Python Builtin Functions

Most Commonly Used Built-in Functions

---

1. `print()`

Prints output to the console.

```
print("Hello, World!")  # Output: Hello, World!
```

---

2. `len()`

Returns the length of an object.

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

---

3. `type()`

Returns the type of an object.

```
print(type(5))  # Output: <class 'int'>
```

---

4. `input()`

Reads a string input from the user.

```
name = input("Enter your name: ")
print(f"Hello, {name}!")
```

---

5. `int()`

Converts a value to an integer.

```python
print(int("42"))  # Output: 42
```

---

6. `float()`

Converts a value to a float.

```python
print(float("3.14"))  # Output: 3.14
```

---

7. `str()`

Converts a value to a string.

```python
print(str(42))  # Output: '42'
```

---

8. `list()`

Creates a list from an iterable.

```python
print(list("hello"))  # Output: ['h', 'e', 'l', 'l', 'o']
```

---

9. `dict()`

Creates a dictionary.

```python
d = dict(name="John", age=30)
print(d)  # Output: {'name': 'John', 'age': 30}
```

---

10. `set()`

Creates a set from an iterable.

```python
s = set([1, 2, 2, 3])
print(s)  # Output: {1, 2, 3}
```

---

11. `tuple()`

Creates a tuple from an iterable.

```
t = tuple([1, 2, 3])
print(t)  # Output: (1, 2, 3)
```

---

12. `range()`

Generates a sequence of numbers.

```
for i in range(5):
    print(i)  # Output: 0, 1, 2, 3, 4
```

---

13. `enumerate()`

Returns an enumerate object with index-value pairs.

```
for i, v in enumerate(["a", "b", "c"]):
    print(i, v)  # Output: 0 a, 1 b, 2 c
```

---

14. `zip()`

Combines multiple iterables into tuples.

```
names = ["Alice", "Bob"]
ages = [25, 30]
print(list(zip(names, ages)))  # Output: [('Alice', 25), ('Bob', 30)]
```

---

15. `map()`

Applies a function to all elements in an iterable.

```
nums = [1, 2, 3]
squares = map(lambda x: x ** 2, nums)
print(list(squares))  # Output: [1, 4, 9]
```

---

16. `filter()`

Filters elements in an iterable based on a condition.

```
nums = [1, 2, 3, 4]
even = filter(lambda x: x % 2 == 0, nums)
print(list(even))  # Output: [2, 4]
```

---

17. `sorted()`

Returns a sorted list from an iterable.

```
nums = [3, 1, 2]
print(sorted(nums))  # Output: [1, 2, 3]
```

---

18. `max()` and `min()`

Returns the largest or smallest item in an iterable.

```
nums = [1, 2, 3]
print(max(nums))  # Output: 3
print(min(nums))  # Output: 1
```

---

19. `sum()`

Returns the sum of an iterable.

```
nums = [1, 2, 3]
print(sum(nums))  # Output: 6
```

---

20. `open()`

Opens a file.

```python
with open("example.txt", "w") as f:
    f.write("Hello, file!")
```

---

21. `isinstance()`

Checks if an object is an instance of a class.

```python
print(isinstance(5, int))  # Output: True
```

---

22. `help()`

Displays the help text for an object.

```python
help(str)  # Outputs help for the `str` class.
```

---

23. `id()`

Returns the memory address of an object.

```python
x = 42
print(id(x))
```

---

24. `round()`

Rounds a number to the specified number of digits.

```python
print(round(3.14159, 2))  # Output: 3.14
```

---

25. `abs()`

Returns the absolute value of a number.

```
print(abs(-5))  # Output: 5
```

Advanced and Less Commonly Used Built-in Functions

---

26. `eval()`

Evaluates a string as Python code.

```
x = 1
print(eval("x + 2"))  # Output: 3
```

---

27. `exec()`

Executes Python code from a string.

```
exec('x = 10\nprint(x)')  # Output: 10
```

---

28. `reversed()`

Returns a reversed iterator.

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

---

29. `delattr()`

Deletes an attribute from an object.

```
class Example:
    x = 10
```

```python
delattr(Example, 'x')
```

---

30. `property()`

Creates a property attribute.

```python
class Example:
    def __init__(self, value):
        self._value = value

    @property
    def value(self):
        return self._value

obj = Example(42)
print(obj.value)  # Output: 42
```

---

31. `globals()` and `locals()`

Returns global and local symbol tables.

```python
print(globals())  # Outputs global variables.
```

---

32. `next()`

Retrieves the next item from an iterator.

```python
it = iter([1, 2, 3])
print(next(it))  # Output: 1
```
