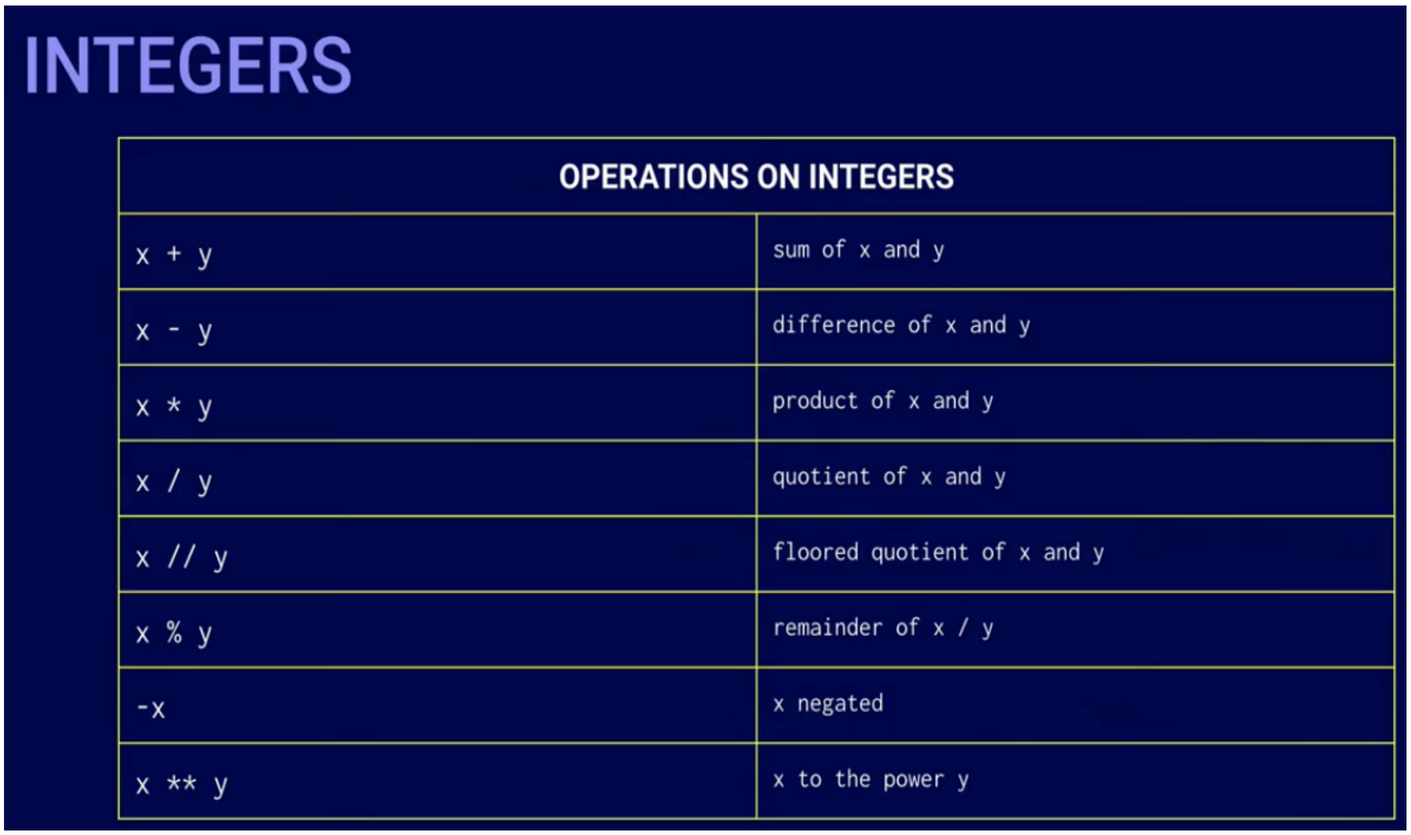
INTEGERS

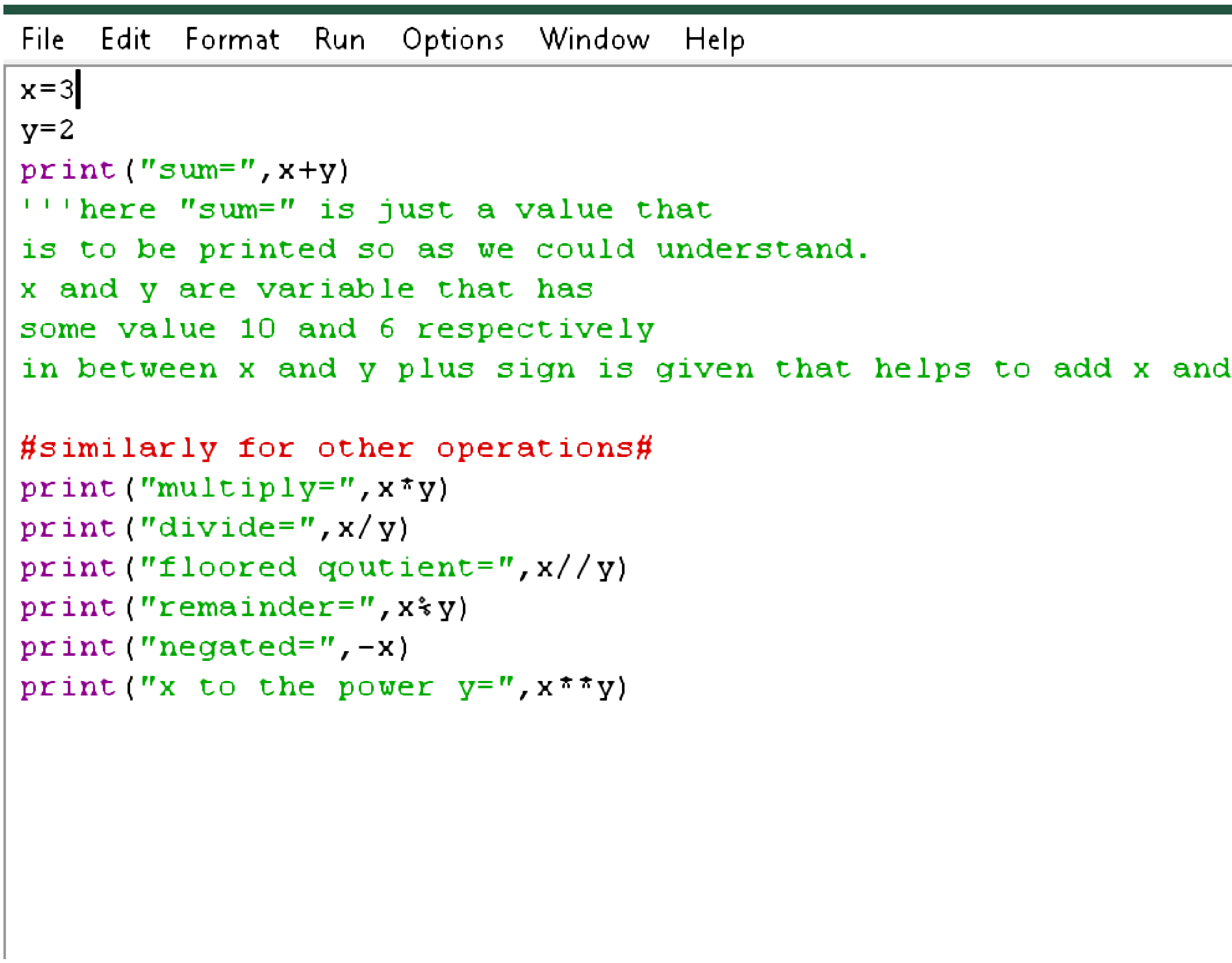
- Represents whole numbers
  3 54763576 0 -56
- Can be zero, positive or negative number
- For negative number we use minus sign (-)
- any quotes (‘’) commas (,) not allowed for example,
  10,000 ✗ 10000 ✓
  ‘5600’ 5600
- instead (,) you can use underscore (_),
  10,000 ✗ 10_000 ✓



INTEGERS

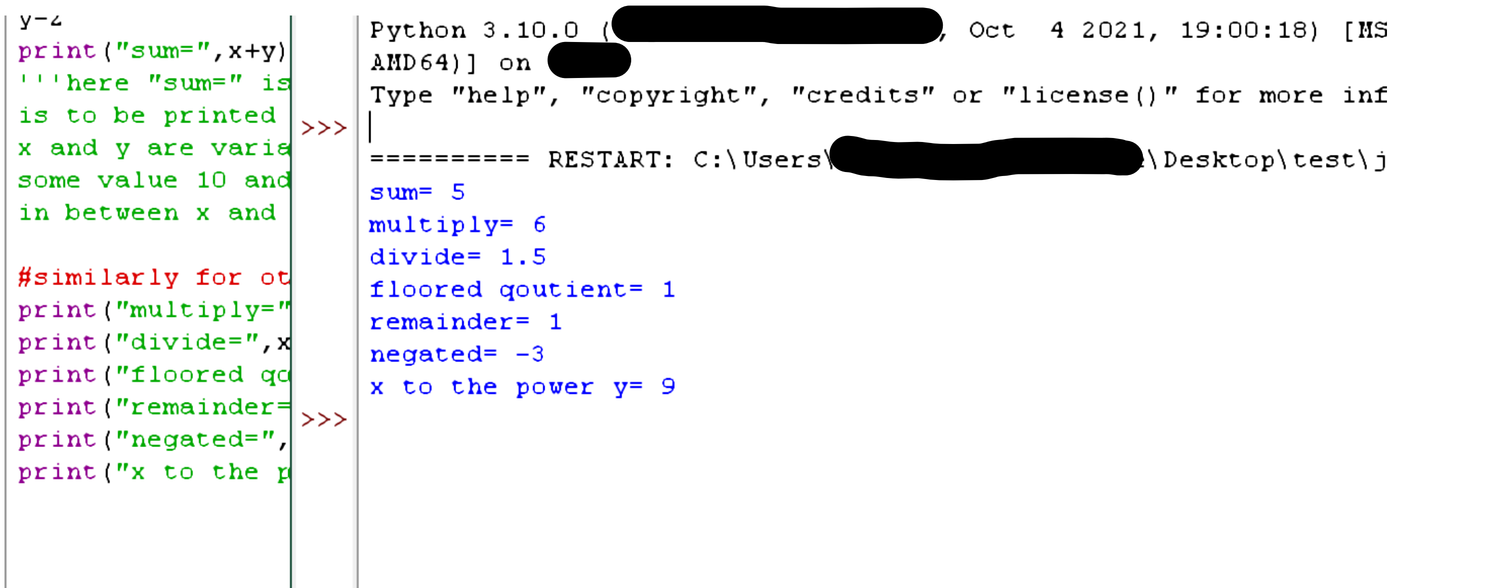
OPERATIONS ON INTEGERS	
`x + y`	sum of x and y
`x - y`	difference of x and y
`x * y`	product of x and y
`x / y`	quotient of x and y
`x // y`	floored quotient of x and y
`x % y`	remainder of x / y
`-x`	x negated
`x ** y`	x to the power y

Quick example on integers



```
x=3
y=2
print("sum=",x+y)
'''here "sum=" is just a value that
is to be printed so as we could understand.
x and y are variable that has
some value 10 and 6 respectively
in between x and y plus sign is given that helps to add x and

#similarly for other operations#
print("multiply=",x*y)
print("divide=",x/y)
print("floored qoutient=",x//y)
print("remainder=",x%y)
print("negated=",-x)
print("x to the power y=",x**y)
```

Output:



```
y=2
print("sum=",x+y)
'''here "sum=" is
is to be printed
x and y are varia
some value 10 and
in between x and

#similarly for ot
print("multiply="
print("divide=",x
print("floored qo
print("remainder=
print("negated=",
print("x to the p
```

```
Python 3.10.0 ([redacted], Oct  4 2021, 19:00:18) [MS
AMD64)] on [redacted]
Type "help", "copyright", "credits" or "license()" for more inf
>>> 
========== RESTART: C:\Users\[redacted]\Desktop\test\j
sum= 5
multiply= 6
divide= 1.5
floored qoutient= 1
remainder= 1
negated= -3
x to the power y= 9
>>>
```

Float

- Represents floating point numbers – numbers that contain a fractional component. Contains decimal point

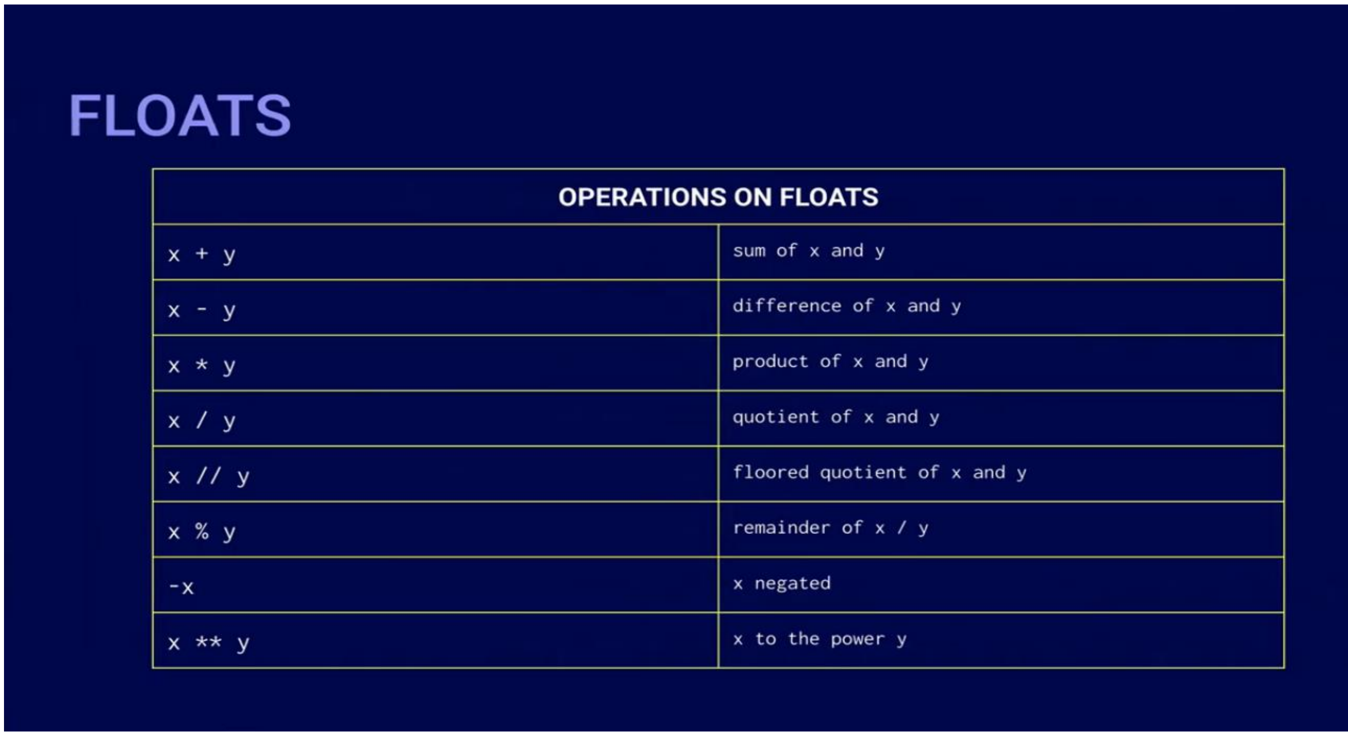
  3.5 547.63576 0.0 -5.56
- Can be zero, positive or negative number
- For negative number we use minus sign (-)
- any quotes (‘’) commas (,) not allowed for example,

  1,000.56 ✗ 1000.56 ✓

  ’86.90’ 86.90
- instead (,) you can use underscore (_),

  1,876.43 ✗ 1_876.43 ✓

Float are also considered as numbers hence it has same operations like integers.



FLOATS

OPERATIONS ON FLOATS	
x + y	sum of x and y
x - y	difference of x and y
x * y	product of x and y
x / y	quotient of x and y
x // y	floored quotient of x and y
x % y	remainder of x / y
-x	x negated
x ** y	x to the power y

Strings

- Represents a sequence of characters
- Syntax:
  - Enclosed in quotation marks ('like this!') or
  - It can also be enclosed in double quotations like,
    ("hello, world! ")
- Quotation marks of strings are not displayed at output screen
  - Example:
    - Print("hello, world!")

    Output:

        Hello, world!

- Every single letter is counted in string including spaces between two words which is identified by index positive and negative numbers

Here 0 to 12 are positive index numbers

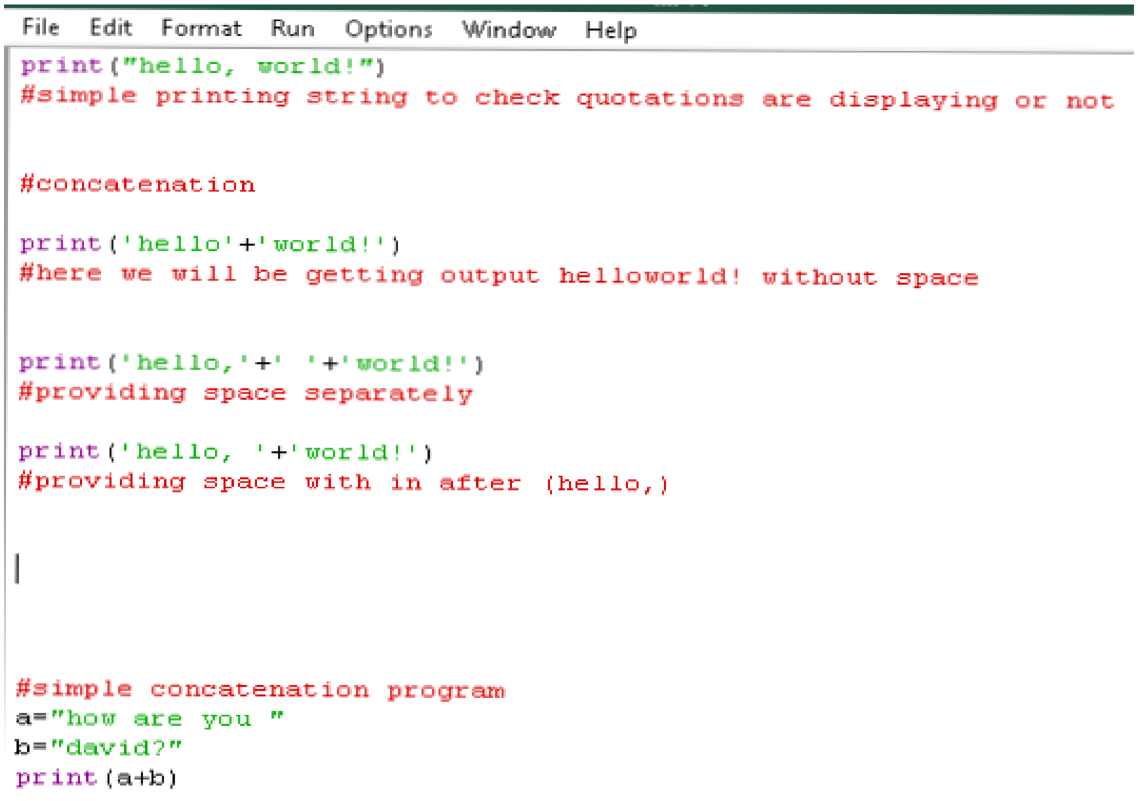
O	1	2	3	4	5	6	7	8	9	10	11	12
H	E	L	L	O	,		W	O	R	L	D	!

Here -13 to -1 are negative index numbers

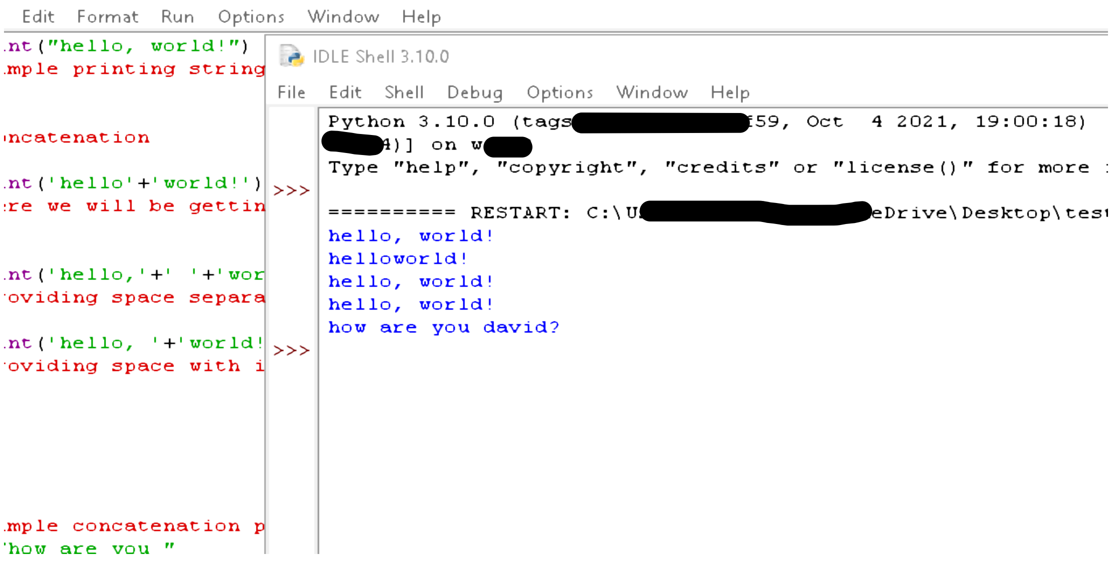
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
H	E	L	L	O	,		W	O	R	L	D	!

- **Concatenation:**
  The method of adding two strings to get together for example.
    - Syntax:
      Print('hello'+'world')

- Quick example:



```
print("hello, world!")
#simple printing string to check quotations are displaying or not


#concatenation

print('hello'+'world!')
#here we will be getting output helloworld! without space


print('hello,'+' '+'world!')
#providing space separately

print('hello, '+'world!')
#providing space with in after (hello,)




#simple concatenation program
a="how are you "
b="david?"
print(a+b)
```


Edit Format Run Options Window Help
IDLE Shell 3.10.0
File Edit Shell Debug Options Window Help

```
.nt("hello, world!")
.mple printing string

ncatenation

.nt('hello'+'world!')
:re we will be gettin

.nt('hello,'+' '+'wor
oviding space separa

.nt('hello, '+'world!
oviding space with i




.mple concatenation p
'how are you "
```

```
Python 3.10.0 (tags[redacted]59, Oct  4 2021, 19:00:18)
[redacted]4)] on w[redacted]
Type "help", "copyright", "credits" or "license()" for more :
>>>
========== RESTART: C:\U[redacted]eDrive\Desktop\test
hello, world!
helloworld!
hello, world!
hello, world!
how are you david?
>>>
```

- **Length:**

  Shows the length of string including spaces by using a function len()

  - Syntax:

    Print(len('hello'))

    Output:

    5

- Depending upon length we can do several operations like slicing, indexing.
- For indexing syntax:
  - Print(a[n])

0	1	2	3	4	5	6	7
B	A	T	H	R	O	O	M

- For slicing syntax:
  - Print(a[start:end])
    - Start = index number to start from
    - End = index number to end till before

0	1	2	3	4	5
P	Y	T	H	O	N

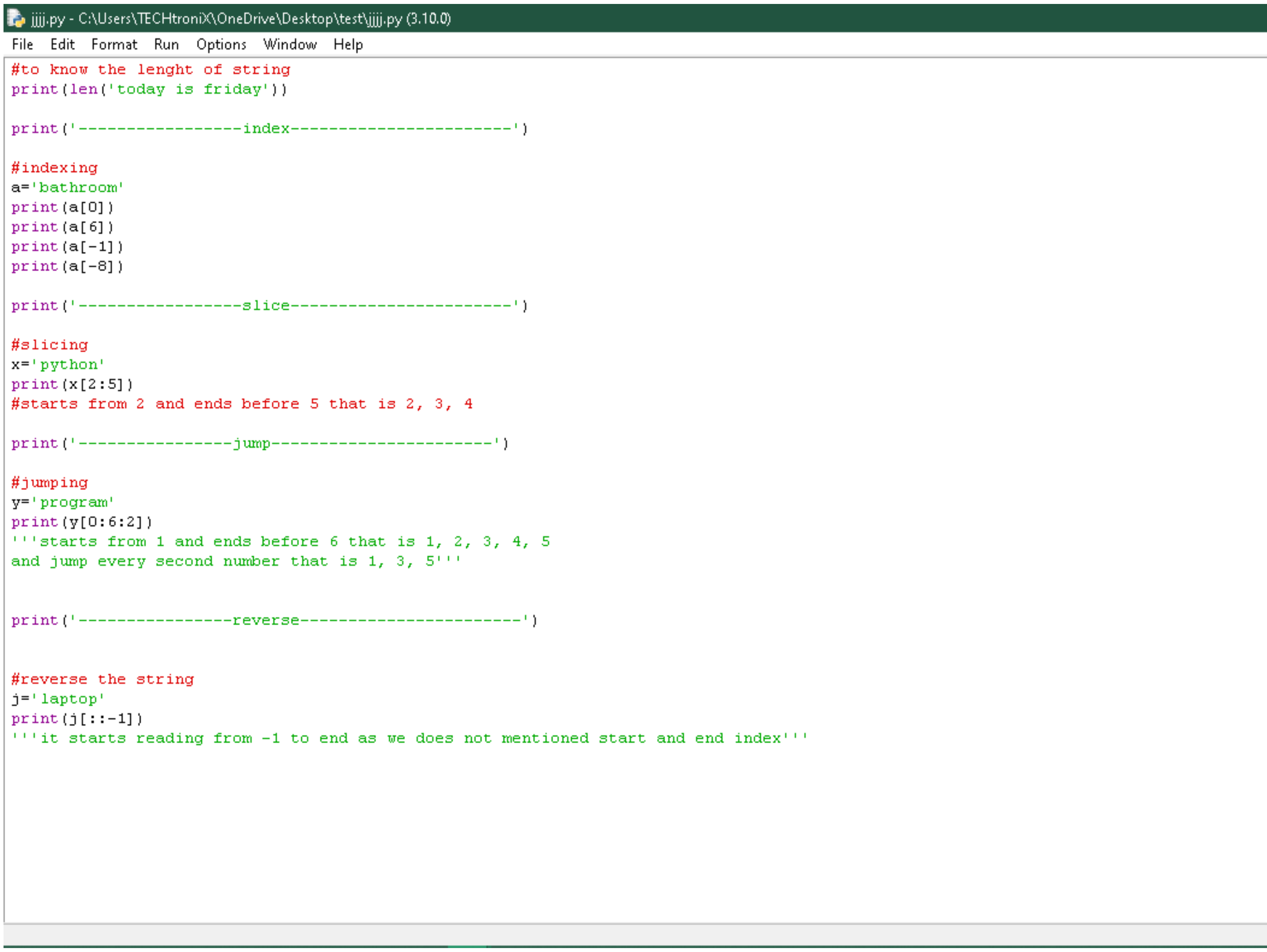
- For jump syntax:
  - Print(a[start:end:jump])
    - Jump = number that jumps to next index

0	1	2	3	4	5	6
P	R	O	G	R	A	m

- For reverse syntax:
  - Print(a[::-1])
    - Leave start and end index empty [::-1]
    - It reads letters from -1 that is negative index.

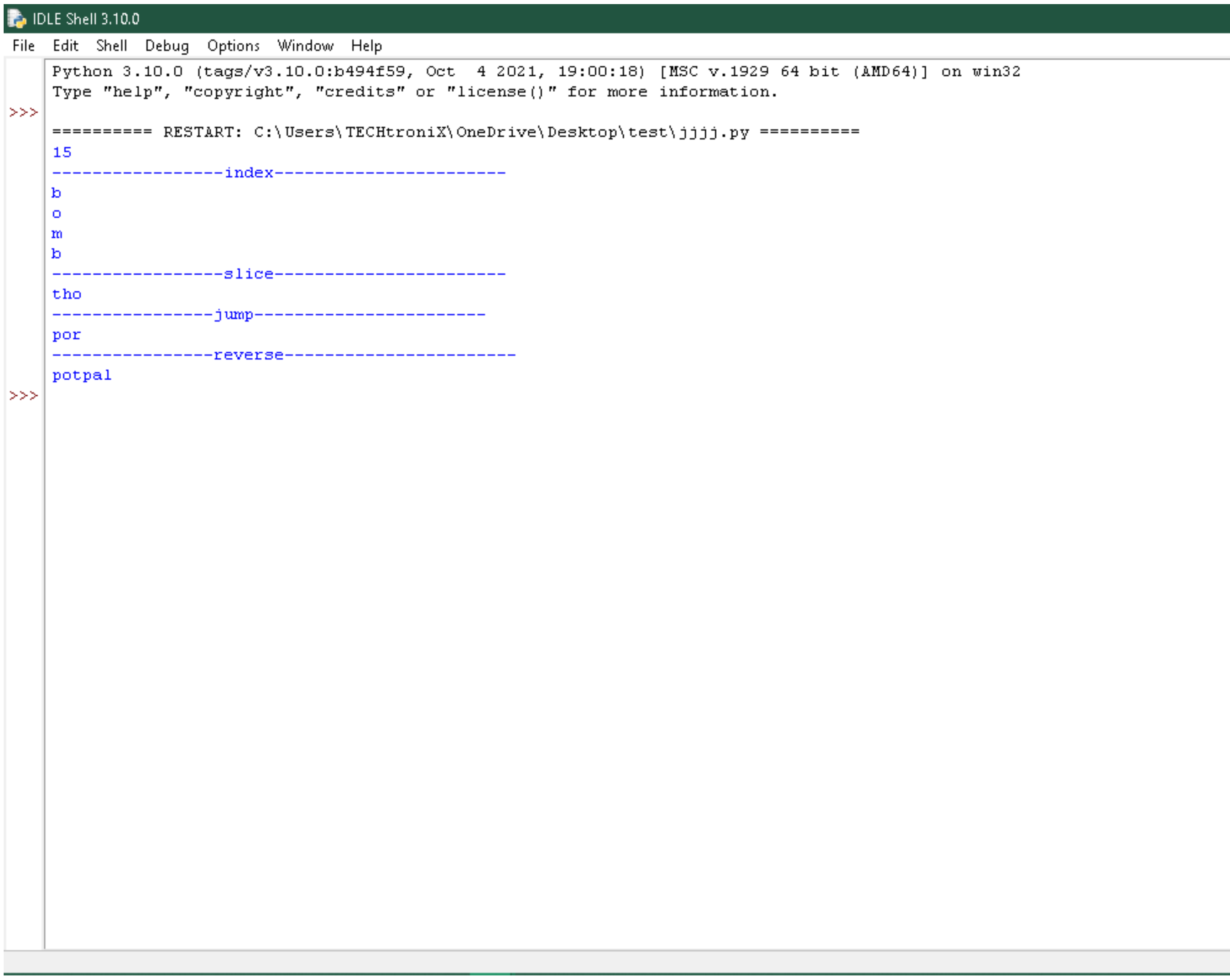
-6	-5	-4	-3	-2	-1
L	A	P	T	O	P

- **Quick example**


jjjj.py - C:\Users\TECHtroniX\OneDrive\Desktop\test\jjjj.py (3.10.0)
File Edit Format Run Options Window Help

```
#to know the lenght of string
print(len('today is friday'))

print('-----------------index-----------------------')

#indexing
a='bathroom'
print(a[0])
print(a[6])
print(a[-1])
print(a[-8])

print('-----------------slice-----------------------')

#slicing
x='python'
print(x[2:5])
#starts from 2 and ends before 5 that is 2, 3, 4

print('----------------jump-----------------------')

#jumping
y='program'
print(y[0:6:2])
'''starts from 1 and ends before 6 that is 1, 2, 3, 4, 5
and jump every second number that is 1, 3, 5'''


print('----------------reverse-----------------------')


#reverse the string
j='laptop'
print(j[::-1])
'''it starts reading from -1 to end as we does not mentioned start and end index'''
```

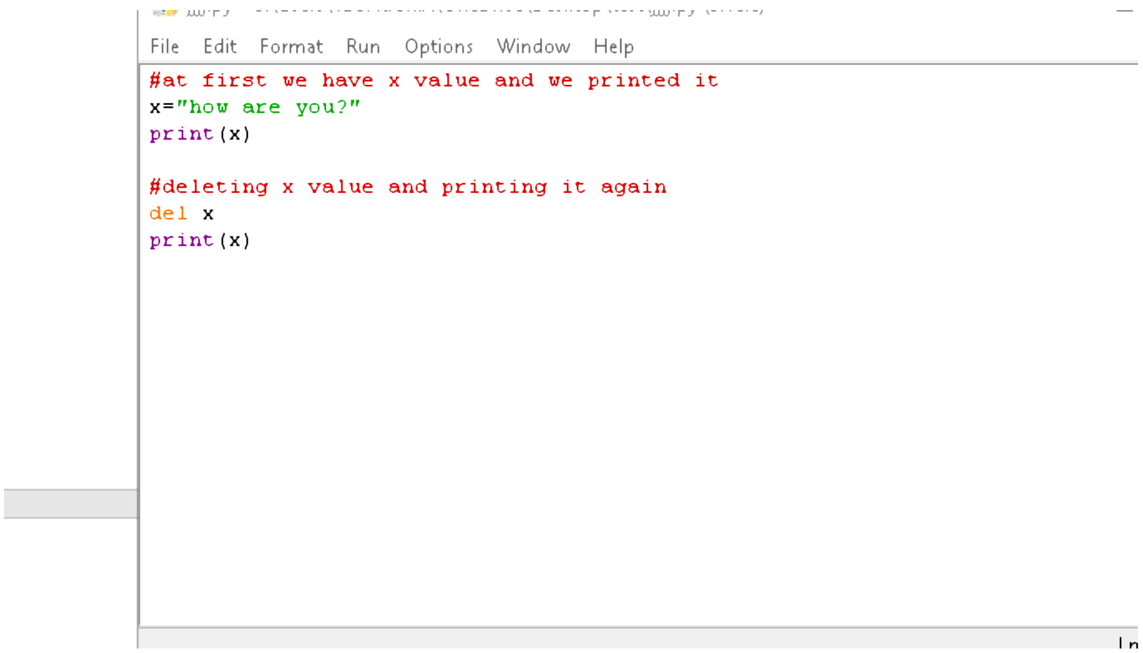
- **Output:**



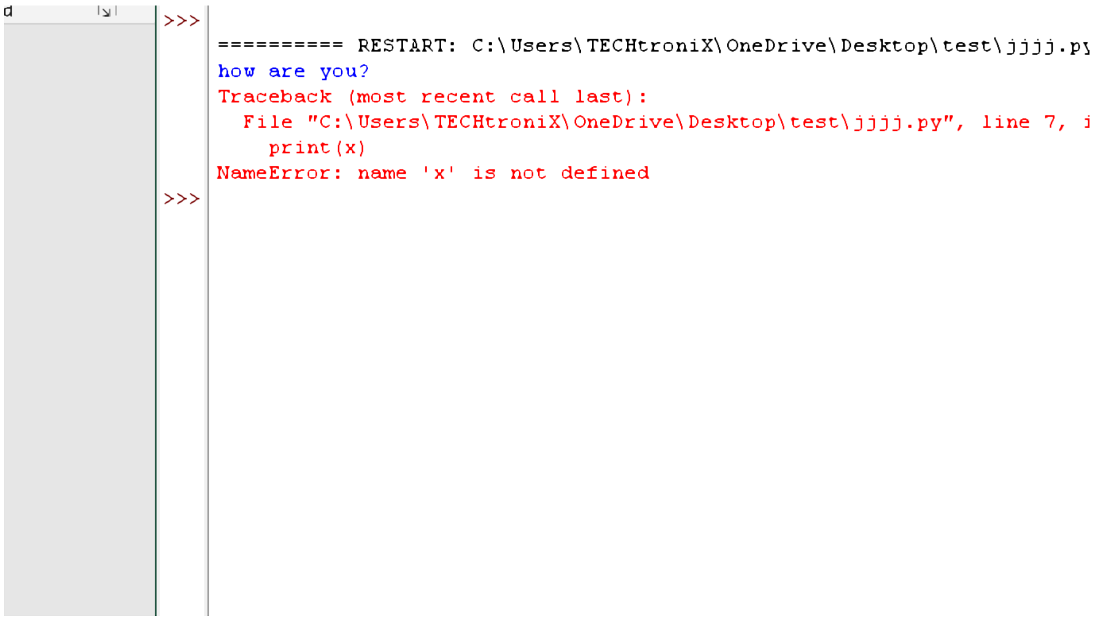
```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\TECHtroniX\OneDrive\Desktop\test\jjjj.py ==========
15
-----------------index----------------------
b
o
m
b
-----------------slice----------------------
tho
----------------jump----------------------
por
----------------reverse----------------------
potpal
>>>
```

- For delete:
    - Syntax:
        - del x

example given below



```
#at first we have x value and we printed it
x="how are you?"
print(x)

#deleting x value and printing it again
del x
print(x)
```

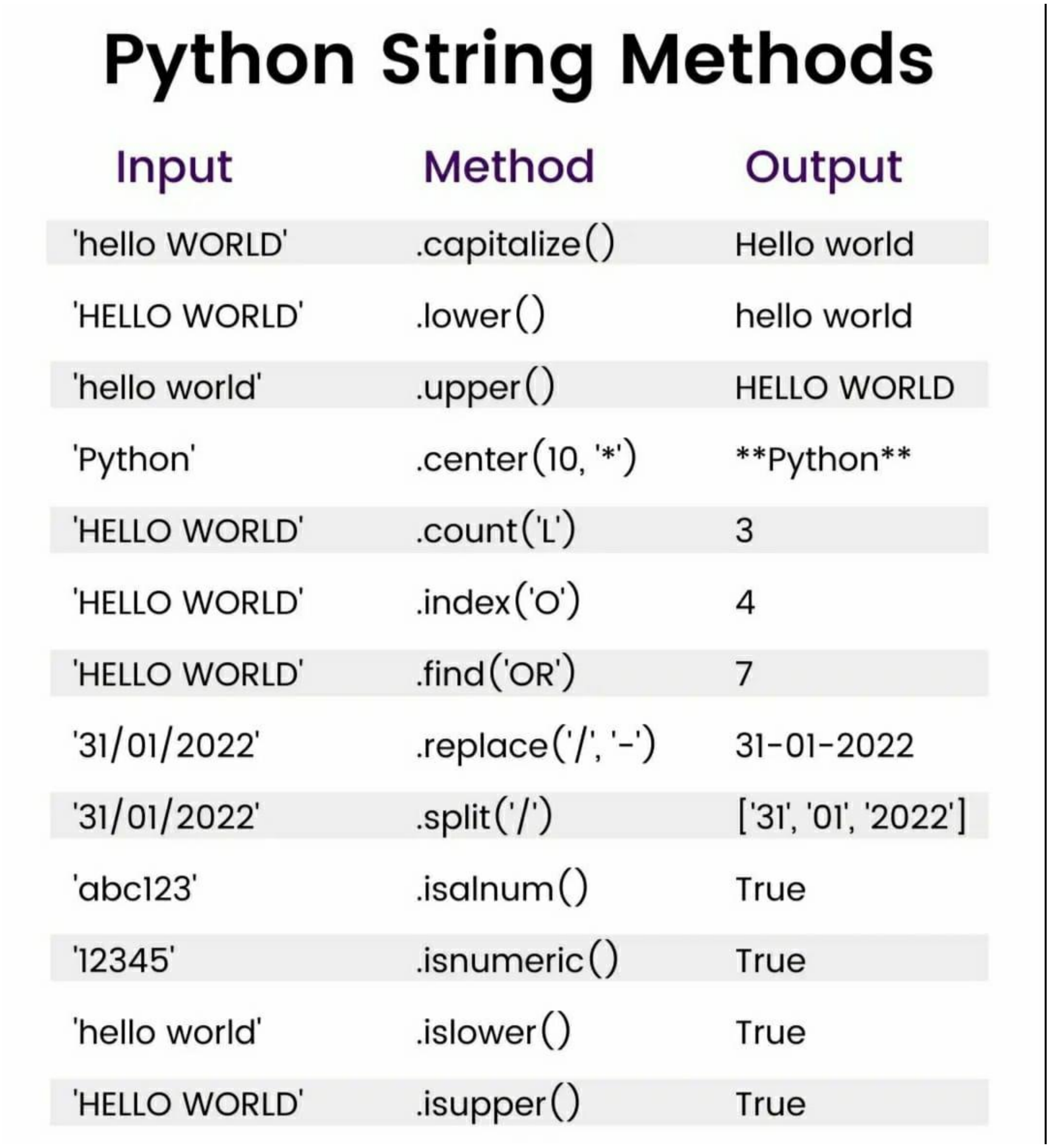


```
>>>
========== RESTART: C:\Users\TECHtroniX\OneDrive\Desktop\test\jjjj.py
how are you?
Traceback (most recent call last):
  File "C:\Users\TECHtroniX\OneDrive\Desktop\test\jjjj.py", line 7, i
    print(x)
NameError: name 'x' is not defined
>>>
```

Here we can see that x is deleted by causing error

Extras:

Syntax: print(‘hello WORLD’.capitalize())



Python String Methods

Input	Method	Output
'hello WORLD'	.capitalize()	Hello world
'HELLO WORLD'	.lower()	hello world
'hello world'	.upper()	HELLO WORLD
'Python'	.center(10, '*')	**Python**
'HELLO WORLD'	.count('L')	3
'HELLO WORLD'	.index('O')	4
'HELLO WORLD'	.find('OR')	7
'31/01/2022'	.replace('/', '-')	31-01-2022
'31/01/2022'	.split('/')	['31', '01', '2022']
'abc123'	.isalnum()	True
'12345'	.isnumeric()	True
'hello world'	.islower()	True
'HELLO WORLD'	.isupper()	True