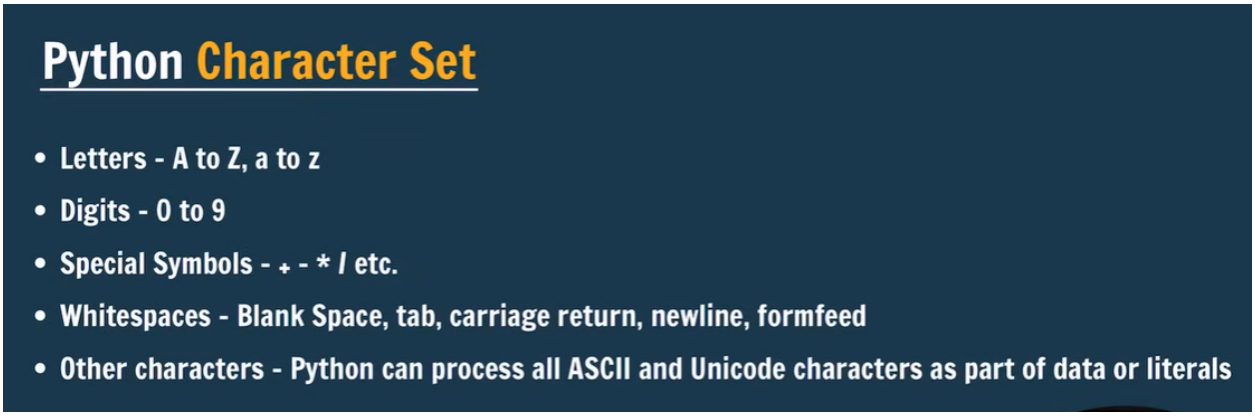
Python Notes For Data Science



Print Statement

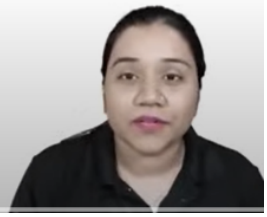
To call a print function in python we just need to write print followed by parentheses () and values written inside quotation marks “”.

Input-

print (“Hello World”)

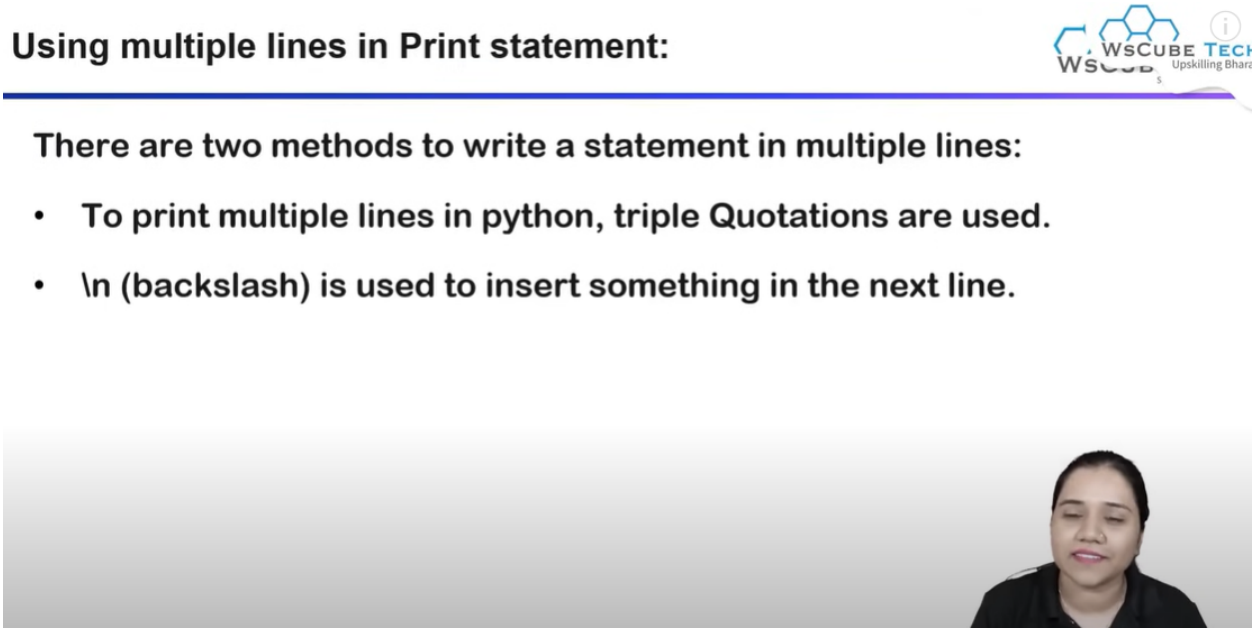
Output-

Hello World

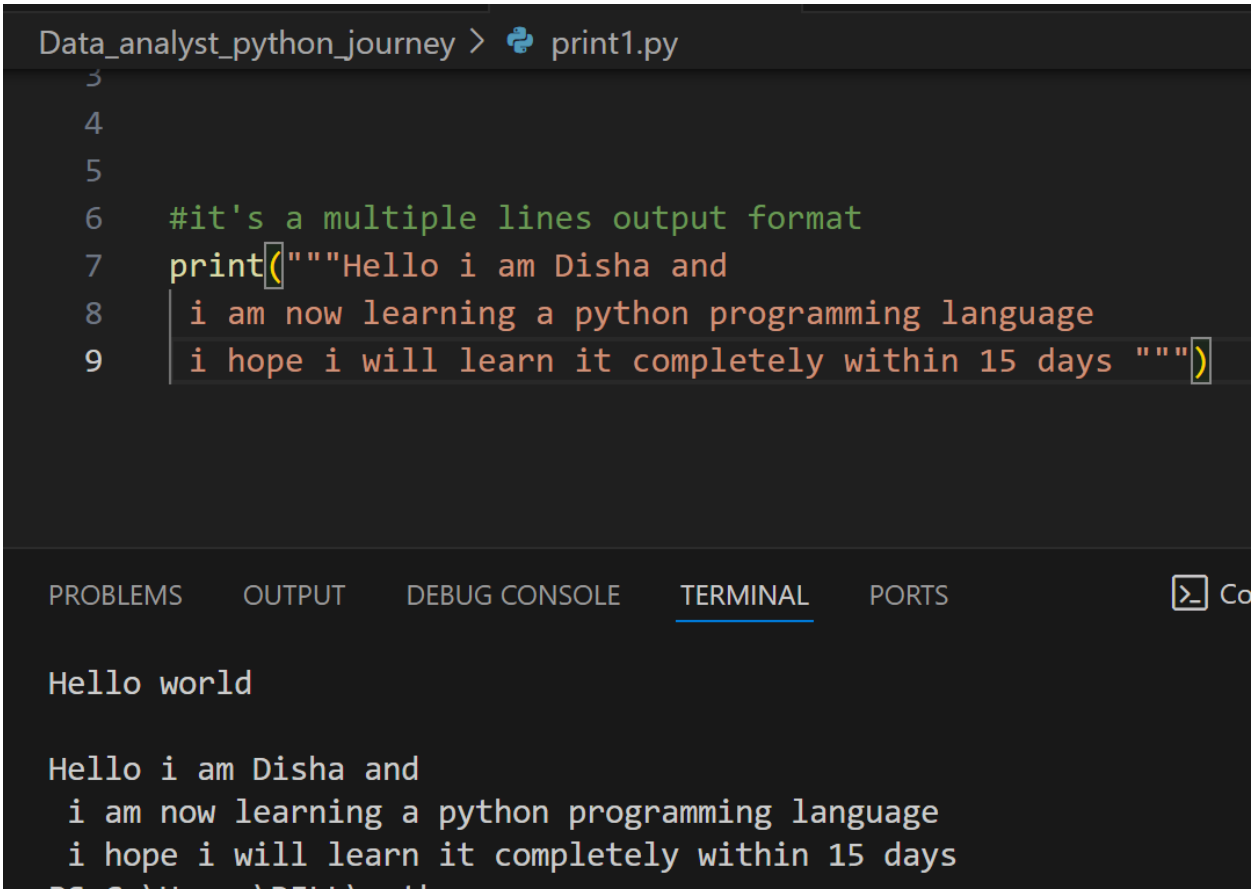


```
Data_analyst_python_journey > print1.py
1   #it's a single line print outcome statement
2   print("Hello world\n")
3
```

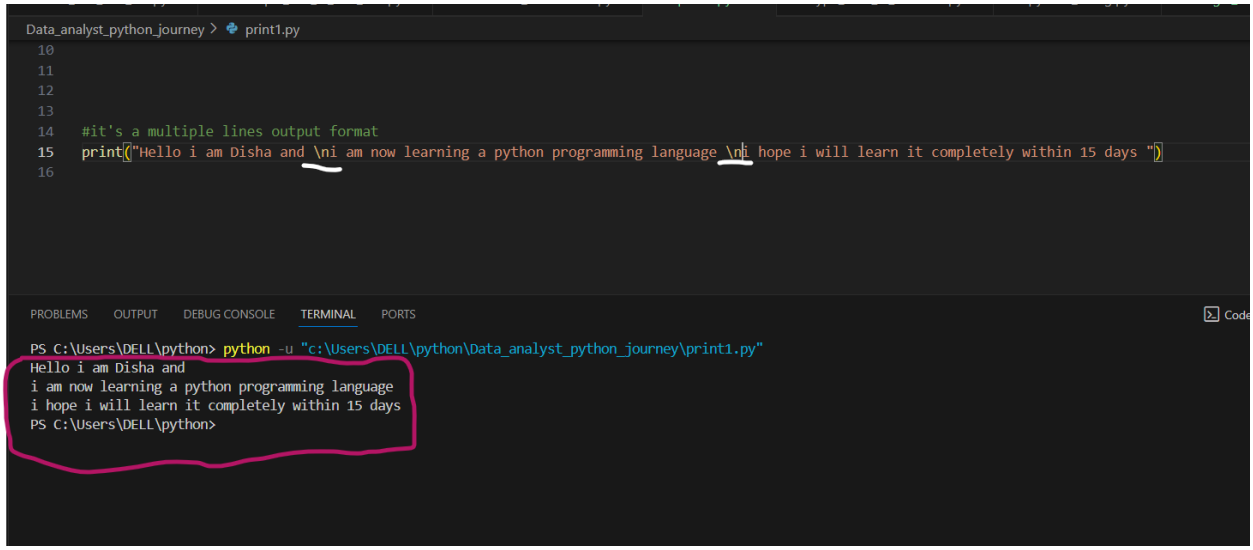
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS        Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\Data_analyst_python_journey\
print1.py"
Hello world

PS C:\Users\DELL\python>
```


Using multiple lines in Print statement:
There are two methods to write a statement in multiple lines:
• To print multiple lines in python, triple Quotations are used.
• \n (backslash) is used to insert something in the next line.

1st method:


Data_analyst_python_journey > print1.py
#it's a multiple lines output format
print("""Hello i am Disha and
i am now learning a python programming language
i hope i will learn it completely within 15 days """)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Hello world
Hello i am Disha and
i am now learning a python programming language
i hope i will learn it completely within 15 days

2nd method:


Data_analyst_python_journey > print1.py
#it's a multiple lines output format
print("Hello i am Disha and \ni am now learning a python programming language \ni hope i will learn it completely within 15 days ")
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\Data_analyst_python_journey\print1.py"
Hello i am Disha and
i am now learning a python programming language
i hope i will learn it completely within 15 days
PS C:\Users\DELL\python>

Single Line comments

To add single line comments, # hash is used.

Python Completely ignores anything written after #.

Multiline comments

To add multiline comments in python, triple Quotations are used.

single_and_multiple_lines_comments.py

```
1       #single_line_comment
2
3   #Hi today i am going to show you
4
5
6       #multiple_lines_comment
7
8   """now i am using multiple lines
9   comments in python"""
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\single_and_multiple_lines_comments.py"
PS C:\Users\DELL\python>
```

Variables

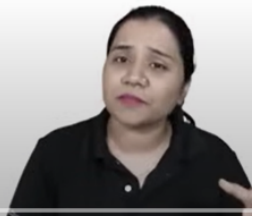
- Variables are placeholders, which can store a value.
- In simple words, Variable is a container that holds data inside it as a value.

Input-

```
a = “hello world”
print (a)
```

Output-

```
hello world
```



variable_print.py > ...

```
1   a="connected to dots"
2   print(a)
```

Terminal (Ctrl+`)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\variable_print.py"
connected to dots
PS C:\Users\DELL\python>
```

Rules for writing a Variables

1. Python is case-sensitive Language, therefore the variables names are case-sensitive as well.

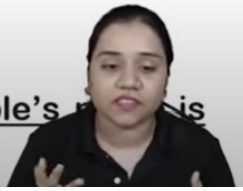
Input-

A = “hello”

print (a)

Output-

It will throw an error as the cases used here for variable’s name is different.



Rules for writing a Variables

2. Make sure to not use spaces while creating a variable.

One can use (_) underscore to separate the names while writing a variable.

3. A variable name should never start with a number or special symbols.

rules_for_writing_variables.py > ...

```python
#rule 1:

Name="John"
print(name)#here Name or name *N and *n  (error)

#rule 2:
even num=2 #error (correct: even_error)  *underscore

#rule 3:
1Num=2  #error (correct: Num1=2) *digit last or first

```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\rules_for_writing_variables.py"
  File "c:\Users\DELL\python\rules_for_writing_variables.py", line 10
    1Num=2  #error (correct: Num1=2) *digit last or first
    ^
SyntaxError: invalid decimal literal
PS C:\Users\DELL\python>
```

Datatypes:

Text- type: String (str)

Numeric Types: integer (int), floating point (float), complex

Sequence Types: list, tuple and range

Mapping Type: Dictionaries (dict)

Set Type: set, frozenset

Boolean Type: bool

Binary Types: bytes, bytearray, memoryview

User-inputs

WSCUBE TECH Upskilling Bharat

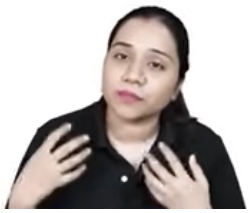
To ask for the input from the user. Default datatype is string.

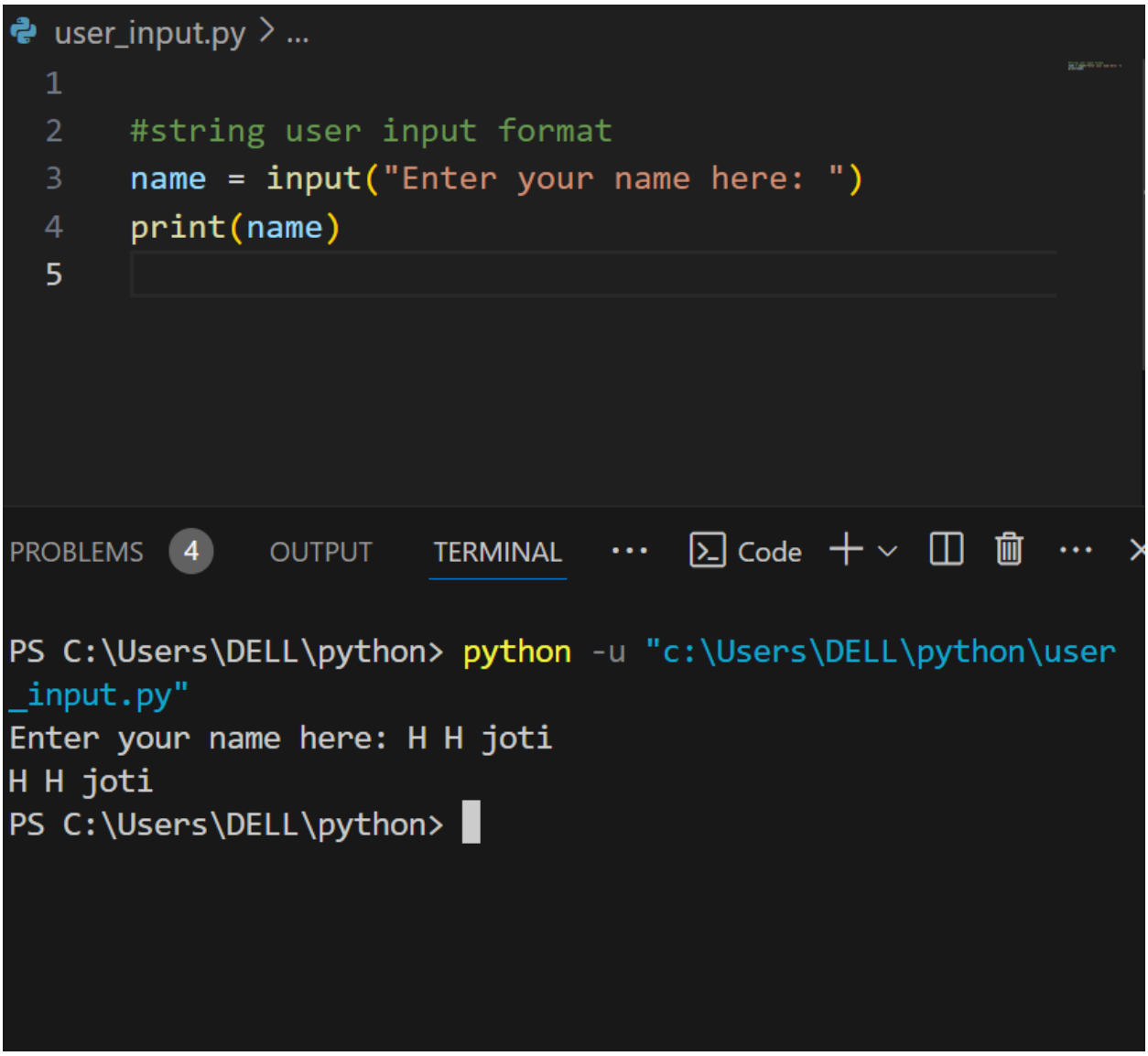
Input:

```
name = input("enter your name here")
print (name)
```

Output:

Entered name by the user




user_input.py > ...
1
2 #string user input format
3 name = input("Enter your name here: ")
4 print(name)
5
PROBLEMS 4
OUTPUT
TERMINAL
Code
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\user_input.py"
Enter your name here: H H joti
H H joti
PS C:\Users\DELL\python>

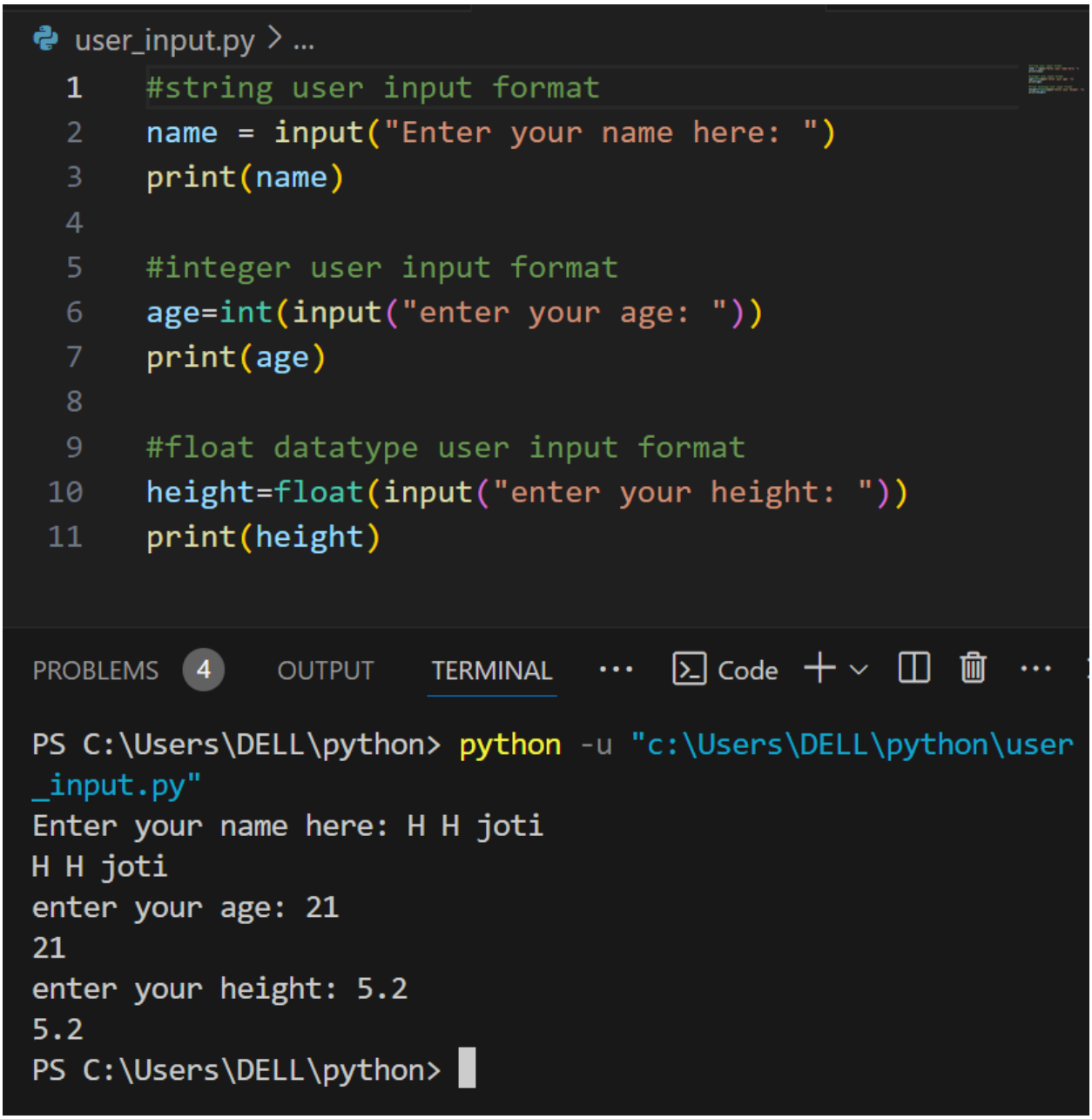
User-inputs

Input:

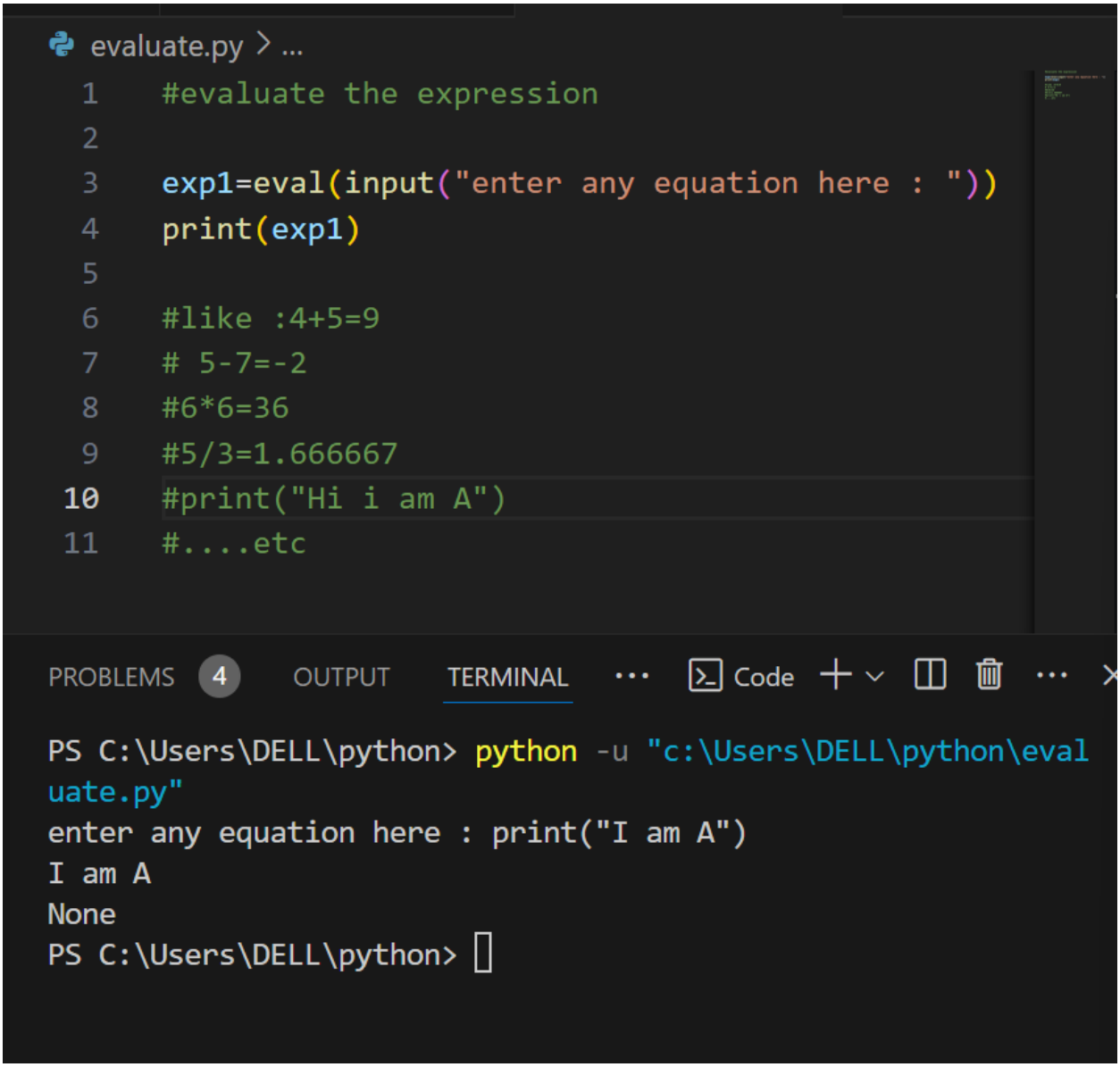
```
age = int(input("enter your name here"))
print (age)
```

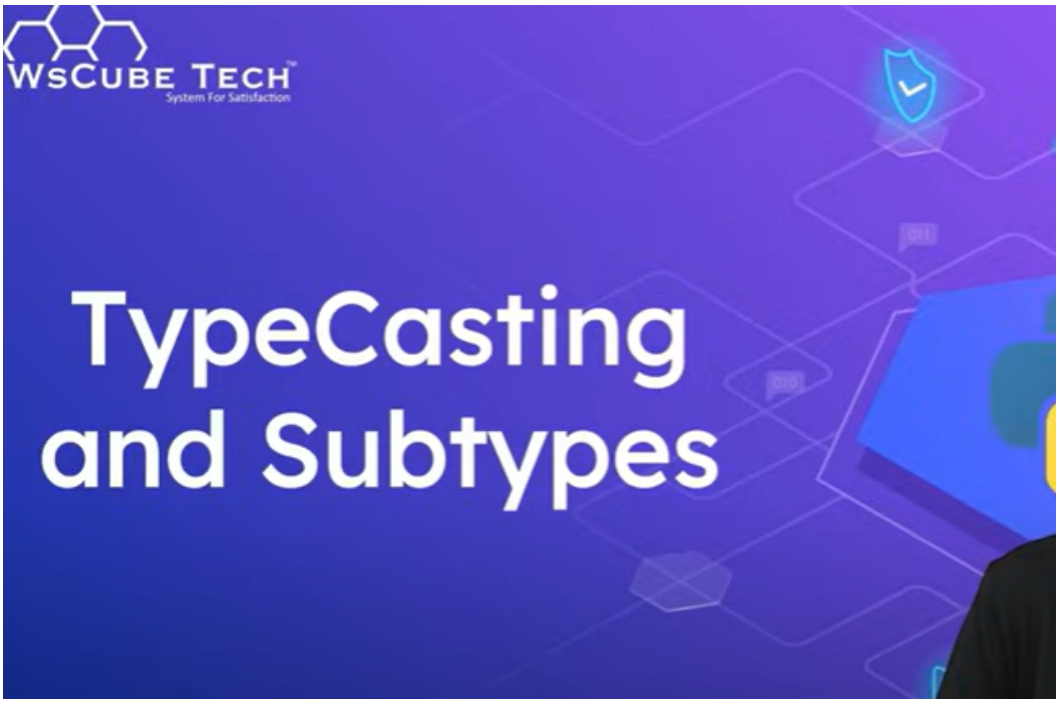
Output:

Entered age by the user



Evaluate function(eval):


evaluate.py > ...
1 #evaluate the expression
2
3 exp1=eval(input("enter any equation here : "))
4 print(exp1)
5
6 #like :4+5=9
7 # 5-7=-2
8 #6*6=36
9 #5/3=1.666667
10 #print("Hi i am A")
11 #....etc
PROBLEMS 4
OUTPUT
TERMINAL
Code
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\evaluate.py"
enter any equation here : print("I am A")
I am A
None
PS C:\Users\DELL\python>

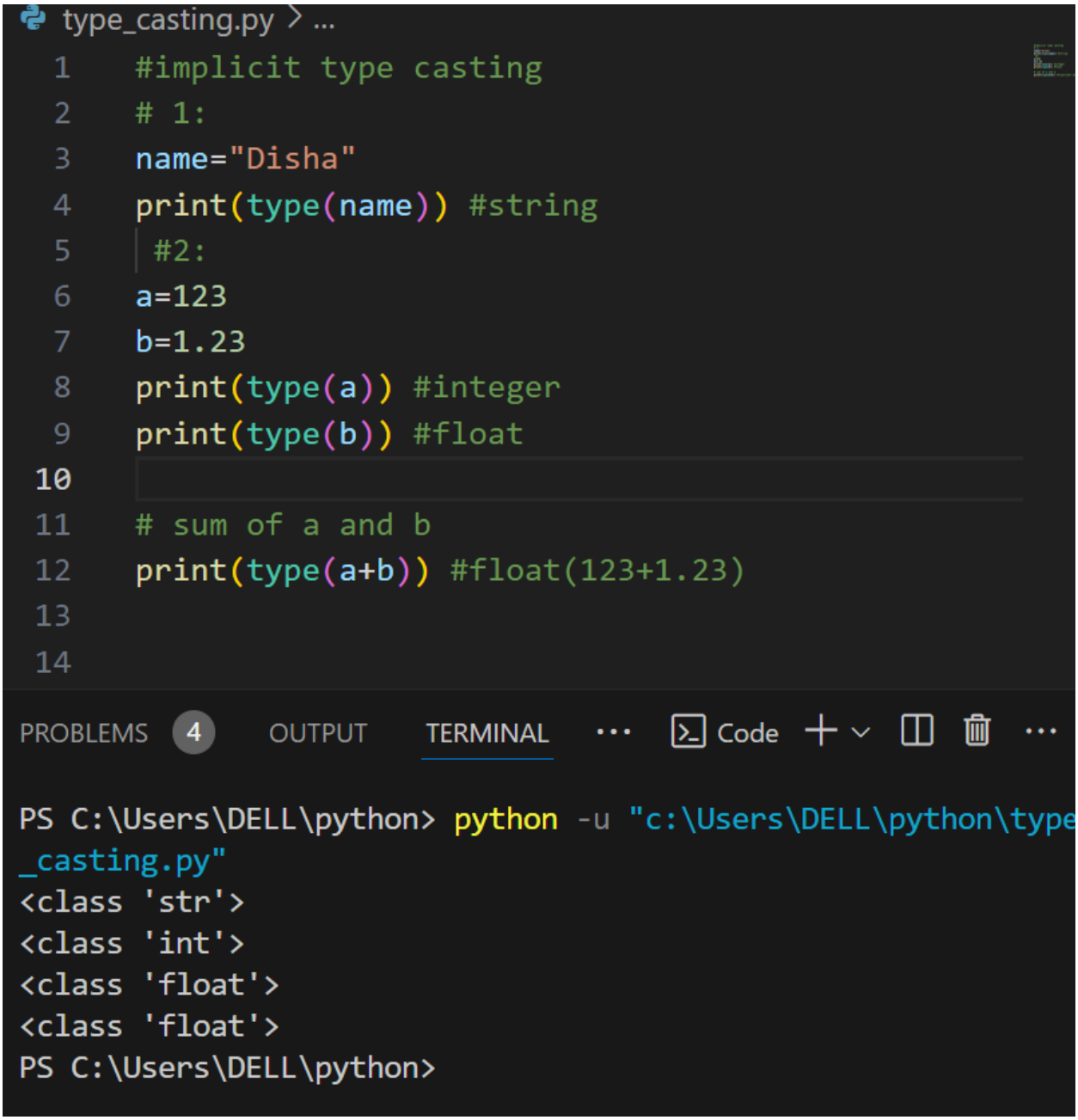


Type Casting

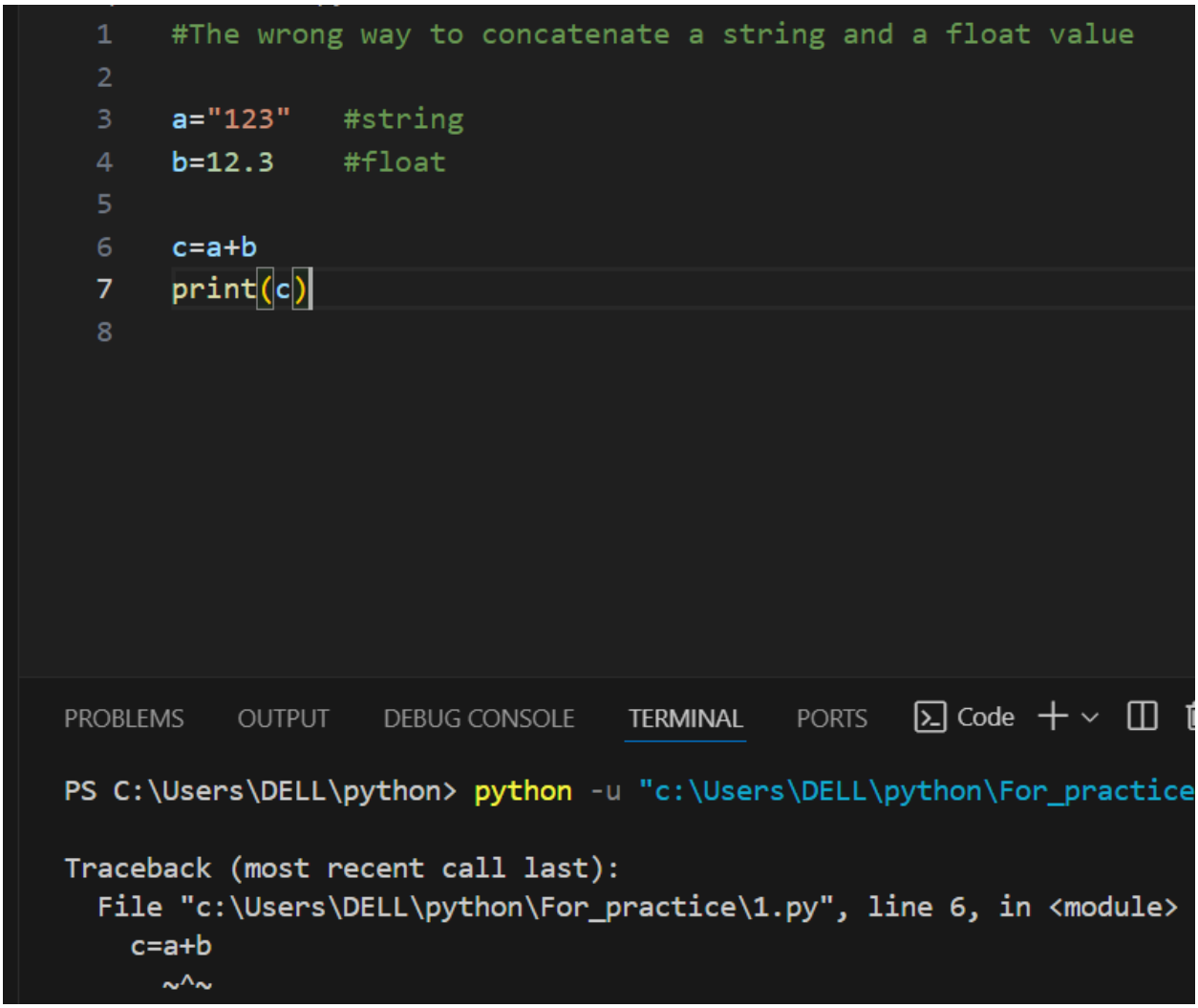
Conversion of one datatype to another is called as type-casting.

There are two types of type-casting:

1. Implicit Type Conversion: where python itself converts one datatype to another.
2. Explicit Type Conversion: where the user converts one da to another.



The wrong way to concatenate a string and a float value


#The wrong way to concatenate a string and a float value
a="123"   #string
b=12.3   #float
c=a+b
print(c)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Code
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice
Traceback (most recent call last):
File "c:\Users\DELL\python\For_practice\1.py", line 6, in <module>
c=a+b
~^~

type_casting2_problem.py > ...

```python
# add string and float
a="123"
b=1.23
c=a+b #error (string and float are never be added)
print(c)

```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\type_casting2_problem.py"
Traceback (most recent call last):
  File "c:\Users\DELL\python\type_casting2_problem.py", line 4, in <module>
    c=a+b #error (string and float are never be added)
      ~^~
TypeError: can only concatenate str (not "float") to str
PS C:\Users\DELL\python>
```

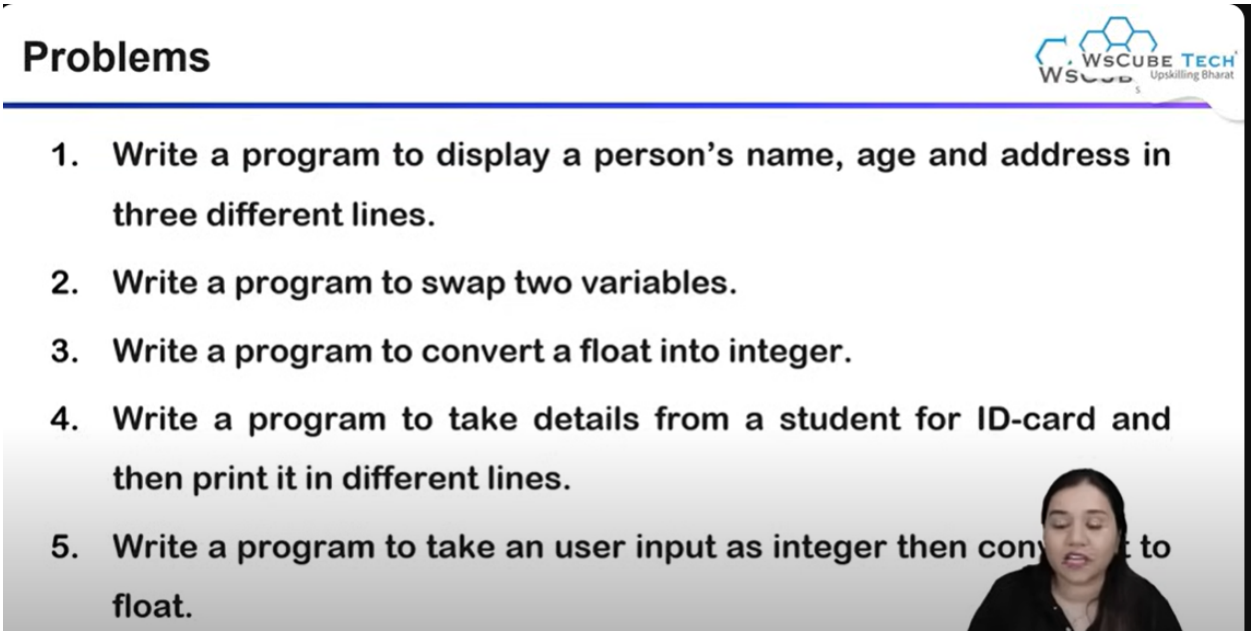
The right way above this error

type_casting2_problem.py > ...

```python
# add string and float
a="123"
b=1.23

a=float(a) #type casting
c=a+b
print(c) #correct
```

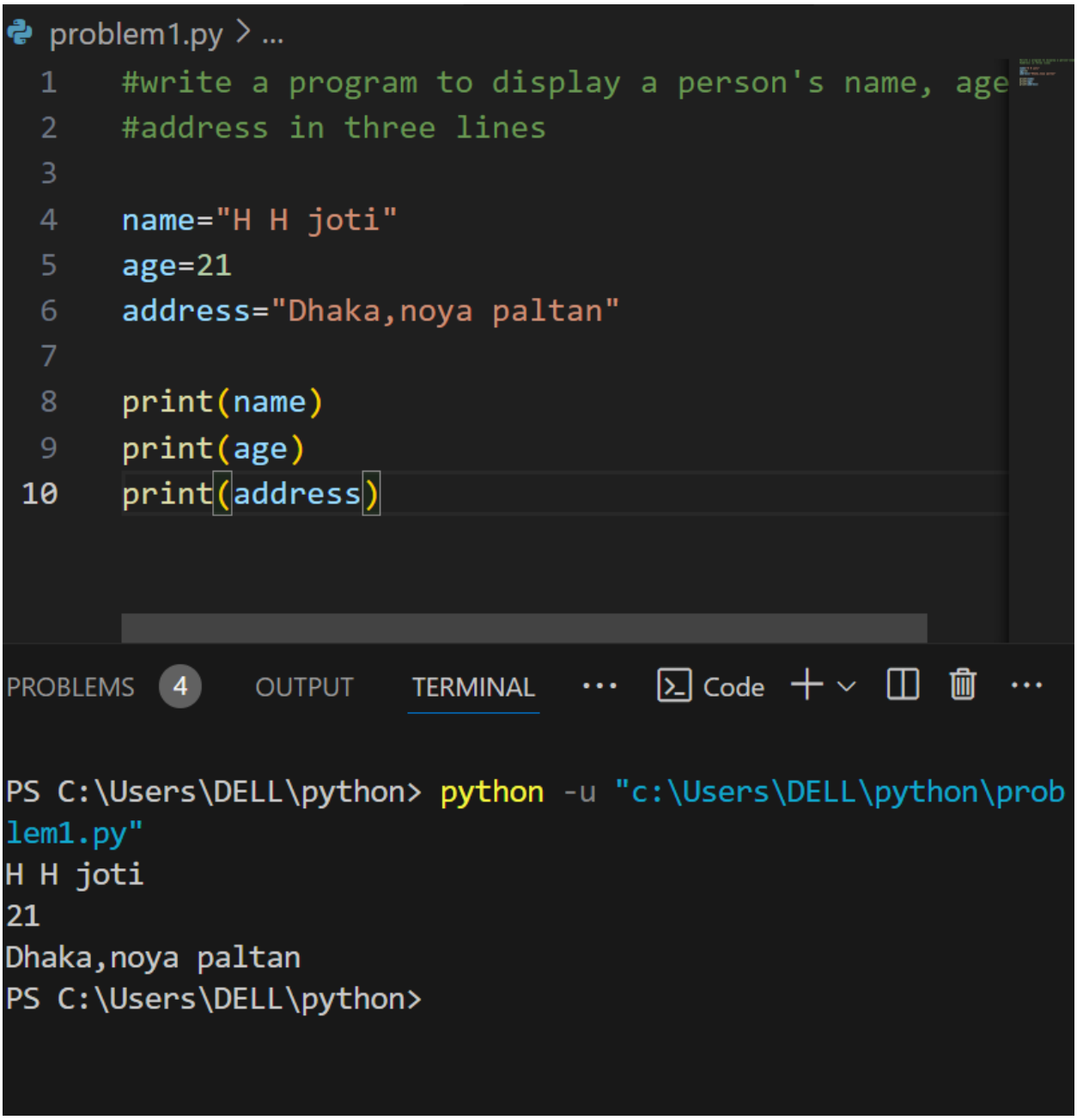
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\type_casting2_problem.py"
124.23
PS C:\Users\DELL\python>
```



Problem 1:

write a program to display a person's name, age , address in three lines



Problem 2:

write a program to swap two variable

problem2.py > ...

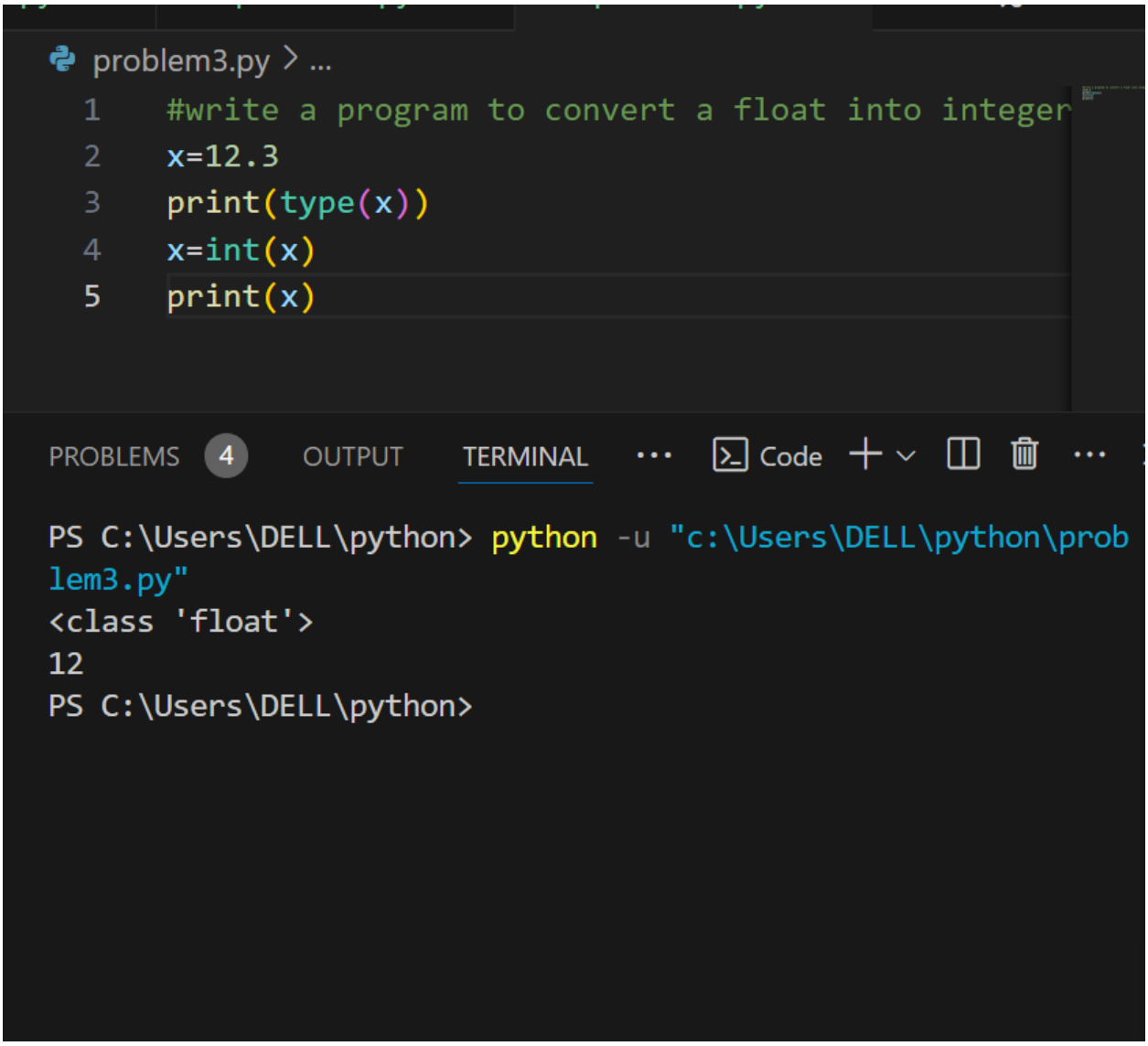
```python
#write a program to swap two variable

#1st method
x=12
y=19
print("Before swapping,print x and y value ",x ,y )
temp=x
x=y
y=temp
print("After swapping ,print x value: ",x)
print("After swapping, print y value: ",y)

#2nd method
a=70
b=80
a,b=b,a
print("After swapping x and y : ",a ,b)
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\problem2.py"
Before swapping,print x and y value  12 19
After swapping ,print x value:  19
After swapping, print y value:  12
After swapping x and y :  80 70
PS C:\Users\DELL\python>
```

problem 3:

write a program to convert a float into integer



Problem 4:

write a program to take details from a student for id card and then print it in different lines

problem4.py > ...

```python
#write a program to take details from a student
#for Id-card and then print it in different
#lines

name=input("enter the name of the student: ")
age=int(input("enter the age of the student: "))
email=input("enter the email of the student")
ph_no=input("enter the phone number of the student")

print("Student Identity Card")
print("print the name: ",name)
print("print the age : ",age)
print ("print the email: ",email)
print("print the phone number: ",ph_no)
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL ... Code

```
enter the age of the student: 21
enter the email of the student h.h.joti87@gmail.com
enter the phone number of the student 01716256333
Student Identity Card
print the name:  H H joti
print the age :  21
print the email:   h.h.joti87@gmail.com
print the phone number:   01716256333
PS C:\Users\DELL\python>
```

problem 5:

write a program to take an user input as integer then convert it to float (1)

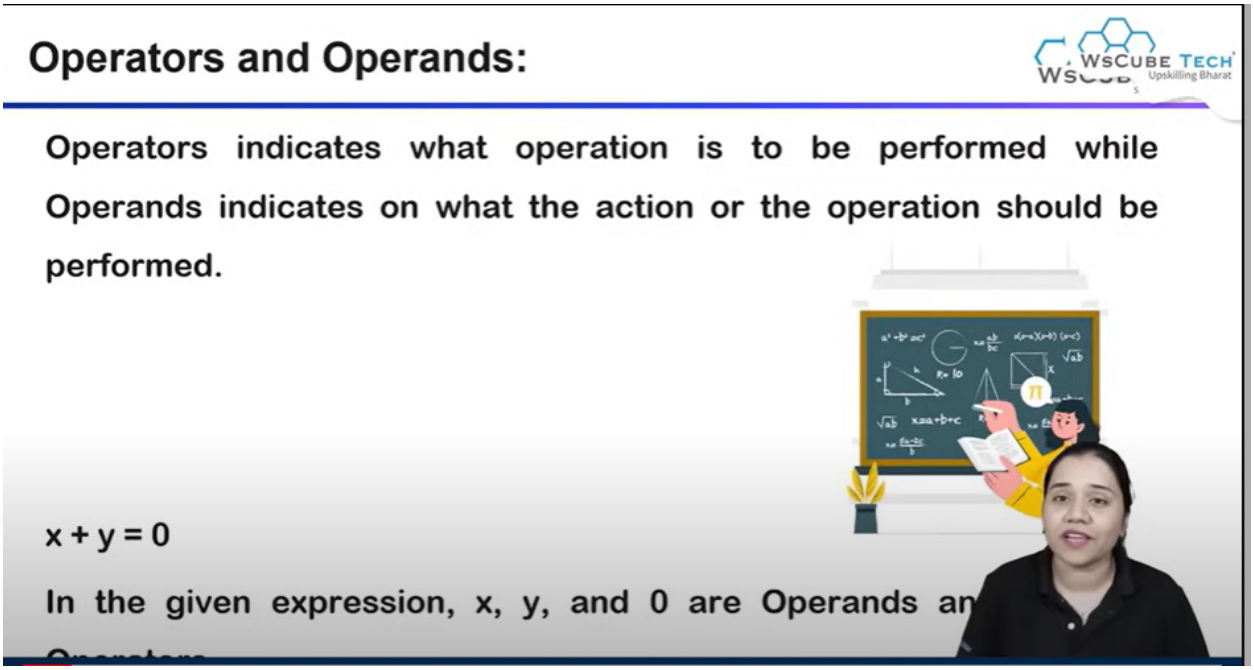
problem5.py > ...

```python
#write a program to take an user input as integer
#then convert it to float

a=int(input("enter a number here: "))
print (a)
print(type(a))

a=float(a)
print("after conversion ",a)
print(type(a))
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL ··· Code

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\problem5.py"
enter a number here: 14
14
<class 'int'>
after conversion  14.0
<class 'float'>
PS C:\Users\DELL\python>
```


Operators and
operands

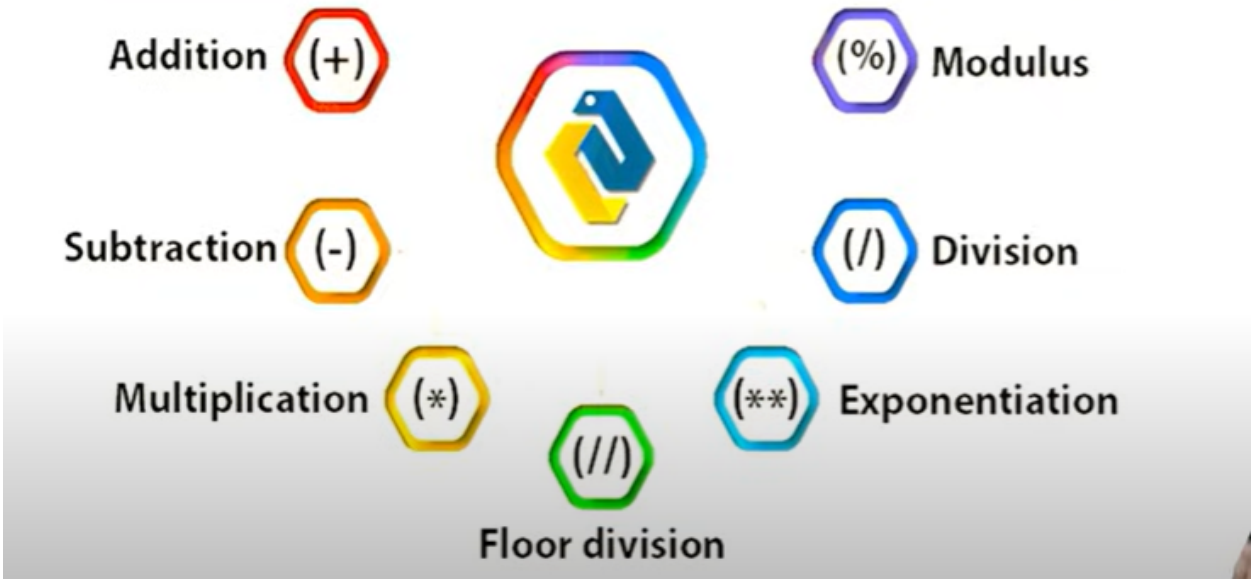

Operators and Operands:
WsCube Tech
Upskilling Bharat
Operators indicates what operation is to be performed while Operands indicates on what the action or the operation should be performed.
x + y = 0
In the given expression, x, y, and 0 are Operands an

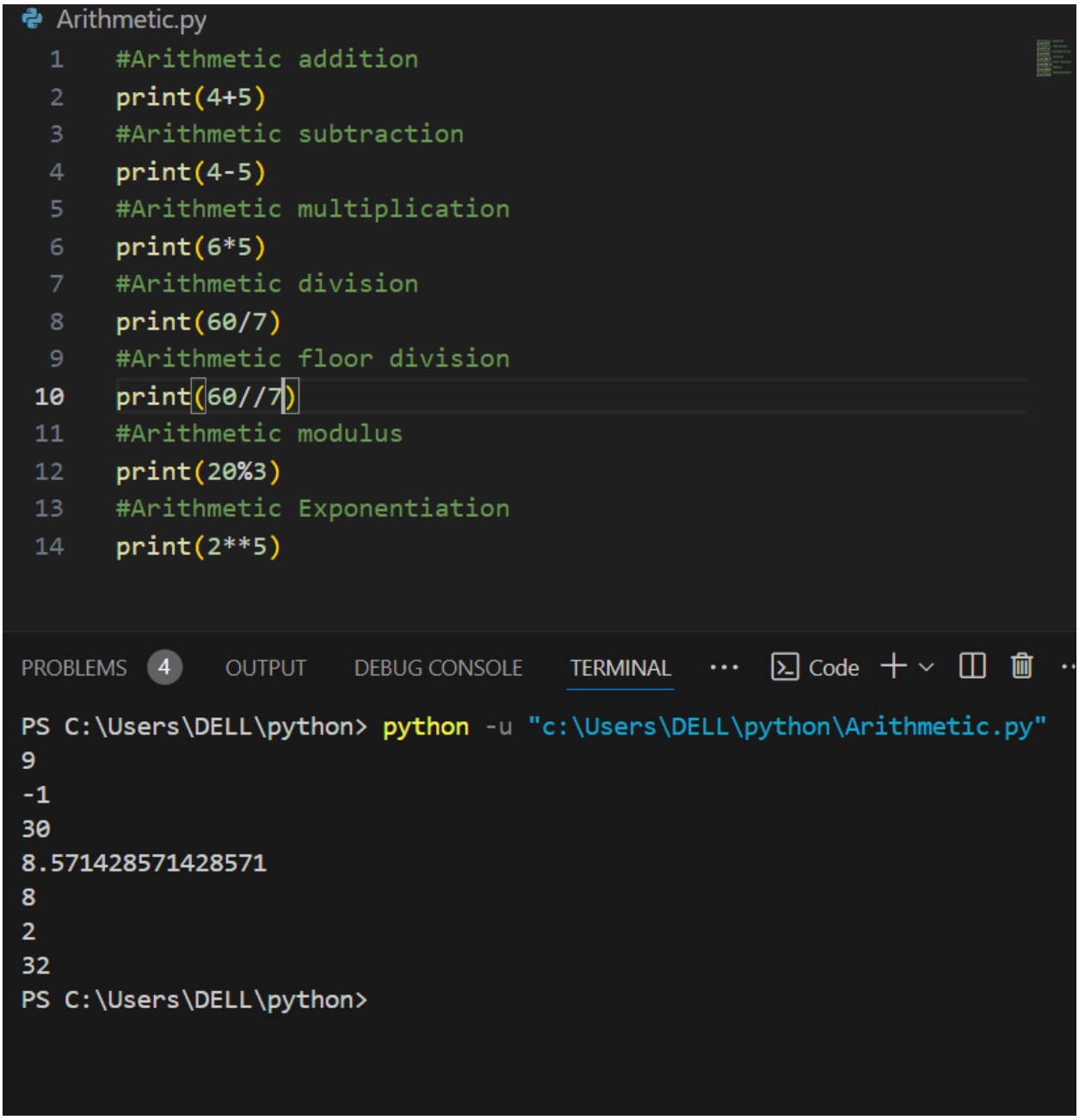
Types of operators:

Operators can be further divided into 6 categories:

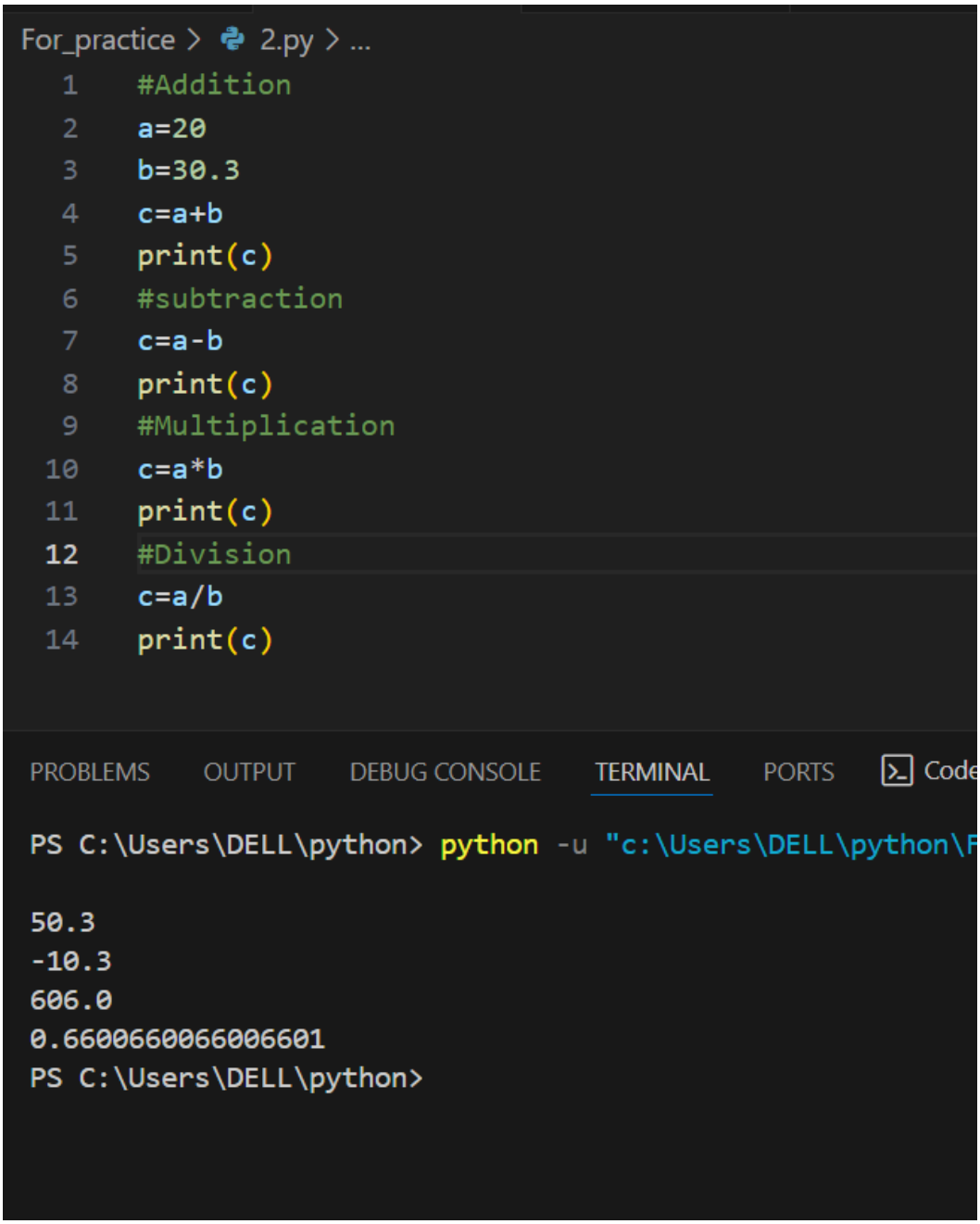
1. Arithmetic operators
2. Comparison Operators
3. Logical Operators
4. Assignment Operators
5. Identity Operators
6. Membership Operators
7. Bitwise Operators

Python Arithmetic Operator

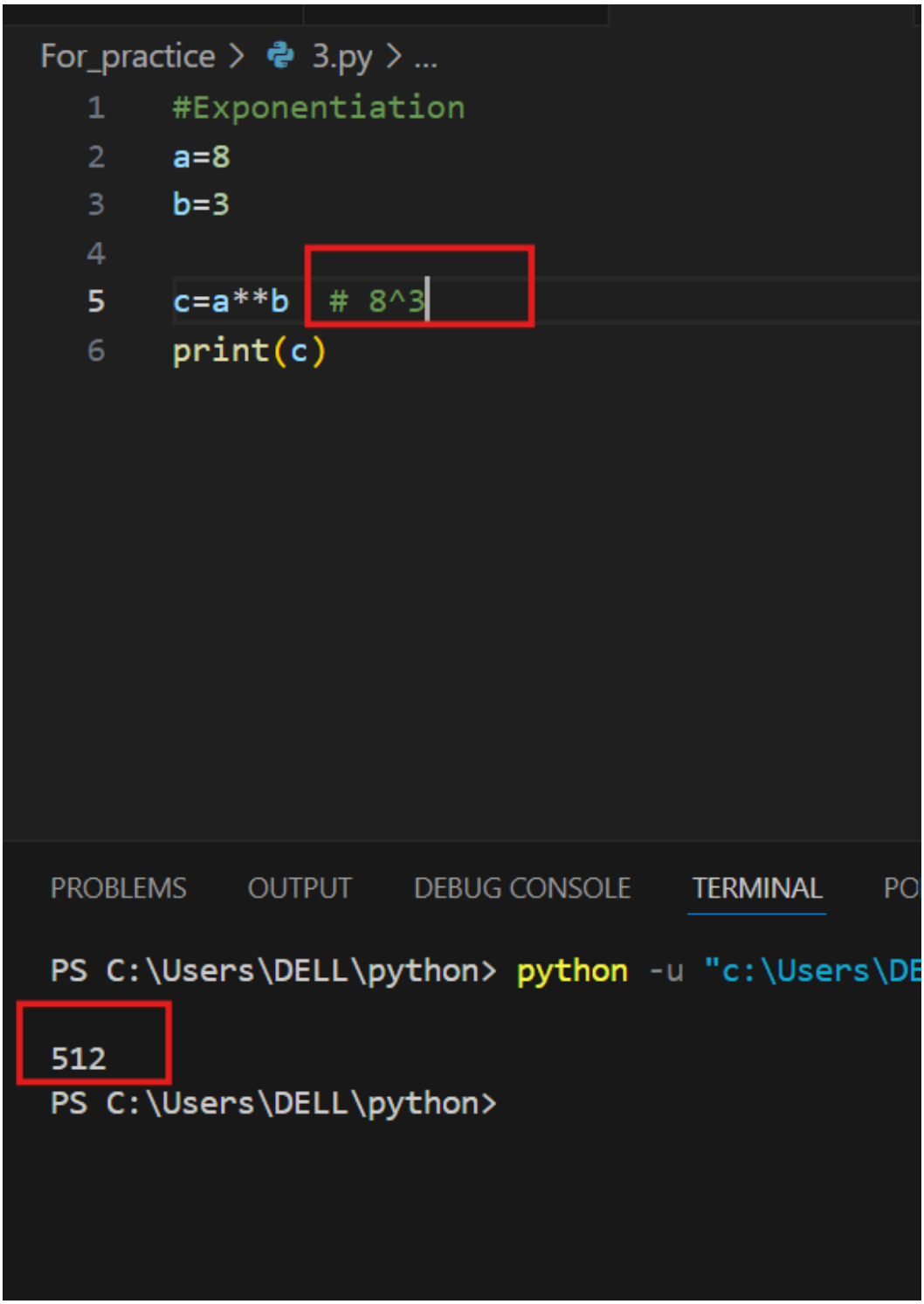




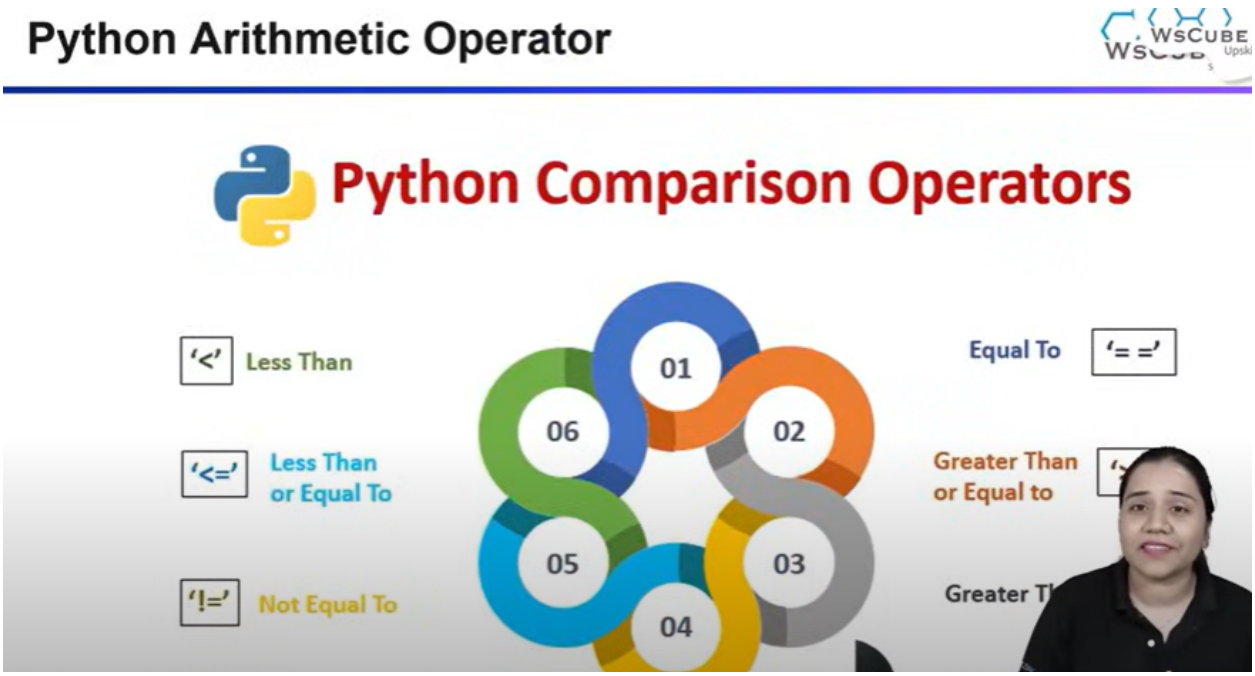
Arithmetic addition , subtraction , multiplication , division



Arithmetic exponential function


For_practice > 3.py > ...
1 #Exponentiation
2 a=8
3 b=3
4
5 c=a**b # 8^3
6 print(c)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PS C:\Users\DELL\python> python -u "c:\Users\DE
512
PS C:\Users\DELL\python>


Comparison
Operators

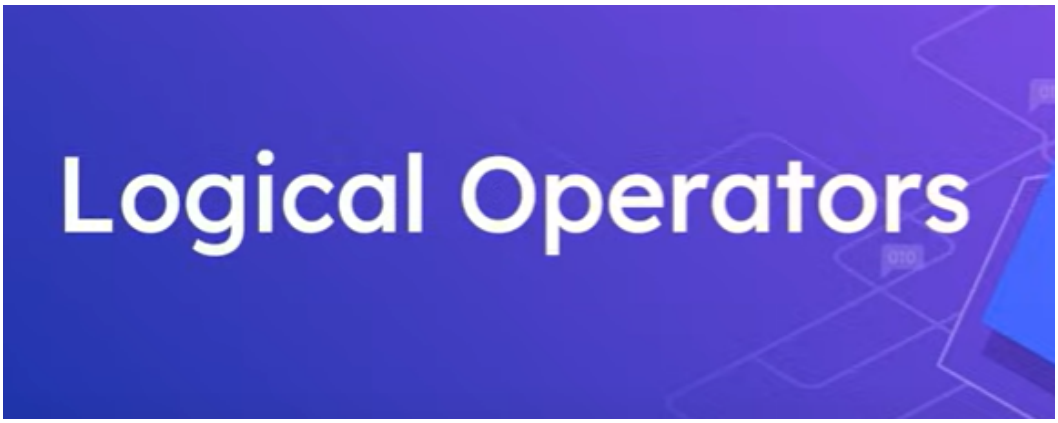

Python Arithmetic Operator
WsCube
Python Comparison Operators
'<' Less Than
'<=' Less Than or Equal To
'!=' Not Equal To
01
02
03
04
05
06
Equal To '= ='
Greater Than or Equal to

comparison.py

```python
#Less than
print(3<6)
#less than or equal to
print(3<=6)
#Not Equal to
print(3!=6)
#Equal to
print(3==3)
#Greater than or Equal to
print(3>=6)
#Greater than
print(3>6)
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\comparison.py"
True
True
True
True
False
False
PS C:\Users\DELL\python>
```



Logical operators

WsCube Tech Upskilling Bharat

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

not_operator.py

```python
#or operator
print(3>4 or 3<7)
#not operator
print(not(3>4 or 3<7))

print("\n")

#and operator
print(3<4 and 3<7)
#not operator
print(not(3<4 and 3<7))

```

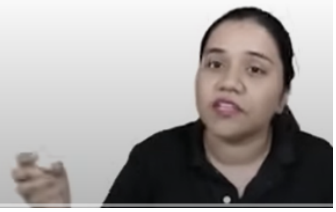
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

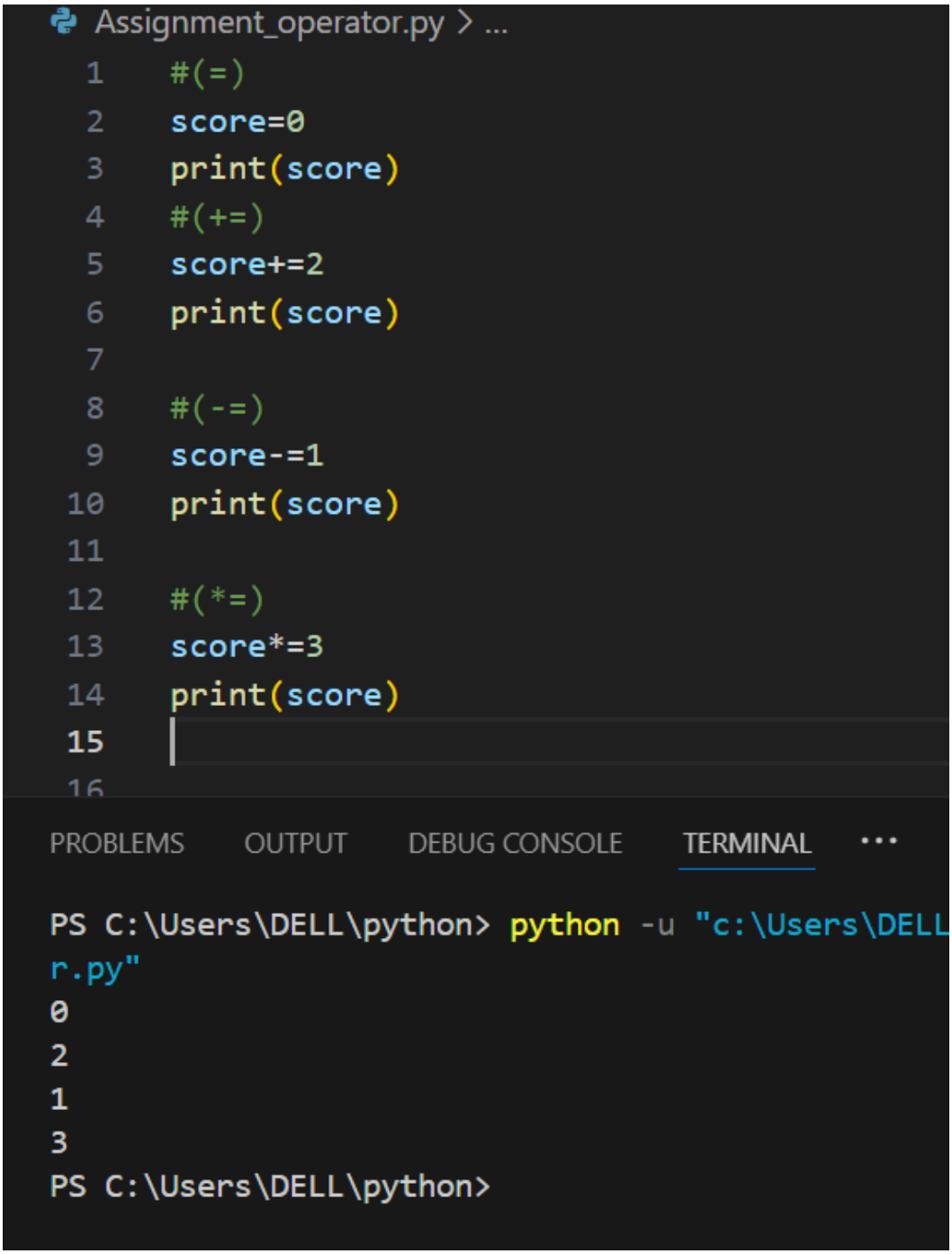
```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\not_operator.py"
True
False

True
False
PS C:\Users\DELL\python>
```

Assignment Operators

- Assignment operators are used in Python to assign values to variables.
- a = 6 is a simple assignment operator that assigns the value 6 on the right to the variable a on the left.

Operator	Example	Equivalent to
=	x = 6	x = 6
+=	x += 6	x = x + 6
-=	x -= 6	x = x - 6
*=	x *= 6	x = x * 6




Assignment_operator.py > ...
#(=)
score=0
print(score)
#(+=)
score+=2
print(score)
#(-=)
score-=1
print(score)
#(*=)
score*=3
print(score)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PS C:\Users\DELL\python> python -u "c:\Users\DELL
r.py"
0
2
1
3
PS C:\Users\DELL\python>

Identity Operators

Identity operators are used to compare items to see if they are the same object with the same memory address.

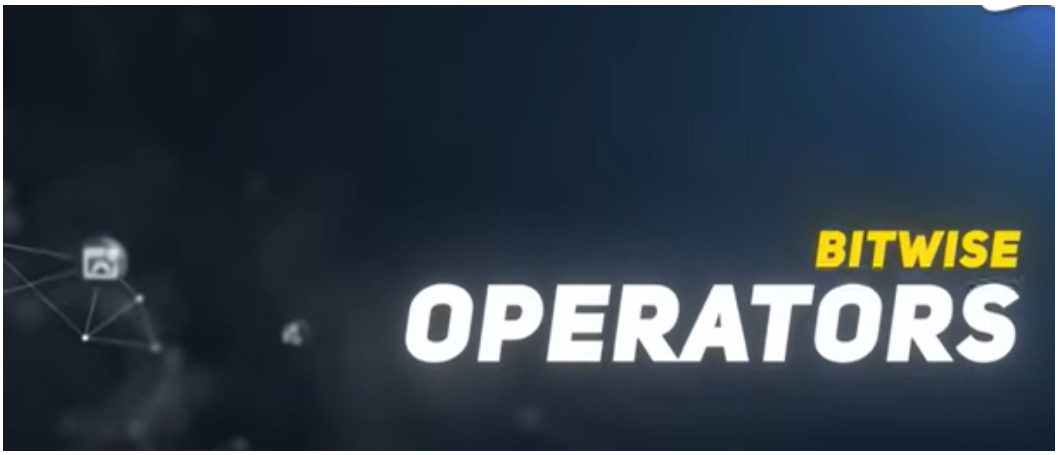
Types:
1. Is
2. Is not

identity_operator.py > ...

```python
#identity operator(is / is not)
a=1234 #int type
b="1234" #string type
print(a is b)
print(a is not b)

print("\n")
p=1234 #int type
q=1234 #int type
print(p is q)
print(p is not q)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\pyt
py"
False
True


True
False
PS C:\Users\DELL\python>
```

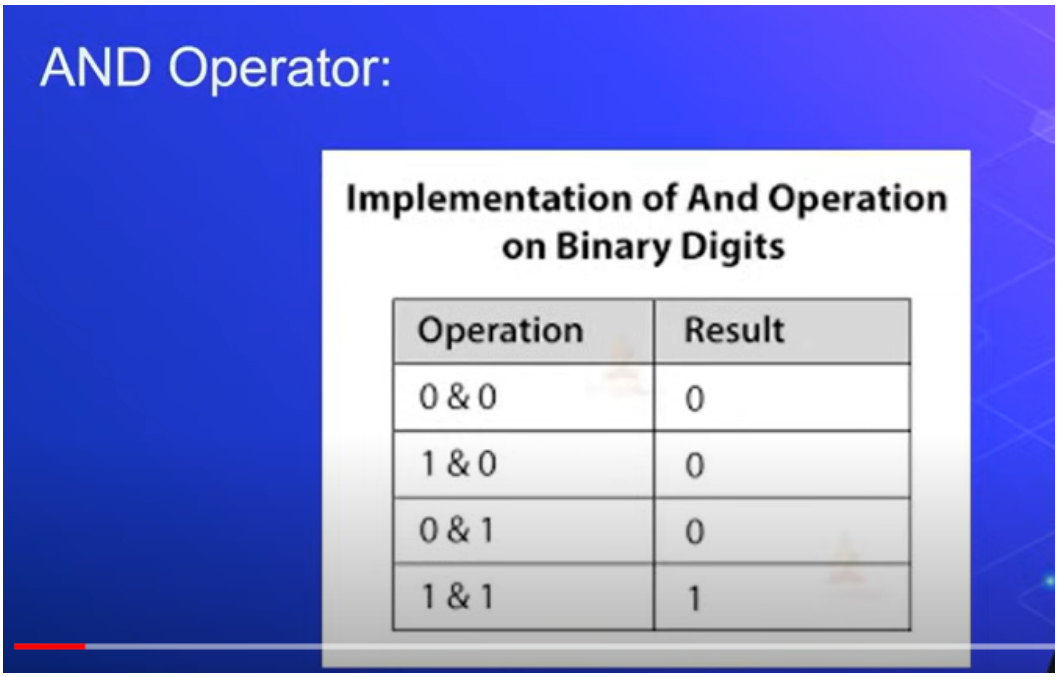


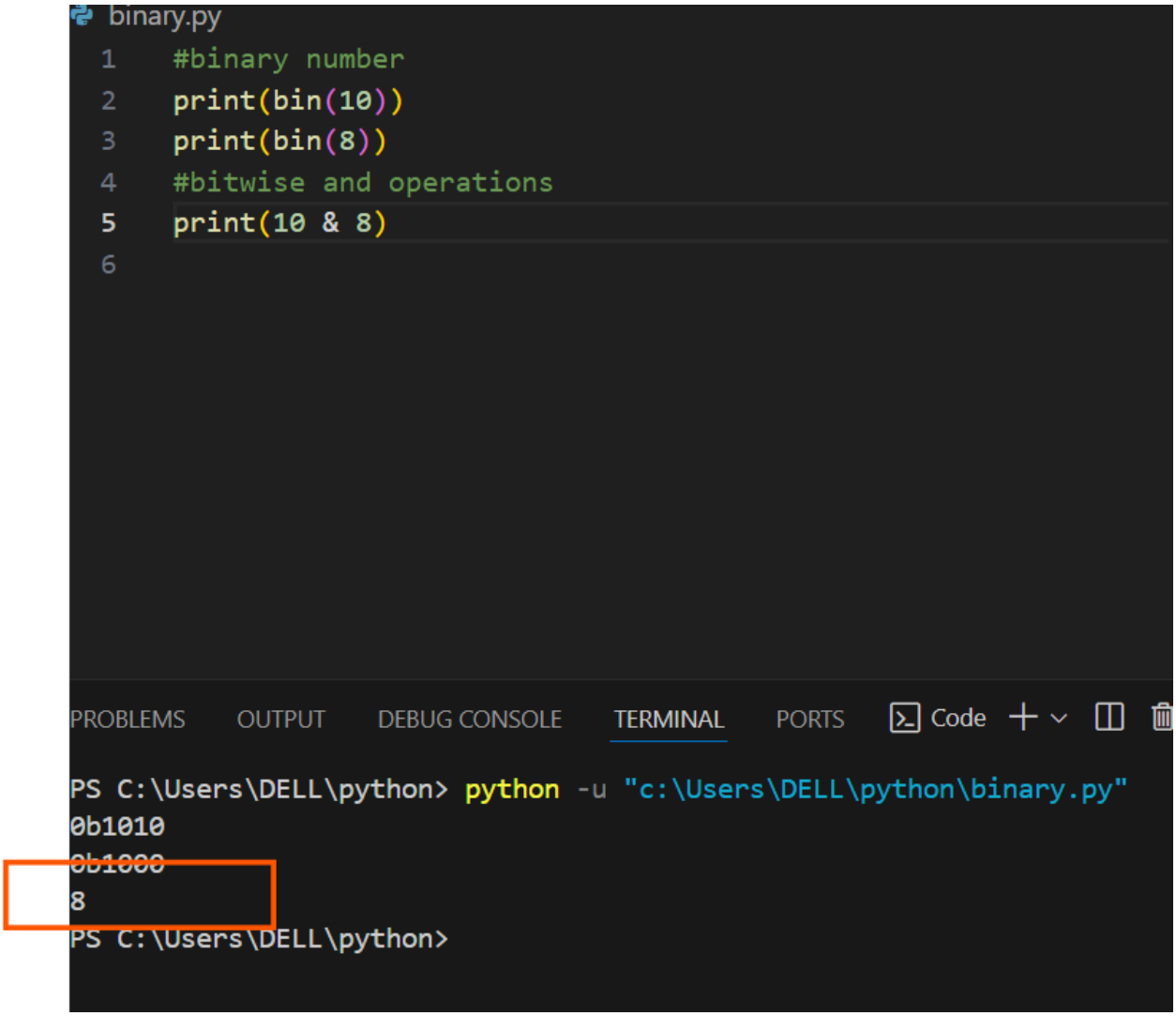
Bitwise Operators:

These Operators are used to compare the Binary numbers
Types:

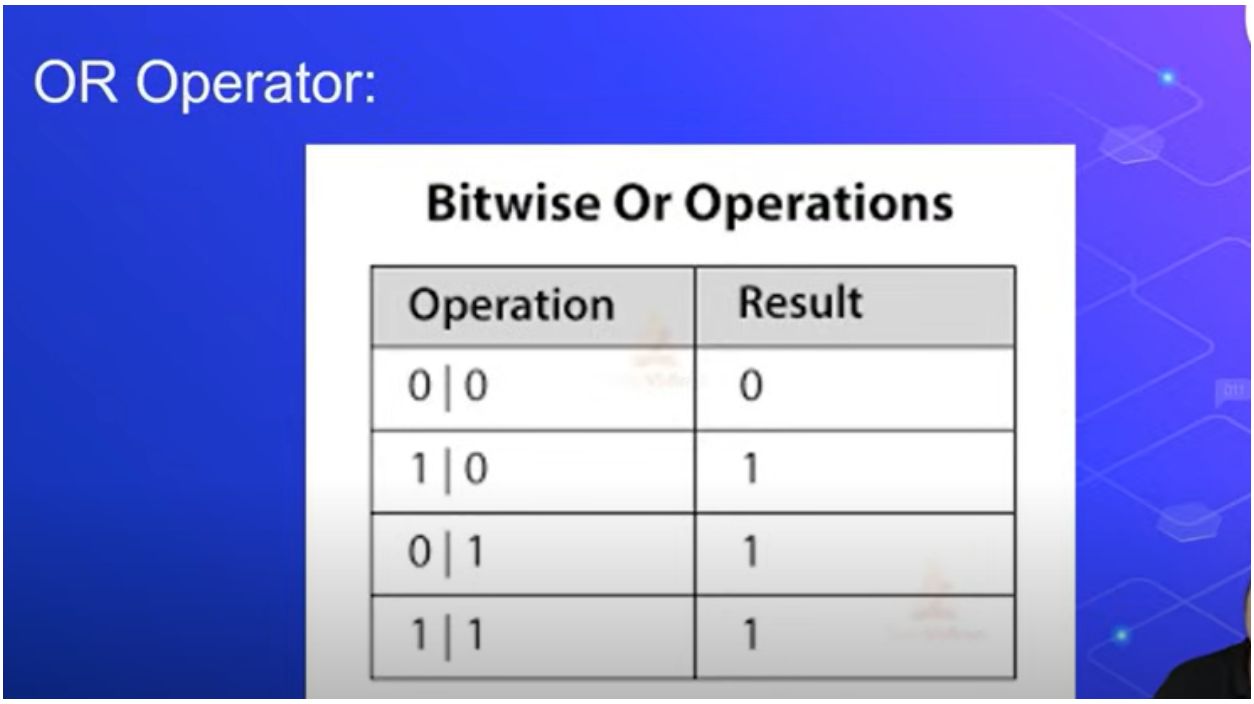
1. AND (&) Operator
2. OR (|)Operator
3. XOR (^) Operator
4. << zero fill left shift
5. >> zero fill right shift

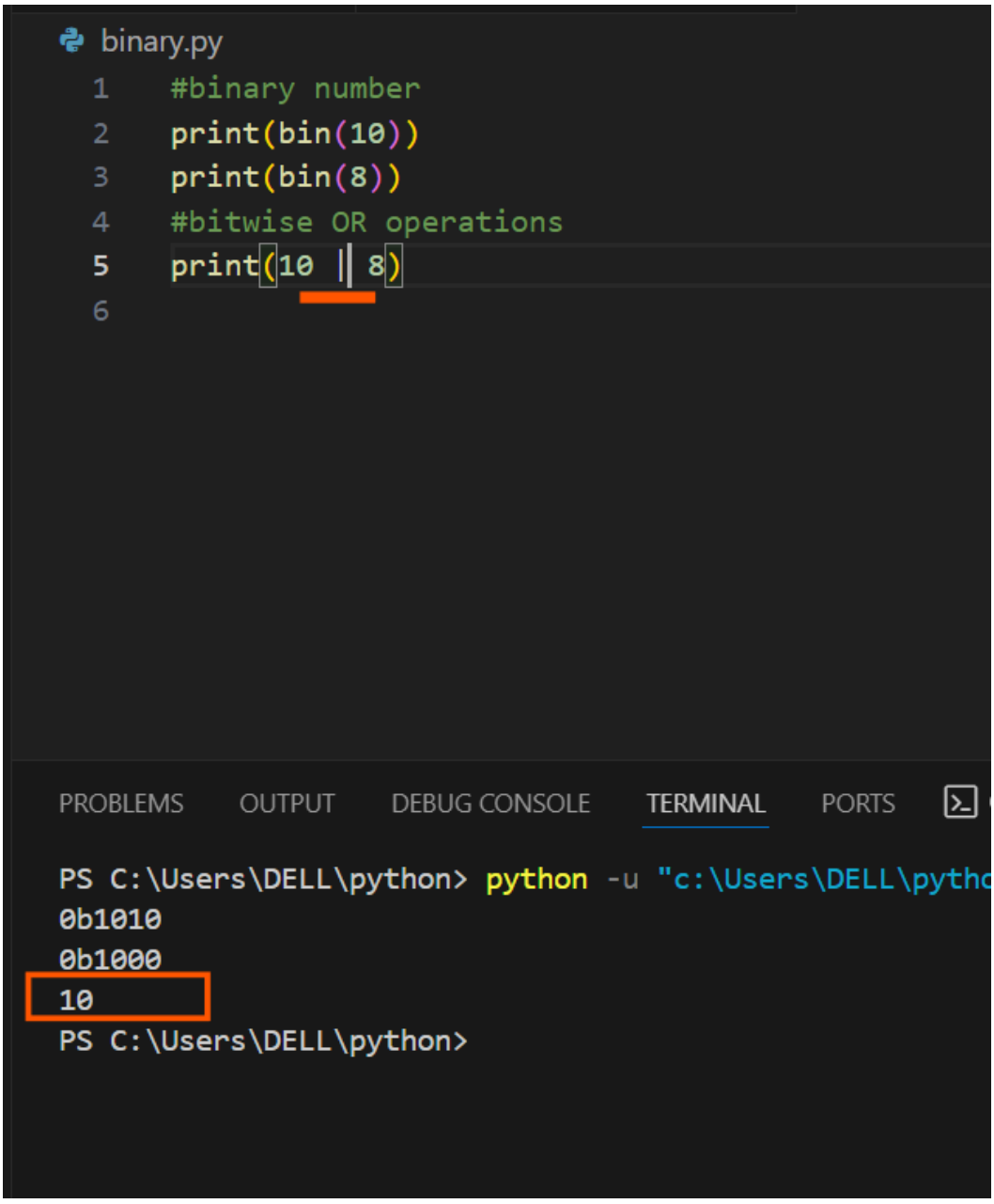
AND OPERATOR


AND Operator:
Implementation of And Operation on Binary Digits
Operation
Result
0 & 0
0
1 & 0
0
0 & 1
0
1 & 1
1

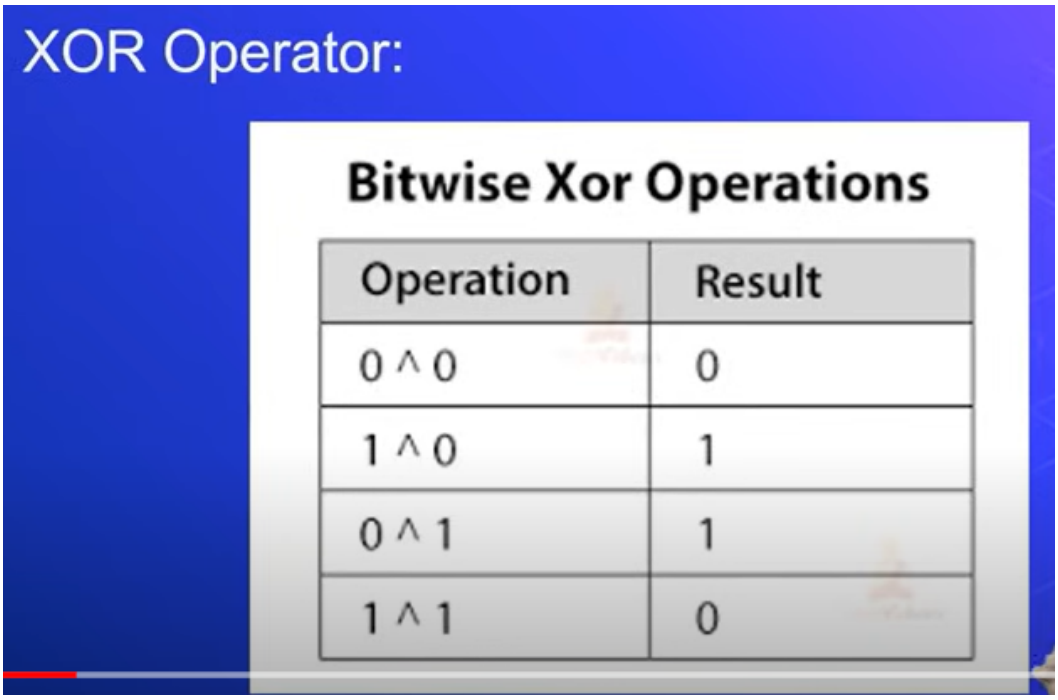

binary.py
1 #binary number
2 print(bin(10))
3 print(bin(8))
4 #bitwise and operations
5 print(10 & 8)
6
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Code
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\binary.py"
0b1010
0b1000
8
PS C:\Users\DELL\python>

OR OPERATOR


OR Operator:
Bitwise Or Operations
Operation
Result
0 | 0
0
1 | 0
1
0 | 1
1
1 | 1
1



XOR OPERATOR


XOR Operator:
Bitwise Xor Operations
Operation
Result
0 ^ 0
0
1 ^ 0
1
0 ^ 1
1
1 ^ 1
0

binary.py

```
1   #binary number
2   print(bin(10))
3   print(bin(8))
4   #bitwise XOR operations
5   print(10 ^ 8)
6
```

PROBLEMS OUTPUT DEBUG CONSOLE

```
PS C:\Users\DELL\python> python -u
0b1010
0b1000
2
PS C:\Users\DELL\python>
```

ZERO FILL RIGHT SHIFT (>>)

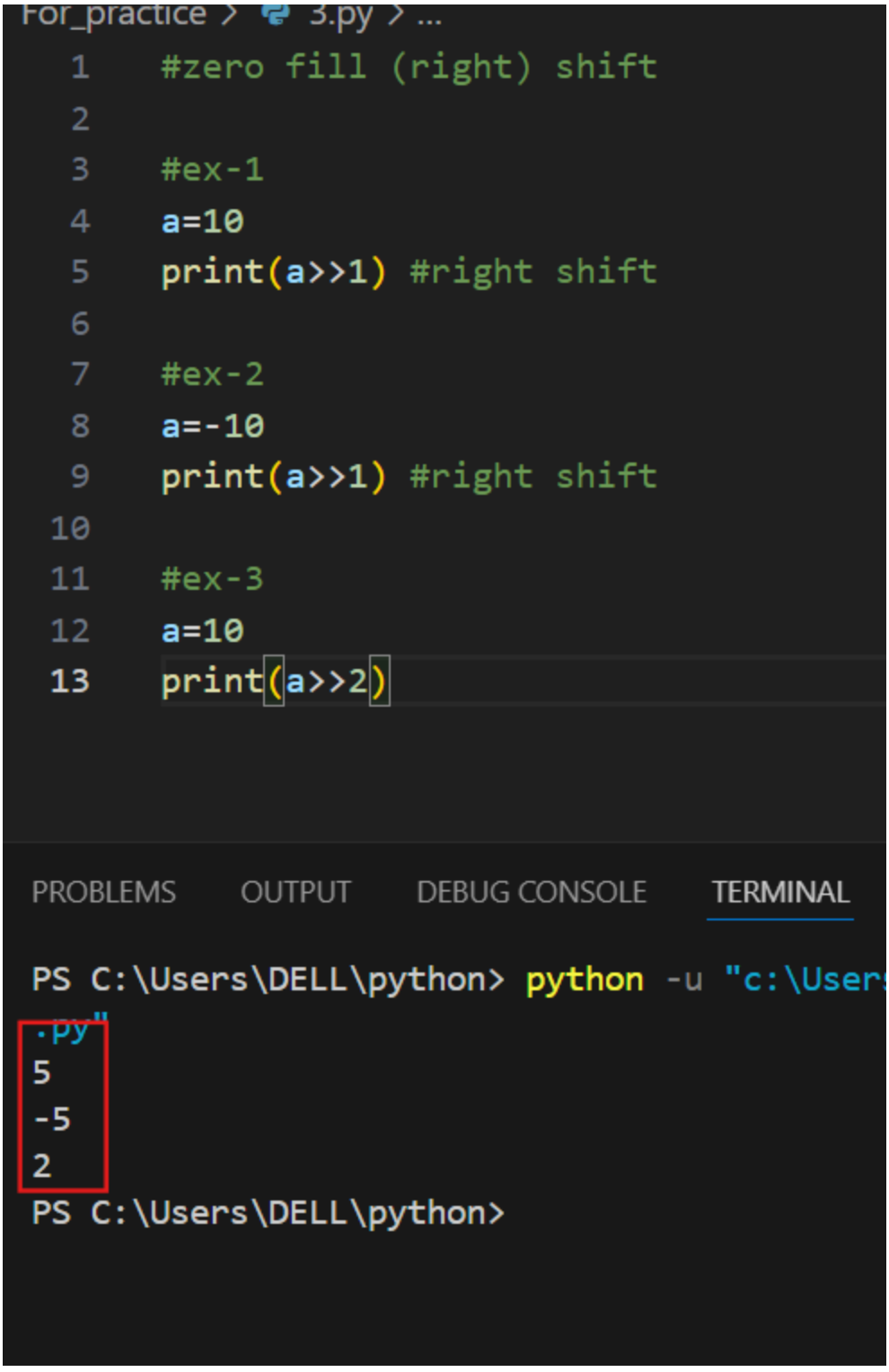
Python Bitwise Right Shift

Shifts the bits of the number to the right and fills 0 on voids left(fills 1 in the case of a negative number) as a result. Similar effect as of dividing the number with some power of two.

```
Example 1:
a = 10 = 0000 1010 (Binary)
a >> 1 = 0000 0101 = 5

Example 2:
a = -10 = 1111 0110 (Binary)
a >> 1 = 1111 1011 = -5
```

negative

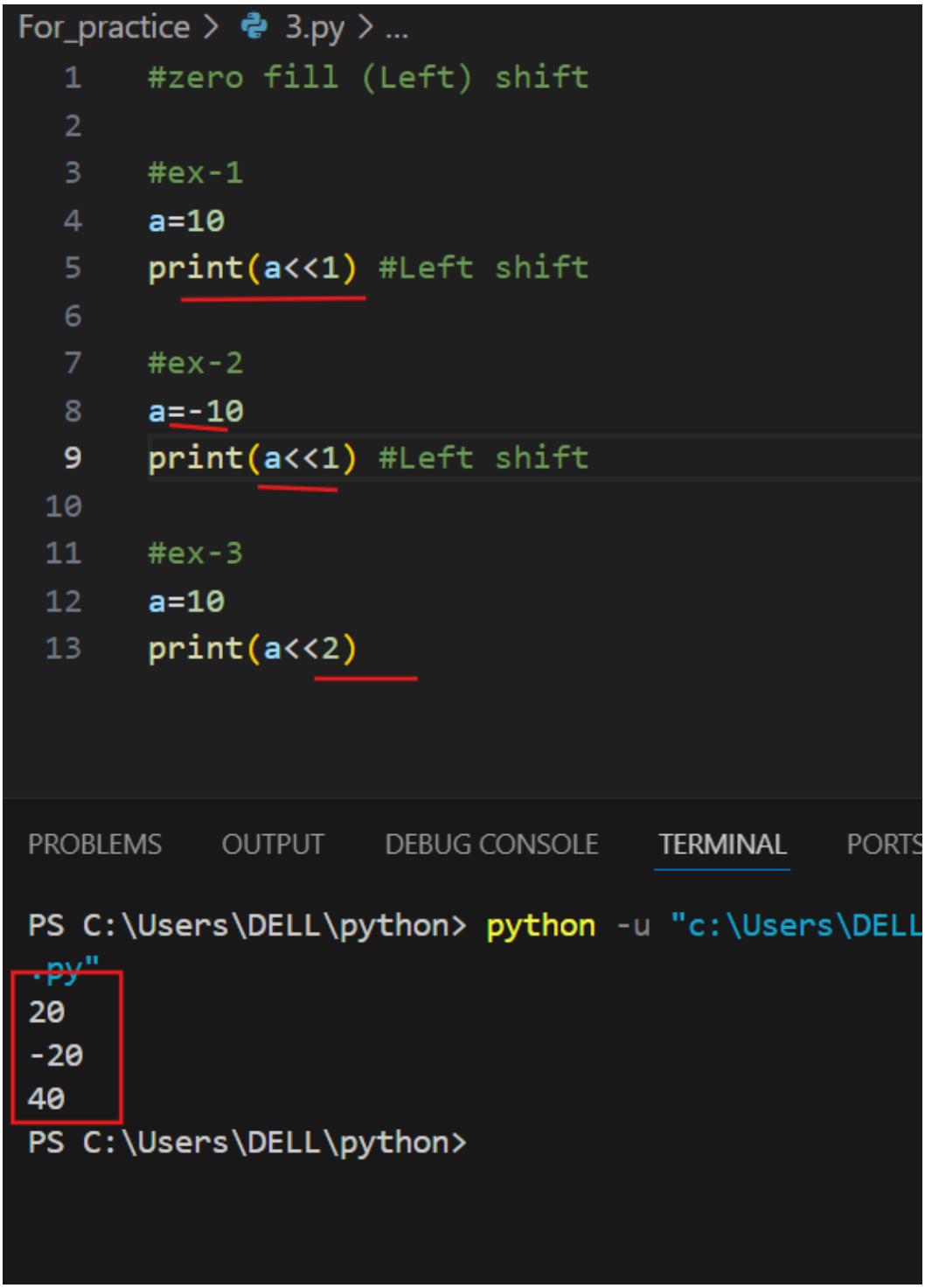
code (right shift)


For_practice > 3.py > ...
1 #zero fill (right) shift
2
3 #ex-1
4 a=10
5 print(a>>1) #right shift
6
7 #ex-2
8 a=-10
9 print(a>>1) #right shift
10
11 #ex-3
12 a=10
13 print(a>>2)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PS C:\Users\DELL\python> python -u "c:\Users
.py"
5
-5
2
PS C:\Users\DELL\python>

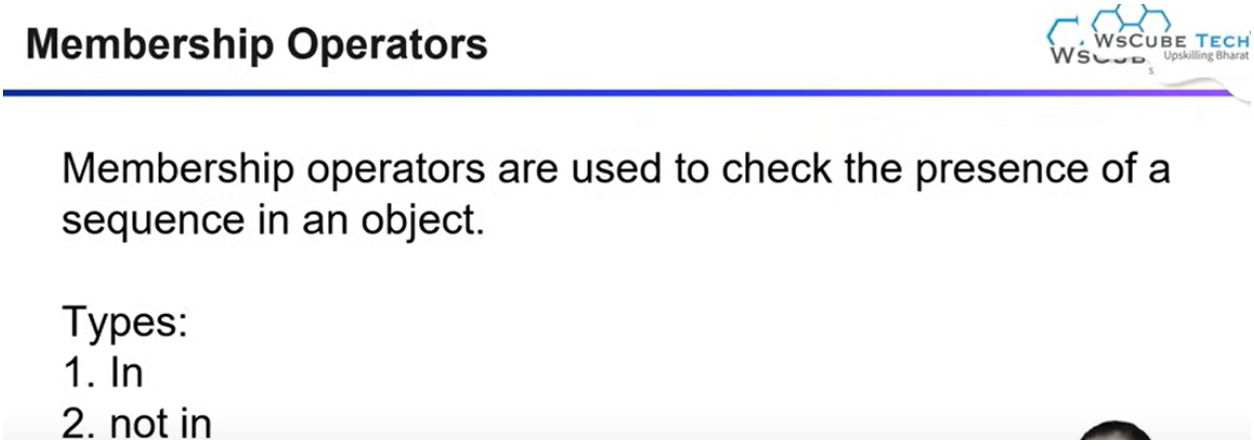
ZERO FILL LEFT SHIFT (<<)

```
Example 1:
a = 5 = 0000 0101 (Binary)
a << 1 = 0000 1010 = 10
a << 2 = 0001 0100 = 20

Example 2:
b = -10 = 1111 0110 (Binary)
b << 1 = 1110 1100 = -20
b << 2 = 1101 1000 = -40
```

code(Left shift)


For_practice > 3.py > ...
#zero fill (Left) shift
#ex-1
a=10
print(a<<1) #Left shift
#ex-2
a=-10
print(a<<1) #Left shift
#ex-3
a=10
print(a<<2)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\DELL\python> python -u "c:\Users\DELL
.py"
20
-20
40
PS C:\Users\DELL\python>

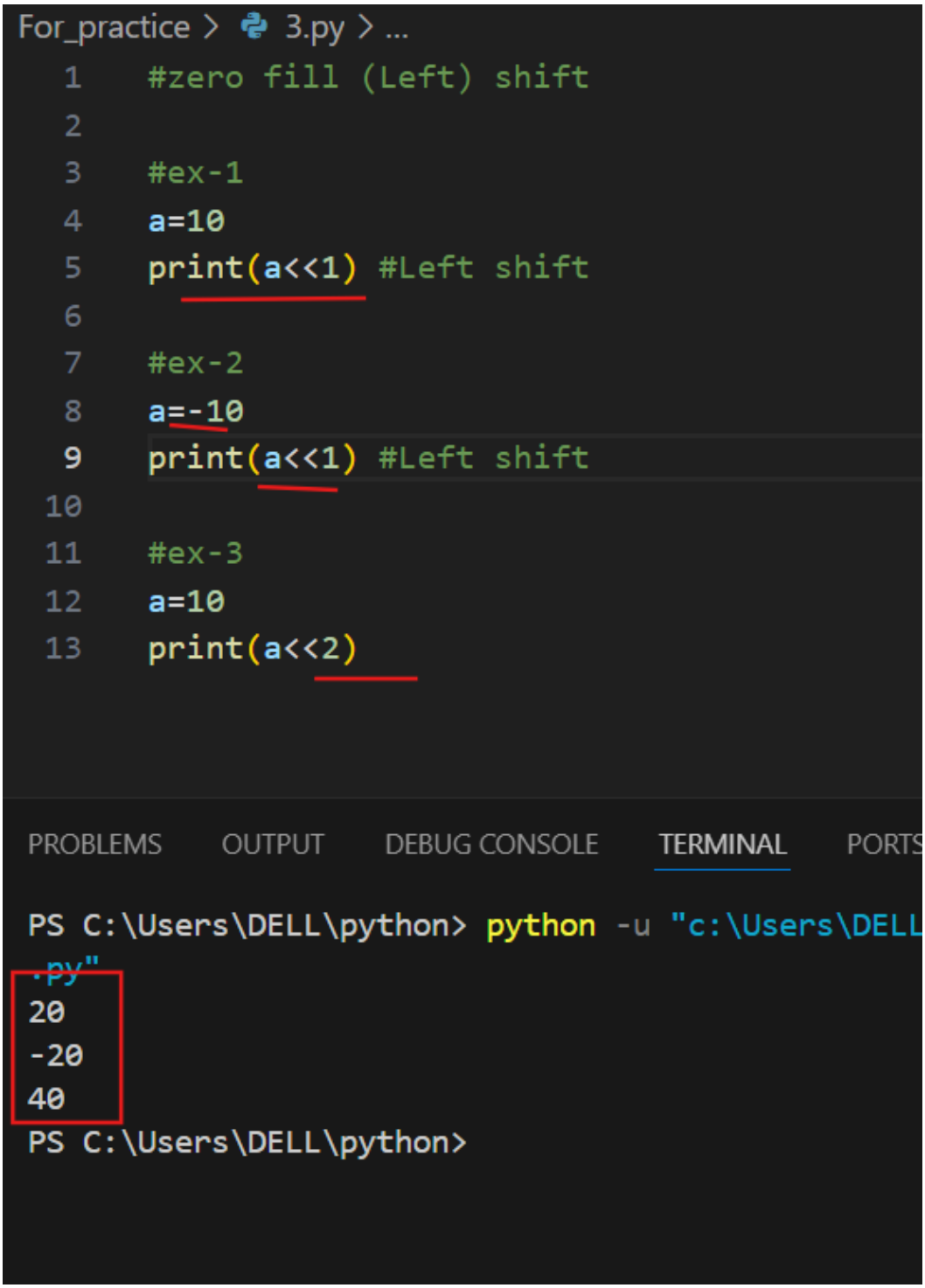
Membership operator(in and not in)

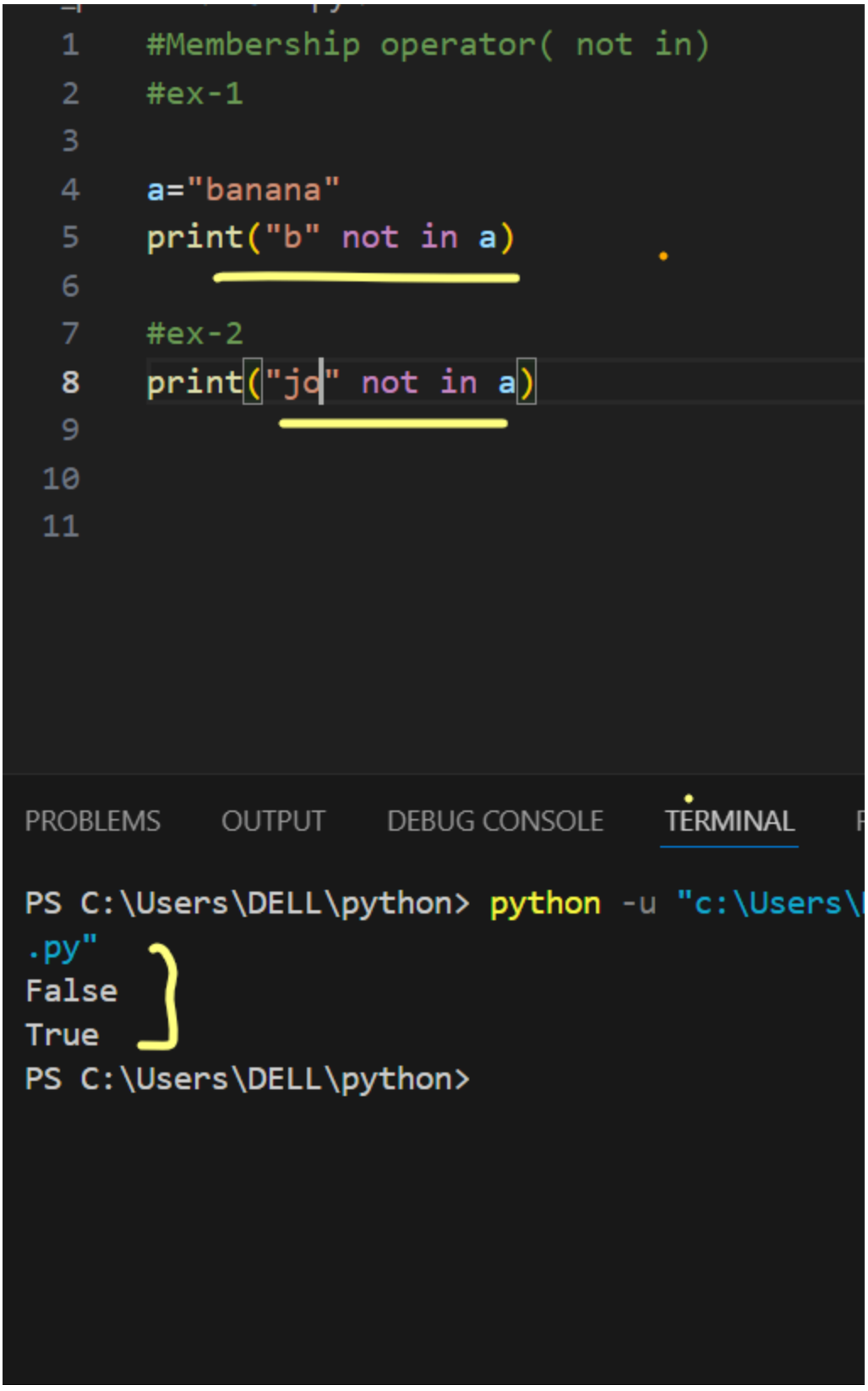


IN (Membership operator)

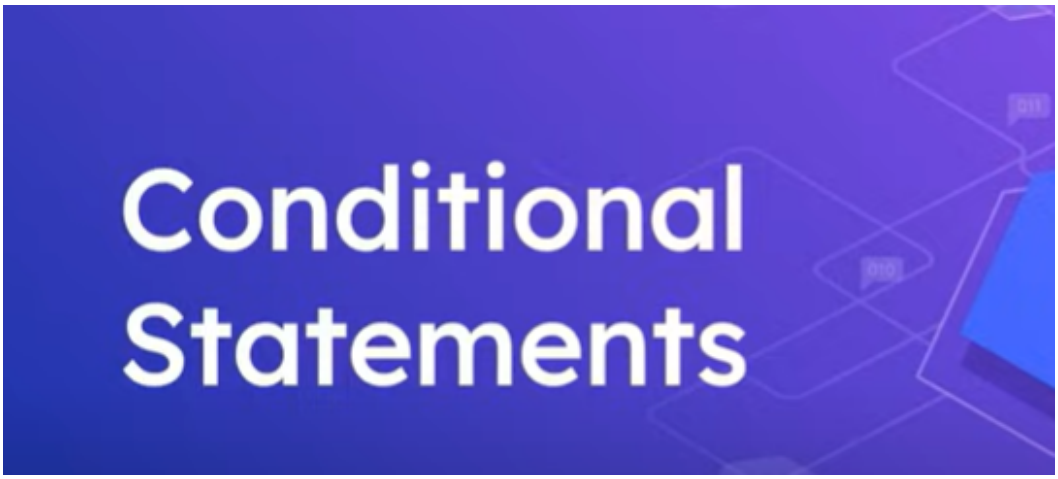
Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y

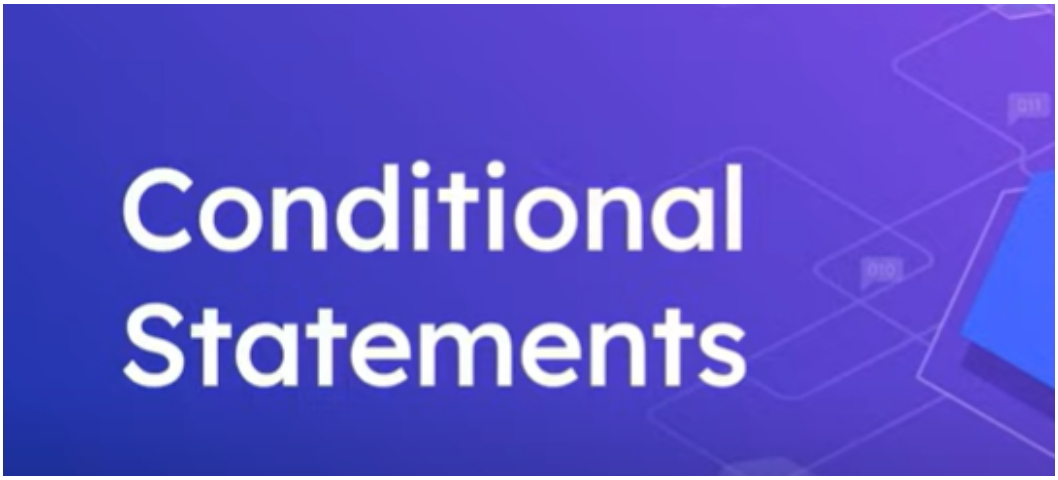
Code


For_practice > 3.py > ...
#zero fill (Left) shift
#ex-1
a=10
print(a<<1) #Left shift
#ex-2
a=-10
print(a<<1) #Left shift
#ex-3
a=10
print(a<<2)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\DELL\python> python -u "c:\Users\DELL
.py"
20
-20
40
PS C:\Users\DELL\python>

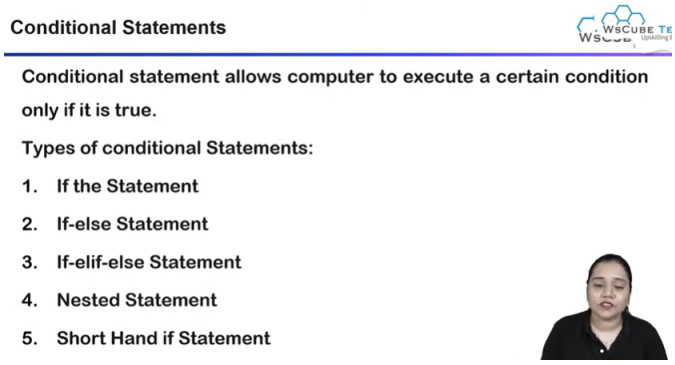

#Membership operator(not in)
#ex-1
a="banana"
print("b" not in a)
#ex-2
print("jo" not in a)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PS C:\Users\DELL\python> python -u "c:\Users\
.py"
False
True
PS C:\Users\DELL\python>

Conditional statement

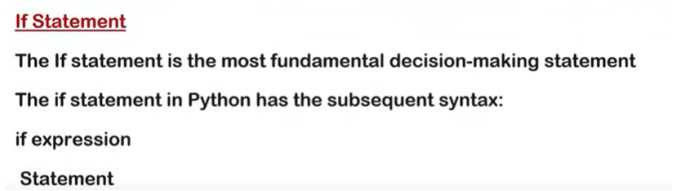




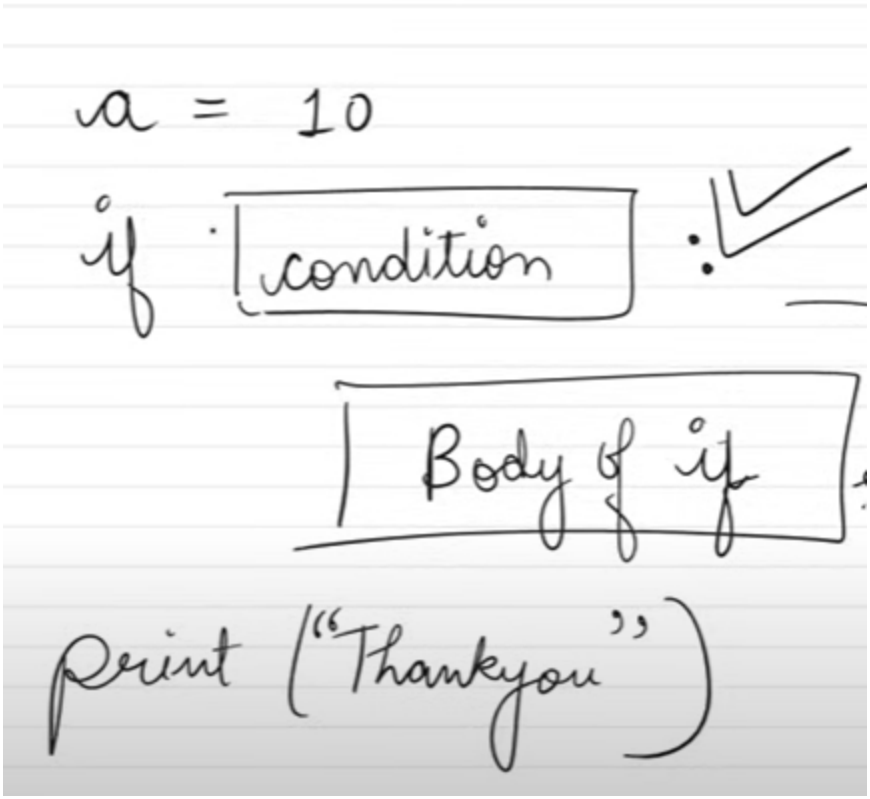
Types of conditional statement



1. If statement



Body of (If statement)



An `if` statement executes a block of code only when the specified condition is met.

Syntax

```
if condition:
    # body of if statement
```

If statement example -1

```python
#if statement example
marks = 87
if marks>=90:
    print("you will get a mobile phone")


print("Thank you!")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\py
b is greater than a
```

If statement example-2

whitespace error

Indentation

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

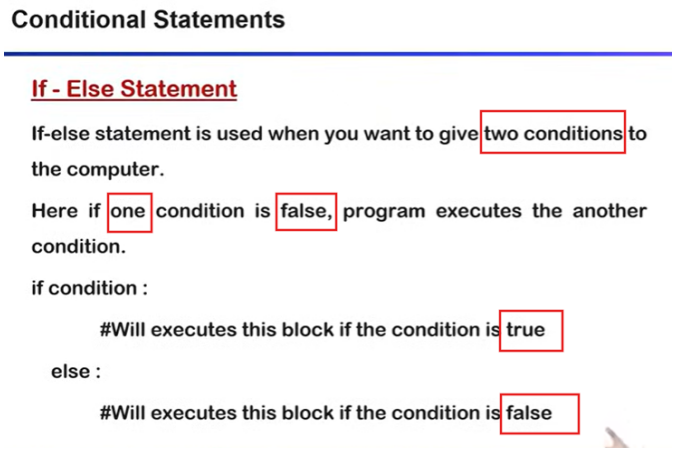
Example

If statement, without indentation (will raise an error):

```
a = 33
b = 200
if b > a:
print("b is greater than a") # you will get an error
```

2.IF-Else statement

If-else condition

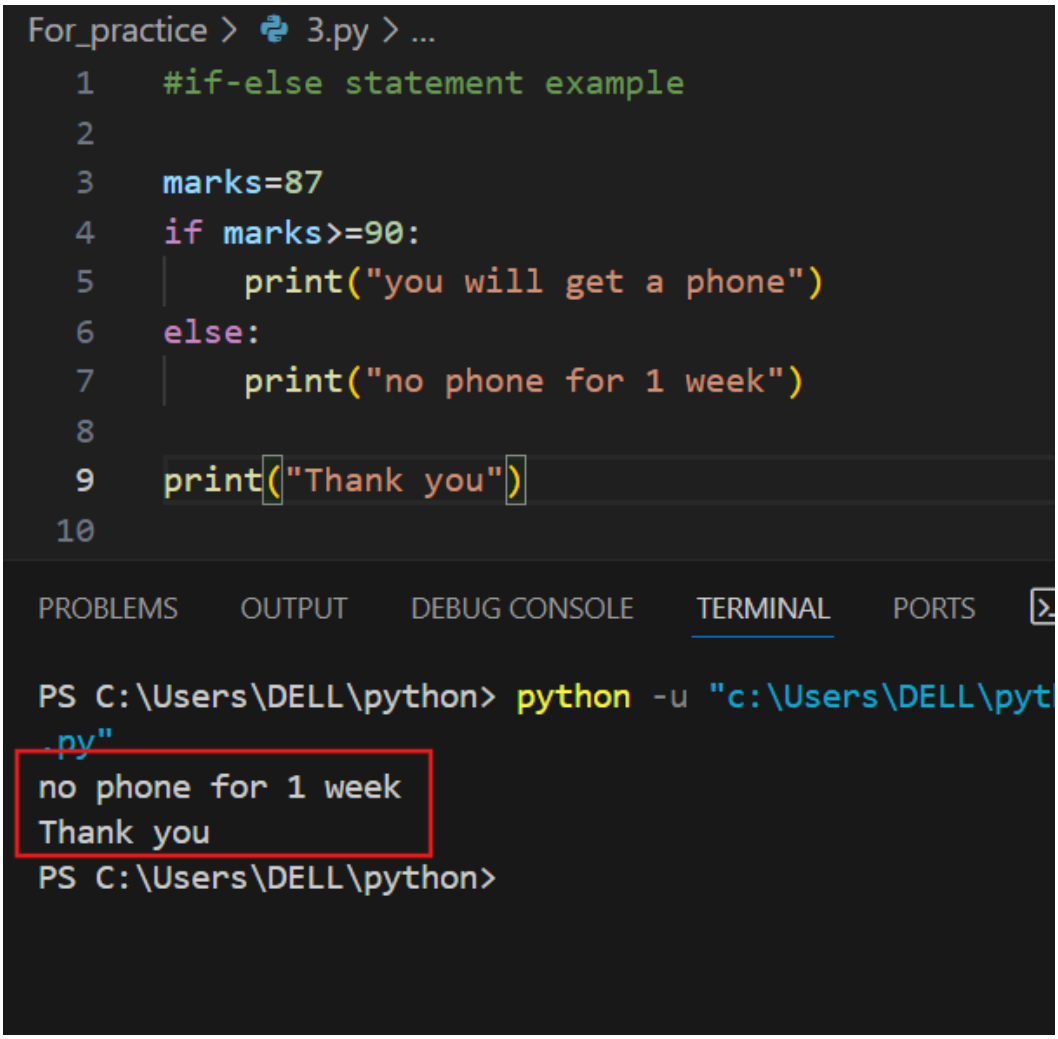


If-else syntax

Syntax

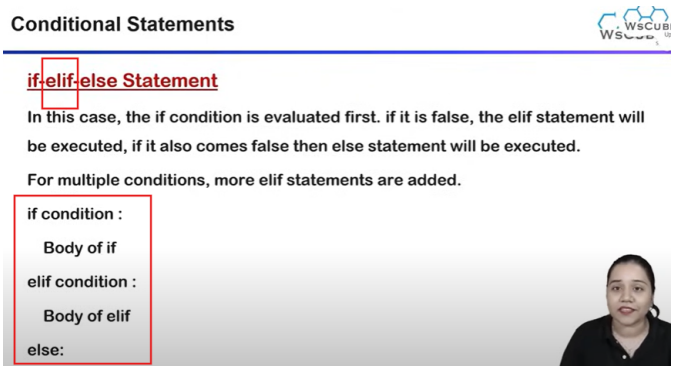
```
if condition:
    # body of if statement
else:
    # body of else statement
```

If-else code


For_practice > 3.py > ...
1 #if-else statement example
2
3 marks=87
4 if marks>=90:
5 print("you will get a phone")
6 else:
7 print("no phone for 1 week")
8
9 print("Thank you")
10
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\DELL\python> python -u "c:\Users\DELL\pyt
.py"
no phone for 1 week
Thank you
PS C:\Users\DELL\python>

If-elif-else

if-elif-else condition

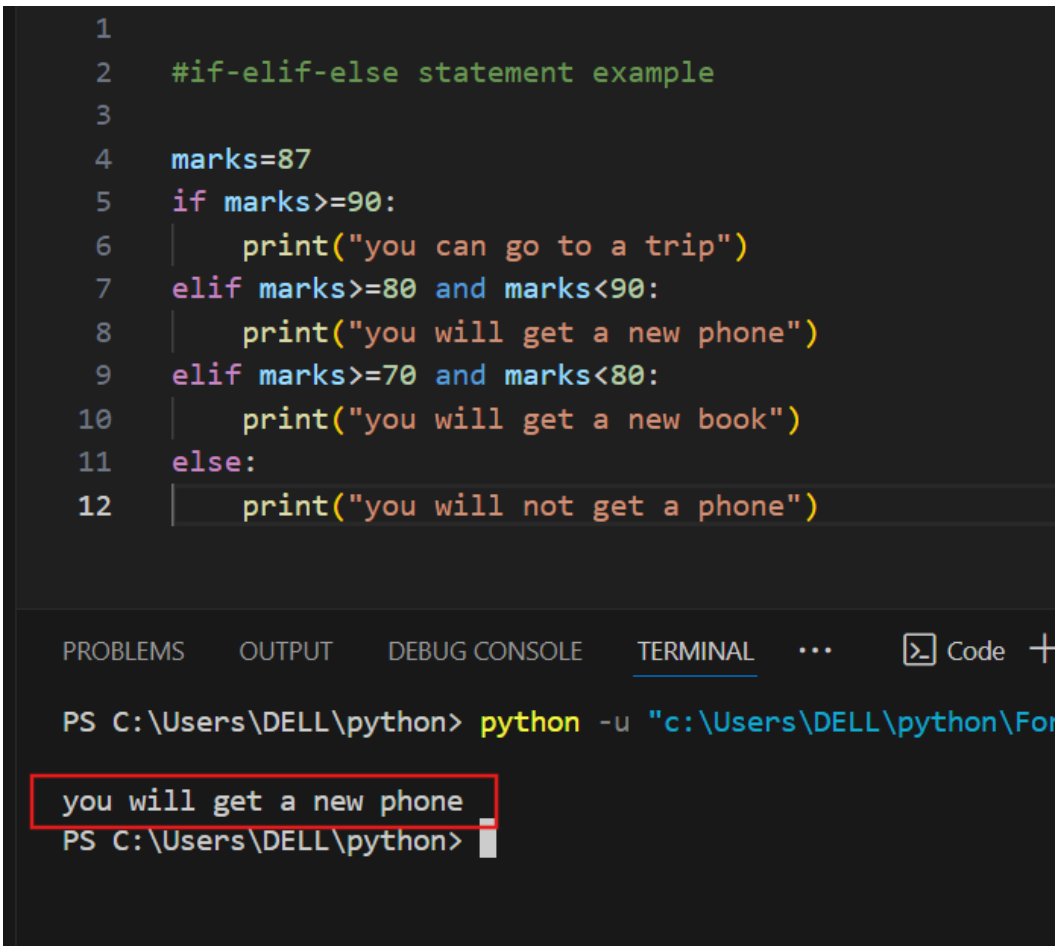


syntax

Syntax

```
if condition1:
    # code block 1

elif condition2:
    # code block 2

else:
    # code block 3
```

code example



Nested if statement

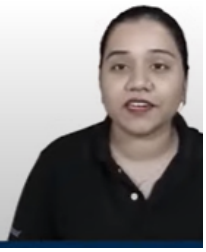
Nested if statement

Conditional Statements

Nested IF Statement

A Nested IF statement is one in which an If statement is nestled inside another If statement. This is used when a variable must be processed more than once. The Nested if statement in Python has the following syntax:

```
if (condition1):
#Executes if condition 1 is true
if (condition 2):
  #Executes if condition 2 is true
  #Condition 2 ends here
#Condition 1 ends here
```



syntax

Syntax

The syntax of the **`nested if construct`** with else condition will be like this –

```
if expression1:
   statement(s)
   if expression2:
      statement(s)
   else
      statement(s)
else:
   if expression3:
      statement(s)
   else:
      statement(s)
```

code sample-1

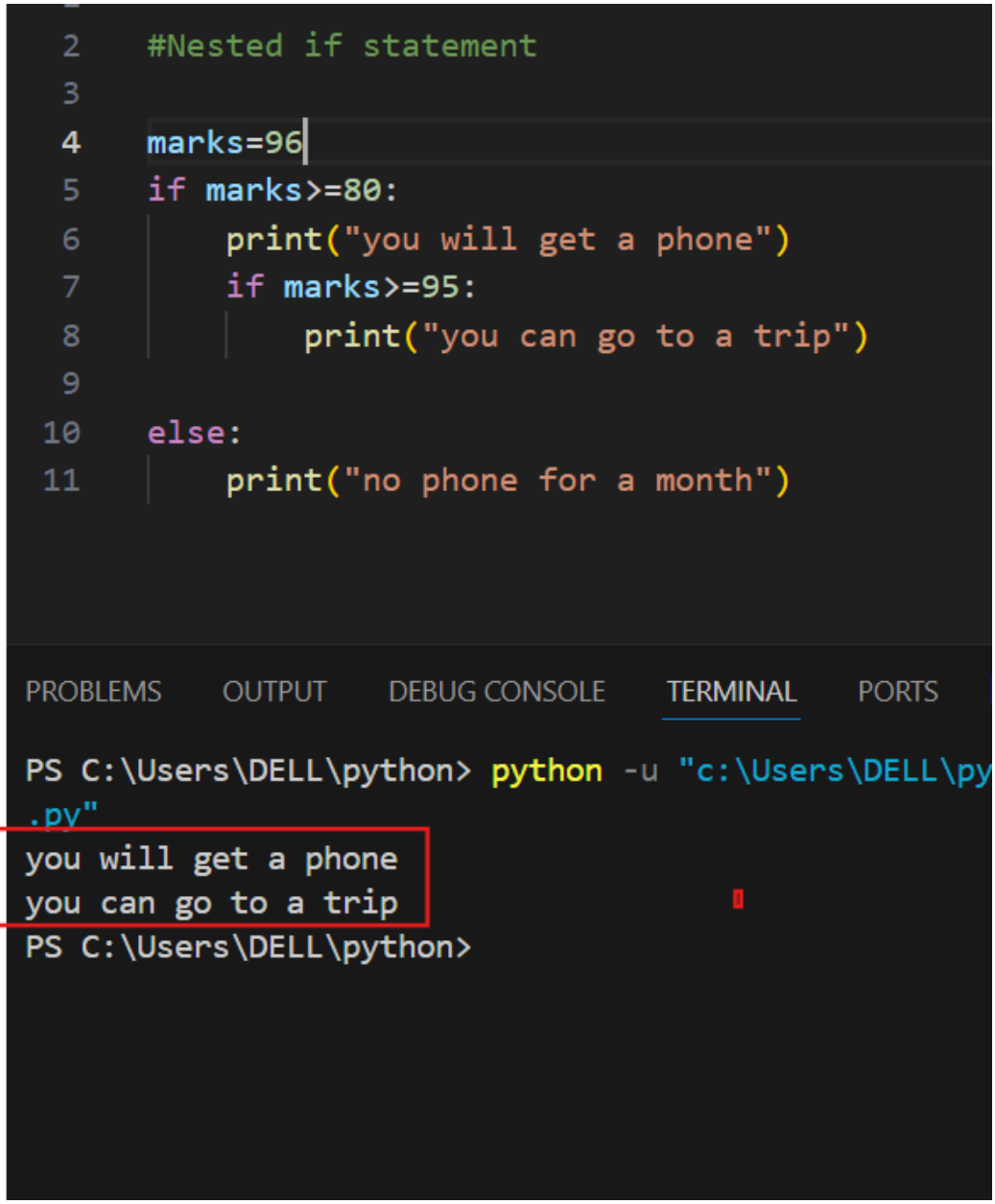
```python

#Nested if statement

marks=78
if marks>=80:
    print("you will get a phone")
    if marks>=95:
        print("you can go to a trip")

else:
    print("no phone for a month")
```

Output (Ctrl+Shift+U)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\
.py"
no phone for a month
PS C:\Users\DELL\python>
```

code sample -2



short Hand if statement

If Statement in One Line

If you have only one statement to execute, you can put it on the same line as the if statement.

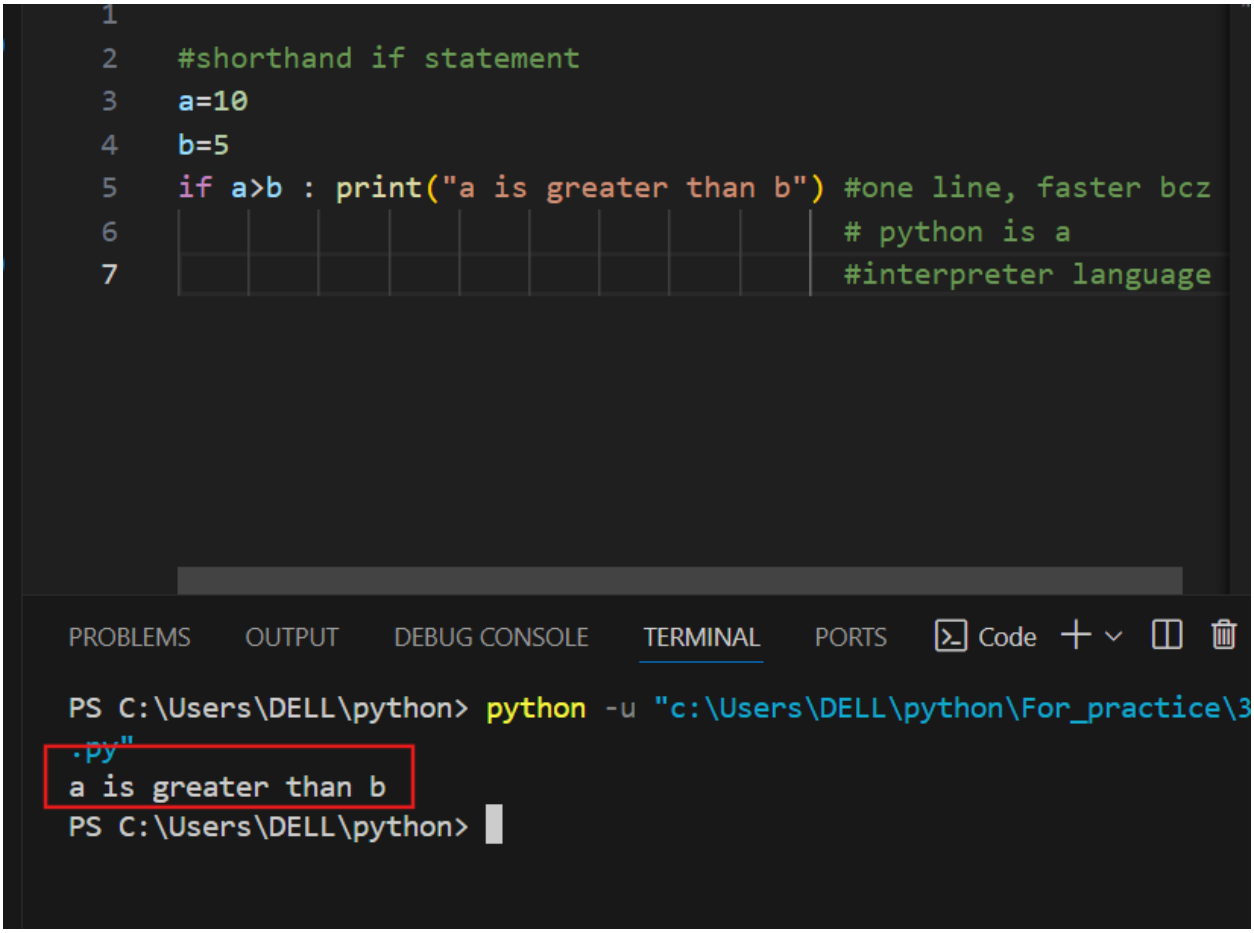
Example

Get your own Python Server

One line if statement:

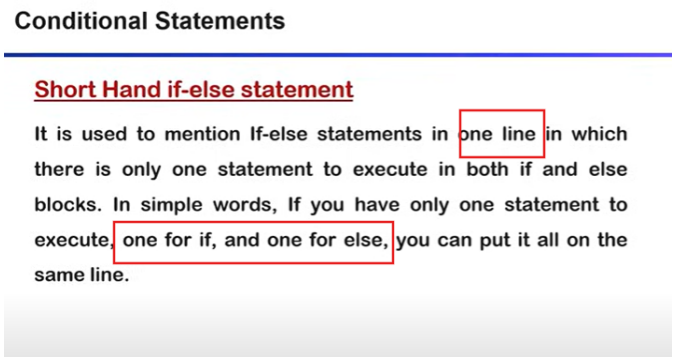
```
if a > b: print("a is greater than b")
```

code(shorthand if statement)



shorthand if-else statement

shorthand if-else condition



Conditional Statements

Short Hand if-else statement

It is used to mention If-else statements in one line in which there is only one statement to execute in both if and else blocks. In simple words, If you have only one statement to execute, one for if, and one for else, you can put it all on the same line.

code (shorthand if-else condition)

```
2   #shorthand if-else statement
3
4   marks=87
5   print("you will go to a trip ") if marks>=90 else print("no phone for a month")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
no phone for a month
PS C:\Users\DELL\python>
```

problem solving

problem-1

1. Write a program to check if a number is positive.

```python

#write a program to check if a number is positive or negative
num=int(input("enter a number here: "))
if num>=0:
    print("The number is positive")
else:
    print("the number is negative")
```

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py"
enter a number here: 23
The number is positive
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py"
enter a number here: -6
the number is negative
PS C:\Users\DELL\python>
```

problem-2

2. Write a program to check whether a number is odd or even.

```python

#write a program to check whether a number is odd or even
num=int(input("enter a number here: "))
if num%2==0:
    print("The number is even")
else:
    print("the number is odd")
```

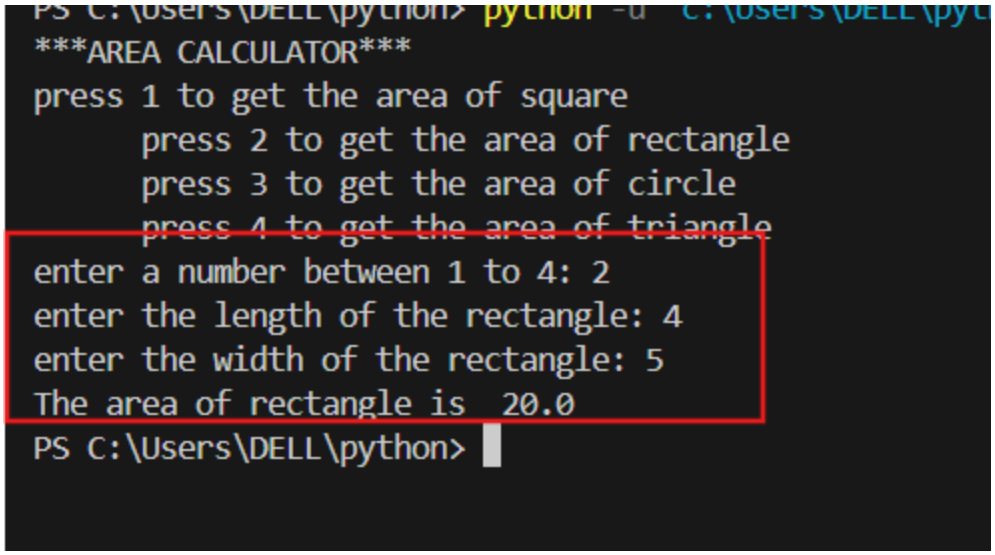
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1
py"
enter a number here: 67
the number is odd
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1
py"
enter a number here: 78
The number is even
PS C:\Users\DELL\python>
```

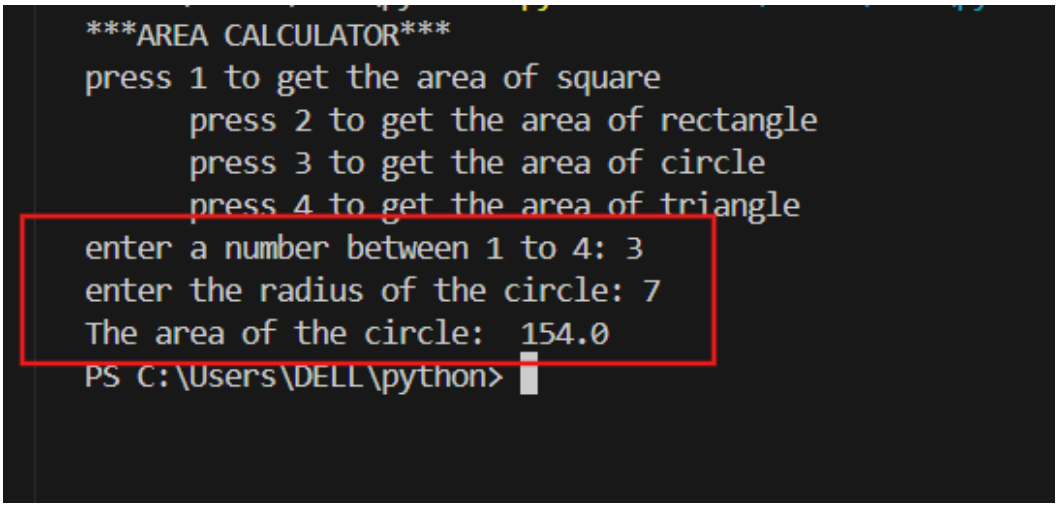
problem-3

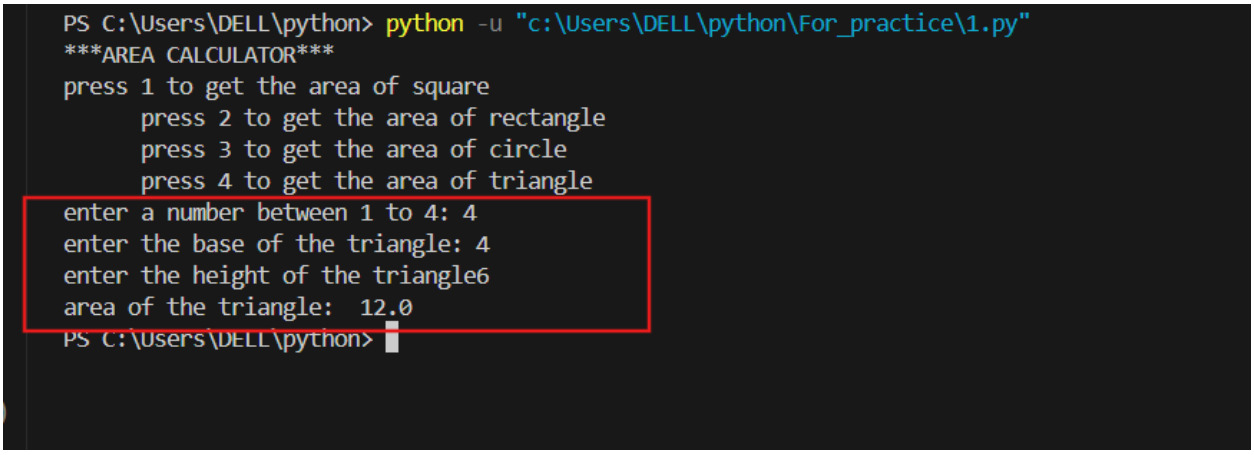
3. Write a program to create area calculator.

```python

#write a program to create area calculator

print("""***AREA CALCULATOR***""")
print("""press 1 to get the area of square
      press 2 to get the area of rectangle
      press 3 to get the area of circle
      press 4 to get the area of triangle""")
choice = int(input("enter a number between 1 to 4: "))
if choice==1:
    side=float(input("enter the length of one side: "))
    area=side**2
    print("area of the square: ",area)

elif choice==2:
    length=float(input("enter the length of the rectangle: "))
    width=float(input("enter the width of the rectangle: "))
    area=length*width
    print("The area of rectangle is ",area)

elif choice==3:
    radius = float(input("enter the radius of the circle: "))
    area=(22/7)*(radius**2)
    print("The area of the circle: ",area)

elif choice==4:
    base=float(input("enter the base of the triangle: "))
    height=float(input("enter the height of the triangle"))
    area=(0.5*base*height)
    print("area of the triangle: ",area)
else:
    print("invalid choice")


```

```
***AREA CALCULATOR***
press 1 to get the area of square
      press 2 to get the area of rectangle
      press 3 to get the area of circle
      press 4 to get the area of triangle
enter a number between 1 to 4: 1
enter the length of one side: 4
area of the square:  16.0
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py"
```


AREA CALCULATOR
press 1 to get the area of square
press 2 to get the area of rectangle
press 3 to get the area of circle
press 4 to get the area of triangle
enter a number between 1 to 4: 2
enter the length of the rectangle: 4
enter the width of the rectangle: 5
The area of rectangle is 20.0
PS C:\Users\DELL\python>


AREA CALCULATOR
press 1 to get the area of square
press 2 to get the area of rectangle
press 3 to get the area of circle
press 4 to get the area of triangle
enter a number between 1 to 4: 3
enter the radius of the circle: 7
The area of the circle: 154.0
PS C:\Users\DELL\python>


PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py"
AREA CALCULATOR
press 1 to get the area of square
press 2 to get the area of rectangle
press 3 to get the area of circle
press 4 to get the area of triangle
enter a number between 1 to 4: 4
enter the base of the triangle: 4
enter the height of the triangle6
area of the triangle: 12.0
PS C:\Users\DELL\python>

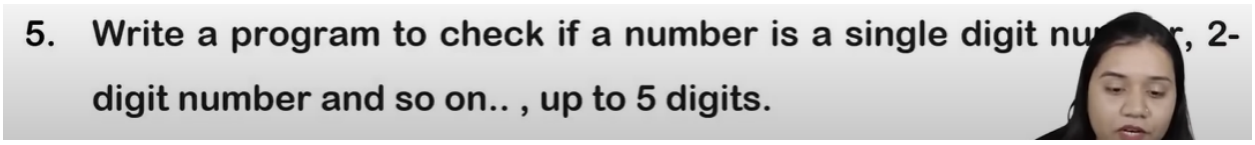
problem-4

4. Write a program check whether the passed letter is a vowel or not.

```
1
2   #write a program check whether the passed is a
3   #vowel or not
4
5   letter = input("enter a letter here: ")
6   if (letter in "aeiou") or (letter in "AEIOU"):
7       print("it is a vowel")
8   else:
9       print("it is not a vowel")
```

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
enter a letter here: a
it is a vowel
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
enter a letter here: E
it is a vowel
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
enter a letter here: W
it is not a vowel
PS C:\Users\DELL\python>
```

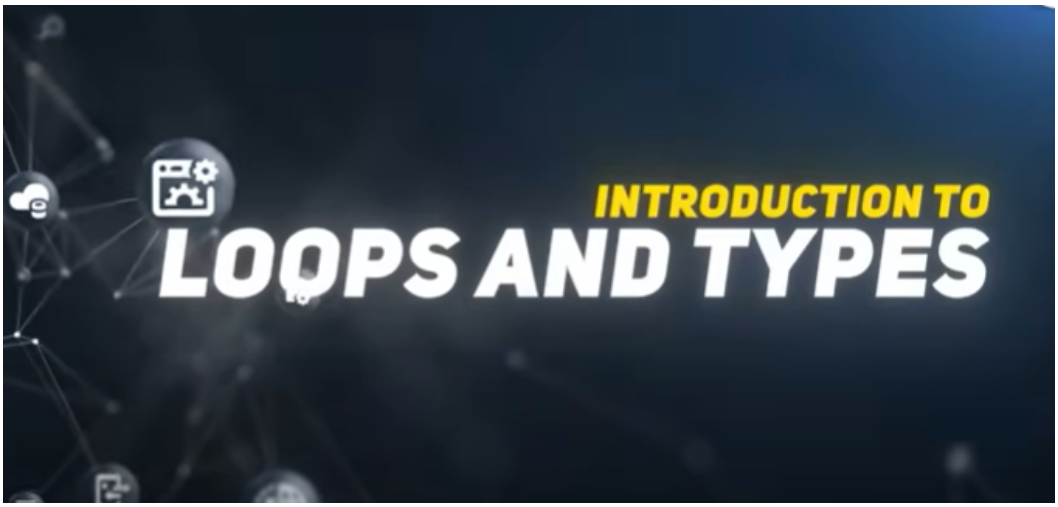
problem-5



```python

#write a program to check if a number is a
#single digit number,2-digit number and so on..,
#upto 5 digits
num=int(input("enter a number upto 5 digit: "))
if num>=0 and num<=9:
    print("It is a single digit")
elif num>=10 and num<=99:
    print("It is a double digit number")
elif num>=100 and num<=999:
    print("It is a triple digit number")
elif num>=1000 and num<=9999:
    print("It is a four digit number")
else:
    print("It is a five digit number")
```

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py"
enter a number upto 5 digit: 123
It is a triple digit number
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py"
enter a number upto 5 digit: 7865
It is a four digit number
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py"
enter a number upto 5 digit: 65
It is a double digit number
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py"
enter a number upto 5 digit: 8
It is a single digit
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py"
enter a number upto 5 digit: 67554
It is a five digit number
```

Loops



Types of loops

A loop means to repeat something in the exact same way.

Types of loops are :

1. For Loop
2. While Loop
3. While True
4. Nested Loop

For loop

For Loop

- For loop is a loop that repeats something in a given range.
- The range has a starting point, ending point and a step/gap in it.
- +1 is added to the ending point while defining a range.

What is range() function

The range() Function

To loop through a set of code a specified number of times, we can use the `range()` function,

The `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Note that `range(6)` is not the values of 0 to 6, but the values 0 to 5.

Code(for loop)

```
#for loop(in range)
for x in range(6):
    print(x)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
0
1
2
3
4
5
PS C:\Users\DELL\python>
```

confusion(in range)

The `range()` function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: `range(2, 6)`, which means values from 2 to 6 (but not including 6):

Example

Using the start parameter:

```
for x in range(2, 6):
  print(x)
```

Code(range-(2 to 6)

```
1  #for loop(in range)
2  for x in range(2,6):
3      print(x)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
2
3
4
5
PS C:\Users\DELL\python>
```

For loop (increment position)

The `range()` function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: `range(2, 30, 3)`:

Example

Increment the sequence with 3 (default is 1):

```
for x in range(2, 30, 3):
  print(x)
```

Code(with increment choice)

```python
#for loop(in range)
for x in range(2,30,3):
    print(x)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
2
5
8
11
14
17
20
23
26
29
PS C:\Users\DELL\python>
```

For loop (- Else -)

Else in For Loop

The `else` keyword in a `for` loop specifies a block of code to be executed when the loop is finished:

Example

Print all numbers from 0 to 5, and print a message when the loop has ended:

```python
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

Code(for loop+else)

```python
#Else in for loop
for x in range(6):
    print(x)
else:
    print("finally finished!")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
0
1
2
3
4
5
finally finished!
PS C:\Users\DELL\python>
```

For loop + (break statement) + else

Break the loop when `x` is 3, and see what happens with the `else` block:

```python
for x in range(6):
  if x == 3: break
  print(x)
else:
  print("Finally finished!")
```

Code (for loop+break+else)

```python
#Else in for loop (with break)
for x in range(6):
    if x==3:break
    print(x)

else:
    print("finally finished!")  #will not be executed
```

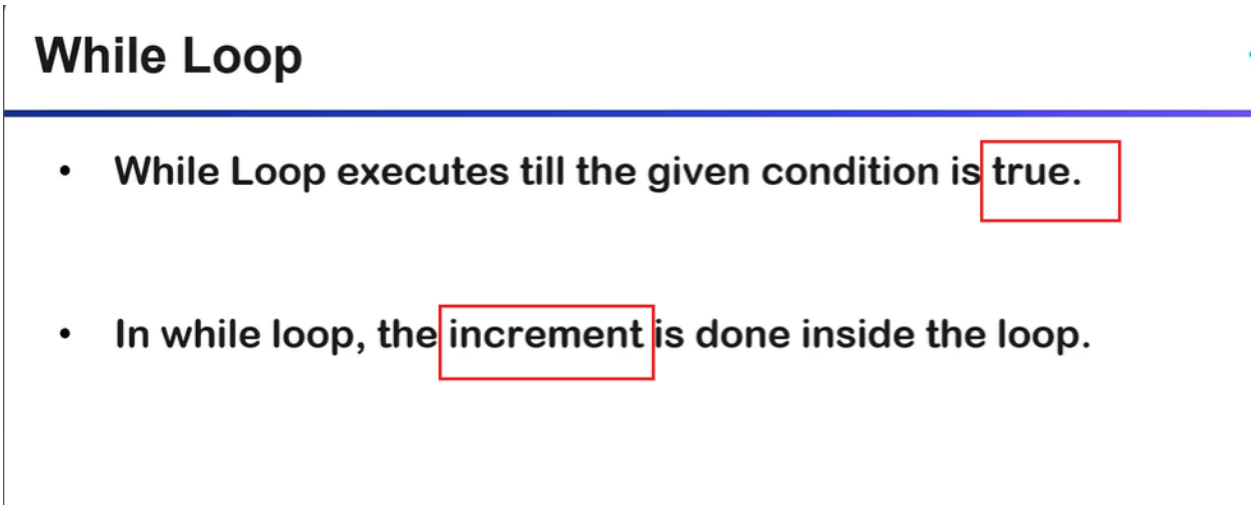
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
0
1
2
PS C:\Users\DELL\python>
```

problem-(multiplication table using for loop)

```python
#multiplication using for loop
n=int(input("enter the number: "))
for i in range (1,11):
    print(n,"x",i, "=" ,n*i)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
enter the number: 12
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
PS C:\Users\DELL\python>
```

While loop ———--Brief introduction



Code(While loop)

```python
#while loop
i=1
while i<10:
    print(i)
    i+=1
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_
1
2
3
4
5
6
7
8
9
PS C:\Users\DELL\python>
```

problem——(multiplication table using while loop

```python
#multiplication table using while loop
i=1
n=int(input("enter the multiplication number: "))
while i<10:
    print(n,"x",i, "=",n*i)
    i+=1
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
enter the multiplication number: 3
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
PS C:\Users\DELL\python>
```

What is while true?

While true

- It is an infinite loop
- To break a while true loop, break statement is used.

Code—-(While true)

```python
while True:
    num1=int(input("enter a number here: "))
    num2=int(input("enter another number here: "))

    print(num1+num2)

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1
enter a number here: 34
enter another number here: 56
90
enter a number here: 66
enter another number here: 77
143
enter a number here:
```

and So ... on

While true———repeat() function

```
For_practice > 1.py > ...
1  while True:
2      num1=int(input("enter a number here: "))
3      num2=int(input("enter another number here: "))
4
5      print(num1+num2)
6      repeat=input("do you stop the program: ")
7      if repeat=="yes":
8          break
9
10
11
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice
"
enter a number here: 45
enter another number here: 67
112
do you stop the program: no
enter a number here: 65
enter another number here: 44
109
do you stop the program: yes
PS C:\Users\DELL\python>
```

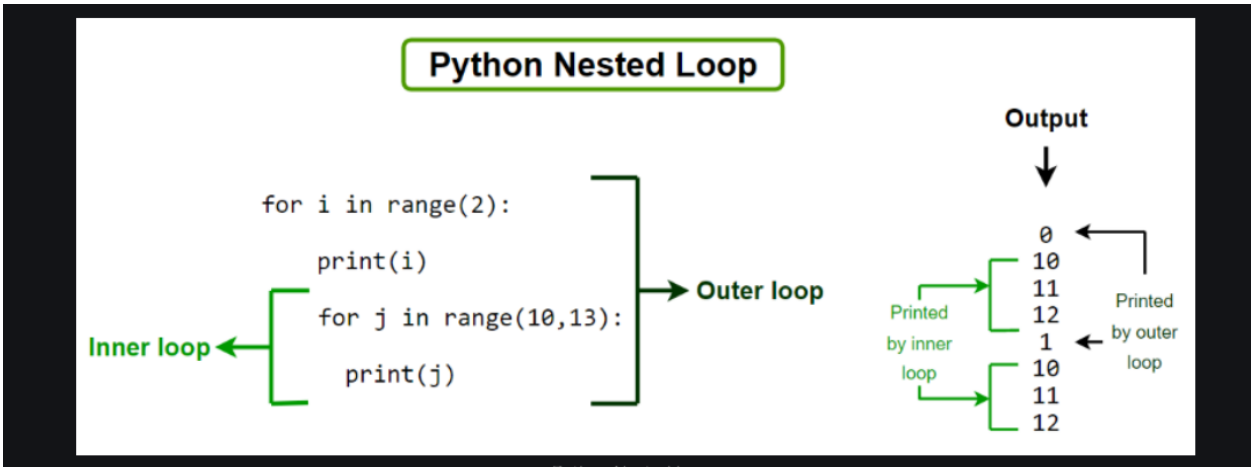
What is nested loops?

Nested Loop

- A loop inside a loop is called as nested loop.

Nested loops are also used to solve pattern problems.

Nested loop structure-example

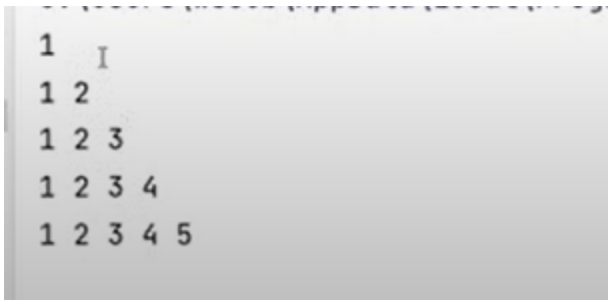


Code-(Nested loop)———-example-1

```
1  for i in range (1,4):
2      for j in range (1,11):
3          print (j ,end=" ")
4      print()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_prac
"
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
PS C:\Users\DELL\python>
```

pattern problem using nested loops



Code(pattern problem)

```python
#pattern problem
for i in range(1,6):
    for j in range(1,i+1):
        print(j,end=" ")

    print()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
PS C:\Users\DELL\python>
```

What for loop with condition statements refer?

For Loop with conditional statements

The use of if-else statements increases the ability of for loop to completes the task effectively. By using if-else statements we can provide with special conditions inside for loop.

Code (for loop with conditional statements)

```
#for loop with conditional statement

for i in range(1,11):
    if(i==3):
        print("add this song to favs")
    else:
        print(i)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_pract
1
2
add this song to favs
4
5
6
7
8
9
10
PS C:\Users\DELL\python>
```

Code (while loop with conditional statements)

```python
#while loop with conditional statements
i=1
while i<=10:
    if(i==3):
        print("add your songs to favs")
    else:
        print(i)
    i+=1
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py
"
1
2
add your songs to favs
4
5
6
7
8
9
10
PS C:\Users\DELL\python>
```

Find the numbers which are divisible both 8 and 12 within 1-100

```python
#for loop with conditional statement

for i in range (1,101):
    if(i%8==0 and i%12==0):
        print(i)

```

Debug Console (Ctrl+Shift+Y)

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py
"
24
48
72
96
PS C:\Users\DELL\python>
```

Continue statement

Continue Statement:

Continue Statement is used when you want to skip a particular condition.

Break statement

Break Statement:

Break Statement is used when you want to destroy a loop at a certain condition and come out of the loop.

Continue ———--code

```python
#continue statement sample code

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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_prac
"
1
2
3
4
6
7
8
9
10
PS C:\Users\DELL\python>
```

5 skip.

Break————--Code

```python
#break statement sample code

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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice
"
1
2
3
4
PS C:\Users\DELL\python>
```

Problem-1

1. Write a program to find a sum of all the even numbers up to 50.

Code

```
#write a program to find a sum of all the even numbers up to 50

sum=0
for i in range(1,51):
    if(i%2==0):
        sum+=i
print(sum)

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
650
PS C:\Users\DELL\python>
```

```python
#write a program to find a sum of all the even numbers up to 50

sum=0
for i in range(1,51):
    if(i%2==0):
        sum+=i
print(sum)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
650
PS C:\Users\DELL\python>
```

problem—2

2. Write a program to write first 20 numbers and their squared numbers.

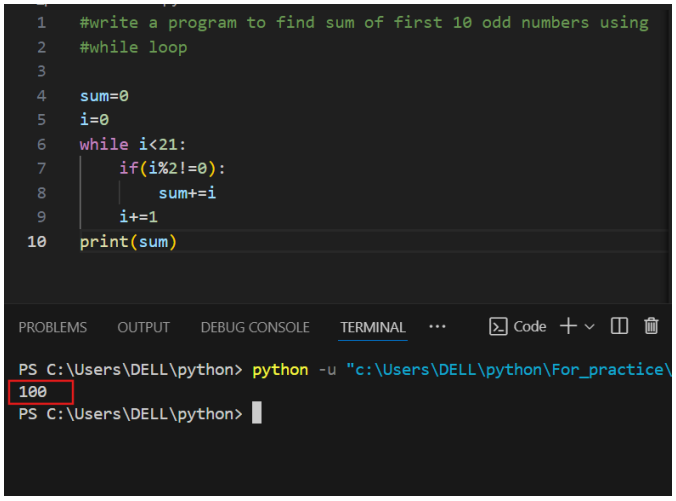
Code

```python
#write a program to write first 20 numbers and their
#squared numbers

for i in range(1,21):
    print(i," ",i**2) #power i**2 =i^2
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  ...  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.
1   1
2   4
3   9
4   16
5   25
6   36
7   49
8   64
9   81
10   100
11   121
12   144
13   169
14   196
15   225
16   256
17   289
18   324
19   361
20   400
PS C:\Users\DELL\python>
```

Problem—-3

3. Write a program to find sum of first 10 odd numbers using while loop.



problem——4

```
# Write a program to check if a number is divisible by
# 8 and 12, up to 100 numbers
```

```python
#write a program to check if a number is divisible
#by 8 and 12, up to numbers

for i in range(1,101):
    if(i%8==0 and i%12==0):
        print(i)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  ...  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practic
24
48
72
96
PS C:\Users\DELL\python>
```

problem——5

```python
# Write a program to create a billing system at supermarket.
```

```python
#write a program to create a billing system at supermarket

while True:
    name=input("enter customer's name: ")
    total=0
    while True:
        print("enter the items you have bought:")
        item_name=input("the item name is : ")
        amount=float(input("enter the amount : "))
        quantity=float(input("enter quantity: "))
        total+=amount*quantity
        repeat=input("do you want to add more items? (yes/no):")
        if repeat=="yes" or repeat=="Yes":
            break
        print("-"*40)
        print("Name: ",name)
        print("Amount to be paid: ",total)
        print("-"*40)
        print("*********Happy shopping**********")

        repeat1=input("do you want to go to the next customer? (yes/no):")
        if repeat1=="no" or repeat1=="No":
            break
```

```
enter customer's name: H H joti
enter the items you have bought:
the item name is : nuts
enter the amount : 70
enter quantity: 2
do you want to add more items? (yes/no):no
----------------------------------------
Name:  H H joti
Amount to be paid:  140.0
----------------------------------------
********Happy shopping*********
do you want to go to the next customer? (yes/no):
```

Strings (problem solving)

A = “Why fit in, When you are Born to Stand Out!”

1. **Write a program to find the length of the following string.**

```
1  #write a program to find the length of the following
2  # string
3  A="why fit in, when you are Born to stand out!"
4  b=len(A)
5  print(b)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  ...  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practic
43
PS C:\Users\DELL\python>
```

2. **Write a program to check how many time alphabet o is occurring.**

```python
# write a program to check how many time
# alphabet o is occurring
A="why fit in, when you are Born to stand out!"
occurring=A.count('o')
print("the number of 'o' is occurring : ",occurring)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.
py"
the number of 'o' is occurring :  4
PS C:\Users\DELL\python>
```

3. **Write a program to convert the whole string into lower and upper cases.**

```python
# write a program to convert the whole
#string into lower and upper cases
A="why fit in, when you are Born to stand out!"
up_case=A.upper()
print("upper case of the string is :",up_case)
lo_case=A.lower()
print("lower case of the string is :",lo_case)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\3.py"
upper case of the string is : WHY FIT IN, WHEN YOU ARE BORN TO STAND OUT!
lower case of the string is : why fit in, when you are born to stand out!
PS C:\Users\DELL\python>
```

title() function

Definition and Usage

The `title()` method returns a string where the first character in every word is upper case. Like a header, or a title.

Problem——4

4. Write a program to convert the following string into a title.

```
#write a program to convert the following
#string into a title
a="why fit in, when you are born to stand out!"
z=a.title()
print(z)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice
py"
Why Fit In, When You Are Born To Stand Out!
PS C:\Users\DELL\python>
```

5. Write a program to find the index number of “fit in”.

```python
#write a program to find the index number of 'fit in'
a="why fit in, when you are born to stand out! "
index=a.find('fit in')
print(index)

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practi
py"
4
PS C:\Users\DELL\python>
```

Pattern problems

Problems:

1

1 2

1 2 3

1 2 3 4

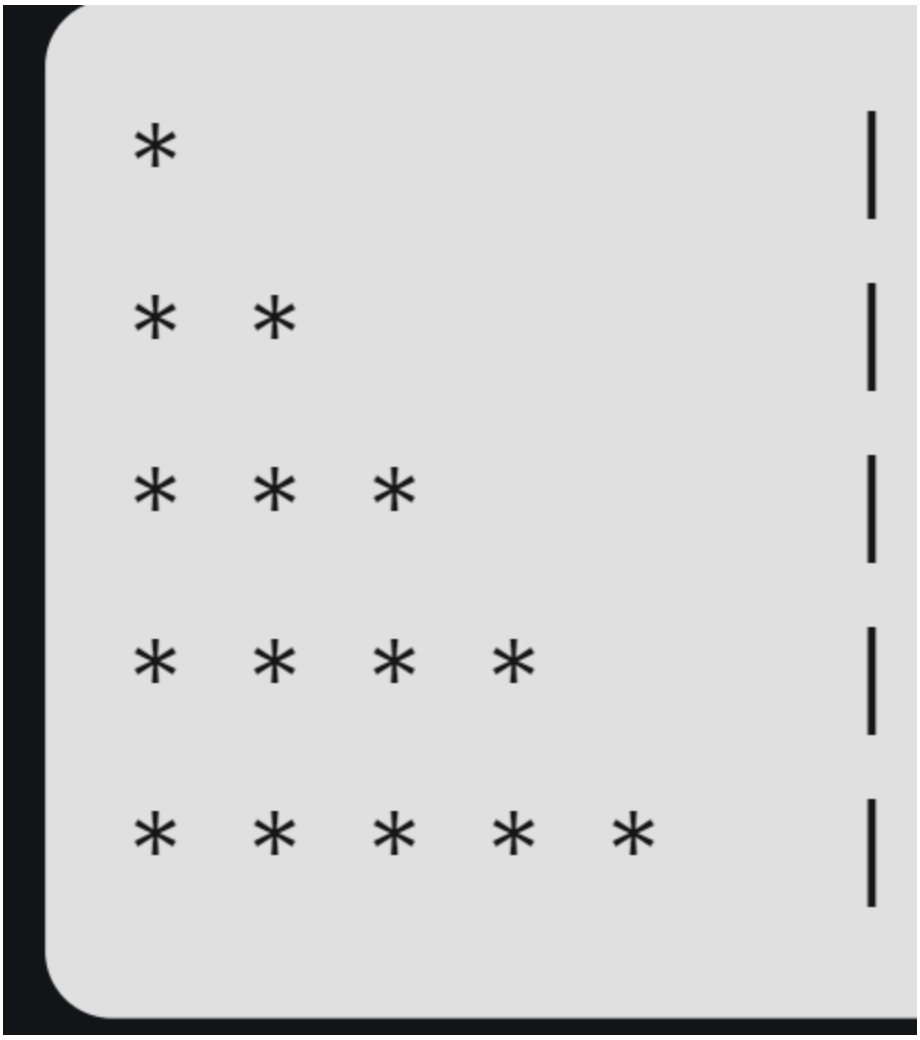
1 2 3 4 5

Write a program to display this pattern.

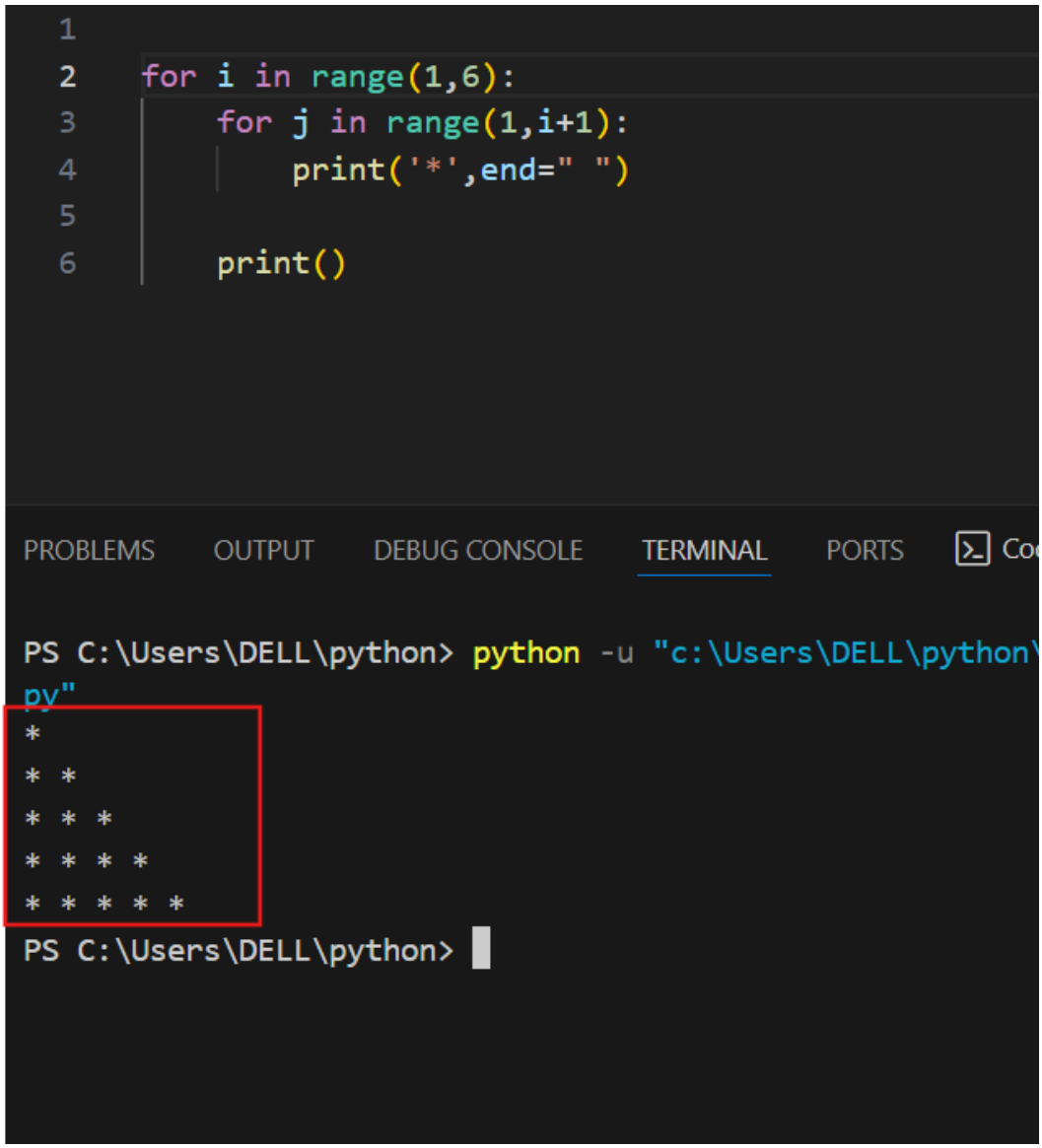
Code

```python
for i in range(1,6):
    for j in range(1,i+1):
        print(j,end=" ")

    print()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\pyth
py"
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
PS C:\Users\DELL\python>
```



Code


1
2 for i in range(1,6):
3 for j in range(1,i+1):
4 print('*',end=" ")
5
6 print()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\
py"
*
* *
* * *
* * * *
* * * * *
PS C:\Users\DELL\python>

Problems:

Write a program to display this pattern.

1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

Code

```python

for i in range(1,6):
    for j in range(1,i+1):
        print(i,end=" ")

    print()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\py
py"
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
PS C:\Users\DELL\python>
```

Problems:

Write a program to display this pattern.

1 1 1 1 1

2 2 2 2

3 3 3

4 4

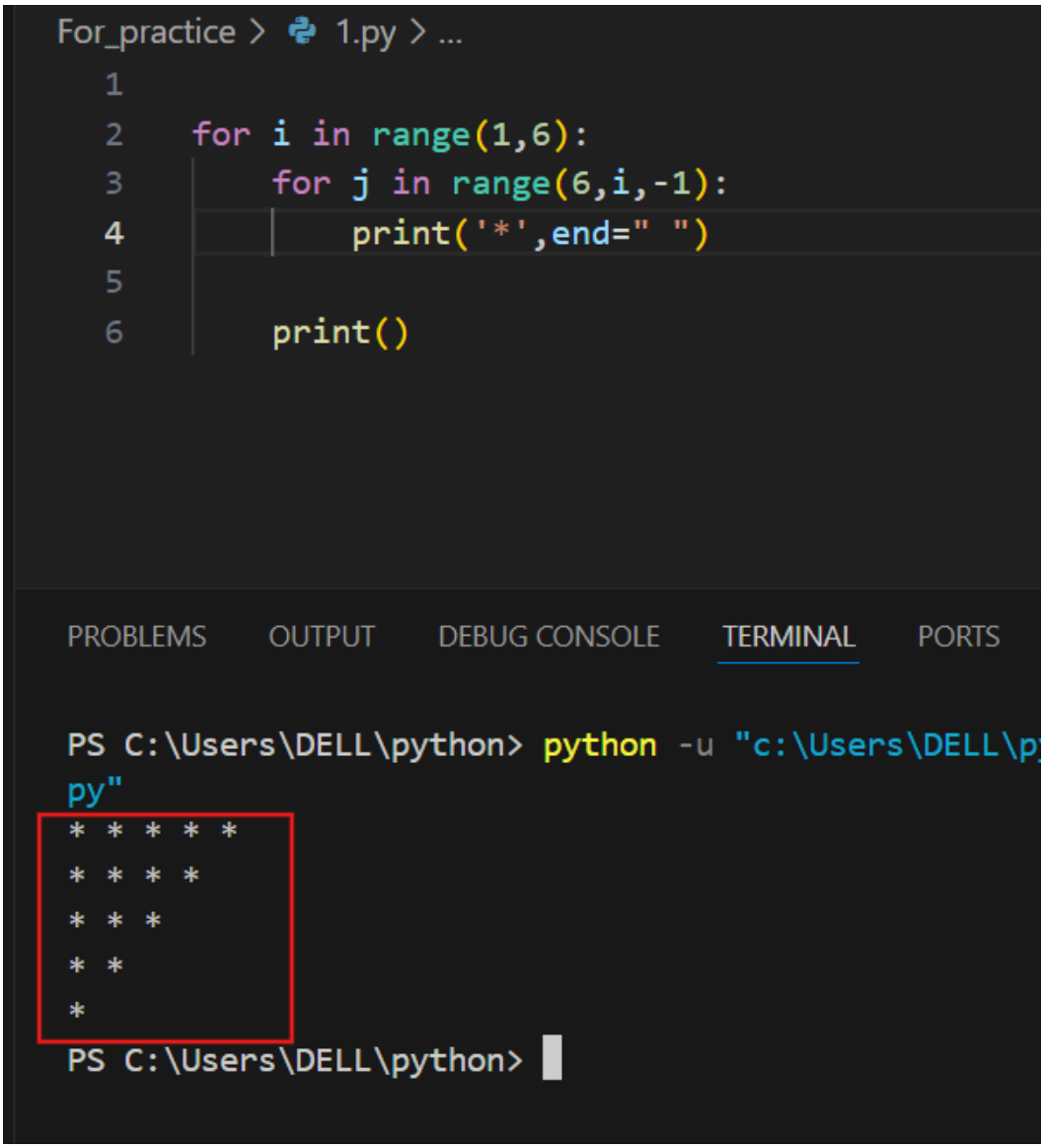
5

Code

```python

for i in range(1,6):
    for j in range(6,i,-1):
        print(i,end=" ")

    print()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\DELL\python> python -u "c:\Users\DELL\pytho
py"
1 1 1 1 1
2 2 2 2
3 3 3
4 4
5
PS C:\Users\DELL\python>
```



Write a program to display this pattern.

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

```
1
2  for i in range(1,6):
3      for j in range(5,i,-1):
4          print(" ",end=" ")
5      for k in range (i):
6          print("*", end=" ")
7      print()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py"
        *
      * *
    * * *
  * * * *
* * * * *
PS C:\Users\DELL\python>
```

Problems:

1

2 1

3 2 1

4 3 2 1

5 4 3 2 1

Write a program to display this pattern.

```
1
2  for i in range(1,6):
3      for j in range(i,0,-1):
4          print(j,end=" ")
5      print()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py"
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
PS C:\Users\DELL\python>
```

Problems:

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

Write a program to display this pattern.

```python
for i in range(1,6):
    for j in range(1,i+1):
        print("*",end=" ")
    print()
for i in range(5,0,-1):
    for k in range(1,i):
        print("*",end=" ")
    print()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py"
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*

PS C:\Users\DELL\python>
```

Problems:

1

2 4

3 6 9

4 8 12 16

5 10 15 20 25

6 12 18 24 30 36

7 14 21 28 35 42 49

8 16 24 32 40 48 56 64

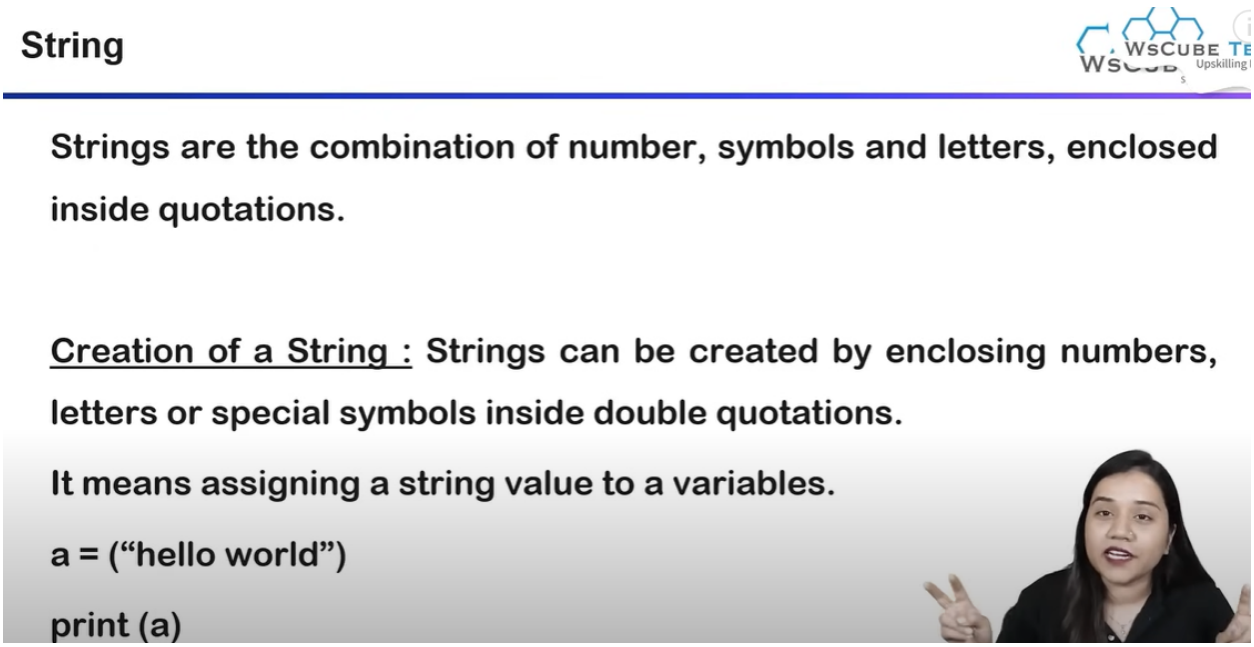
Write a program to display the following pattern.

```python

for i in range(1,9):
    for j in range(1,i+1):
        print(i*j,end=" ")
    print()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_practice\1.py"
1
2 4
3 6 9
4 8 12 16
5 10 15 20 25
6 12 18 24 30 36
7 14 21 28 35 42 49
8 16 24 32 40 48 56 64
PS C:\Users\DELL\python>
```

STRING MANIPULATION

WHAT IS STRING?



Strings

Strings in python are surrounded by either single quotation marks, or double quotation marks.

`'hello'` is the same as `"hello"`.

SINGLE QUOTATION OR DOUBLE QUOTATION ,BOTH ARE SAME

Example

```
print("Hello")
print('Hello')
```

```
Hello
Hello
```

QUOTATION INSIDE QUOTES(NO CHANGE)

Quotes Inside Quotes

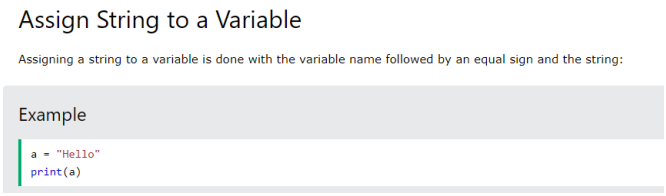
You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

Example

```
print("It's alright")
print("He is called 'Johnny'")
print('He is called "Johnny"')
```

```
It's alright
He is called 'Johnny'
He is called "Johnny"
```

ASSIGN STRING TO A VARIABLE



Assign String to a Variable

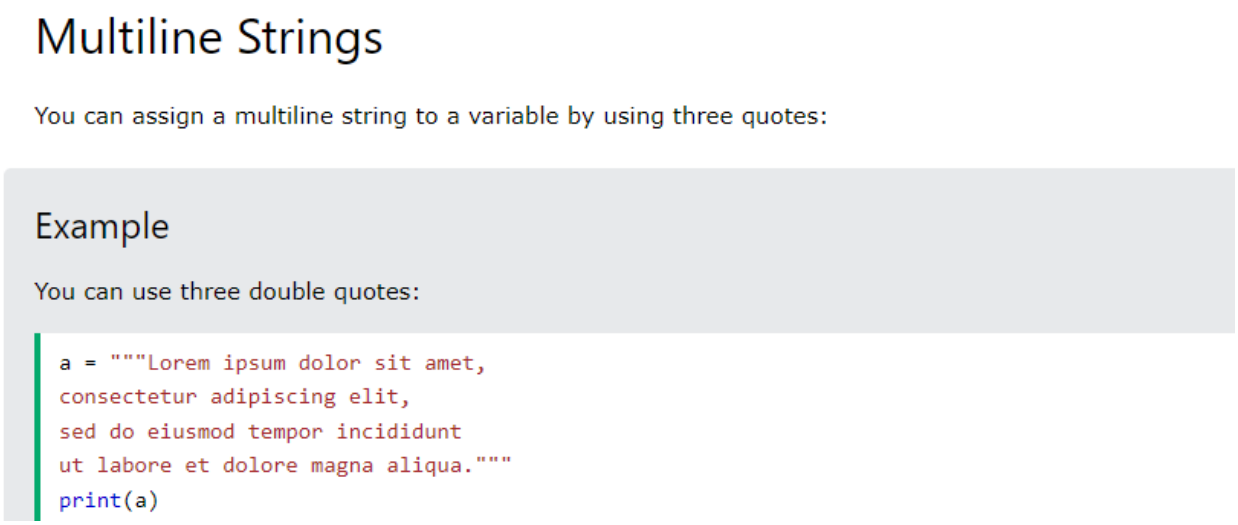
Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

Example

```
a = "Hello"
print(a)
```

```
Hello
```

Multiple string(three quotes)



Multiline Strings

You can assign a multiline string to a variable by using three quotes:

Example

You can use three double quotes:

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
```

Three single quotes

Or three single quotes:

Example

```
a = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''
print(a)
```

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
```

string [————-len() ————function]

Python len() Function

Definition—-len()———-function

Definition and Usage

The `len()` function returns the number of items in an object.

When the object is a string, the `len()` function returns the number of characters in the string.

syntax

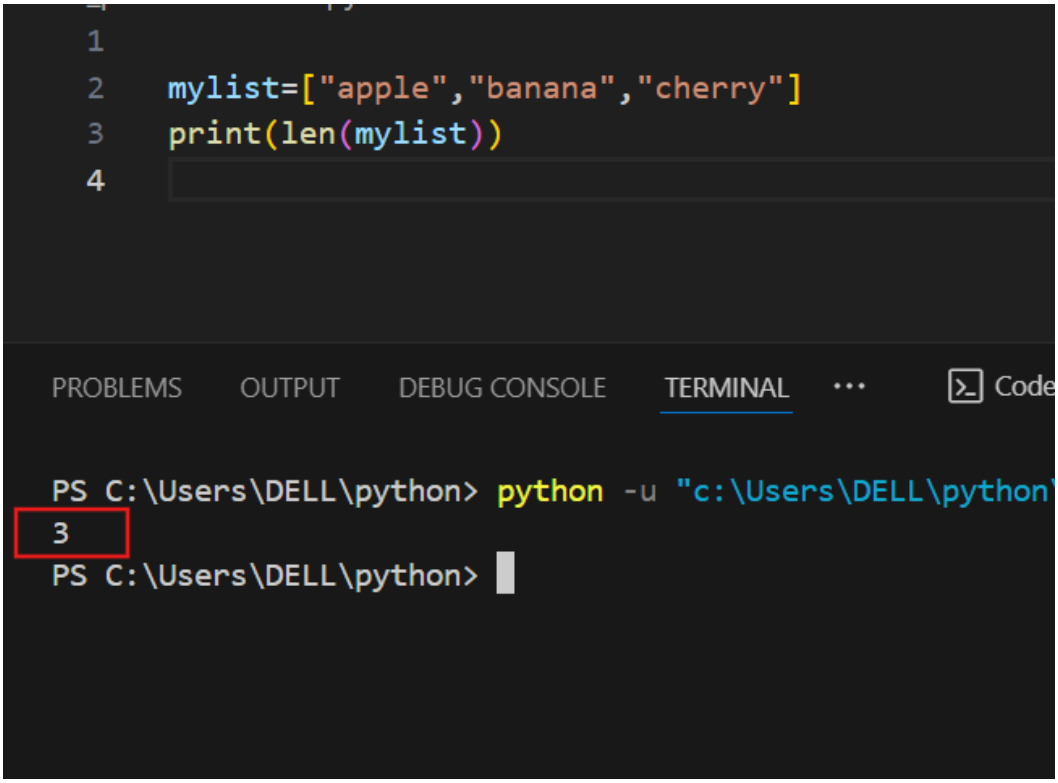
Syntax

```
len(object)
```

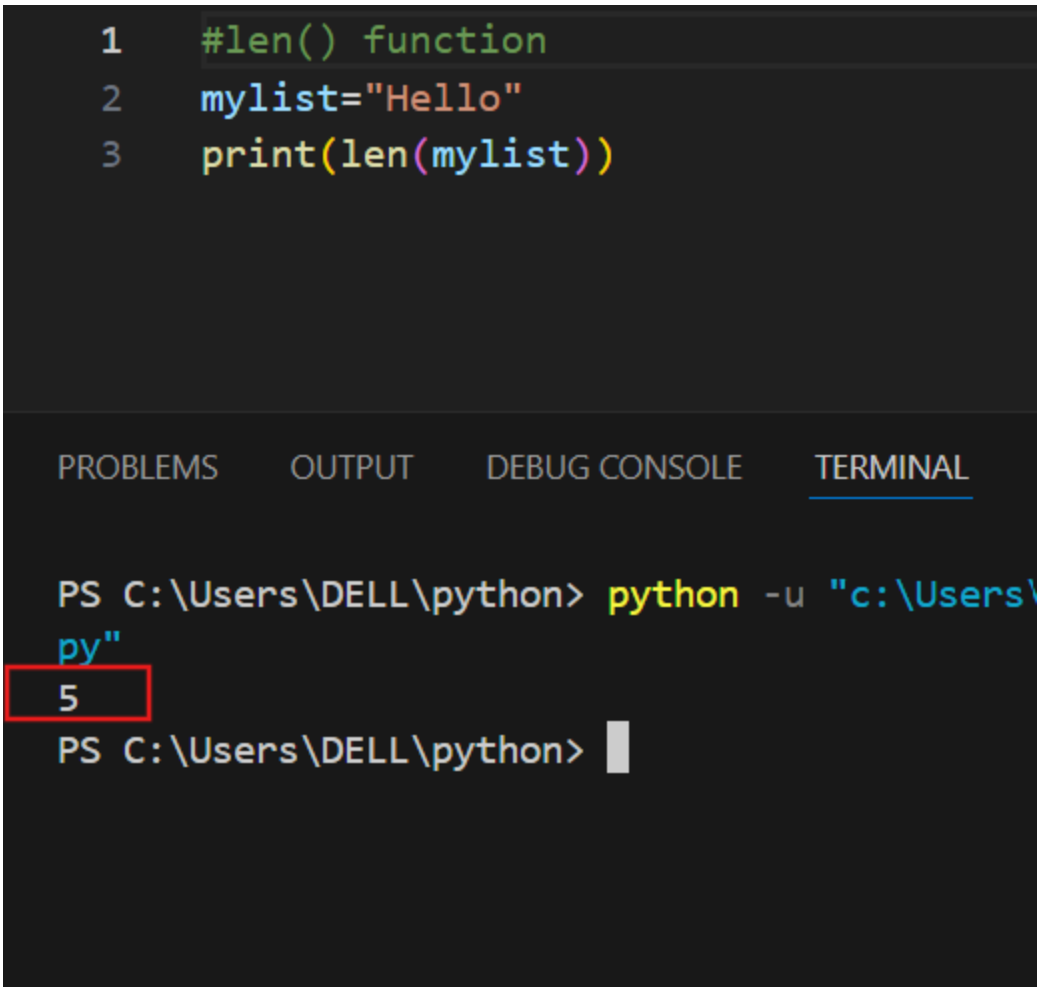
Parameter Values

Parameter	Description
object	Required. An object. Must be a sequence or a collection

Example-1


1
2 mylist=["apple","banana","cherry"]
3 print(len(mylist))
4
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
PS C:\Users\DELL\python> python -u "c:\Users\DELL\python
3
PS C:\Users\DELL\python>

Example-2


1 #len() function
2 mylist="Hello"
3 print(len(mylist))
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PS C:\Users\DELL\python> python -u "c:\Users\
py"
5
PS C:\Users\DELL\python>

Example-3

```python
#To find the length of the string

a="Harry potter and the Goblet of Fire"
print(len(a))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python
py"
35
PS C:\Users\DELL\python>
```

string

Count() method

Python List count() Method

Definition

Definition and Usage

The `count()` method returns the number of elements with the specified value.

Syntax

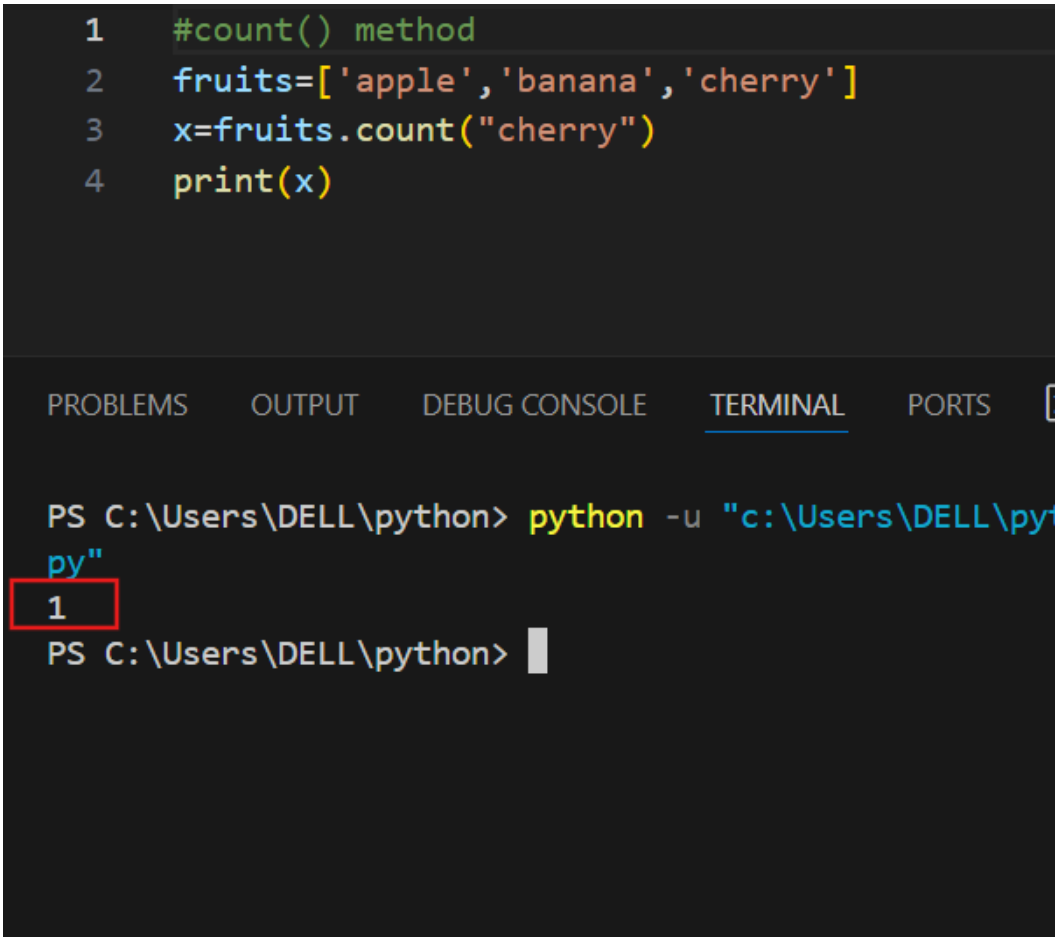
Syntax

```
list.count(value)
```

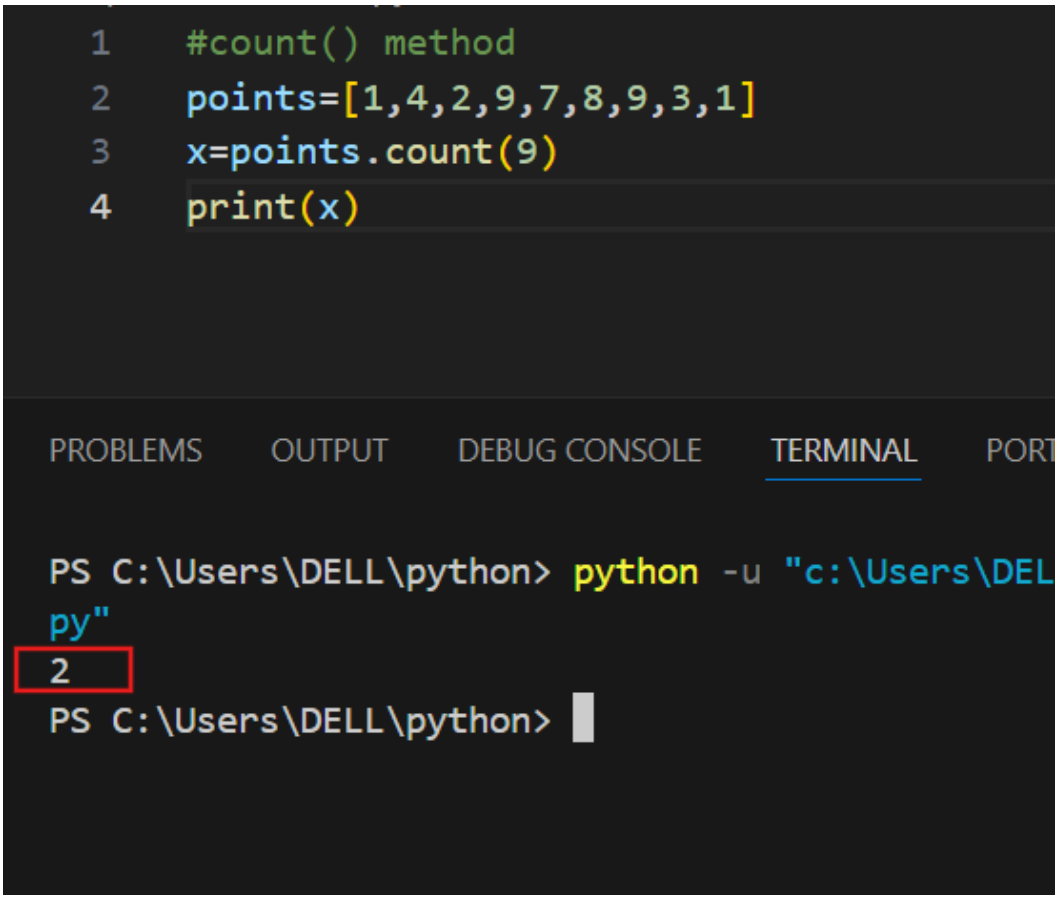
Parameter Values

Parameter	Description
value	Required. Any type (string, number, list, tuple, etc.). The value to search for.

Example-1


1 #count() method
2 fruits=['apple','banana','cherry']
3 x=fruits.count("cherry")
4 print(x)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\DELL\python> python -u "c:\Users\DELL\py
py"
1
PS C:\Users\DELL\python>

Example-2


1 #count() method
2 points=[1,4,2,9,7,8,9,3,1]
3 x=points.count(9)
4 print(x)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORT
PS C:\Users\DELL\python> python -u "c:\Users\DEL
py"
2
PS C:\Users\DELL\python>

Example-3

```
1   #count() method
2   a="Harry potter and the Goblet of Fire"
3   print(a.count("o"))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\pyth
py"
3
PS C:\Users\DELL\python>
```

upper() method

Python String upper() Method

Definition

Definition and Usage

The `upper()` method returns a string where all characters are in upper case.

Symbols and Numbers are ignored.

Syntax

Syntax

```
string.upper()
```

Example-1

Example

Upper case the string:

```
txt = "Hello my friends"

x = txt.upper()

print(x)
```

```
HELLO MY FRIENDS
```

Example-2

```
1   #upper() method
2   a="Harry potter and the Goblet of Fire"
3   print(a.upper())
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For_pract
py"
HARRY POTTER AND THE GOBLET OF FIRE
PS C:\Users\DELL\python>
```

lower() Method

Python String lower() Method

Definition

Definition and Usage

The `lower()` method returns a string where all characters are lower case.

Symbols and Numbers are ignored.

Syntax

Syntax

```
string.lower()
```

Example—-1

Example

Lower case the string:

```
txt = "Hello my FRIENDS"

x = txt.lower()

print(x)
```

```
hello my friends
```

Example——2

```python
#lower() method
a="Harry potter and the Goblet of Fire"
print(a.lower())
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\For
py"
harry potter and the goblet of fire
PS C:\Users\DELL\python>
```

index() method

Python String index() Method

Definition

Definition and Usage

The `index()` method finds the first occurrence of the specified value.

The `index()` method raises an exception if the value is not found.

Syntax

Syntax

```
string.index(value, start, end)
```

Parameter Values

Parameter	Description
value	Required. The value to search for
start	Optional. Where to start the search. Default is 0
end	Optional. Where to end the search. Default is to the end of the string

Example-1

Example

Where in the text is the first occurrence of the letter "e"?:

```
txt = "Hello, welcome to my world."

x = txt.index("e")

print(x)
```

```
1
```

Example-2

Example

Where in the text is the first occurrence of the letter "e" when you only search between position 5 and 10?:

```
txt = "Hello, welcome to my world."

x = txt.index("e", 5, 10)

print(x)
```

```
8
```

Difference between [index() and find() method]

The `index()` method is almost the same as the `find()` method, the only difference is that the `find()` method returns -1 if the value is not found. (See example below)

let's test

Example

If the value is not found, the find() method returns -1, but the index() method will raise an exception:

```
txt = "Hello, welcome to my world."

print(txt.find("q"))
print(txt.index("q"))
```

```
-1
Traceback (most recent call last):
  File "demo_ref_string_find_vs_index.py", line 4 in <module>
    print(txt.index("q"))
ValueError: substring not found
```

Example-3

```python
#index() method
a="Harry potter and the Goblet of Fire"
print(a.index("o",15,34))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python\Fc
py"
22
PS C:\Users\DELL\python>
```

Capitalize() method

Python String capitalize() Method

Definition

Definition and Usage

The `capitalize()` method returns a string where the first character is upper case, and the rest is lower case.

Syntax

Syntax

```
string.capitalize()
```

Example-1

Example

Upper case the first letter in this sentence:

```
txt = "hello, and welcome to my world."

x = txt.capitalize()

print (x)
```

```
Hello, and welcome to my world.
```

Example-2

Example

The first character is converted to upper case, and the rest are converted to lower case:

```
txt = "python is FUN!"

x = txt.capitalize()

print (x)
```

```
Python is fun!
```

Example-3

Example

See what happens if the first character is a number:

```
txt = "36 is my age."

x = txt.capitalize()

print (x)
```

```
36 is my age
```

Example-4

```
1   #capitalize() method
2   a="Harry potter and the Goblet of Fire"
3   print(a.capitalize())
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Co

PS C:\Users\DELL\python> python -u "c:\Users\DELL\python
py"
Harry potter and the goblet of fire
PS C:\Users\DELL\python>
```

casefold() method

Python String casefold() Method

Definition

Definition and Usage

The `casefold()` method returns a string where all the characters are lower case.

Syntax

Syntax

```
string.casefold()
```

Example-1

Example

Make the string lower case:

```
txt = "Hello, And Welcome To My World!"

x = txt.casefold()

print(x)
```

```
hello, and welcome to my world!
```

Are lower() and casefold() same? ——not exactly, casefold() method is more stronger than lower()

This method is similar to the `lower()` method, but the `casefold()` method is stronger, more aggressive, meaning that it will convert more characters into lower case, and will find more matches when comparing two strings and both are converted using the `casefold()` method.

format() method

Python String format() Method

Definition

Definition and Usage

The `format()` method formats the specified value(s) and insert them inside the string's placeholder.

The placeholder is defined using curly brackets: {}. Read more about the placeholders in the Placeholder section below.

The `format()` method returns the formatted string.

Syntax

Syntax

```
string.format(value1, value2...)
```

parameters

Parameter Values

Parameter	Description
value1, value2...	Required. One or more values that should be formatted and inserted in the string. The values are either a list of values separated by commas, a key=value list, or a combination of both. The values can be of any data type.

Here different formats(all are given the same result)

The Placeholders

The placeholders can be identified using named indexes `{price}`, numbered indexes `{0}`, or even empty placeholders `{}`.

Example

Using different placeholder values:

```
txt1 = "My name is {fname}, I'm {age}".format(fname = "John", age = 36)
txt2 = "My name is {0}, I'm {1}".format("John",36)
txt3 = "My name is {}, I'm {}".format("John",36)
```

Example-1

```python
#named indexes:
txt1 = "My name is {fname}, I'm {age}".format(fname = "John", age = 36)
#numbered indexes:
txt2 = "My name is {0}, I'm {1}".format("John",36)
#empty placeholders:
txt3 = "My name is {}, I'm {}".format("John",36)

print(txt1)
print(txt2)
print(txt3)
```

```
My name is John, I'm 36
My name is John, I'm 36
My name is John, I'm 36
```

Example-2

Example

Insert the price inside the placeholder, the price should be in fixed point, two-decimal format:

```python
txt = "For only {price:.2f} dollars!"
print(txt.format(price = 49))
```

```
For only 49.00 dollars!
```

Example-3

```python
#format() method
name="Joti"
age=24
b="my name is {}.and age is {}"
print(b.format(name,age))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\DELL\python> python -u "c:\Users\DELL\pytho
py"
my name is Joti.and age is 24
PS C:\Users\DELL\python>
```

Center() method

Python String center() Method

Definition

Definition and Usage

The `center()` method will center align the string, using a specified character (space is default) as the fill character.

Syntax

Syntax

```
string.center(length, character)
```

Parameter Values

Parameter	Description
length	Required. The length of the returned string
character	Optional. The character to fill the missing space on each side. Default is " " (space)

Example-1

Example

Using the letter "O" as the padding character:

```
txt = "banana"

x = txt.center(20, "O")

print(x)
```

```
0000000banana0000000
```

Example-2

```
1   #center()
2
3   name="john"
4   print(name.center(20))
```

```
TERMINAL  ...                    Code  +  v  
PS C:\Users\DELL\python> python -u "c:\U
L\python\For_practice\2.py"
        john
PS C:\Users\DELL\python>
```

isalnum() method

Python String isalnum() Method

Definition

Definition and Usage

The `isalnum()` method returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9).

Example of characters that are not alphanumeric: (space)!#%&? etc.

Syntax

Syntax

```
string.isalnum()
```

Parameter Values

No parameters.

Example-1

Example

Get your own Pyt

Check if all the characters in the text are alphanumeric:

```
txt = "Company12"

x = txt.isalnum()

print(x)
```

```
True
```

Example-2(space counted—-that's why false)

Example

Check if all the characters in the text is alphanumeric:

```
txt = "Company 12"

x = txt.isalnum()

print(x)
```

```
False
```

sample

```
a = "hello"
b = "Hello123"
c = "123456"
d = "HELLO"
e = " "
f = "Hello 123@"
g = "1.234"
```

Example-3

```
a="hello"
b="Hello123"
c="123456"
d="HELLO"
e=" "
f="Hello 123@"
g="1.234"

#isalnum()
print(a,a.isalnum())
print(b,b.isalnum())
print(c,c.isalnum())
print(d,d.isalnum())
print(e,e.isalnum())
print(f,f.isalnum())
print(g,g.isalnum())
```

```
hello True
Hello123 True
123456 True
HELLO True
  False
Hello 123@ False
1.234 False
```

isalpha() method

Python String isalpha() Method

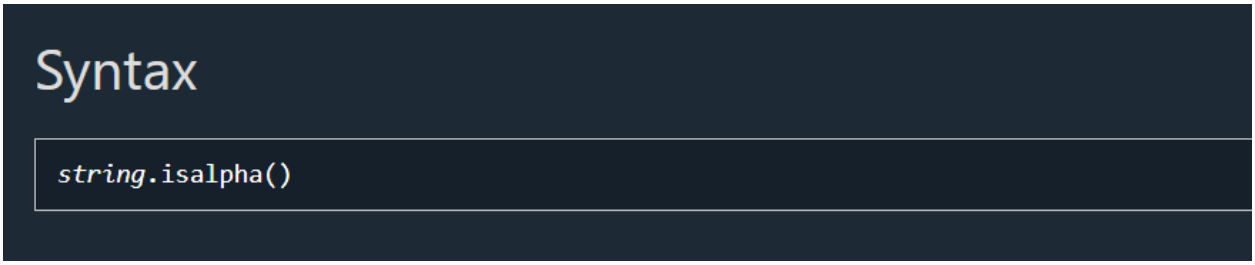
Definition

Definition and Usage

The `isalpha()` method returns True if all the characters are alphabet letters (a-z).

Example of characters that are not alphabet letters: (space)!#%&? etc.

Syntax



Example-1

```
txt = "CompanyX"

x = txt.isalpha()

print(x)
```

```
True
```

Example-2

Example

Check if all the characters in the text is alphabetic:

```
txt = "Company10"

x = txt.isalpha()

print(x)
```

```
txt = "Company10"

x = txt.isalpha()

print(x)
```

```
False
```