**Python Inbuilt Functions with Examples**

Python provides a rich set of **inbuilt functions** that can simplify a lot of tasks. These functions are part of Python's standard library, and you don't need to import anything to use them.

Here’s a list of some commonly used inbuilt functions in Python with examples:

---

**1. print()**

- **Purpose**: Prints the given object to the console.
- **Syntax**: print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)

**Example:**

python

Copy code

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

---

**2. len()**

- **Purpose**: Returns the length (the number of items) of an object like a list, string, or tuple.
- **Syntax**: len(object)

**Example:**

python

Copy code

```
text = "Python"

print(len(text))  # Output: 6
```

---

**3. type()**

- **Purpose**: Returns the type of an object.
- **Syntax**: type(object)

**Example:**

python

Copy code

```
print(type(42))  # Output: <class 'int'>
print(type("Hello"))  # Output: <class 'str'>
```

---

**4. sum()**

- **Purpose**: Returns the sum of a sequence (such as a list or tuple) of numbers.
- **Syntax**: sum(iterable, start=0)

**Example:**

python

Copy code

```
numbers = [1, 2, 3, 4]
print(sum(numbers))  # Output: 10
```

---

**5. sorted()**

- **Purpose**: Returns a sorted list of the specified iterable's items.
- **Syntax**: sorted(iterable, key=None, reverse=False)

**Example:**

python

Copy code

```
numbers = [3, 1, 4, 1, 5, 9, 2]
print(sorted(numbers))  # Output: [1, 1, 2, 3, 4, 5, 9]
```

---

**6. min() and max()**

- **Purpose**: Return the smallest and largest item in an iterable or among two or more arguments.
- **Syntax**:
    - min(iterable, *args, key=None, default=None)
    - max(iterable, *args, key=None, default=None)

**Example:**

python

Copy code

```
numbers = [3, 5, 2, 8]
print(min(numbers))  # Output: 2
print(max(numbers))  # Output: 8
```

---

**7. abs()**

- **Purpose**: Returns the absolute value of a number.
- **Syntax**: abs(x)

**Example:**

python

Copy code

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

---

**8. range()**

- **Purpose**: Returns a sequence of numbers, which is commonly used for looping a specific number of times.
- **Syntax**: range(start, stop, step)

**Example:**

python

Copy code

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

# 3

# 4

---

**9. input()**

- **Purpose**: Reads a line of text from user input and returns it as a string.
- **Syntax**: input(prompt)

**Example:**

python

Copy code

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

---

**10. int(), float(), str()**

- **Purpose**: Convert a value to an integer, float, or string, respectively.
- **Syntax**:
    - int(x)
    - float(x)
    - str(x)

**Examples:**

python

Copy code

```
print(int("42"))  # Output: 42
print(float("3.14"))  # Output: 3.14
print(str(100))  # Output: '100'
```

---

**11. isinstance()**

- **Purpose**: Checks if an object is an instance of a specified class or a subclass thereof.
- **Syntax**: isinstance(object, classinfo)

**Example:**

python

Copy code

```
x = 5

print(isinstance(x, int))  # Output: True
```

---

**12. all()**

- **Purpose**: Returns True if all elements in an iterable are true (or if the iterable is empty).
- **Syntax**: all(iterable)

**Example:**

python

Copy code

```
numbers = [True, True, False]

print(all(numbers))  # Output: False
```

---

**13. any()**

- **Purpose**: Returns True if any element in the iterable is true.
- **Syntax**: any(iterable)

**Example:**

python

Copy code

```
numbers = [False, False, True]

print(any(numbers))  # Output: True
```

---

**14. zip()**

- **Purpose**: Combines multiple iterables (like lists or tuples) element-wise into pairs.
- **Syntax**: zip(*iterables)

**Example:**

python

Copy code

```
names = ["Alice", "Bob", "Charlie"]

ages = [25, 30, 35]

result = zip(names, ages)

print(list(result))  # Output: [('Alice', 25), ('Bob', 30), ('Charlie', 35)]
```

---

**15. map()**

- **Purpose**: Applies a function to all items in an iterable (such as a list or tuple).
- **Syntax**: map(function, iterable)

**Example:**

python

Copy code

```
numbers = [1, 2, 3, 4]

squared_numbers = map(lambda x: x**2, numbers)

print(list(squared_numbers))  # Output: [1, 4, 9, 16]
```

---

**16. filter()**

- **Purpose**: Filters elements from an iterable using a function that returns True or False.
- **Syntax**: filter(function, iterable)

**Example:**

python

Copy code

```
numbers = [1, 2, 3, 4, 5]

even_numbers = filter(lambda x: x % 2 == 0, numbers)

print(list(even_numbers))  # Output: [2, 4]
```

---

**Conclusion**

These inbuilt functions provide a wide variety of utility, allowing developers to easily handle basic operations, data manipulation, and more complex tasks without needing to implement such functions manually. This results in cleaner, more efficient, and more readable code.