```
In [1]: 15 % 2

Out[1]: 1

In [2]: 15 %% 2

  Cell In[2], line 1
    15 %% 2
        ^
SyntaxError: invalid syntax

In [3]: 3 + 'nit'

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[3], line 1
----> 1 3 + 'nit'

TypeError: unsupported operand type(s) for +: 'int' and 'str'

In [4]: print('C:\Users')

  Cell In[4], line 1
    print('C:\Users')
          ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position
2-3: truncated \UXXXXXXXX escape

In [8]: print('D:\NIT')

  Cell In[8], line 1
    print('D:\NIT')
          ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position
2-3: malformed \N character escape

In [9]: name1 = 'fine'
        name1

Out[9]: 'fine'

In [10]: name1[0:1]

Out[10]: 'f'

In [11]: name1

Out[11]: 'fine'

In [12]: name1[1:]

Out[12]: 'ine'

In [14]: 'd' + name1[1:]

Out[14]: 'dine'
```

In [16]:
```
help()
```

```
Welcome to Python 3.12's help utility! If this is your first time using
Python, you should definitely check out the tutorial at
https://docs.python.org/3.12/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To get a list of available
modules, keywords, symbols, or topics, enter "modules", "keywords",
"symbols", or "topics".

Each module also comes with a one-line summary of what it does; to list
the modules whose name or summary contain a given string such as "spam",
enter "modules spam".

To quit this help utility and return to the interpreter,
enter "q" or "quit".

You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)".  Executing "help('string')"
has the same effect as typing a particular string at the help> prompt.
```

RANGE

In [17]:
```
range(0,10)
```

Out[17]:
```
range(0, 10)
```

In [18]:
```
list(range(0,10))
```

Out[18]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [2]:
```
r = range(10)
```

In [3]:
```
list(range(10,20))
```

Out[3]:
```
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

In [4]:
```
for i in r:
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [5]:
```
list(range(10,100,5))
```

Out[5]:
```
[10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95]
```

```
In [6]: list(r)

Out[6]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [9]: r1 = range(10,50,7)
        r1

Out[9]: range(10, 50, 7)

In [11]: for i in r1:
             print(i) #prints all numbers with escaping

10
17
24
31
38
45

In [12]: r1[5]

Out[12]: 45

In [13]: range(10,400,50,2)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[13], line 1
----> 1 range(10,400,50,2)

TypeError: range expected at most 3 arguments, got 4

In [14]: list(range(20,130,10))

Out[14]: [20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120]

In [15]: range(all)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[15], line 1
----> 1 range(all)

TypeError: 'builtin_function_or_method' object cannot be interpreted as an intege
r

In [16]: range(any)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[16], line 1
----> 1 range(any)

TypeError: 'builtin_function_or_method' object cannot be interpreted as an intege
r
```

arithmetic operattors

```
In [24]: x1, y1 = 10,15
```

```
In [25]: x1 + y1
```

```
Out[25]: 25
```

```
In [26]: x1 - y1
```

```
Out[26]: -5
```

```
In [27]: x1 * y1
```

```
Out[27]: 150
```

```
In [28]: x1 / y1
```

```
Out[28]: 0.6666666666666666
```

```
In [29]: x1 // y1
```

```
Out[29]: 0
```

```
In [30]: x1 ** y1
```

```
Out[30]: 1000000000000000
```

```
In [31]: x1 % y1
```

```
Out[31]: 10
```

```
In [ ]:
```