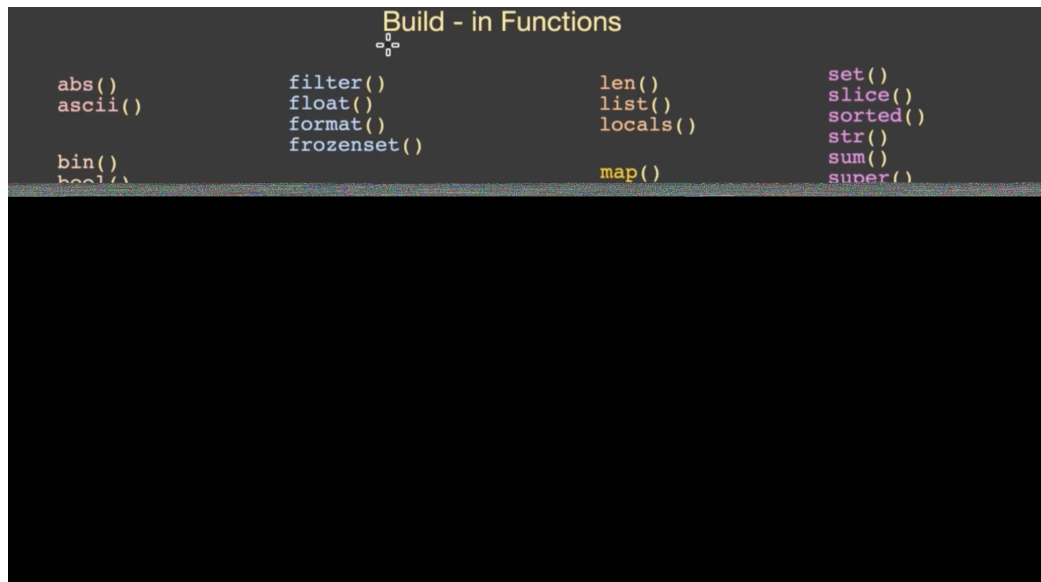


Build-in Function #1

- They are many **Build-in Functions / global functions** available in python
- The Build-in Functions are



- Lets see the working of these functions
- **abs() , ascii()**

```
>>>
>>> a = -15
>>> abs(a)
15
>>> b = -17.86
>>>
>>> abs(b)
17.86
>>>
>>> abs(3+4j)
5.0
>>> ascii('A')
'"A"'
>>> ascii(10)
'10'
>>> letter = '\u0521'
>>> letter
'ܐ'
>>> ascii(letter)
'"\\u0521"'
>>>
```

- `bin()` it takes number and convert it into binary form
- `bool()` convert anything into bool type
- `bytearray()` and `bytes()` they both are similar the difference is byte array is mutable where as bytes is immutable

```
>>>
>>> ba = bytearray(5)
>>> ba
bytearray(b'\x00\x00\x00\x00\x00')
>>> s1 = 'abcde'
>>> ba = bytearray(s1.encode())
>>> ba
bytearray(b'abcde')
>>>
>>> for i in ba:
>>>     print(i)
```

```
97
98
99
100
101
>>> ba.append(102)
>>> ba
bytearray(b'abcdef')
>>> b = bytes(s1.encode())
>>> b
b'abcde'
>>> |
```

- `callable()` we can know if the given identifier is a function or not
- `chr()` gives you the character for any given ascii code
- `complex()` used for creating complex datatype

```
>>> def add(a,b):  
        return a+b
```

```
>>> s1 = 'abcd'
```

```
>>>
```

```
>>> n = 10
```

```
>>>
```

```
>>> callable(n)
```

```
False
```

```
>>> callable(s1)
```

```
False
```

```
>>> callable(add)
```

```
True
```

```
>>>
```

```
>>> chr(65)
```

```
'A'
```

```
>>> ord('A')
```

```
65
```

```
>>>
```

- `dict()` used for creating a Dictionary
- `dir()` give details of particular class
- `divmod()` takes 2 parameters and gives division as well as modulus as result

```
>>>
>>> divmod(11,3)
(3, 2)
>>>
>>> q, r = divmod(13,4)
>>> q
3
>>> r
1
>>> divmod(14.3,3.2)
(4.0, 1.5)
>>> |
      |
```

- `enumerate()` gives indexing for all items in given sequence

```
>>>
>>> L = ['A', 'B', 'C', 'D', 'E']
>>> e = enumerate(L)
>>>
>>> e
<enumerate object at 0x7f9e7bc3ed80>
>>>
>>> for i in e:
>>>     print(i)

(0, 'A')
(1, 'B')
(2, 'C')
(3, 'D')
(4, 'E')
>>>
```

- `eval()` evaluates an expression
- `exec()` execute python statements

```
>>>
>>> eval('3 * 10 + 15 / 3')
35.0
>>> eval('2 ** 4 + 9')
25
>>>
>>> s = 'x=10\ny=20\nprint(x+y)'
>>>
>>> exec(s)
30
>>> x
10
>>> |
```