

```
In [ ]: #28th Oct- Basic Python
```

```
In [ ]: #Rename the string 'fine' to 'dine'
```

```
In [1]: name1 = 'fine'
name1
```

```
Out[1]: 'fine'
```

```
In [5]: name1[0 : 1] = 'd'
```

```
-----
TypeError                                Traceback (most recent call last)
Cell In[5], line 1
----> 1 name1[0 : 1] = 'd'

TypeError: 'str' object does not support item assignment
```

```
In [6]: name1[1:]
```

```
Out[6]: 'ine'
```

```
In [7]: 'd' + name1[1:]
```

```
Out[7]: 'dine'
```

```
In [ ]: #help () options
```

```
In [8]: help()
```

Welcome to Python 3.13's help utility! If this is your first time using Python, you should definitely check out the tutorial at <https://docs.python.org/3.13/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", "quit" or "exit".

Here is a list of the Python keywords. Enter any keyword to get more help.

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

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.

In [10]: `help(list)`

Help on class list in module builtins:

```
class list(object)
| list(iterable=(), /)
|
| Built-in mutable sequence.
|
| If no argument is given, the constructor creates a new empty list.
| The argument must be an iterable if specified.
|
| Methods defined here:
|
| __add__(self, value, /)
|     Return self+value.
|
| __contains__(self, key, /)
|     Return bool(key in self).
|
| __delitem__(self, key, /)
|     Delete self[key].
|
| __eq__(self, value, /)
|     Return self==value.
|
| __ge__(self, value, /)
|     Return self>=value.
|
| __getattr__(self, name, /)
|     Return getattr(self, name).
|
| __getitem__(self, index, /)
|     Return self[index].
|
| __gt__(self, value, /)
|     Return self>value.
|
| __iadd__(self, value, /)
|     Implement self+=value.
|
| __imul__(self, value, /)
|     Implement self*=value.
|
| __init__(self, /, *args, **kwargs)
|     Initialize self. See help(type(self)) for accurate signature.
|
| __iter__(self, /)
|     Implement iter(self).
|
| __le__(self, value, /)
|     Return self<=value.
|
| __len__(self, /)
|     Return len(self).
|
| __lt__(self, value, /)
|     Return self<value.
```

```

__mul__(self, value, /)
    Return self*value.

__ne__(self, value, /)
    Return self!=value.

__repr__(self, /)
    Return repr(self).

__reversed__(self, /)
    Return a reverse iterator over the list.

__rmul__(self, value, /)
    Return value*self.

__setitem__(self, key, value, /)
    Set self[key] to value.

__sizeof__(self, /)
    Return the size of the list in memory, in bytes.

append(self, object, /)
    Append object to the end of the list.

clear(self, /)
    Remove all items from list.

copy(self, /)
    Return a shallow copy of the list.

count(self, value, /)
    Return number of occurrences of value.

extend(self, iterable, /)
    Extend list by appending elements from the iterable.

index(self, value, start=0, stop=9223372036854775807, /)
    Return first index of value.

    Raises ValueError if the value is not present.

insert(self, index, object, /)
    Insert object before index.

pop(self, index=-1, /)
    Remove and return item at index (default last).

    Raises IndexError if list is empty or index is out of range.

remove(self, value, /)
    Remove first occurrence of value.

    Raises ValueError if the value is not present.

reverse(self, /)

```

```

Reverse *IN PLACE*.

sort(self, /, *, key=None, reverse=False)
    Sort the list in ascending order and return None.

    The sort is in-place (i.e. the list itself is modified) and stable (i.e. the
    order of two equal elements is maintained).

    If a key function is given, apply it once to each list item and sort them,
    ascending or descending, according to their function values.

    The reverse flag can be set to sort in descending order.

-----
Class methods defined here:

__class_getitem__(object, /)
    See PEP 585

-----
Static methods defined here:

__new__(*args, **kwargs)
    Create and return a new object.  See help(type) for accurate signature.

-----
Data and other attributes defined here:

__hash__ = None

```

```
In [ ]: #Swap variable
```

```
In [13]: a = 5
         b = 6
```

```
In [14]: #Wrong method:
         a = b
         b = a
         print(a)
         print(b)
```

```
6
6
```

```
In [17]: a = 5
         b = 6
```

```
In [18]: #Using3rd variable/temp variable :
         temp = a
         a = b
         b = temp
         print(a)
         print(b)
```

6  
5

```
In [22]: a = 5  
        b = 6
```

```
In [23]: #Without using 3rd variable (Addition and subtraction)  
        a = a+b  
        b = a-b  
        a = a-b  
        print(a)  
        print(b)
```

6  
5

```
In [26]: a = 5  
        b = 6
```

```
In [27]: #Simple and best method  
        a, b = b, a  
        print(a)  
        print(b)
```

6  
5

```
In [28]: #math module:  
        x = sqrt(25)
```

```
-----  
NameError                                Traceback (most recent call last)  
Cell In[28], line 2  
      1 #math module:  
----> 2 x = sqrt(25)  
  
NameError: name 'sqrt' is not defined
```

```
In [29]: #sqrt is a part of math library  
        import math  
        x = math.sqrt(25)  
        x
```

Out[29]: 5.0

```
In [30]: x1 = math.sqrt(15)  
        x1
```

Out[30]: 3.872983346207417

```
In [31]: print(math.floor(3.8729))
```

3

```
In [32]: print(math.ceil(3.8729))
```

4

```
In [33]: print(math.pow(3,2))    #pow is also in math library
```

9.0

```
In [34]: print(math.pow(3,3))
```

27.0

```
In [35]: print(math.pi)
```

3.141592653589793

```
In [36]: print(math.e)
```

2.718281828459045

```
In [37]: #Optimized way  
import math as m    #m is called Alias  
m.sqrt(10)
```

Out[37]: 3.1622776601683795

```
In [38]: #Using only specific functions from library  
from math import sqrt, pow  
pow(2,3)
```

Out[38]: 8.0

```
In [39]: round(pow(2,3))
```

Out[39]: 8

```
In [ ]: #Run python code using cli
```