```

```
Enter n value100
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 4
6 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88
89 90 91 92 93 94 95 96 97 98 99 100
```

In [ ]:



Home Page - Select or create a | Untitled12 - Jupyter Notebook

localhost:8888/notebooks/Untitled12.ipynb?kernel_name=python3

jupyter Untitled12 Last Checkpoint: 23 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

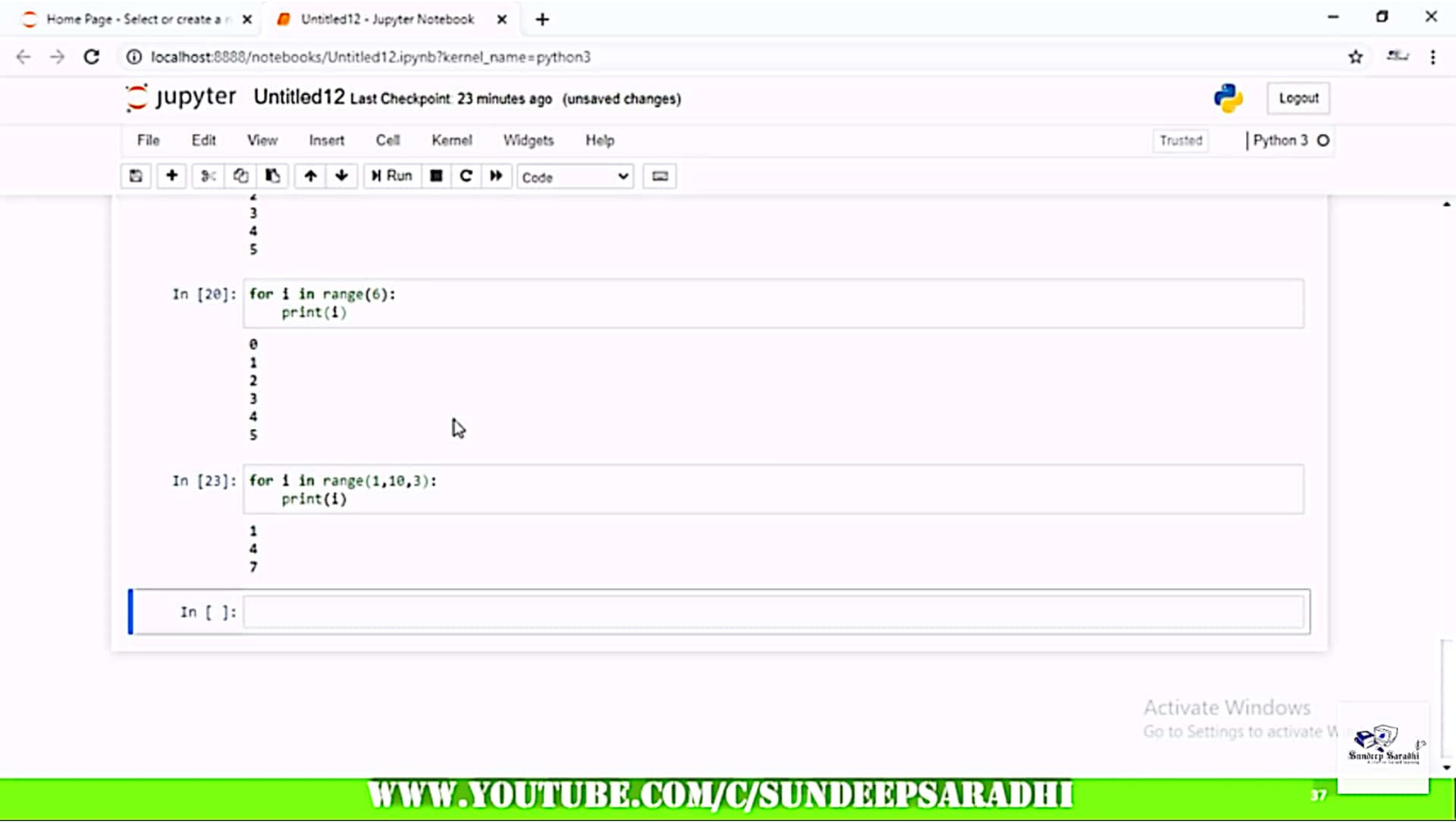
In [18]:
```
for i in range(0,6,1):
    print(i)
```

```
0
1
2
3
4
5
```

In [20]:
```
for i in range(6):
    print(i)
```

```
0
1
2
3
4
5
```

In [ ]:
```
for i in range(1,10,2):
    print(i)
```



5) Extended Sequence unpacking: It allows us to be more flexible in how we select portions of a sequence to assign.

```
>>> p, *q="Hello"
```

Here, p is matched with the first character in the string on the right and q with the rest. The starred name (*q) is assigned a list, which collects all items in the sequence not assigned to other names.

```
>>> p
'H'
>>> q
['e', 'l', 'l', 'o']
```

- **Numeric**
  - Integer - Whole Numbers Ex: 3, 400, 500
  - Complex Number – Numbers with real and imaginary values , Ex: a+bj
  - Float - Numbers with decimal point , Ex: 3.14, 4.15
- **Dictionary** - Unordered Key: Value pairs { key1 : value1, key2 : value2…. }
- **Boolean** - True or False
- **Set** - Unordered Collection of unique object , Ex: {10, 20, 30}
- **Sequence Type**
  - Strings - Ordered sequence of characters , Ex: “python”, “sundeep”
  - List - Ordered sequence of object , Ex: [10 , 20 , 30]
  - Tuple - Ordered immutable sequence of objects , Ex: (10, 20, 30)

Home Page - Select or create a | Untitled12 - Jupyter Notebook

localhost:8888/notebooks/Untitled12.ipynb?kernel_name=python3

jupyter Untitled12 Last Checkpoint: 26 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

```
4
5
```

In [25]:
```
for i in range(1,100,3):
    print(i,end=" ")
```

```
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97
```

In [26]:
```
str="PYTHON"
for letter in str:
    print(letter)
```

```
P
Y
T
H
O
N
```

In [27]:
```
l=[10,20,30,40,50]
```

In [ ]:
```
for ele in l:
    print(ele)
```

Activate Windows
Go to Settings to activate W

Sandeep Saradhi

Home Page - Select or create a | Untitled12 - Jupyter Notebook

localhost:8888/notebooks/Untitled12.ipynb?kernel_name=python3

jupyter Untitled12 Last Checkpoint: 39 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [28]:
```
for ele in l:
    print(ele)
```

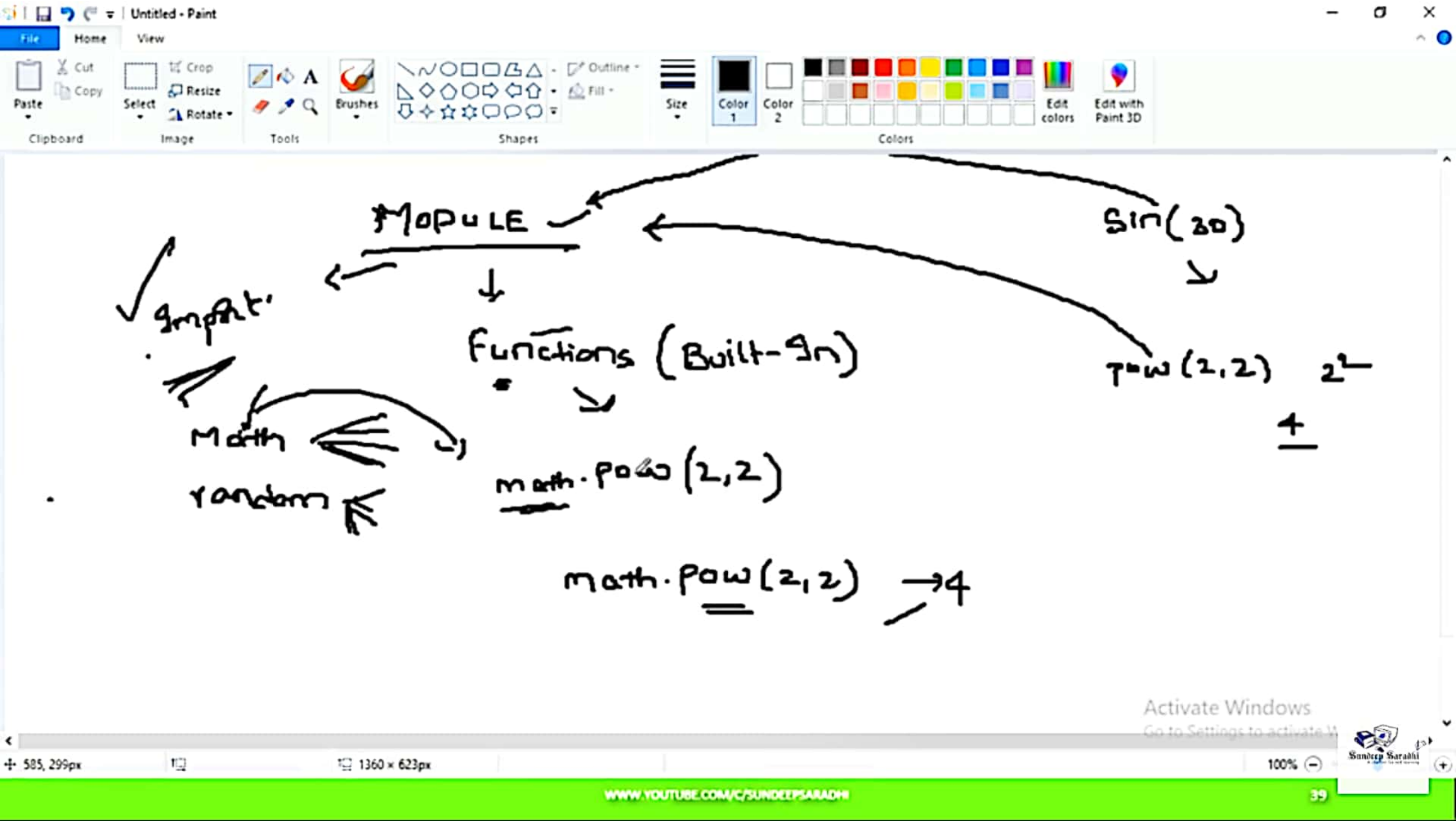
```
10
20
30
40
50
```

In [43]:
```
for i in range(5): # i = 0 to 4
    for j in range(5): # j = 0 to 4
        print("*",end=" ")
    print(" ")
```

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

In [ ]:
```
for i in range(10):
    if i==5:
        break
    else:
        print(i)
```

Math & Random Modules



Mathematical Functions

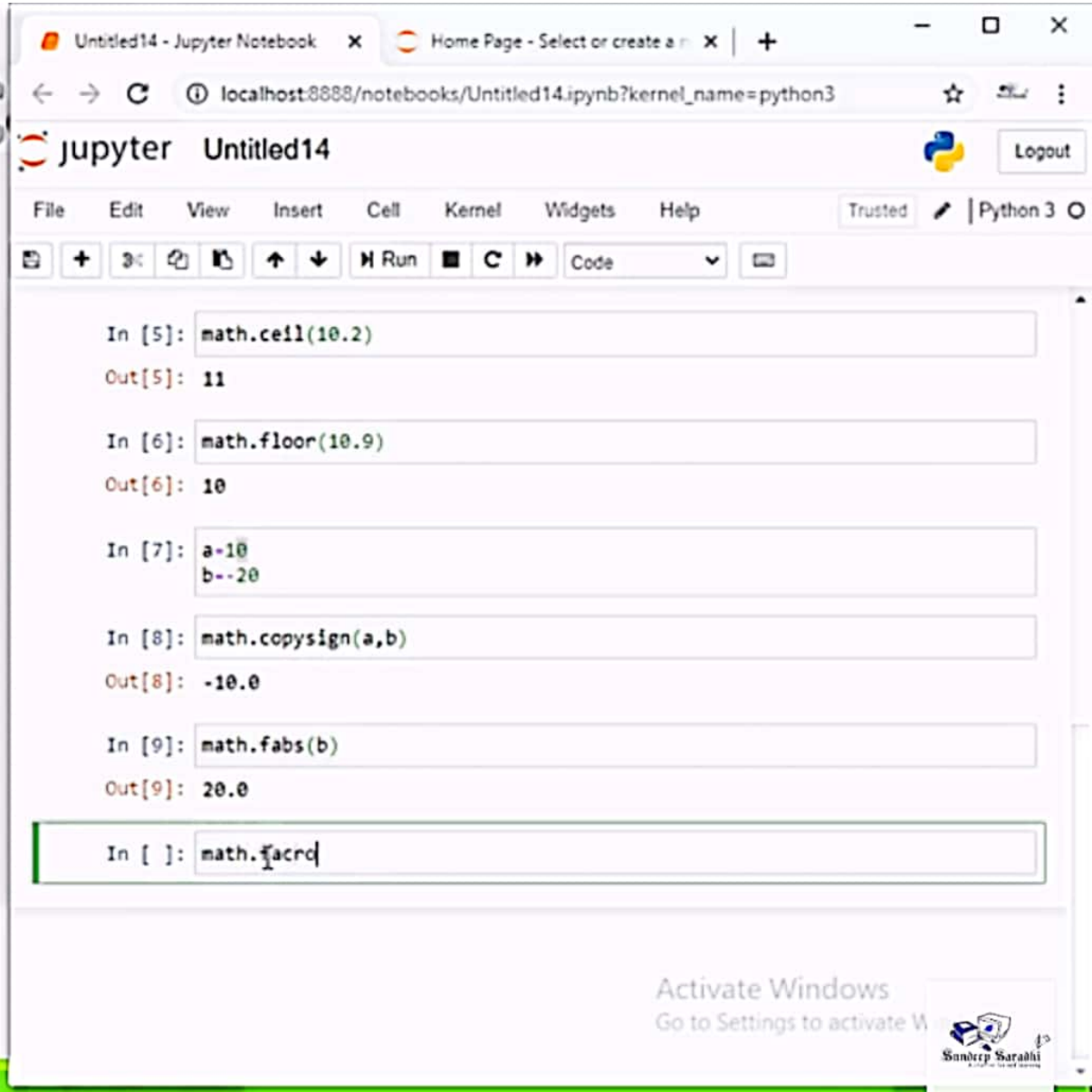
- ceil(x)
- copysign(x,y)
- fabs(x)
- factorial(x)
- floor(x)
- fsum(iterable)
- gcd(x,y)
- pow(x,y)
- sqrt(x)
- sin(x)
- cos(x)
- tan(x)
- pi
- e
- tau
- inf
- nan

Random Functions

- choice(seq)
- randrange(start , stop , step)
- random( )
- shuffle(list)
- uniform(x , y)

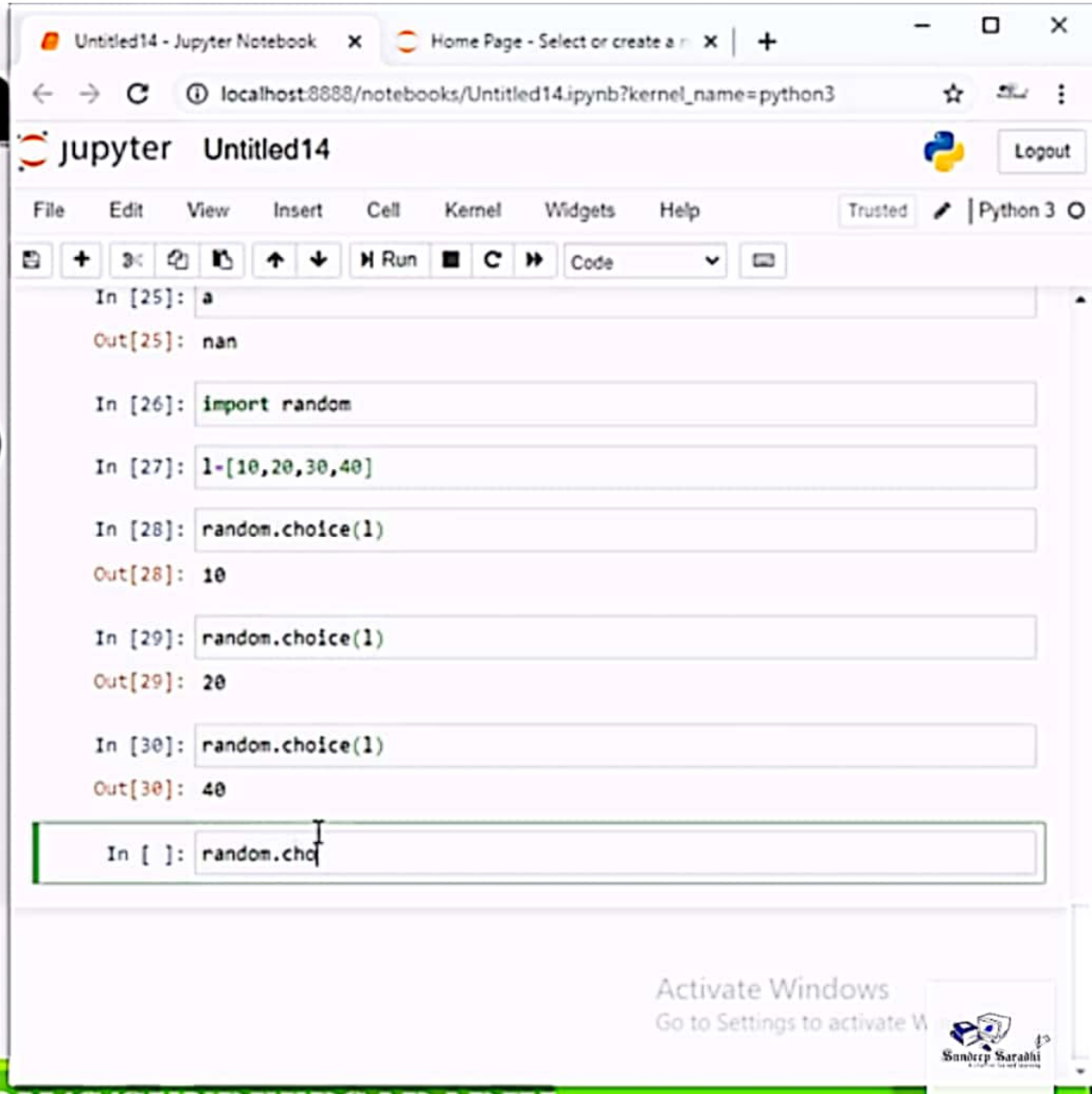
Mathematic

- ceil(x)
- copysign(x,y)
- fabs(x)
- factorial(x)
- floor(x)
- fsum(iterable)
- gcd(x,y)
- pow(x,y)
- sqrt(x)



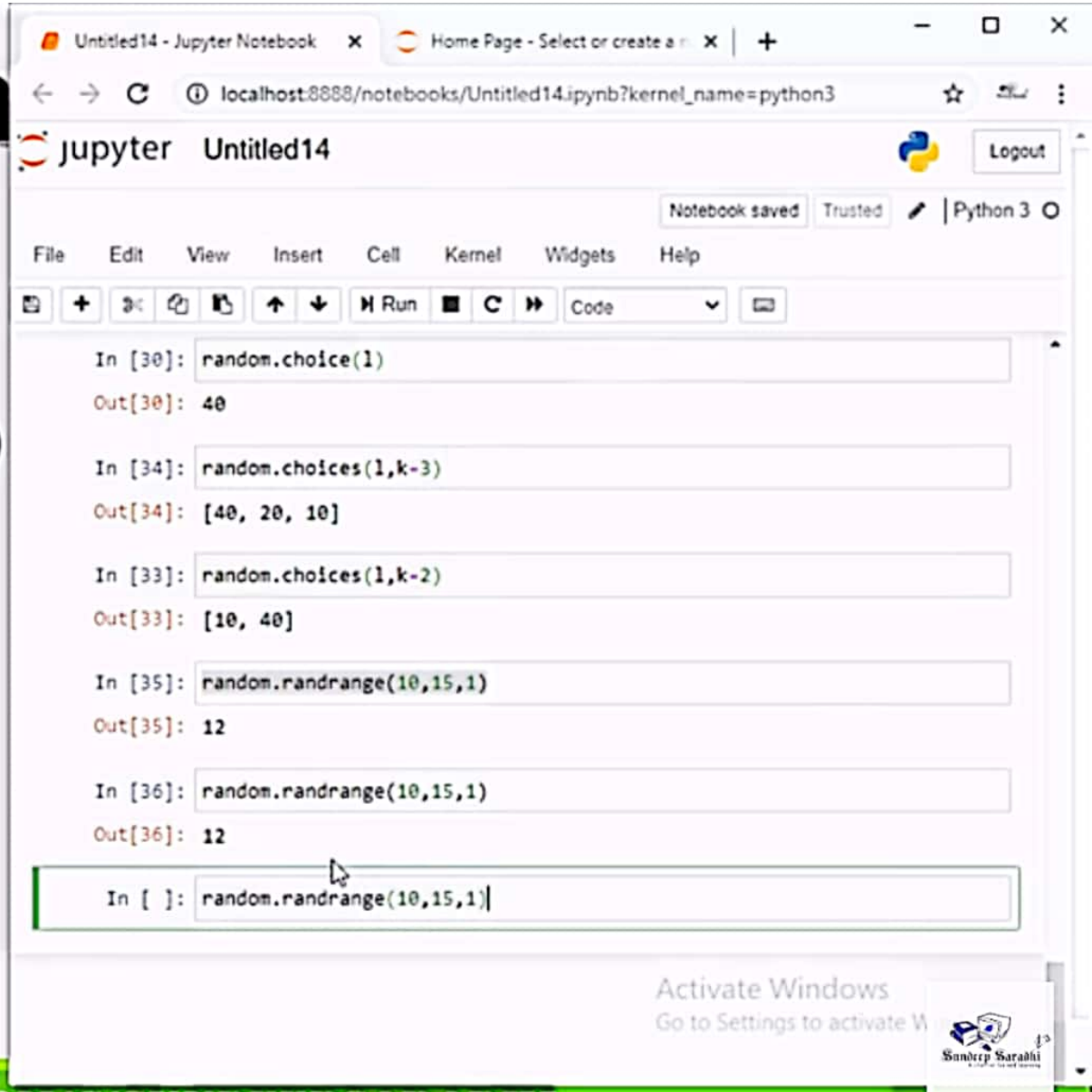
Random

- choice(seq)
- randrange(start , stop , step)
- random( )
- shuffle(list)
- uniform(x , y)



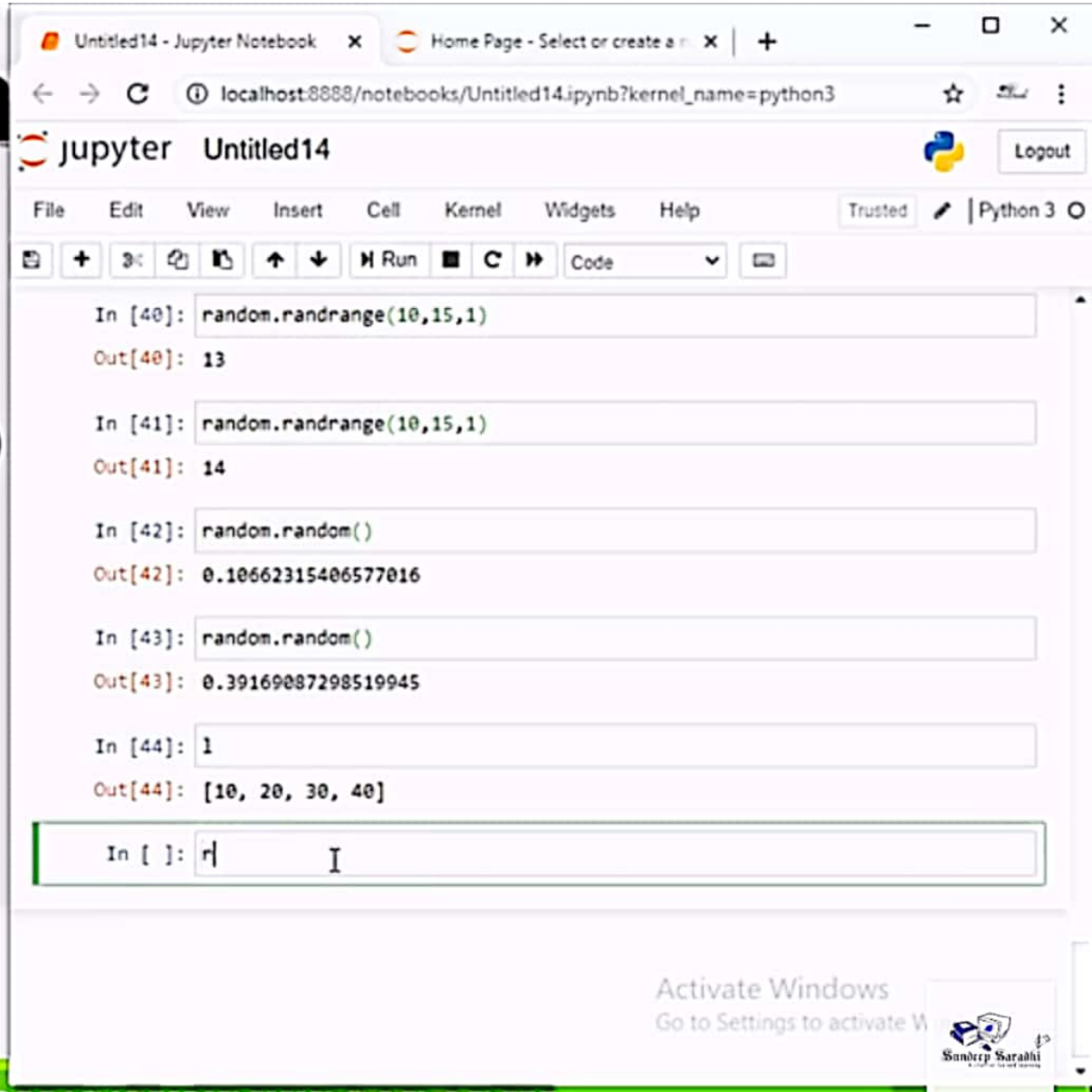
Random

- choice(seq)
- randrange(start , stop , step)
- random( )
- shuffle(list)
- uniform(x , y)



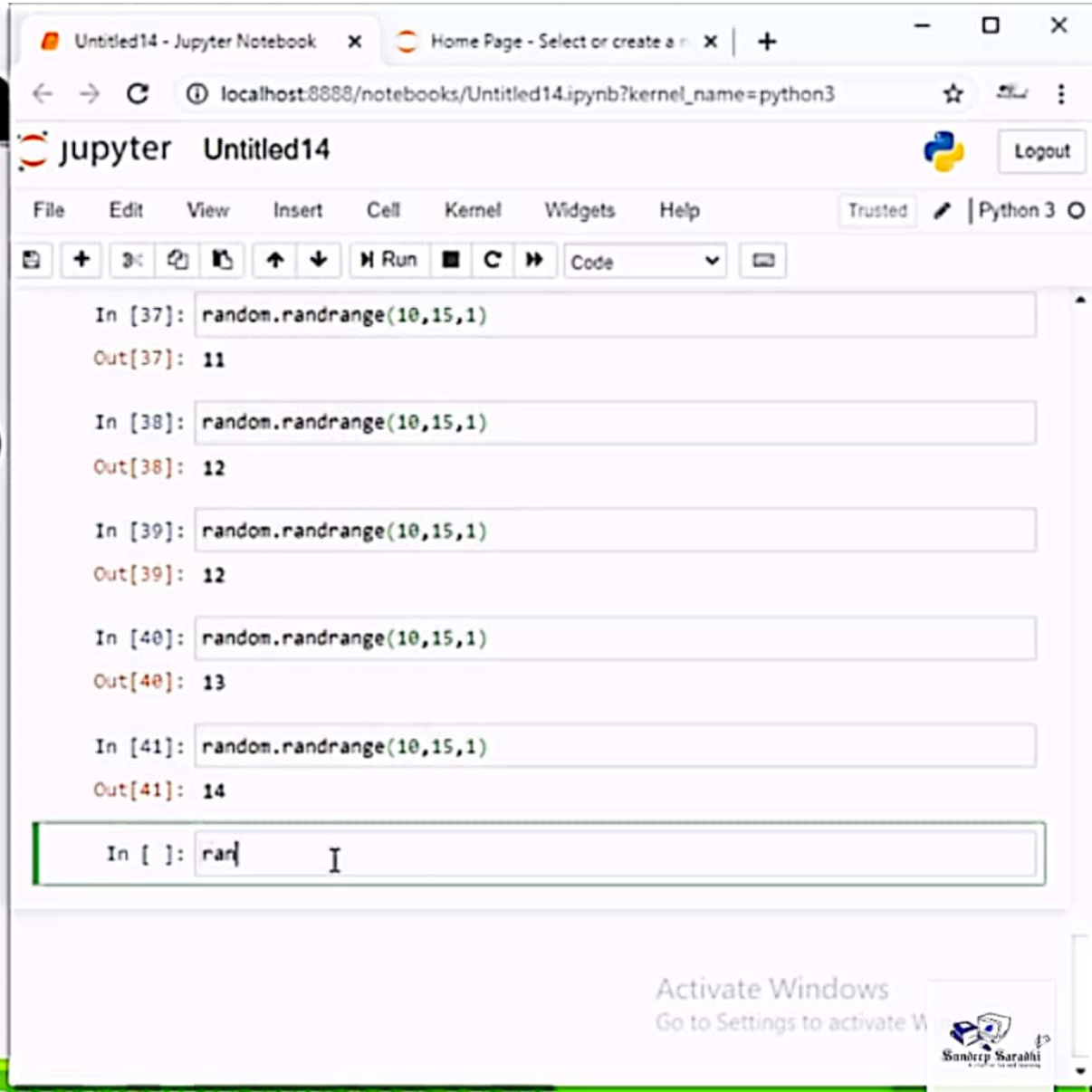
Random

- choice(seq)
- randrange(start , stop , step)
- random( )
- shuffle(list)
- uniform(x , y)



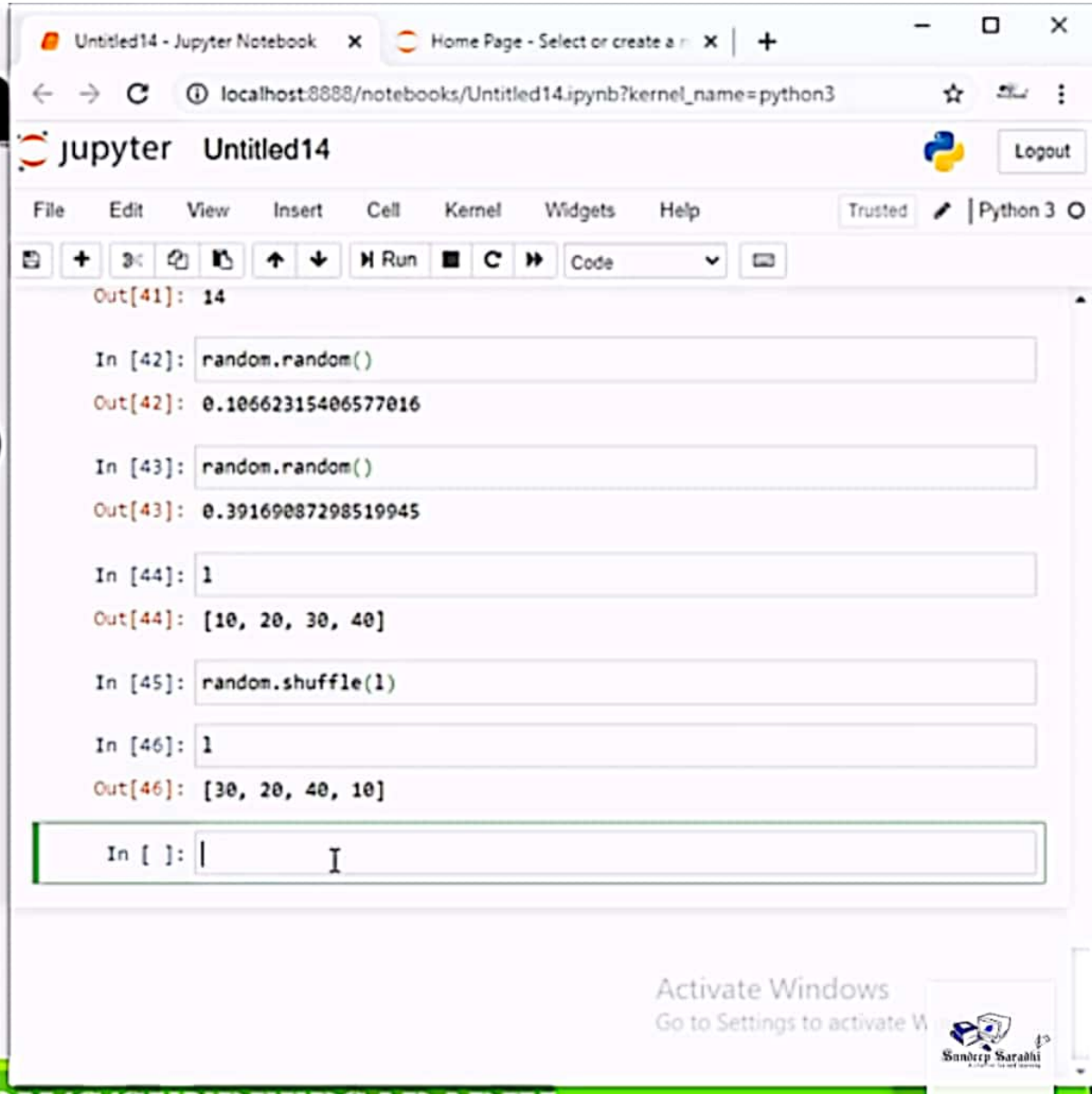
Random

- choice(seq)
- randrange(start , stop , step)
- random( )
- shuffle(list)
- uniform(x , y)



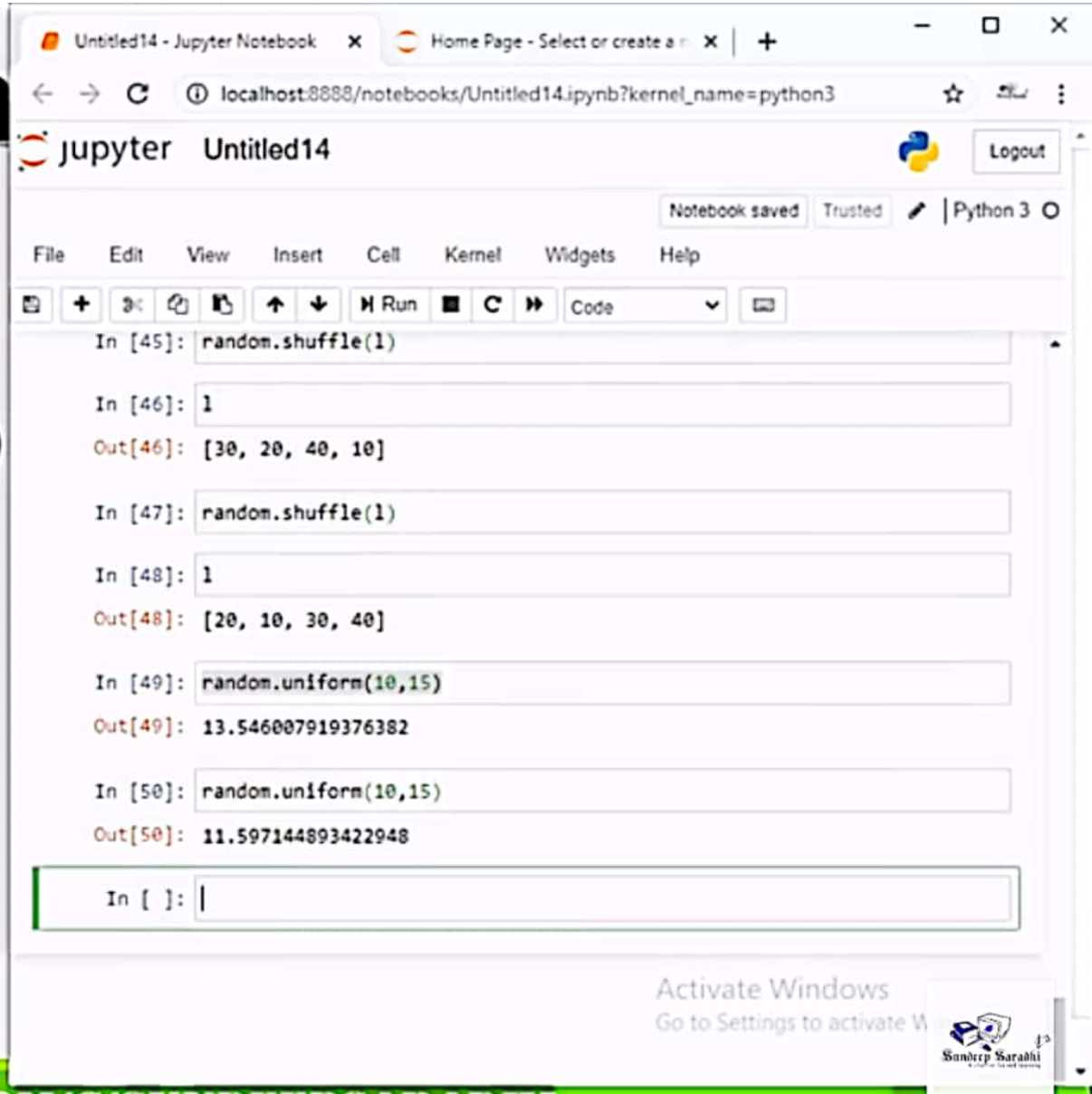
Random

- choice(seq)
- randrange(start , stop , step)
- random( )
- shuffle(list)
- uniform(x , y)

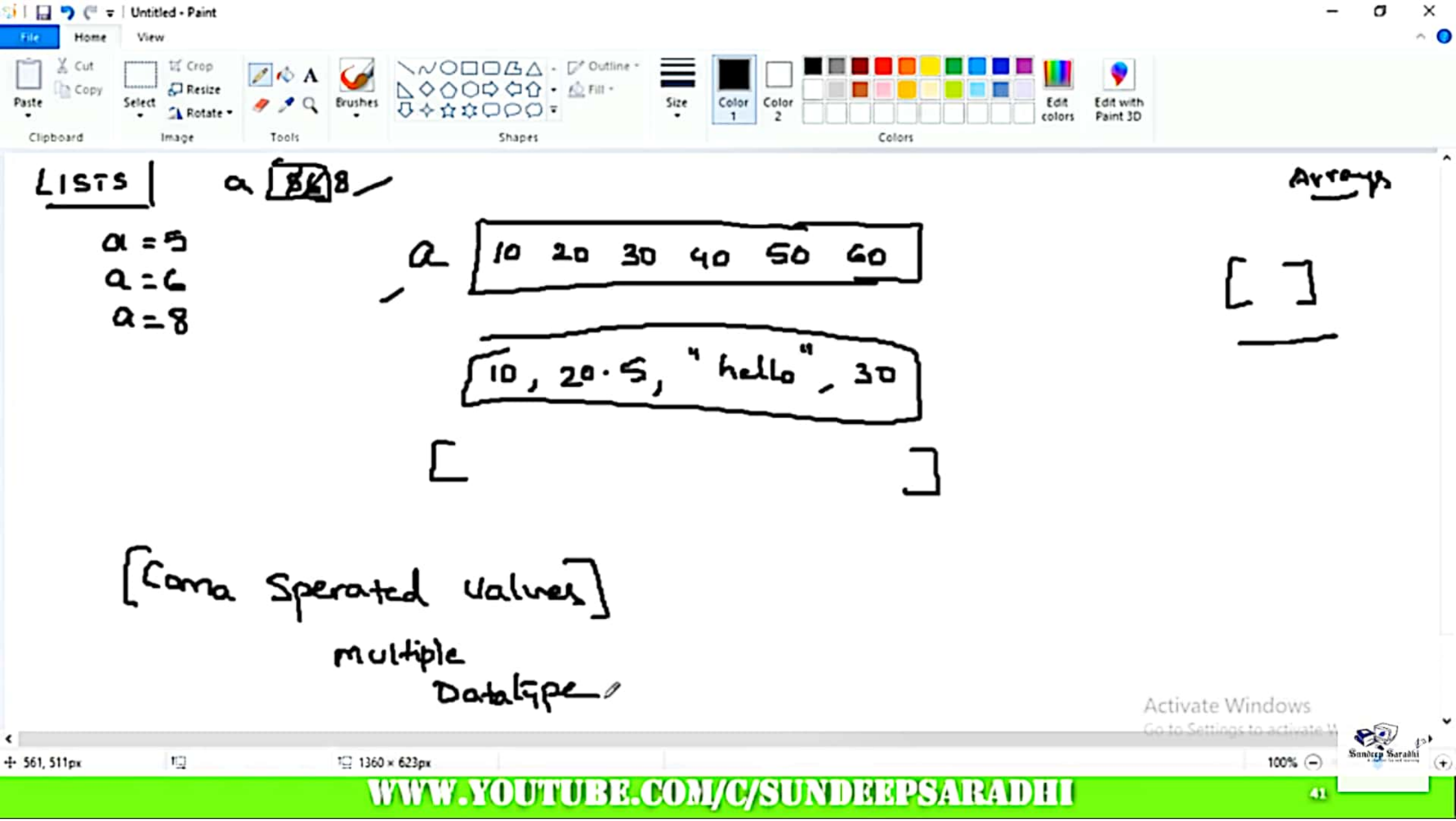


Random

- choice(seq)
- randrange(start , stop , step)
- random( )
- shuffle(list)
- uniform(x , y)



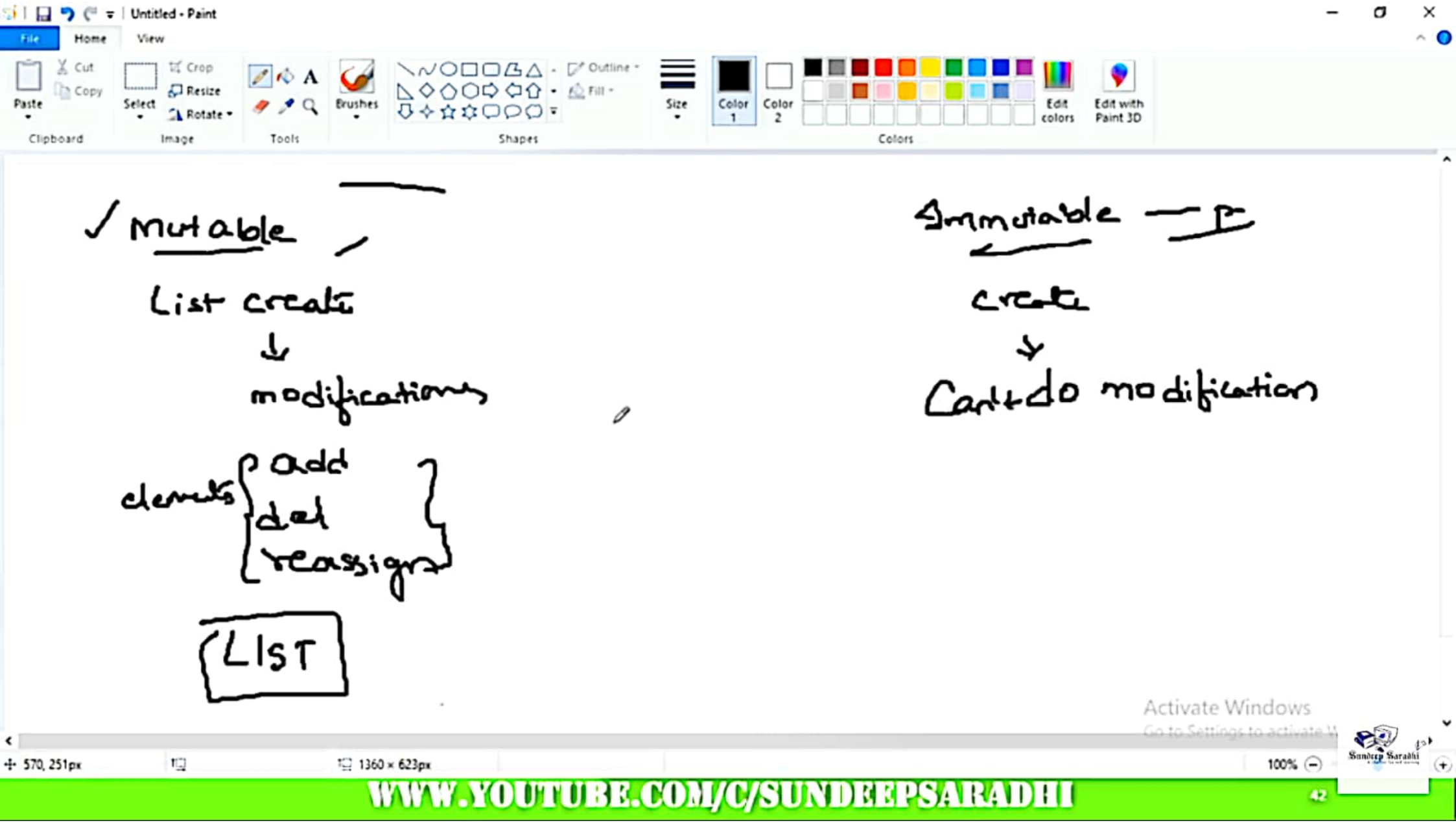
Lists

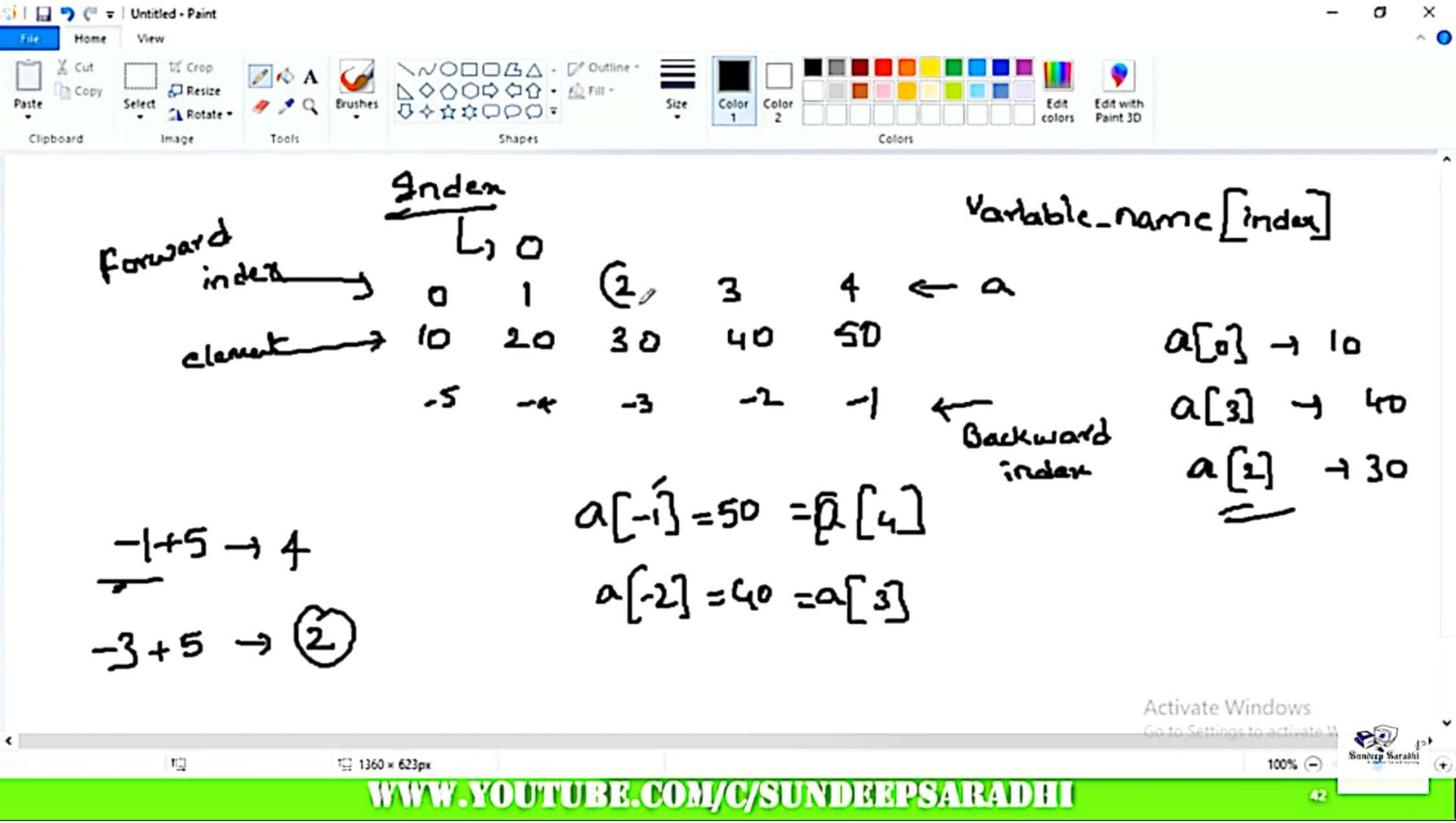


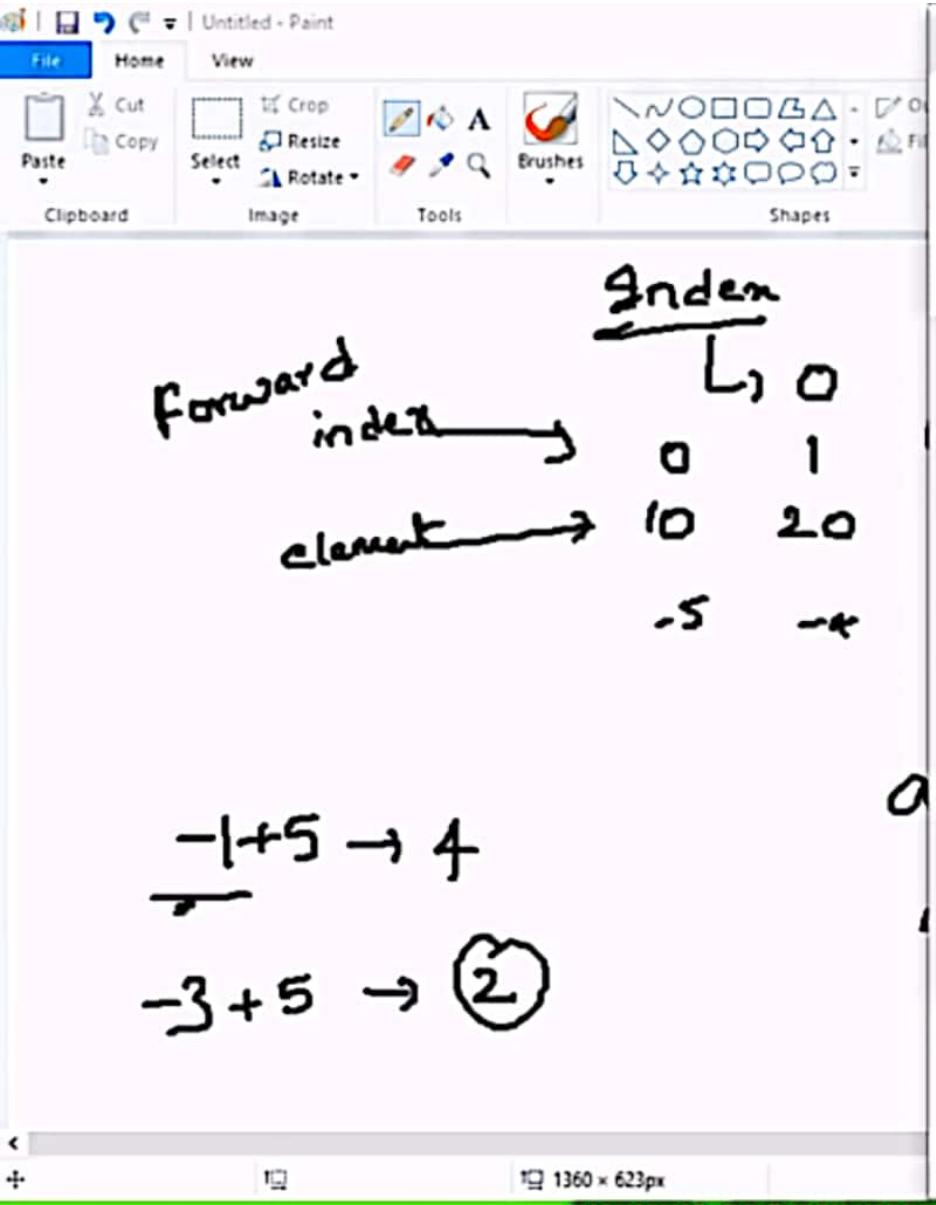
- Creating Lists
- Accessing elements from Lists
- Slicing
- Reassigning List elements
- Deleting elements
- Multidimensional Lists
- Basic Operations

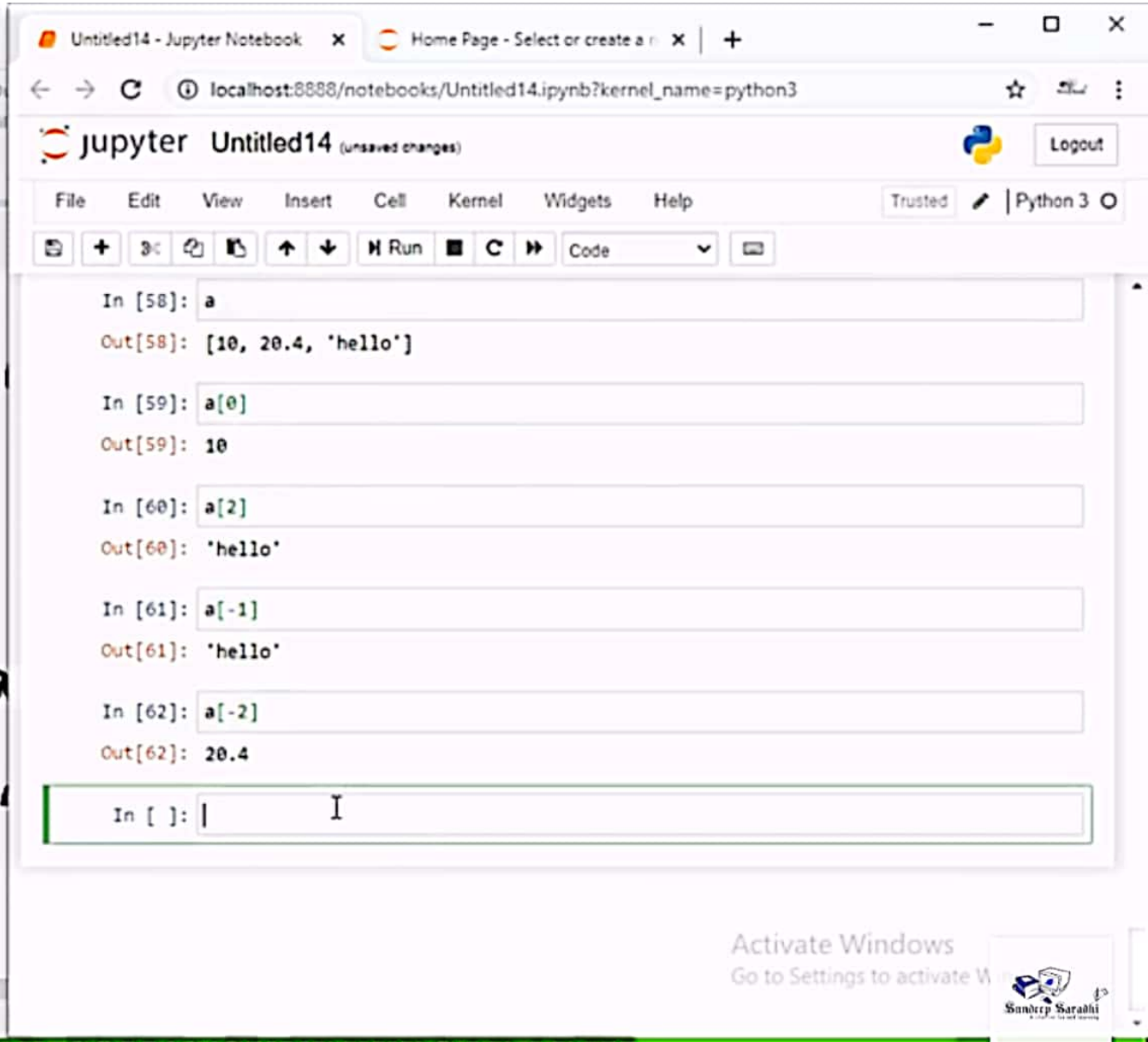
  + , * , len , min , max , sum , membership , iterations
- List Comprehension
- Built-In Methods

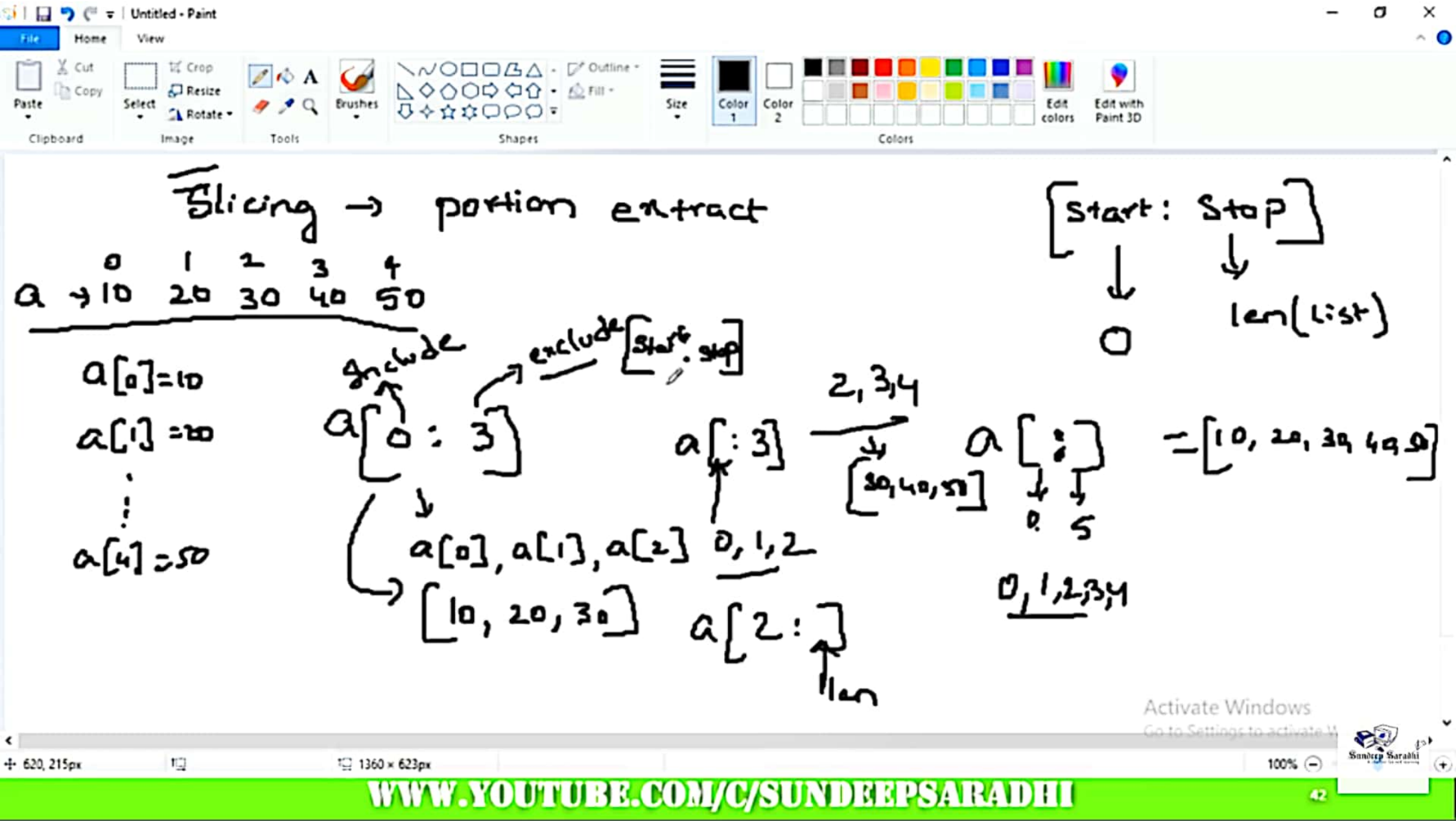
  append , extend , insert , remove , pop , sort , count , index , reverse







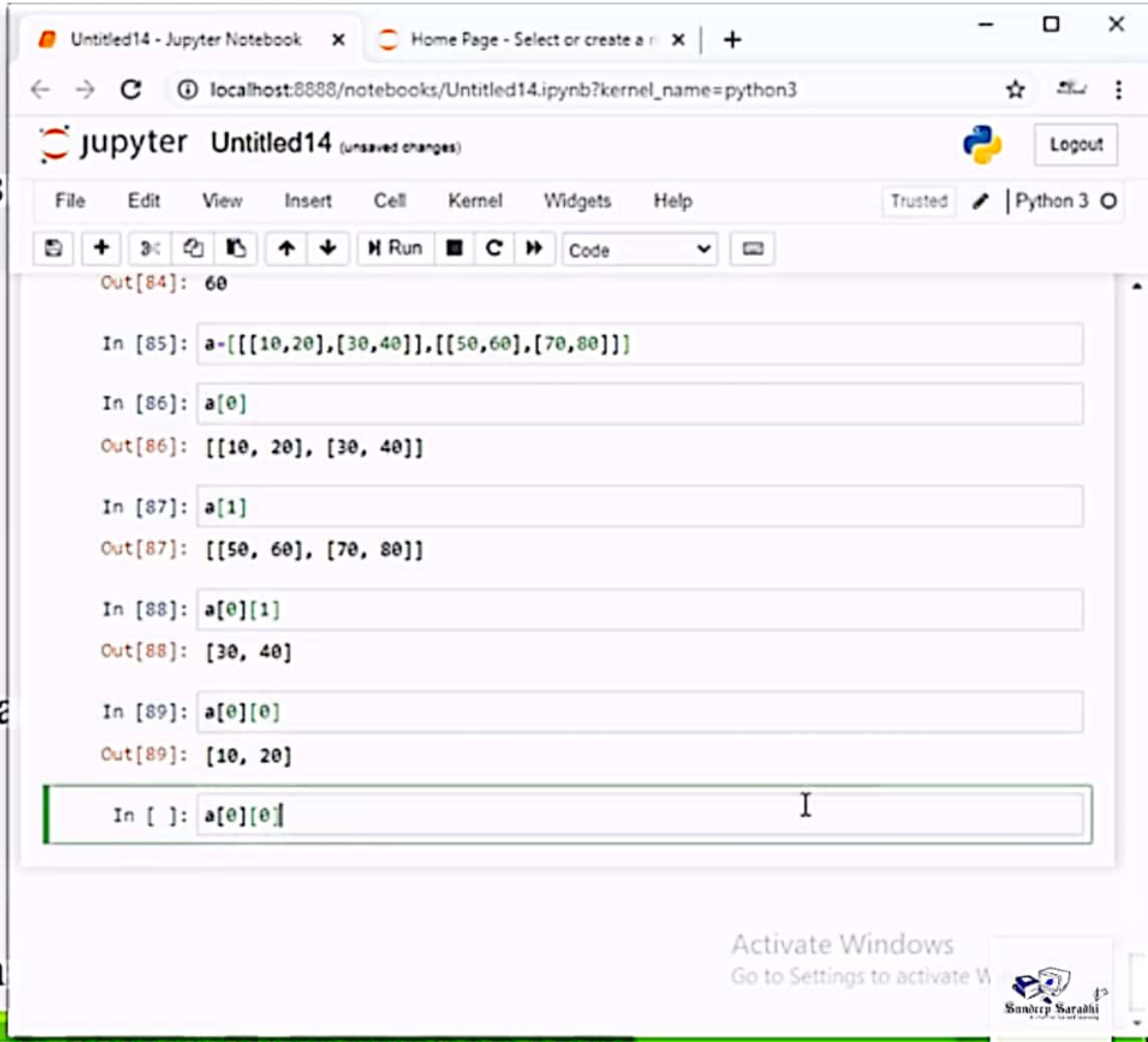




- ❑ Creating Lists
- ❑ Accessing elements from Lists
- ❑ Slicing
- ❑ Reassigning List elements
- ❑ Deleting elements
- ❑ Multidimensional Lists
- ❑ Basic Operations

  + , * , len , min , ma
- ❑ List Comprehension
- ❑ Built-In Methods

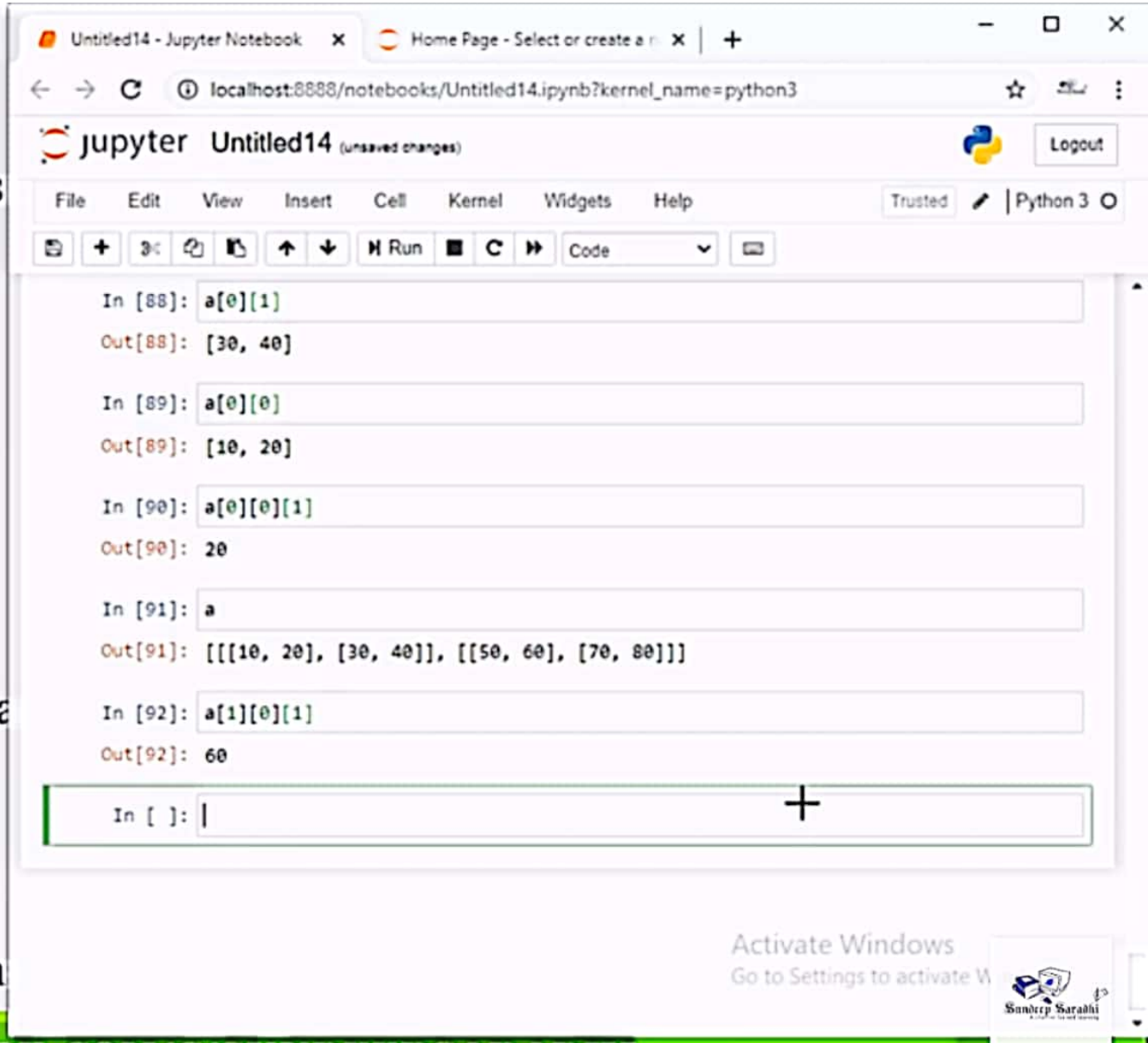
  append , extend , in



- ❑ Creating Lists
- ❑ Accessing elements from Lists
- ❑ Slicing
- ❑ Reassigning List elements
- ❑ Deleting elements
- ❑ Multidimensional Lists
- ❑ Basic Operations

  + , * , len , min , ma
- ❑ List Comprehension
- ❑ Built-In Methods

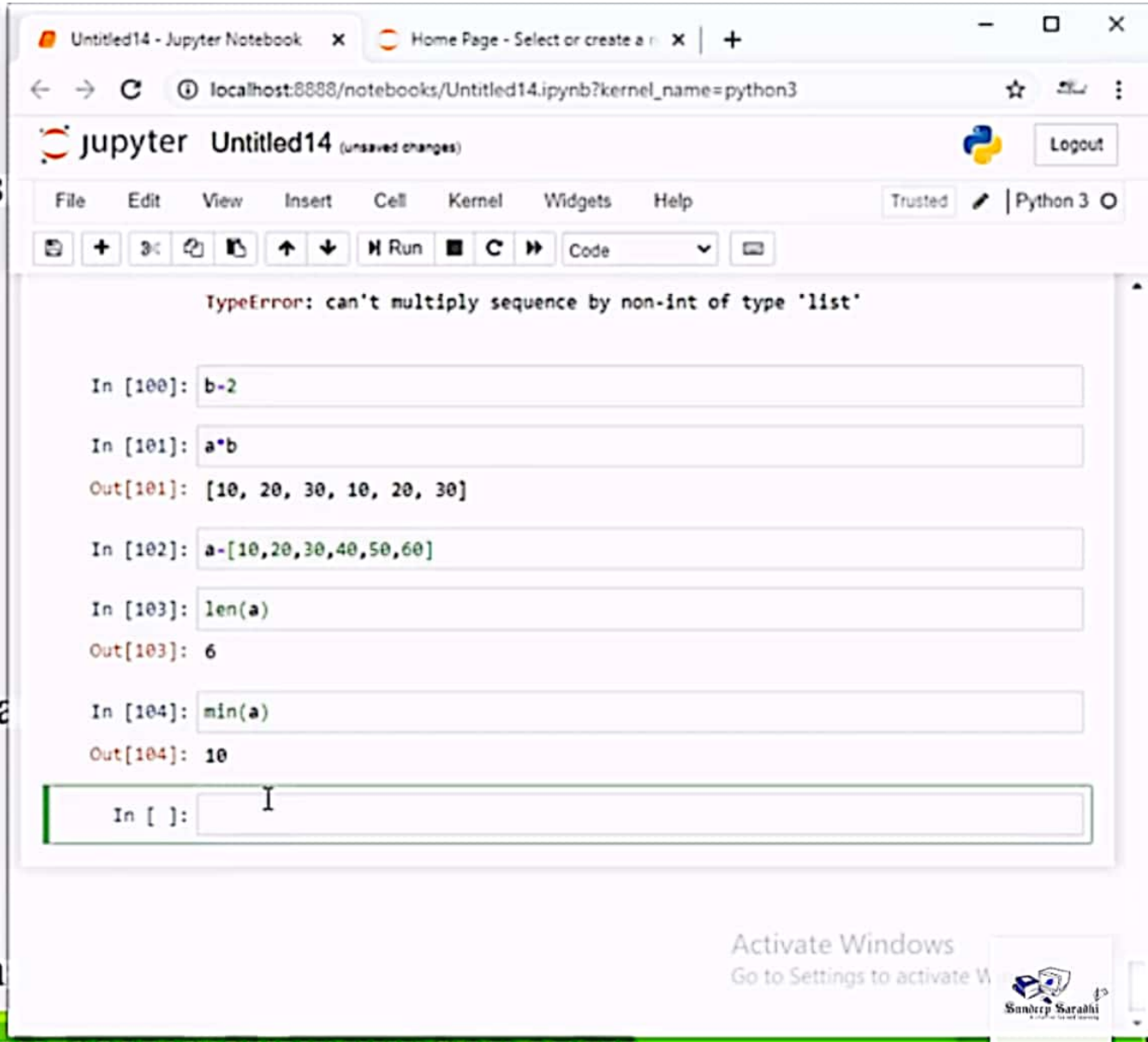
  append , extend , in



- ❑ Creating Lists
- ❑ Accessing elements from Lists
- ❑ Slicing
- ❑ Reassigning List elements
- ❑ Deleting elements
- ❑ Multidimensional Lists
- ❑ Basic Operations

  + , * , len , min , ma
- ❑ List Comprehension
- ❑ Built-In Methods

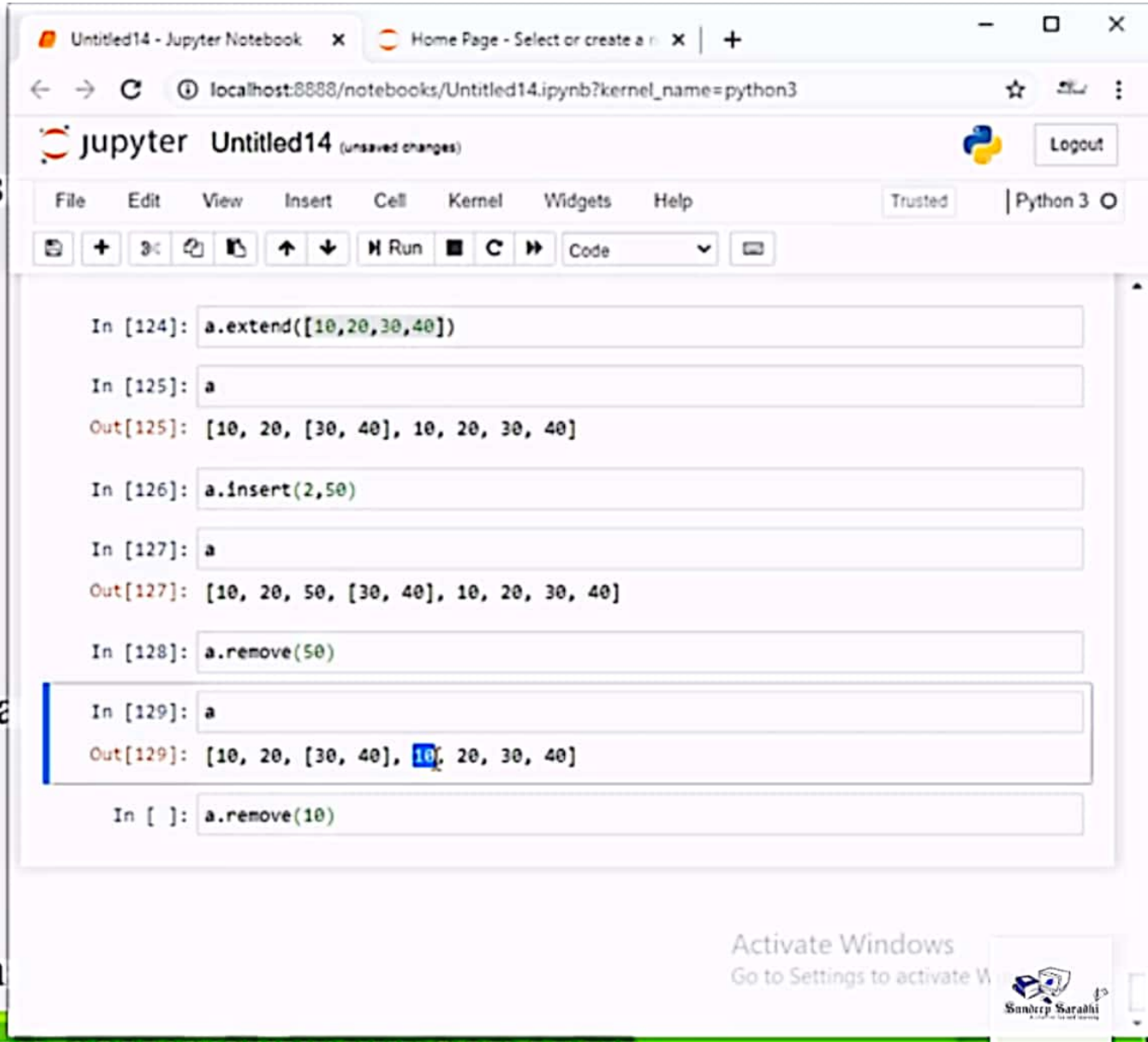
  append , extend , in



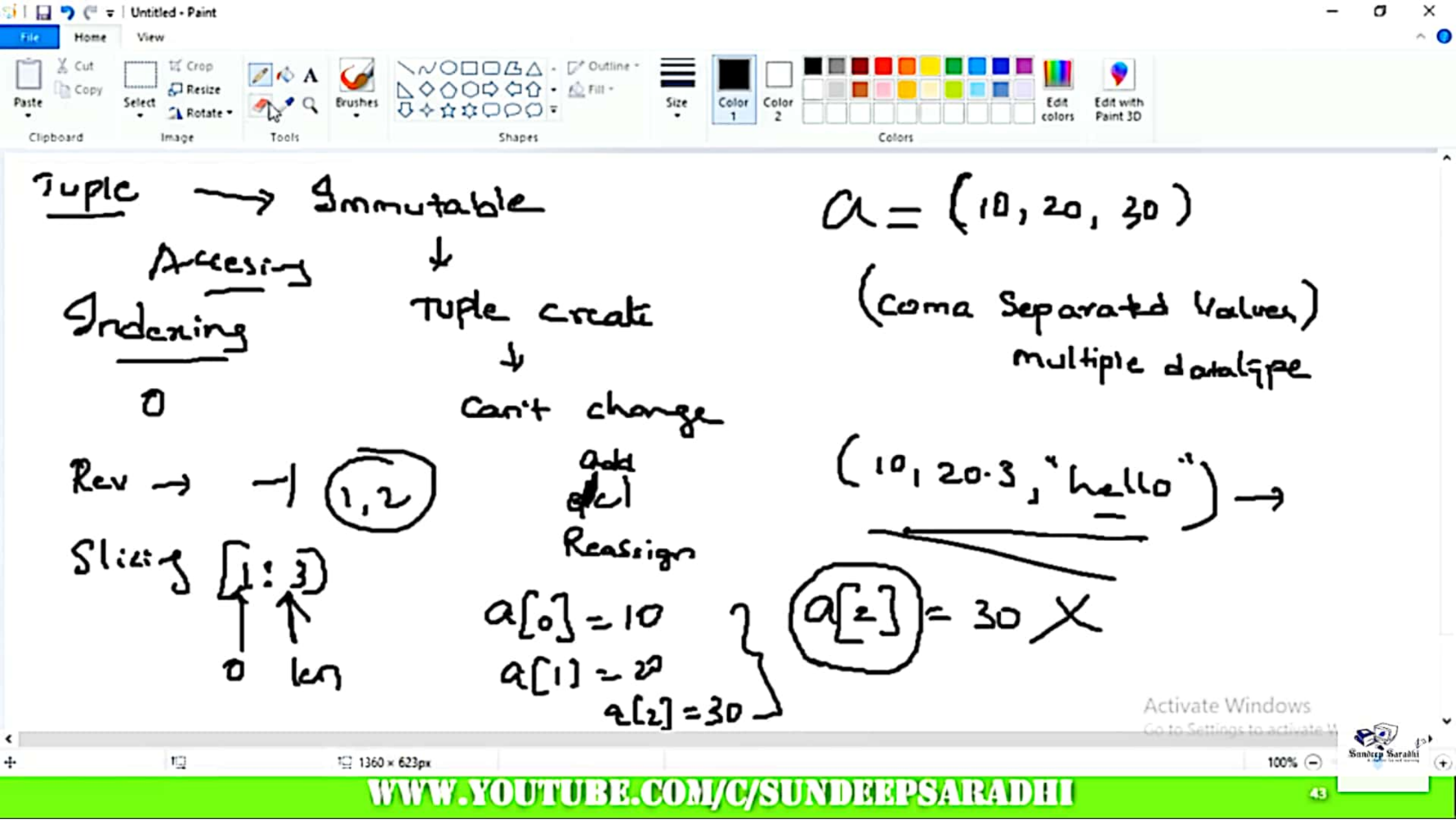
- ❑ Creating Lists
- ❑ Accessing elements from Lists
- ❑ Slicing
- ❑ Reassigning List elements
- ❑ Deleting elements
- ❑ Multidimensional Lists
- ❑ Basic Operations

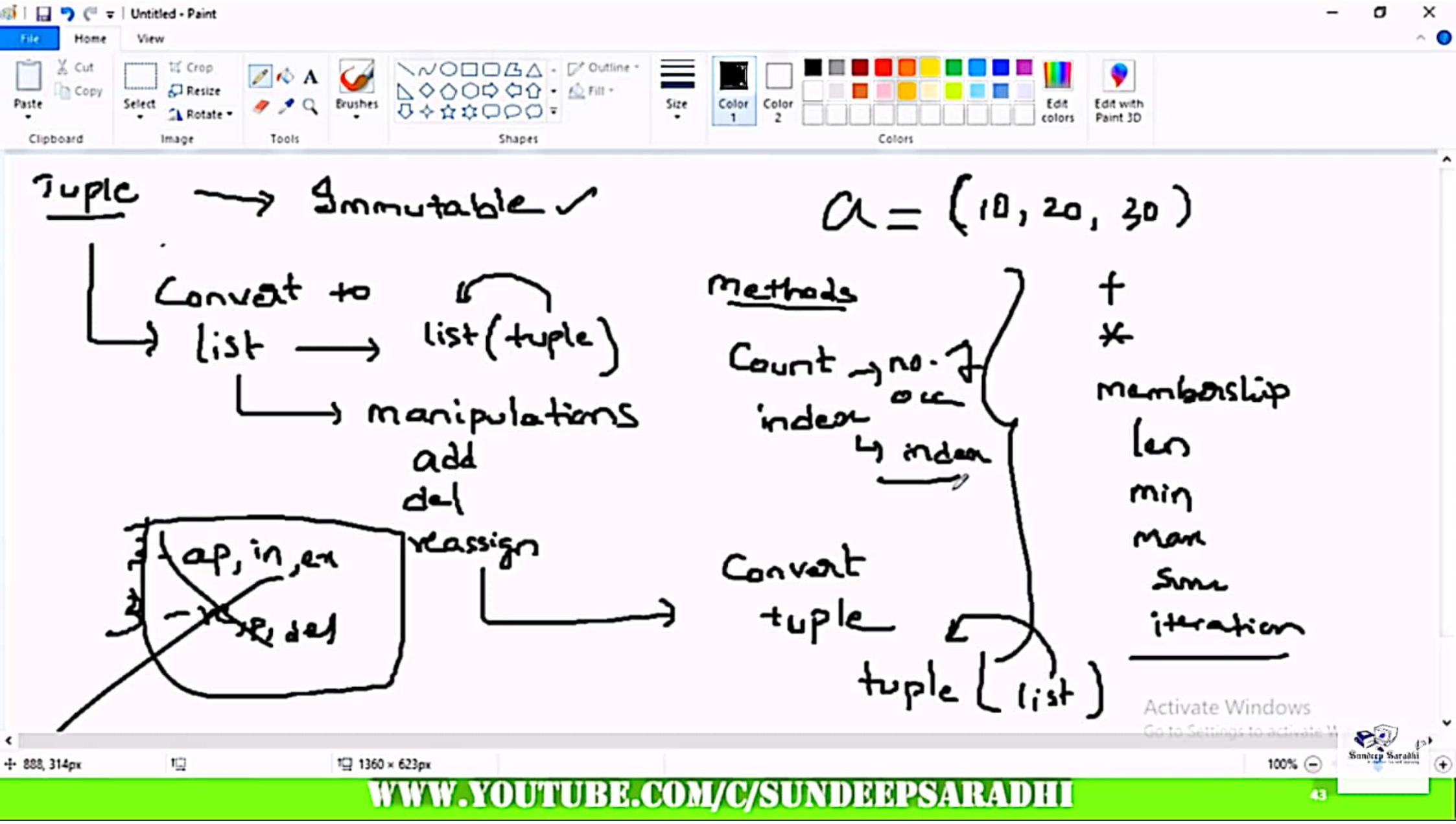
  + , * , len , min , ma
- ❑ List Comprehension
- ❑ Built-In Methods

  append , extend , in



Tuples





- Creating Tuple
- Accessing elements from Tuple
- Slicing
- Reassigning Tuple elements
- Deleting elements
- Nested Tuples
- Basic Operations

  + , * , len , min , max , sum , membership , iterations
- Built-In Methods

  count , index

Strings

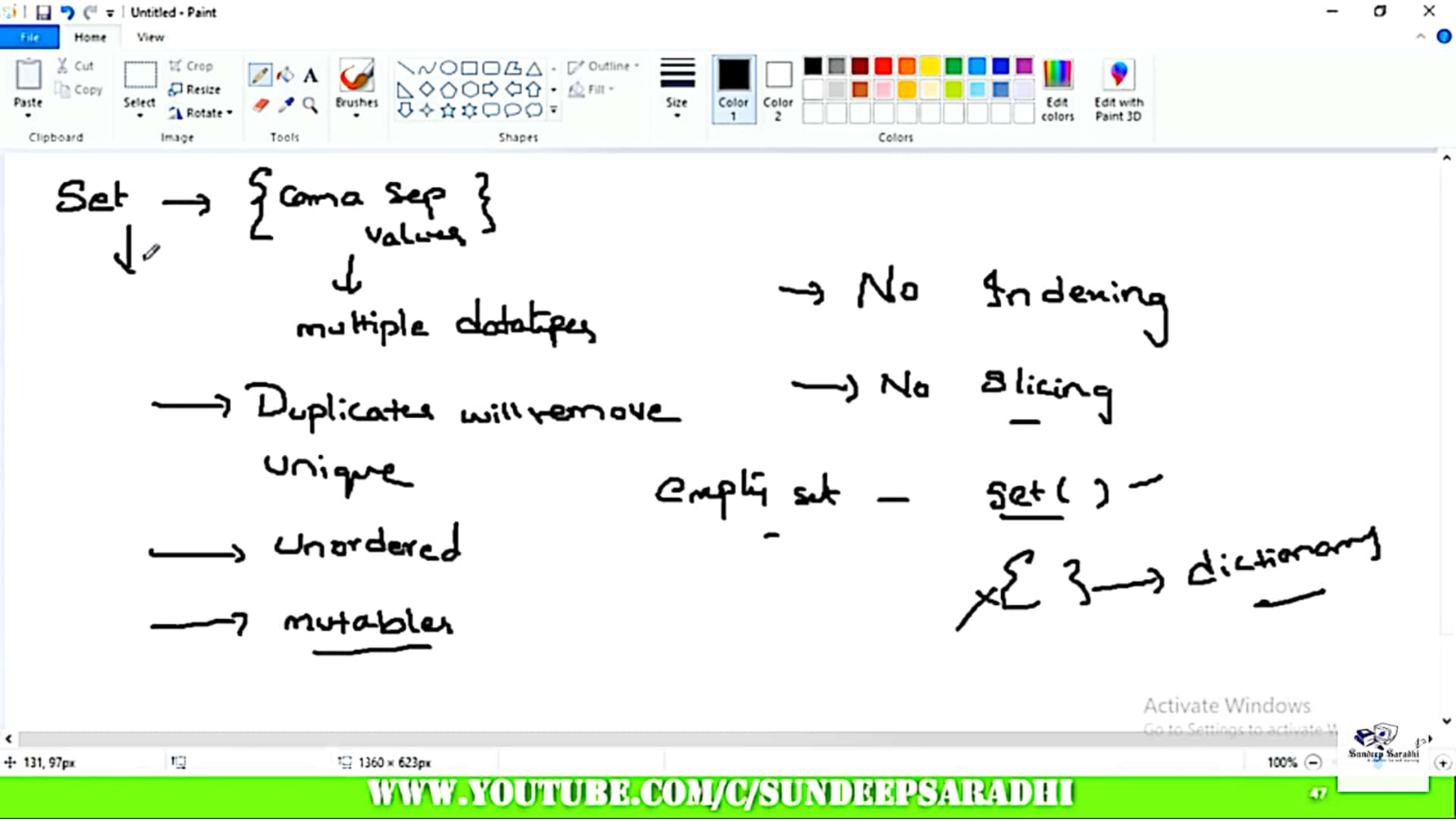
- Creating Strings
- Accessing elements from Strings
- Slicing
- Single , Double , Triple Quotes
- Format Method
- Basic Operations

  + , * , len , min , max , sum , membership , iterations

- Built-In Methods

  capitalize , center , count , endswith , startswith , find , index , rfind , rindex , isalnum , isalpha , isdigit , isspace , islower , isupper , istitle , ljust , rjust , lower , upper , strip , lstrip , rstrip , max , min , replace , split , swapcase , title , zfill

Sets



- Creating Sets
- Set as an iterable
- Basic Operations

  add , remove , discard , pop , clear len , membership ,

  issubset , issuperset , union , intersection , difference , copy ,

  symmetric_difference , update , intersection_update ,

  difference_update , symmetric_difference_update

Dictionaries

Untitled - Paint

File Home View

Paste Cut Copy | Select Crop Resize Rotate | Brushes | Shapes Outline Fill | Size | Color 1 Color 2 | Edit colors | Edit with Paint 3D

Clipboard Image Tools Colors

Dictionary = { }

{ key1 : Value1 , key2 : Val2 , ... }

↳ items

Key : Value

Access

Indexing

Key - index

{ } dict

d[key]

Keys — No Duplicates

Values — Duplicates

d = {'a' : 2 , 'b' : 3 }

d['a'] = 2

Activate Windows

390, 203px 1360 × 623px 100%

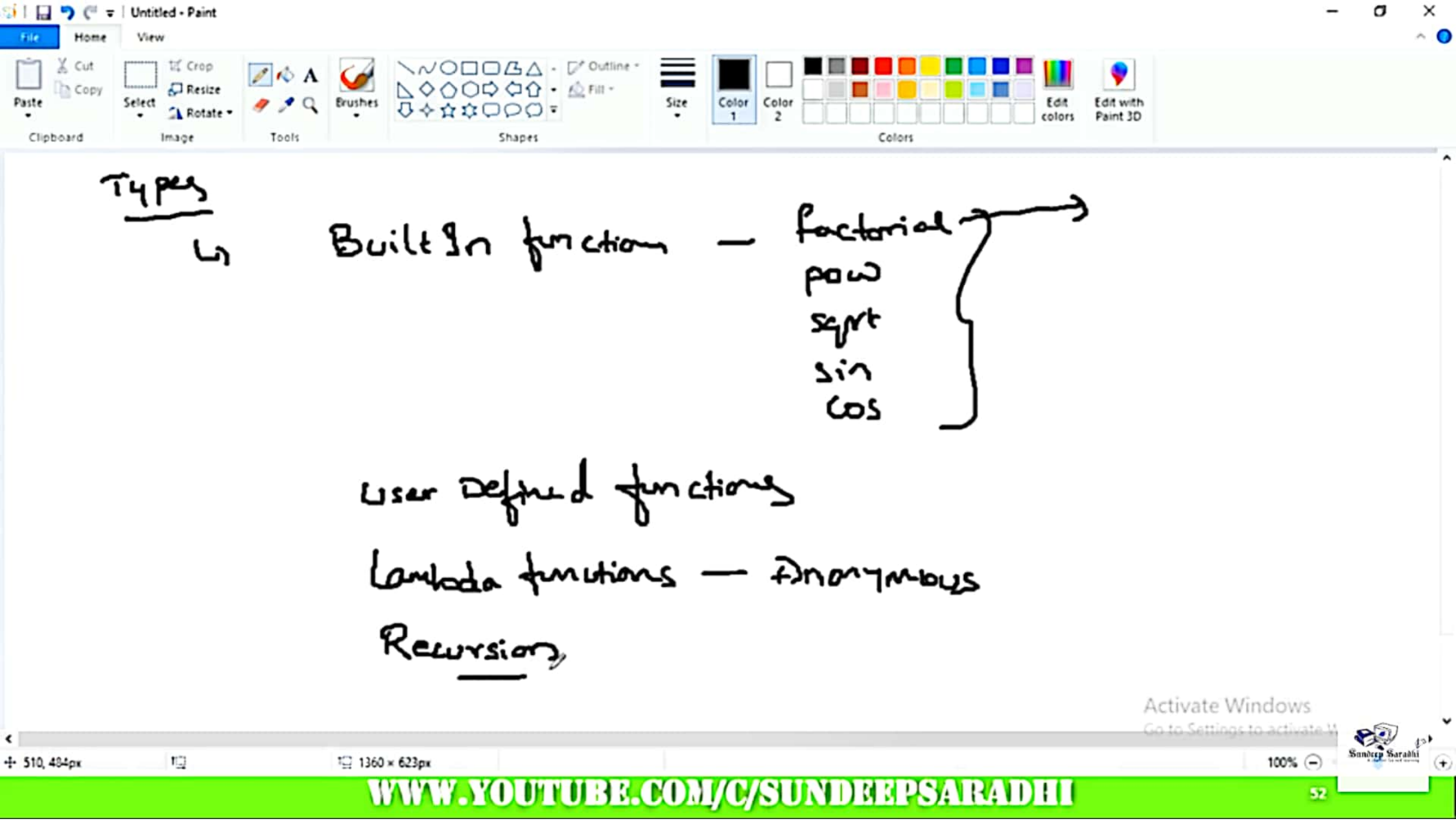
WWW.YOUTUBE.COM/C/SUNDEEPSARADHI

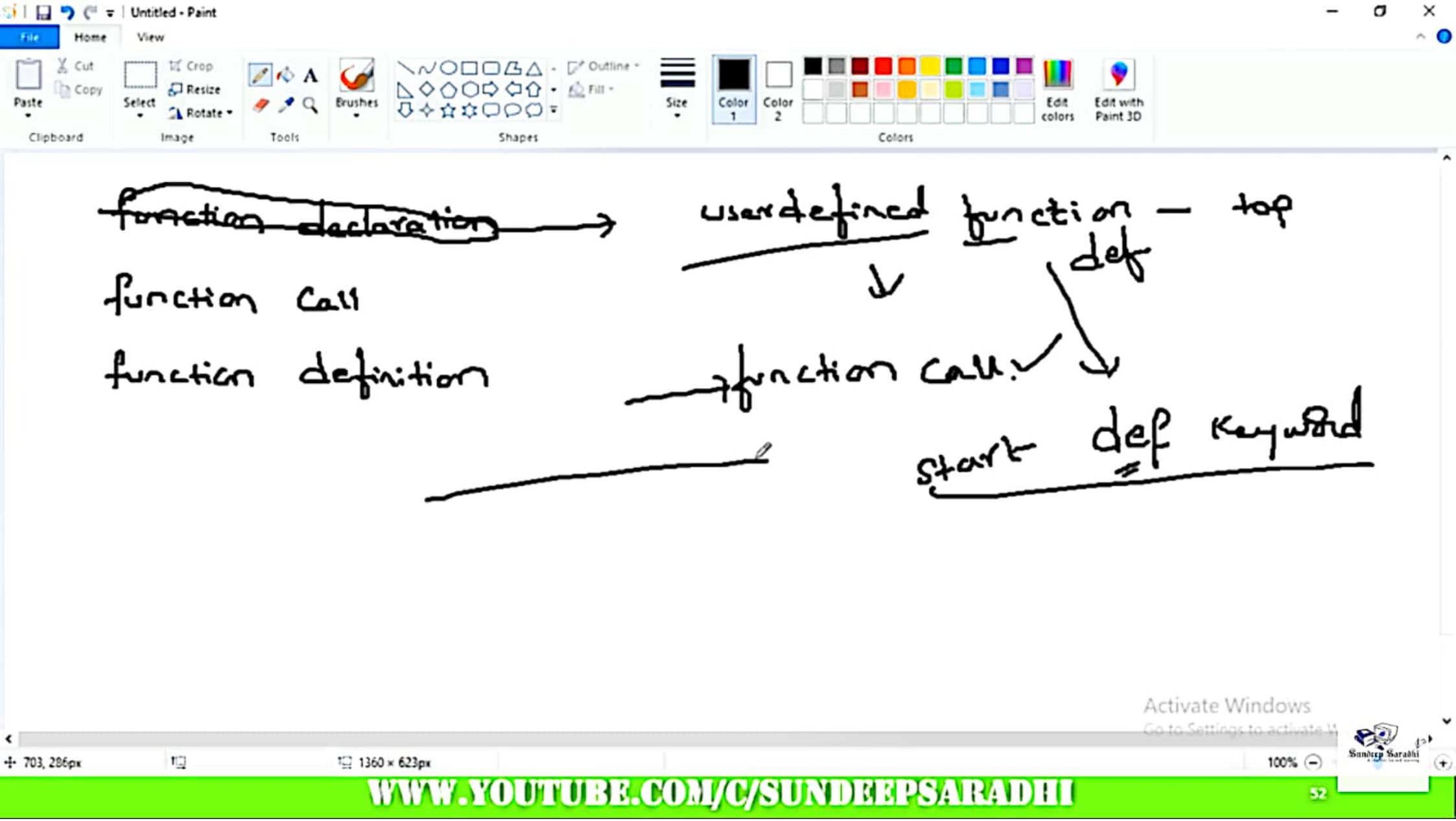
- Creating Dictionary
- Accessing items from Dictionary
- Updating Dictionary
- Reassigning items
- Deleting items
- Dictionary Comprehension
- Built-In Methods

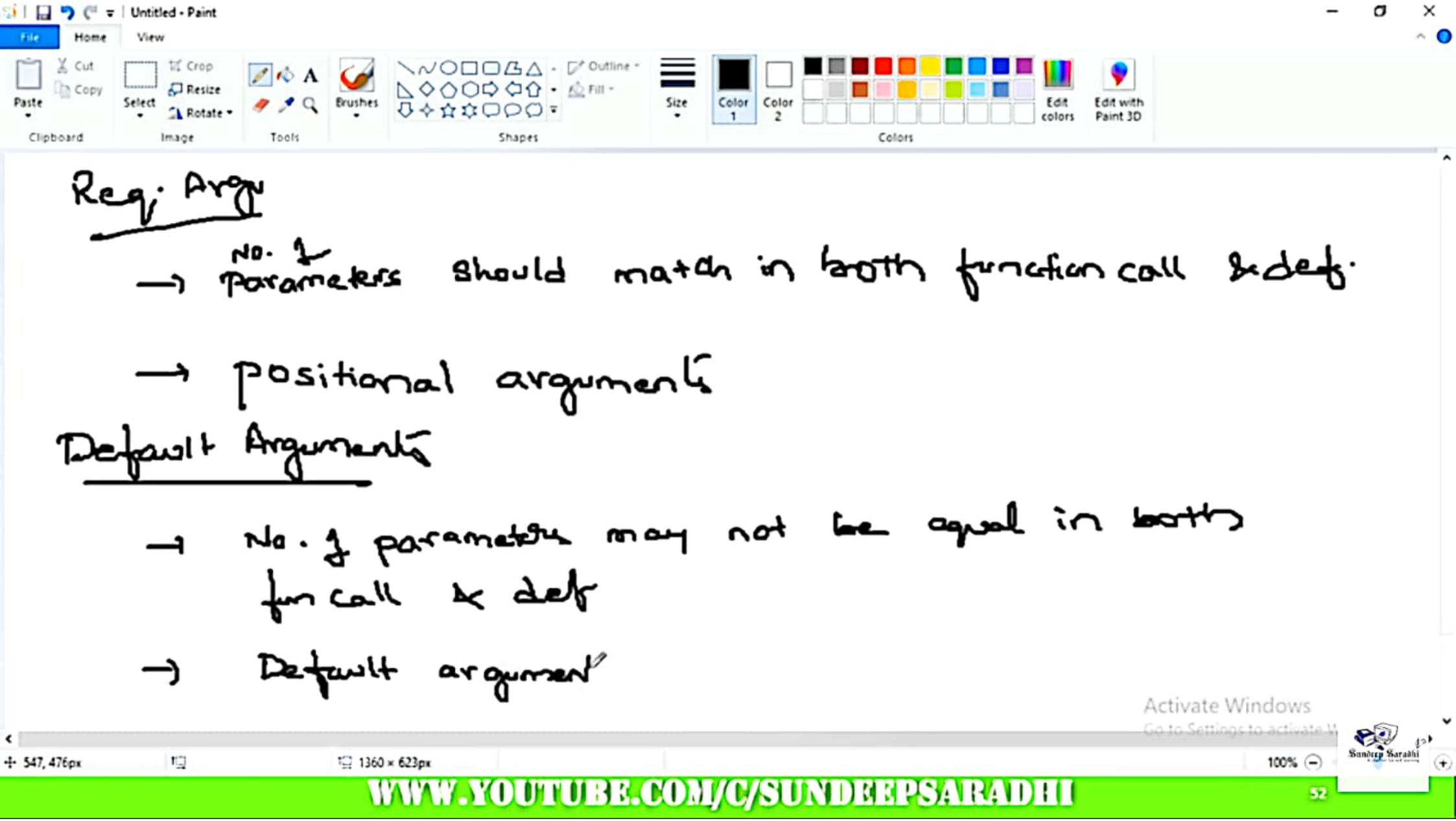
clear , copy , items , values , keys , update

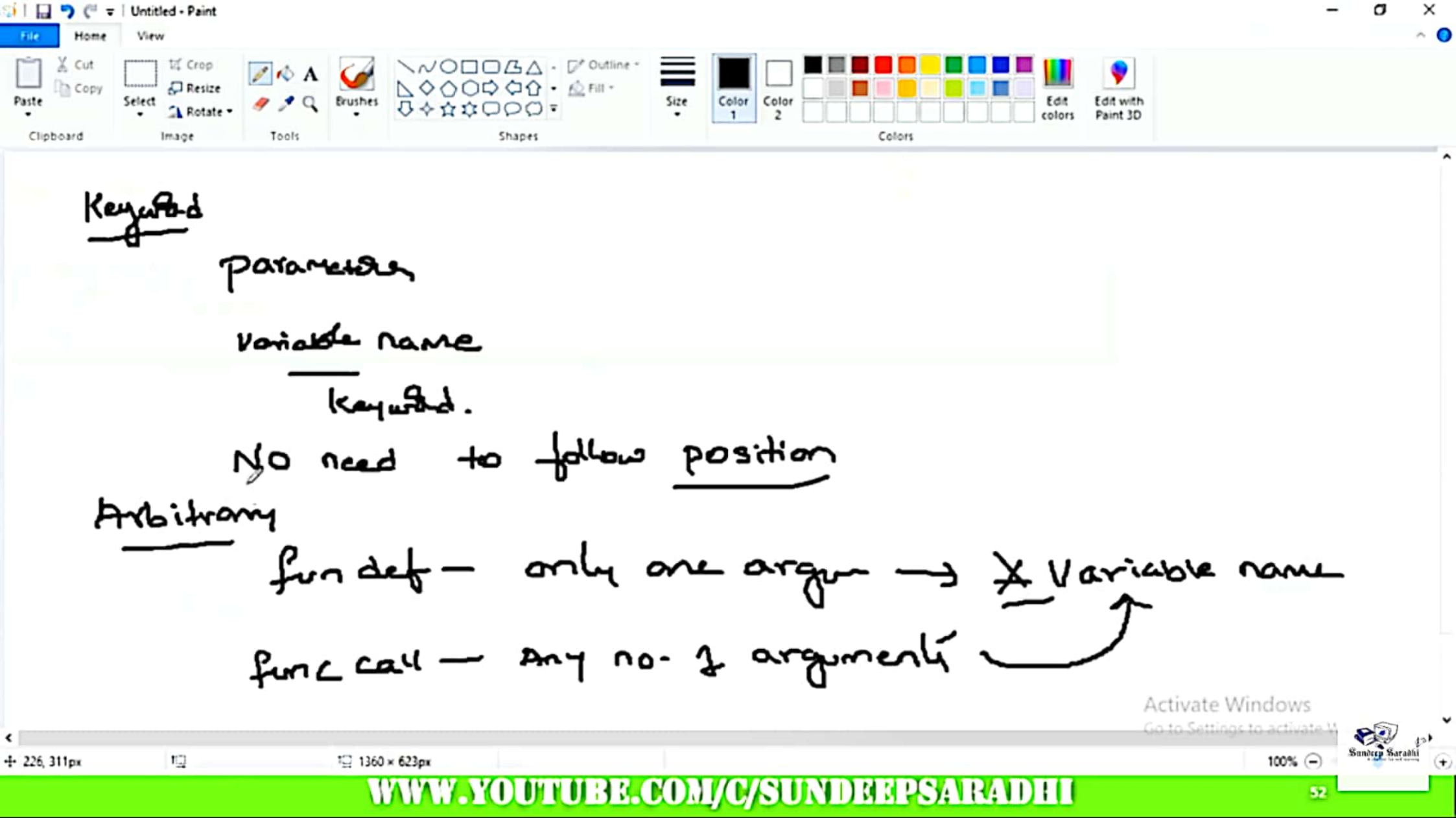
Functions

- Types of Functions
- Function Call
- Function Definition
- Types of Arguments
  - Required Arguments , Default Arguments , Keyword Arguments , Arbitrary Arguments
- Multiple Returns
- Local and Global Variables
- Lambda Function
- Recursion









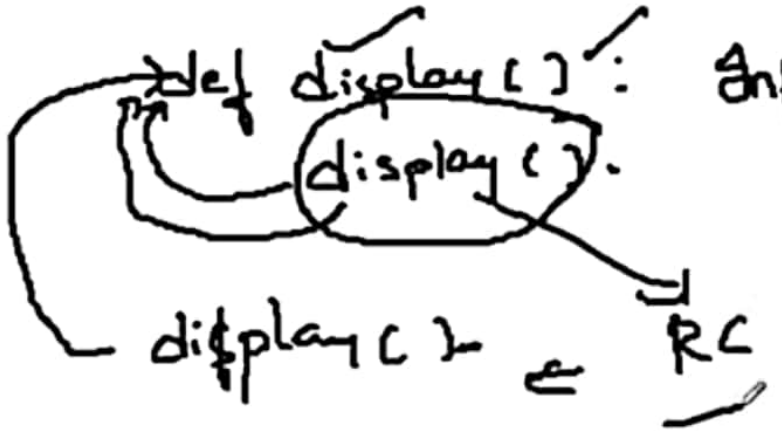
Untitled - Paint

File Home View

Cut Copy Paste Clipboard | Crop Resize Rotate Select Image | Tools | Brushes | Shapes | Outline Fill | Size | Color 1 Color 2 | Colors | Edit colors | Edit with Paint 3D

Recursion

Calling function itself.



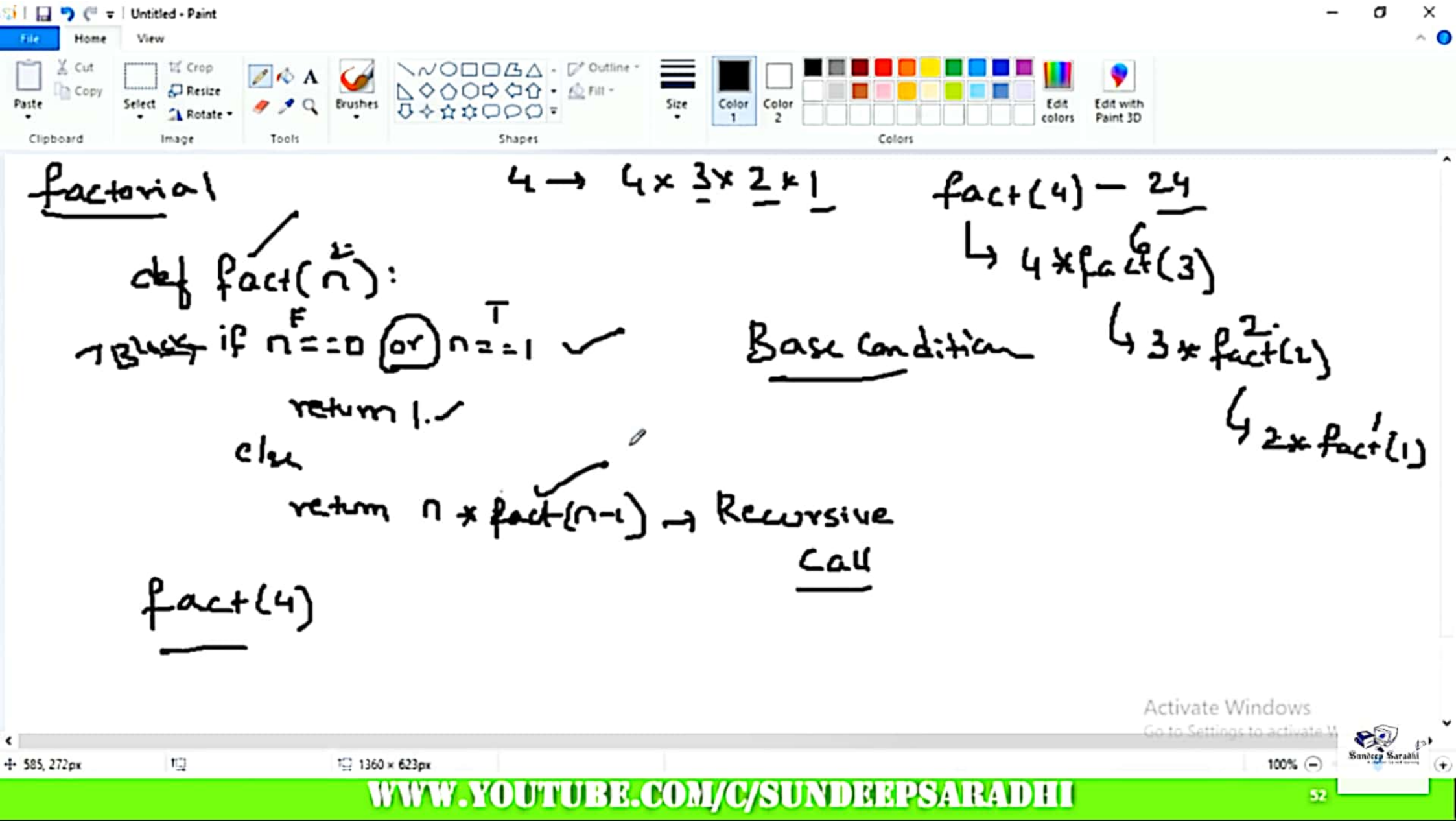
Infinite recursion → terminate

Base condition

recursive call

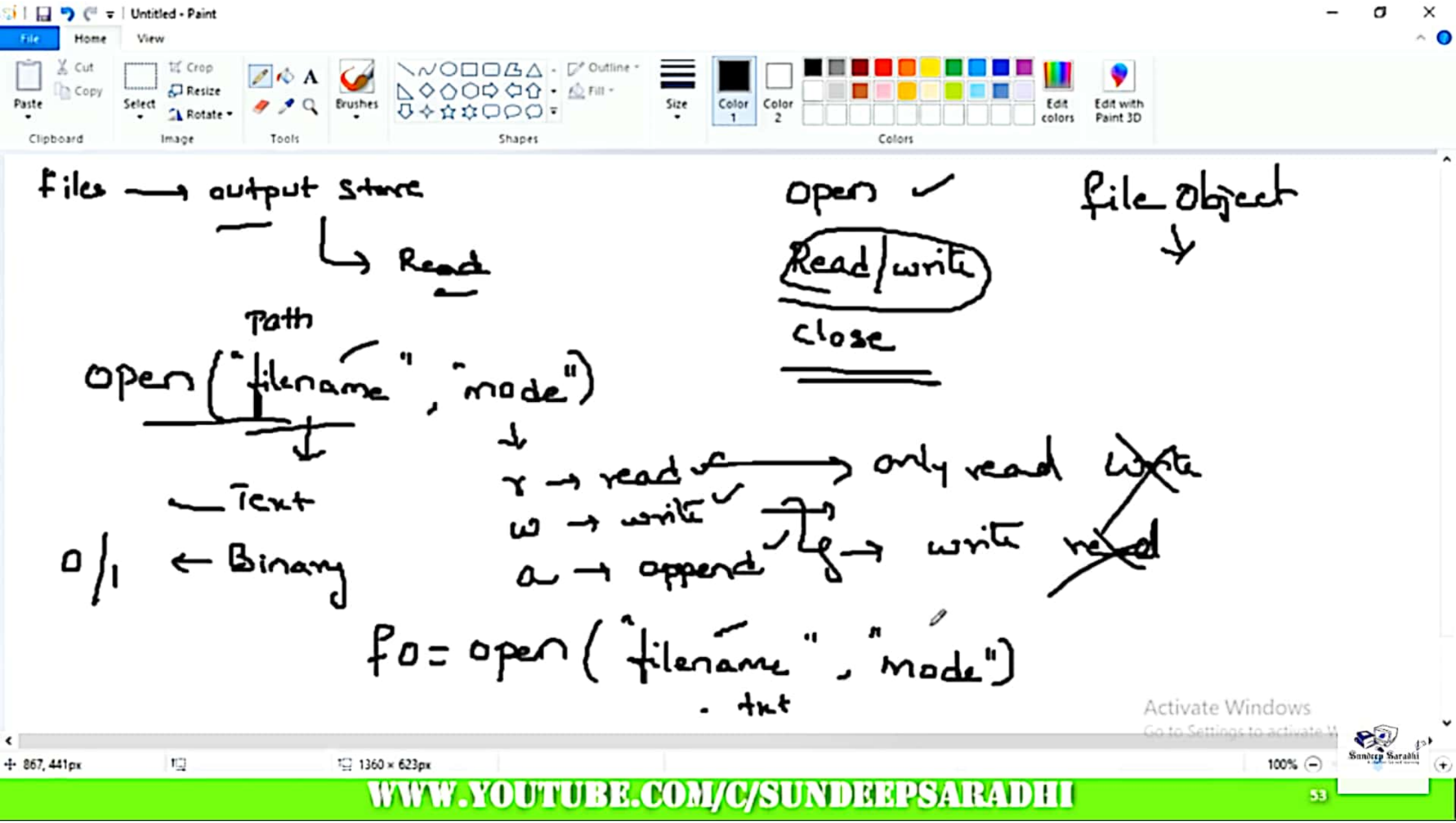
Activate Windows

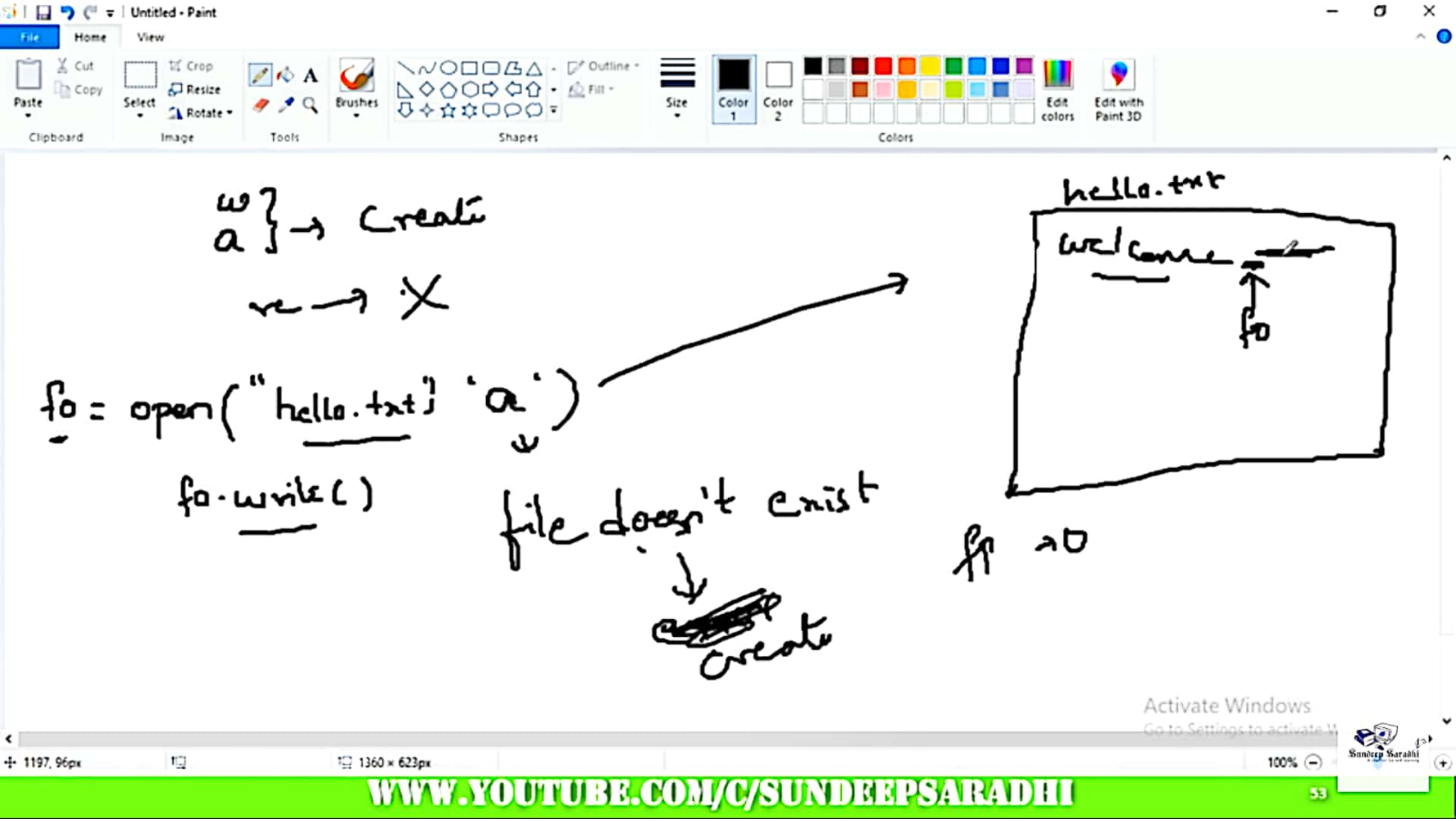
588, 408px 1360 × 623px 100%

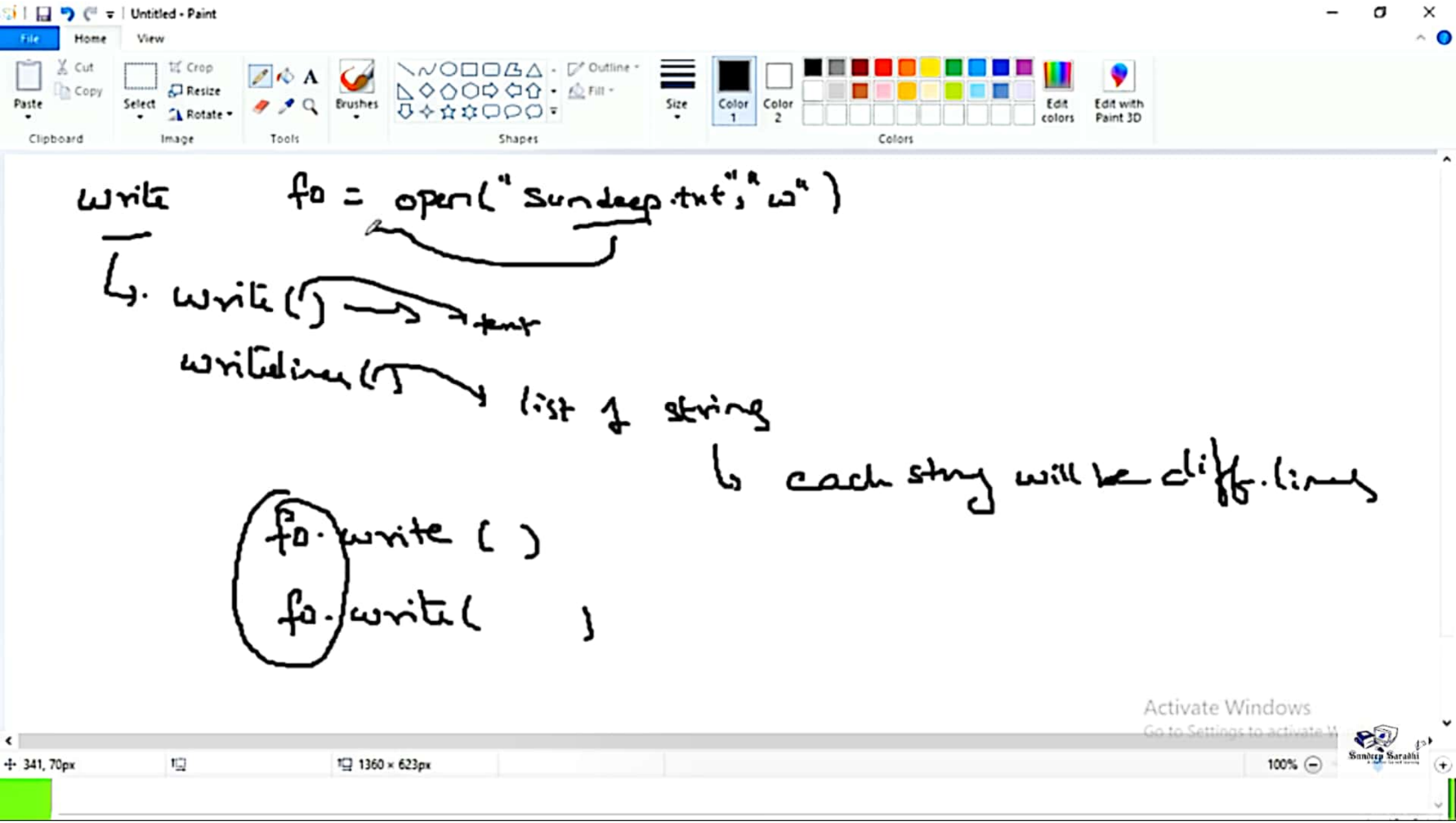


Files

- What is File ?
- Opening a File
- Access Modes
- Closing File
- Writing Data into File
  - write , writelines
- Reading Data from File
  - read , readline , readlines
- File Handling functions
  - tell , seek
- Usage of WITH keyword in Files



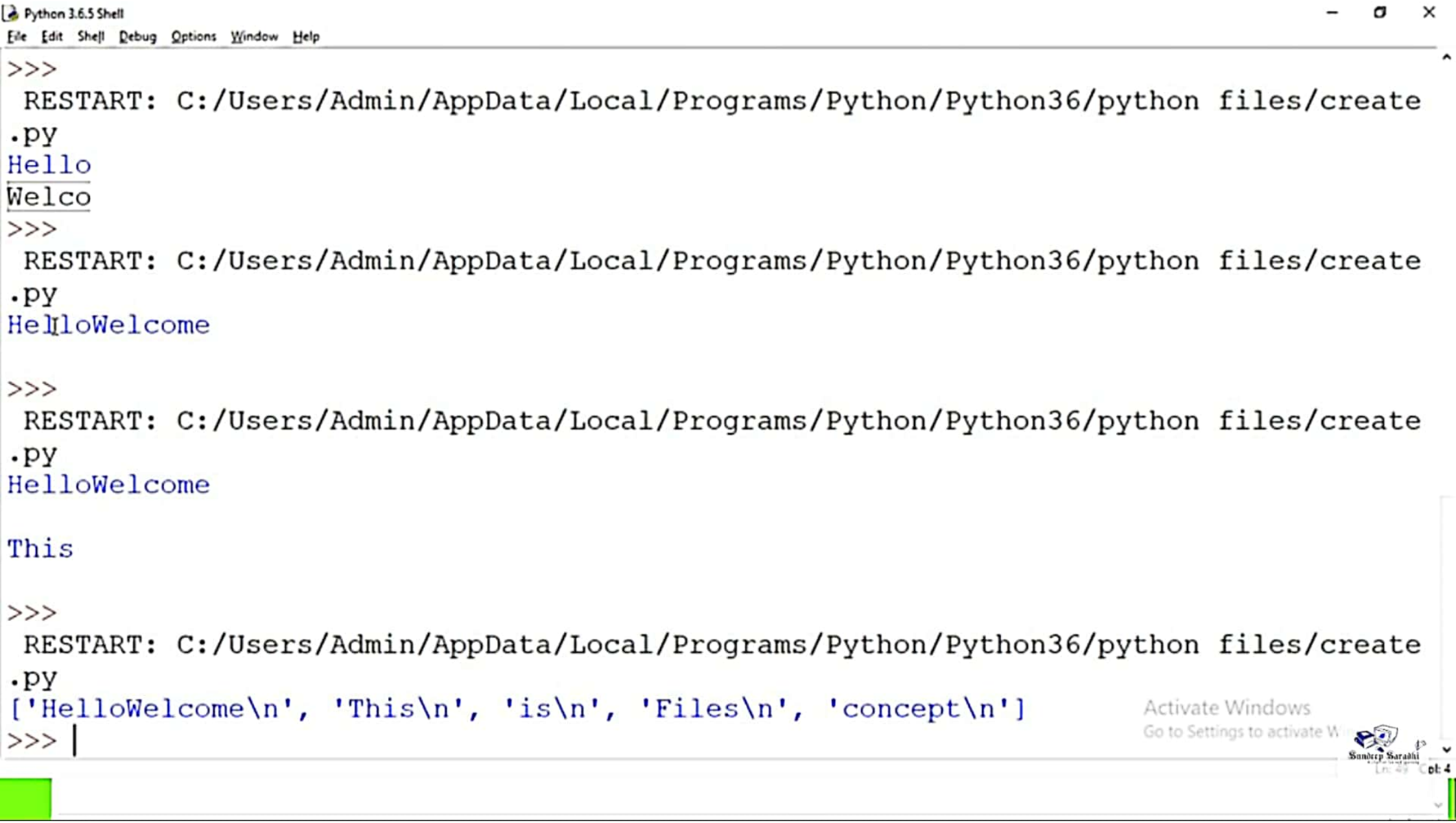




create.py - C:/Users/Admin/AppData/Local/Programs/Python/Python36/python files/create.py (3.6.5)

File Edit Format Run Options Window Help

```
fo=open("Sundeep.txt","w")
fo.write("Hello")
fo.write("Welcome")
l=["\nThis\n","is\n","Files\n","concept\n"]
fo.writelines(l)
fo.close()
fo1=open("Sundeep.txt","r")
print(fo1.readlinse())

fo1.close()
```



Libraries in Python

- **Data Science**
  - Pandas , NumPy , SciPy , Scrapy , Matplotlib , Seaborn , Scikit-Learn , TensorFlow , Scikit-Image , Librosa
- **Data Visualization**
  - Matplotlib , ggplot , Ploty , Altair , Seaborn , Bokeh , Pygal
- **Data Manipulation**
  - Pandas , NumPy
- **Database Access**
  - SQLAlchemy , Quandl
- **Data Modeling**
  - Tensorflow , PyTorch , Scikit – Learn , Keras

- **Web Development**
  - Django , Bottle , Pyramid , Turbo Gears , Web2py , Grok , Flask , CherryPy , Hug , Falcon
- **Gaming**
  - PyGame
- **GUI Framework**
  - Tkinter , wxPython , PyQt5
- **Robotics**
  - PyRobot , DART , SOFA , Pyro
- **Network Programming**
  - NetworkX
- **Natural Language Processing**
  - NLTK , Scikit-Learn , polyglot , genism , spaCy , rasaNLU

- **Deep Learning**
  - TensorFlow , Torch , Caffe , Theano , Deepmat , ML.NET , Neon , Microsoft CNTK , Deeplearning 4j
- **Image Processing**
  - OpenCV , Scikit-Image , Scikit-Learn , SciPy , NumPy
- **AI & Machine Learning**
  - Scikit-Learn , Theano , TensorFlow , Keras