🐍 Python Built-in Functions Notes – By Rushil Chauhan

🔢 Type Conversion Functions

1. int() – kisi bhi value ko integer (poornaankh) me convert karta hai. Float ya string ko bhi integer bana sakta hai (decimal hata kar).

ex.,

```
a = int("10")

b = int(5.9)

print(a, b)  # Output: 10 5
```

---

2. float() – kisi bhi value ko float (decimal sankhya) me badal deta hai.

ex.,

```
x = float("12")

y = float(5)

print(x, y)  # Output: 12.0 5.0
```

---

3. str() – kisi bhi value ko string (text) me convert karta hai.

ex.,

```
s = str(123)

print(s + " is a number")  # Output: 123 is a number
```

---

4. bool() – kisi bhi value ko boolean me convert karta hai (True/False). Empty → False, non-empty → True.

ex.,

```
print(bool(0))       # False

print(bool("Hello")) # True
```

---

5. complex() – complex number banata hai (real + imaginary).

ex.,

```
z = complex(3, 4)
print(z)  # Output: (3+4j)
```

---

6. bytes() – immutable byte array banata hai.

ex.,

```
b = bytes("hello", "utf-8")
print(b)  # Output: b'hello'
```

---

7. bytearray() – mutable (badalne yogya) byte object banata hai.

ex.,

```
ba = bytearray("abc", "utf-8")
print(ba)  # Output: bytearray(b'abc')
```

---

8. memoryview() – memory-efficient object banata hai, jo data ko bina copy kiye access karta hai.

ex.,

```
mv = memoryview(b"Python")
print(mv[0])  # Output: 80
```

🗄 Sequence Functions

1. len() – kisi bhi string, list, tuple, dict ya sequence ka length (kitne elements hain) batata hai.

ex.,

```
print(len("hello"))      # Output: 5
print(len([1, 2, 3]))    # Output: 3
```

---

2. sum() – list ya tuple ke andar jitne numeric values hain unka total batata hai.

ex.,

```
print(sum([10, 20, 30]))  # Output: 60
```

---

3. max() – list, tuple ya string me se sabse badi value return karta hai.

ex.,

```
print(max([3, 9, 2]))     # Output: 9
print(max("python"))     # Output: y
```

---

4. min() – list, tuple ya string me se sabse chhoti value return karta hai.

ex.,

```
print(min([3, 9, 2]))     # Output: 2
print(min("python"))     # Output: h
```

---

5. sorted() – kisi sequence ko sorted (ascending) order me return karta hai. Original change nahi hoti.

ex.,

```
print(sorted([3, 1, 2]))     # Output: [1, 2, 3]
print(sorted("rushil"))     # Output: ['h', 'i', 'l', 'r', 's', 'u']
```

---

6. reversed() – sequence ko reverse karta hai (iterator deta hai).

ex.,

```
print(list(reversed([1, 2, 3])))  # Output: [3, 2, 1]
```

---

7. range() – numbers ka sequence banata hai (start to end tak). Mostly loops me use hota hai.

ex.,

```
for i in range(1, 5):
    print(i)
# Output: 1 2 3 4
```

---

8. enumerate() – kisi iterable me index ke sath values deta hai (useful in loops).

ex.,

```
for i, val in enumerate(['a', 'b']):
    print(i, val)
# Output: 0 a  1 b
```

---

9. zip() – multiple lists ko ek saath combine karta hai (tuple pairs banaata hai).

ex.,

```
a = [1, 2]
b = ['x', 'y']
print(list(zip(a, b)))  # Output: [(1, 'x'), (2, 'y')]
```

---

10. map() – har item par ek function apply karta hai (like converting all to string).

ex.,

```
print(list(map(str, [1, 2, 3])))  # Output: ['1', '2', '3']
```

---

11. filter() – condition ke basis par kuch hi elements ko nikaalta hai.

ex.,

```
print(list(filter(lambda x: x > 2, [1, 2, 3, 4])))  # Output: [3, 4]
```

➕ Math Functions

1. abs() – kisi bhi number ka absolute value (positive form) return karta hai.

ex.,

```
print(abs(-10))     # Output: 10
print(abs(5.7))     # Output: 5.7
```

---

2. round() – number ko round karta hai nearest value pe. Optional second argument: decimal places.

ex.,

```
print(round(3.567))      # Output: 4
print(round(3.567, 2))   # Output: 3.57
```

---

3. pow() – exponentiation karta hai (x ki power y). pow(x, y) = x^y

ex.,

```
print(pow(2, 3))        # Output: 8
print(pow(10, 2))       # Output: 100
```

---

4. divmod() – do number ka quotient aur remainder ek sath tuple me deta hai.

ex.,

```
print(divmod(10, 3))    # Output: (3, 1)
```

---

5. bin() – number ko binary string me convert karta hai.

ex.,

```
print(bin(10))          # Output: '0b1010'
```

---

6. hex() – number ko hexadecimal string me convert karta hai.

ex.,

```
print(hex(255))         # Output: '0xff'
```

---

7. oct() – number ko octal string me convert karta hai.

ex.,

```
print(oct(8))           # Output: '0o10'
```

🔍 Object/Type Handling Functions

1. type() – kisi bhi object ka type return karta hai.

ex.,

```
print(type("Rushil"))   # Output: <class 'str'>
print(type(5.0))        # Output: <class 'float'>
```

---

2. id() – kisi bhi object ka memory address return karta hai.

ex.,

```
a = 10
print(id(a))          # Output: unique memory location (int)
```

---

3. isinstance() – check karta hai ki object kisi particular type ka hai ya nahi.

ex.,

```
print(isinstance(10, int))      # Output: True
print(isinstance("hi", float))  # Output: False
```

---

4. issubclass() – check karta hai ki ek class dusri class ki subclass hai ya nahi.

ex.,

```
print(issubclass(bool, int))    # Output: True
```

---

5. getattr() – object ke andar se kisi attribute ko fetch karta hai (string ke through).

ex.,

```
class A:
   x = 10


print(getattr(A, 'x'))   # Output: 10
```

---

6. setattr() – object me naye attribute add/set karta hai.

ex.,

```
class A: pass
setattr(A, 'y', 20)
print(A.y)            # Output: 20
```

---

7. hasattr() – check karta hai ki object me koi attribute hai ya nahi.

ex.,

```
class B:
    name = "Rushil"


print(hasattr(B, 'name'))  # Output: True
```

---

8. delattr() – object ke kisi attribute ko delete karta hai.

ex.,

```
class C:
    age = 25


delattr(C, 'age')
print(hasattr(C, 'age'))  # Output: False
```

✅ Utility Functions

1. all() – agar list ya iterable ke saare elements True hain to True return karta hai.

ex.,

```
print(all([True, 1, "hi"]))     # Output: True
print(all([True, 0, "hi"]))     # Output: False
```

---

2. any() – agar list ya iterable me koi ek bhi True hua to True return karta hai.

ex.,

```
print(any([False, 0, "", 5]))   # Output: True
print(any([0, False]))          # Output: False
```

---

3. chr() – integer (Unicode) ko character me convert karta hai.

ex.,

```
print(chr(65))     # Output: 'A'
```

---

4. ord() – character ko Unicode number me convert karta hai.

ex.,

```
print(ord('A'))    # Output: 65
```

---

5. ascii() – non-ASCII characters ko escape form me convert karta hai.

ex.,

```
print(ascii("नमस्ते"))  # Output: '\u0928\u092e\u0938\u094d\u0924\u0947'
```

---

6. repr() – object ka official string representation deta hai.

ex.,

```
x = "Rushil"
print(repr(x))     # Output: "'Rushil'"
```

□ Evaluation & Code Execution

---

7. eval() – string me likha expression evaluate karta hai (sirf expressions).

ex.,

```
x = eval("2 + 3")
print(x)   # Output: 5
```

---

8. exec() – string ke andar likha hua **pure Python code** execute karta hai.

ex.,

```
exec("a = 5\nprint(a)")   # Output: 5
```

---

9. compile() – code string ko compile karta hai, jise eval() ya exec() se chala sakte hain.

ex.,

```
code = compile("print('Hello')", "", "exec")
exec(code)
```

□ Help & Debugging Tools

10. help() – kisi bhi object ya function ki help/instructions deta hai (console me).

ex.,

```
help(str)
```

---

11. dir() – kisi bhi object ke methods aur attributes ki list deta hai.

ex.,

```
print(dir([]))    # List ke sab methods
```

---

12. vars() – kisi object ke attributes ko dict form me return karta hai.

ex.,

```
class A:
    x = 10


print(vars(A))    # Output: {'x': 10}
```

---

13. globals() – current program ke global variables ka dictionary deta hai.

ex.,

```
print(globals().keys())  # Shows all global names
```

---

14. locals() – local scope ke variables return karta hai.

ex.,

```
def test():
    x = 5
    print(locals())


test()
```

---

15. callable() – check karta hai ki object ko call kiya ja sakta hai ya nahi (jaise function).

ex.,

```
print(callable(print))   # Output: True
```

🗀 File Handling

16. open() – file ko read/write/edit mode me open karta hai.

ex.,

```
file = open("data.txt", "w")
file.write("Hello")
file.close()
```

□ Class-related Built-ins

17. property() – getter/setter create karta hai bina method call kiye.

ex.,

```
class A:
   def __init__(self): self._x = 0
   @property
   def x(self): return self._x

a = A()
print(a.x)  # Output: 0
```

---

18. staticmethod() – class ke andar aisa method banata hai jise object ke bina call kar sakte ho.

ex.,

```
class A:
   @staticmethod
   def greet(): print("Hello")
```

```
A.greet()  # Output: Hello
```

---

19. classmethod() – method jo class ko as first argument leta hai.

ex.,

```
class A:
    @classmethod
    def show(cls): print(cls)

A.show()
```

---

20. super() – parent class ke method ko call karta hai child class ke andar.

ex.,

```
class A:
    def show(self): print("Parent")

class B(A):
    def show(self):
        super().show()
        print("Child")

B().show()
```

---

21. object() – Python ka base class jisse sab kuch inherit karta hai.

ex.,

```
o = object()
print(type(o))  # Output: <class 'object'>
```

---

22. format() – strings ko dynamic tarike se format karta hai.

ex.,

```
name = "Rushil"
print("Hello, {}".format(name))  # Output: Hello, Rushil
```

---

23. breakpoint() – debugging ke liye execution stop karta hai (Python 3.7+).

ex.,

```
x = 10
breakpoint()
print(x)
```