You said:

GET INBULIT FUNCTIONS AND BASICS OF PYTHON THAT WITH EXAMPLES THAT ARE NEEDED TO WRITE AD CODE FOR CODING EXAM

ChatGPT said:

Here are some **important built-in functions and Python basics** that are commonly used in coding exams, along with examples:

🔹 Basic Data Types

python
CopyEdit

```
x = 10       # Integer
y = 3.14     # Float
name = "Python"  # String
is_valid = True  # Boolean
```

🔹 Input and Output

python
CopyEdit

```
name = input("Enter your name: ")  # Taking input (Always string)
print("Hello,", name)  # Output
```

🔹 Type Conversion

python
CopyEdit

```
num = int(input("Enter a number: "))  # Convert input to integer
flt = float("3.14")  # Convert string to float
txt = str(100)  # Convert number to string
```

🔹 Math Operations

python
CopyEdit

```
a, b = 10, 3
print(a + b)  # Addition
print(a - b)  # Subtraction
print(a * b)  # Multiplication
print(a / b)  # Division (float)
print(a // b) # Floor division
print(a % b)  # Modulus (remainder)
print(a ** b) # Exponentiation (power)
```

🔹 Control Flow (if-else)

python
CopyEdit

```
num = int(input("Enter a number: "))
if num > 0:
    print("Positive")
elif num < 0:
    print("Negative")
else:
    print("Zero")
```

🔹 Loops (for, while)

python
CopyEdit

```
# For Loop
for i in range(1, 6):
    print(i)

# While Loop
n = 5
while n > 0:
    print(n)
    n -= 1  # Decrement
```

🔹 Functions

python
CopyEdit

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

print(add(5, 3))  # Output: 8
```

🔹 List Operations

python
CopyEdit

```
nums = [1, 2, 3, 4, 5]
print(nums[0])   # Access first element
nums.append(6)   # Add element
nums.remove(3)   # Remove element
print(len(nums)) # Length of list
```

🔹 Tuple (Immutable List)

python
CopyEdit

```
tup = (10, 20, 30)
print(tup[1])  # Access second element
```

🔹 Dictionary (Key-Value Pair)

python
CopyEdit

```
person = {"name": "Alice", "age": 25}
print(person["name"])  # Access value
person["city"] = "New York"  # Add new key-value pair
```

🔹 String Operations

python
CopyEdit

```
text = "hello world"
print(text.upper())  # HELLO WORLD
print(text.replace("hello", "hi"))  # hi world
print(text.split())  # ['hello', 'world']
```

🔹 Set (Unique Elements)

python
CopyEdit

```
s = {1, 2, 3, 2, 1}
print(s)  # Output: {1, 2, 3}
s.add(4)  # Add element
```

🔹 Built-in Functions

Function	Description
`len()`	Returns length of a sequence
`max()`	Returns the maximum value
`min()`	Returns the minimum value
`sum()`	Returns sum of elements
`sorted()`	Returns sorted sequence
`abs()`	Returns absolute value
`round()`	Rounds a number

python

CopyEdit

```
nums = [3, 1, 4, 1, 5]
print(len(nums))  # Output: 5
print(max(nums))  # Output: 5
print(min(nums))  # Output: 1
print(sum(nums))  # Output: 14
print(sorted(nums))  # Output: [1, 1, 3, 4, 5]
```

🔹 Lambda Function (Short Anonymous Function)

python
CopyEdit

```
square = lambda x: x ** 2
print(square(4))  # Output: 16
```

🔹 List Comprehension

python
CopyEdit

```
nums = [x for x in range(10) if x % 2 == 0]
print(nums)  # Output: [0, 2, 4, 6, 8]
```

These **basics and built-in functions** will help you write efficient code in your exam. 🚀 Let me know if you need more!

You said:

MORE

ChatGPT said:

Here are **more essential Python concepts and built-in functions** to help you ace your coding exam! 🚀

🔹 More on Loops

Looping Through a List

python
CopyEdit

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

Looping with Index (`enumerate`)

python
CopyEdit

```
fruits = ["apple", "banana", "cherry"]
for index, fruit in enumerate(fruits):
    print(index, fruit)
```

Looping Through a Dictionary

python
CopyEdit

```
person = {"name": "Alice", "age": 25, "city": "New York"}
for key, value in person.items():
    print(key, ":", value)
```

Looping with `range()`

python
CopyEdit

```
for i in range(5):  # 0 to 4
    print(i)

for i in range(2, 10, 2):  # Start=2, Stop=10, Step=2
    print(i)
```

Using `break` and `continue`

python
CopyEdit

```
for i in range(5):
    if i == 3:
```

```
        break  # Stops loop at 3
    print(i)

for i in range(5):
    if i == 3:
        continue  # Skips 3
    print(i)
```

🔹 More on Functions

Default Arguments

python
CopyEdit

```
def greet(name="Guest"):
    print("Hello,", name)

greet()          # Output: Hello, Guest
greet("Alice")   # Output: Hello, Alice
```

Variable-Length Arguments (`*args`, `**kwargs`)

python
CopyEdit

```
# *args (Multiple Positional Arguments)
def add(*numbers):
    return sum(numbers)

print(add(1, 2, 3, 4))  # Output: 10

# **kwargs (Multiple Keyword Arguments)
def details(**info):
    for key, value in info.items():
        print(key, ":", value)

details(name="Alice", age=25, city="New York")
```

🔹 More on Strings

python
CopyEdit

```
s = "  hello world  "
print(s.strip())     # Removes spaces -> "hello world"
print(s.startswith("h"))  # True
print(s.endswith("d"))    # True
print(s.count("l"))  # Count occurrences of 'l'
print(s.find("world"))  # Find index of "world"
```

String Formatting (`f-strings`)

python
CopyEdit

```
name = "Alice"
age = 25
print(f"My name is {name} and I am {age} years old.")
```

🔹 More on Lists

Slicing a List

python
CopyEdit

```
nums = [10, 20, 30, 40, 50]
print(nums[1:4])  # Output: [20, 30, 40]
print(nums[:3])   # Output: [10, 20, 30]
print(nums[-2:])  # Output: [40, 50]
```

List Methods

python
CopyEdit

```
nums = [1, 2, 3]
nums.append(4)   # Add element at end
nums.insert(1, 10)  # Insert at index 1
nums.pop()  # Remove last element
nums.remove(2)  # Remove first occurrence of 2
```

```
nums.reverse()  # Reverse list
nums.sort()  # Sort list
print(nums)
```

🔹 More on Tuples

python
CopyEdit

```
tup = (10, 20, 30, 40)
print(tup.count(20))  # Count occurrences of 20
print(tup.index(30))  # Find index of 30
```

🔹 More on Sets

python
CopyEdit

```
A = {1, 2, 3, 4}
B = {3, 4, 5, 6}

print(A | B)  # Union: {1, 2, 3, 4, 5, 6}
print(A & B)  # Intersection: {3, 4}
print(A - B)  # Difference: {1, 2}
print(A ^ B)  # Symmetric Difference: {1, 2, 5, 6}
```

🔹 More on Dictionaries

python
CopyEdit

```
d = {"name": "Alice", "age": 25}
d.update({"city": "New York"})  # Add new key-value pair
print(d.keys())  # Output: dict_keys(['name', 'age', 'city'])
print(d.values())  # Output: dict_values(['Alice', 25, 'New York'])
print(d.items())  # Output: dict_items([('name', 'Alice'), ('age',
25), ('city', 'New York')])
```

🔹 More Built-in Functions

Function	Description
`zip()`	Combines multiple iterables element-wise
`map()`	Applies a function to each item in an iterable
`filter()`	Filters elements based on a condition
`reduce()`	Reduces an iterable to a single value

Using `zip()`

python
CopyEdit

```
names = ["Alice", "Bob", "Charlie"]
scores = [85, 90, 88]

zipped = zip(names, scores)
print(list(zipped))  # Output: [('Alice', 85), ('Bob', 90),
('Charlie', 88)]
```

Using `map()`

python
CopyEdit

```
nums = [1, 2, 3, 4]
squared = list(map(lambda x: x ** 2, nums))
print(squared)  # Output: [1, 4, 9, 16]
```

Using `filter()`

python
CopyEdit

```
nums = [1, 2, 3, 4, 5, 6]
evens = list(filter(lambda x: x % 2 == 0, nums))
```

```
print(evens)  # Output: [2, 4, 6]
```

Using `reduce()` (from `functools`)

python
CopyEdit

```
from functools import reduce

nums = [1, 2, 3, 4]
product = reduce(lambda x, y: x * y, nums)
print(product)  # Output: 24
```

🔹 Exception Handling (Try-Except)

python
CopyEdit

```
try:
    num = int(input("Enter a number: "))
    print(10 / num)
except ZeroDivisionError:
    print("Cannot divide by zero!")
except ValueError:
    print("Invalid input! Please enter a number.")
finally:
    print("Execution finished.")
```

🔹 Working with Files

python
CopyEdit

```
# Writing to a file
with open("test.txt", "w") as file:
    file.write("Hello, world!")

# Reading from a file
with open("test.txt", "r") as file:
```

```
    content = file.read()
    print(content)
```

🔹 Random Module

python
CopyEdit

```
import random

print(random.randint(1, 10))  # Random integer between 1 and 10
print(random.choice(["apple", "banana", "cherry"]))  # Random choice
from list
print(random.shuffle([1, 2, 3, 4, 5]))  # Shuffle a list
```

🔹 Time and Date

python
CopyEdit

```
import datetime

now = datetime.datetime.now()
print(now.strftime("%Y-%m-%d %H:%M:%S"))  # Format date-time
```

These **advanced concepts and built-in functions** will be very useful in your **coding exam**! 💯
Let me know if you need any **practice problems** or **more explanations**! 🚀🔥

4o

You said:

MORE

ChatGPT said:

Here are **more advanced Python concepts and built-in functions** to help you excel in coding exams! 🚀

🔹 More on Loops

Using `else` with Loops

python
CopyEdit

```
for i in range(5):
    print(i)
else:
    print("Loop completed without break")  # Executes only if loop
finishes without `break`

for i in range(5):
    if i == 3:
        break
    print(i)
else:
    print("This won't print because loop was broken!")
```

🔹 Advanced List Techniques

List Comprehensions with Conditionals

python
CopyEdit

```
nums = [1, 2, 3, 4, 5, 6]
even_squares = [x**2 for x in nums if x % 2 == 0]
print(even_squares)  # Output: [4, 16, 36]
```

Flattening a Nested List

python
CopyEdit

```
nested = [[1, 2], [3, 4], [5, 6]]
flat = [num for sublist in nested for num in sublist]
```

```
print(flat)  # Output: [1, 2, 3, 4, 5, 6]
```

🔹 Advanced Dictionary Techniques

Dictionary Comprehension

python
CopyEdit

```
squares = {x: x**2 for x in range(1, 6)}
print(squares)  # Output: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Merging Two Dictionaries

python
CopyEdit

```
dict1 = {'a': 1, 'b': 2}
dict2 = {'b': 3, 'c': 4}
merged = {**dict1, **dict2}  # `b` gets overwritten
print(merged)  # Output: {'a': 1, 'b': 3, 'c': 4}
```

🔹 Advanced String Techniques

Reversing a String

python
CopyEdit

```
s = "Python"
print(s[::-1])  # Output: "nohtyP"
```

Checking for Palindromes

python
CopyEdit

```python
def is_palindrome(s):
    return s == s[::-1]

print(is_palindrome("madam"))  # Output: True
print(is_palindrome("hello"))  # Output: False
```

🔹 Using `zip()` in Advanced Ways

Unzipping a List of Tuples

python
CopyEdit

```python
pairs = [(1, 'a'), (2, 'b'), (3, 'c')]
numbers, letters = zip(*pairs)
print(numbers)  # Output: (1, 2, 3)
print(letters)  # Output: ('a', 'b', 'c')
```

🔹 Using `map()`, `filter()`, and `reduce()`

`map()` for Element-Wise Operations

python
CopyEdit

```python
nums = [1, 2, 3, 4]
squared = list(map(lambda x: x**2, nums))
print(squared)  # Output: [1, 4, 9, 16]
```

`filter()` for Conditional Filtering

python
CopyEdit

```python
nums = [1, 2, 3, 4, 5, 6]
```

```
evens = list(filter(lambda x: x % 2 == 0, nums))
print(evens)  # Output: [2, 4, 6]
```

`reduce()` for Accumulative Computation

python
CopyEdit

```
from functools import reduce
nums = [1, 2, 3, 4]
product = reduce(lambda x, y: x * y, nums)
print(product)  # Output: 24
```

🔹 Sorting in Python

Sorting Lists

python
CopyEdit

```
nums = [5, 2, 9, 1, 5, 6]
nums.sort()  # Sorts in-place
print(nums)  # Output: [1, 2, 5, 5, 6, 9]

sorted_nums = sorted(nums, reverse=True)
print(sorted_nums)  # Output: [9, 6, 5, 5, 2, 1]
```

Sorting a List of Tuples

python
CopyEdit

```
students = [("Alice", 25), ("Bob", 22), ("Charlie", 23)]
students.sort(key=lambda x: x[1])  # Sort by age
print(students)  # Output: [('Bob', 22), ('Charlie', 23), ('Alice',
25)]
```

🔹 Custom Exception Handling

python
CopyEdit

```
class NegativeNumberError(Exception):
    pass

def check_positive(n):
    if n < 0:
        raise NegativeNumberError("Negative number not allowed!")
    return n

try:
    print(check_positive(-5))
except NegativeNumberError as e:
    print(e)  # Output: Negative number not allowed!
```

🔹 Using Generators for Memory Efficiency

Generator Function (`yield`)

python
CopyEdit

```
def count_up_to(n):
    i = 1
    while i <= n:
        yield i
        i += 1

for num in count_up_to(5):
    print(num)
```

Output:

CopyEdit

```
1
2
3
4
5
```

Generator Expression

python
CopyEdit

```
gen = (x**2 for x in range(5))
print(next(gen))  # Output: 0
print(next(gen))  # Output: 1
```

🔹 Multithreading & Multiprocessing

Using `threading`

python
CopyEdit

```
import threading

def print_numbers():
    for i in range(5):
        print(i)

t1 = threading.Thread(target=print_numbers)
t1.start()
t1.join()  # Wait for thread to finish
```

Using `multiprocessing`

python
CopyEdit

```
import multiprocessing

def print_numbers():
    for i in range(5):
        print(i)

p1 = multiprocessing.Process(target=print_numbers)
p1.start()
p1.join()
```

🔹 Working with JSON

python
CopyEdit

```
import json

data = {"name": "Alice", "age": 25, "city": "New York"}

# Convert dictionary to JSON string
json_data = json.dumps(data)
print(json_data)  # Output: {"name": "Alice", "age": 25, "city": "New
York"}

# Convert JSON string back to dictionary
parsed_data = json.loads(json_data)
print(parsed_data["name"])  # Output: Alice
```

🔹 Datetime and Timedelta

python
CopyEdit

```
from datetime import datetime, timedelta

now = datetime.now()
```

```
print(now.strftime("%Y-%m-%d %H:%M:%S"))  # Format datetime

future_date = now + timedelta(days=5)
print(future_date.strftime("%Y-%m-%d"))  # Output: Date 5 days ahead
```

🔹 Regular Expressions (`re` module)

python
CopyEdit

```
import re

text = "My phone number is 9876543210"

# Check if text contains a 10-digit number
match = re.search(r"\b\d{10}\b", text)
if match:
    print("Phone number found:", match.group())
```

🔹 File Handling (Appending & Reading)

python
CopyEdit

```
# Writing to a file
with open("test.txt", "a") as file:
    file.write("Appending new text!\n")

# Reading from a file
with open("test.txt", "r") as file:
    content = file.readlines()
    print(content)
```

These **advanced Python topics** will help you **crack your coding exams!** 💯🔥

Do you need **practice problems or real-world coding challenges** to prepare? 😃

4o

You said:
MORE

ChatGPT said:

You're really diving deep into Python! 🚀 Here are **even more advanced Python concepts, techniques, and built-in functions** that will help you in your coding exams and beyond.

🔹 More Advanced Data Structures

Deque (Double-Ended Queue) - `collections.deque`

Faster than lists for appending and popping elements from both ends.

python
CopyEdit

```
from collections import deque

dq = deque([1, 2, 3])
dq.append(4)  # Add to right
dq.appendleft(0)  # Add to left
print(dq)  # Output: deque([0, 1, 2, 3, 4])

dq.pop()  # Remove from right
dq.popleft()  # Remove from left
print(dq)  # Output: deque([1, 2, 3])
```

Default Dictionary - `collections.defaultdict`

Handles missing keys gracefully.

python
CopyEdit

```python
from collections import defaultdict

d = defaultdict(int)  # Default value is 0
d["a"] += 1
print(d["a"])  # Output: 1
print(d["b"])  # Output: 0 (no KeyError)
```

Counter - Counting Elements Efficiently

python
CopyEdit

```python
from collections import Counter

nums = [1, 2, 2, 3, 3, 3, 4]
count = Counter(nums)
print(count)  # Output: Counter({3: 3, 2: 2, 1: 1, 4: 1})

# Get most common elements
print(count.most_common(2))  # Output: [(3, 3), (2, 2)]
```

🔹 Heap (Priority Queue) - heapq

Finding K Smallest or K Largest Elements

python
CopyEdit

```python
import heapq

nums = [3, 1, 4, 1, 5, 9, 2]
print(heapq.nlargest(3, nums))  # Output: [9, 5, 4]
print(heapq.nsmallest(3, nums))  # Output: [1, 1, 2]
```

Implementing a Min Heap

python
CopyEdit

```
heap = []
heapq.heappush(heap, 3)
heapq.heappush(heap, 1)
heapq.heappush(heap, 4)

print(heapq.heappop(heap))  # Output: 1 (Smallest element)
```

🔹 More Advanced Sorting

Sorting by Custom Function

python
CopyEdit

```
students = [("Alice", 90), ("Bob", 80), ("Charlie", 85)]
students.sort(key=lambda x: x[1], reverse=True)
print(students)  # Output: [('Alice', 90), ('Charlie', 85), ('Bob',
80)]
```

🔹 Binary Search - `bisect`

Efficient way to insert elements in sorted order.

python
CopyEdit

```
import bisect

arr = [10, 20, 30, 40]
pos = bisect.bisect_left(arr, 25)  # Find position to insert 25
print(pos)  # Output: 2

bisect.insort(arr, 25)  # Insert 25 in sorted order
print(arr)  # Output: [10, 20, 25, 30, 40]
```

🔹 Matrix Manipulation (2D Lists)

python
CopyEdit

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

# Transpose a matrix
transposed = [[row[i] for row in matrix] for i in
range(len(matrix[0]))]
print(transposed)
# Output: [[1, 4, 7], [2, 5, 8], [3, 6, 9]]
```

🔹 Graph Algorithms

Graph Representation using Adjacency List

python
CopyEdit

```
graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B', 'F'],
    'F': ['C', 'E']
}
```

Depth-First Search (DFS)

python
CopyEdit

```
def dfs(graph, node, visited=None):
    if visited is None:
```

```
        visited = set()
    if node not in visited:
        print(node)
        visited.add(node)
        for neighbor in graph[node]:
            dfs(graph, neighbor, visited)

dfs(graph, 'A')
```

🔹 Multithreading & Multiprocessing

Multithreading (Running Multiple Tasks in Parallel)

python
CopyEdit

```
import threading

def print_numbers():
    for i in range(5):
        print(i)

t1 = threading.Thread(target=print_numbers)
t1.start()
t1.join()
```

Multiprocessing (Using Multiple CPU Cores)

python
CopyEdit

```
import multiprocessing

def print_numbers():
    for i in range(5):
        print(i)

p1 = multiprocessing.Process(target=print_numbers)
```

```
p1.start()
p1.join()
```

🔹 Working with Databases (`sqlite3`)

python
CopyEdit

```
import sqlite3

# Connect to database (or create it)
conn = sqlite3.connect("test.db")
cursor = conn.cursor()

# Create table
cursor.execute("CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY
KEY, name TEXT)")
conn.commit()

# Insert data
cursor.execute("INSERT INTO users (name) VALUES ('Alice')")
conn.commit()

# Retrieve data
cursor.execute("SELECT * FROM users")
print(cursor.fetchall())  # Output: [(1, 'Alice')]

conn.close()
```

🔹 Decorators (Functions that Modify Other Functions)

python
CopyEdit

```
def decorator(func):
    def wrapper():
        print("Before function execution")
        func()
        print("After function execution")
    return wrapper

@decorator
def say_hello():
    print("Hello!")

say_hello()
```

Output:

pgsql
CopyEdit

```
Before function execution
Hello!
After function execution
```

🔹 Working with APIs (Requests Module)

python
CopyEdit

```
import requests

response =
requests.get("https://jsonplaceholder.typicode.com/todos/1")
data = response.json()
print(data)  # Output: {'userId': 1, 'id': 1, 'title': 'delectus aut
autem', 'completed': False}
```

🔹 Path Manipulation (`os` and `pathlib`)

python
CopyEdit

```
import os

# Get current directory
print(os.getcwd())

# List files in a directory
print(os.listdir("."))

# Check if a file exists
print(os.path.exists("test.txt"))
```

🔹 Thread Pool (Efficient Parallel Processing)

python
CopyEdit

```
from concurrent.futures import ThreadPoolExecutor

def square(n):
    return n * n

with ThreadPoolExecutor(max_workers=4) as executor:
    results = executor.map(square, [1, 2, 3, 4, 5])

print(list(results))  # Output: [1, 4, 9, 16, 25]
```

🎯 Summary

🔥 You now have **a complete roadmap** of Python techniques needed for coding exams and real-world projects! 💯

- **Data Structures**: Lists, Tuples, Sets, Dicts, Deque
- **Algorithms**: Sorting, Searching, Graphs, DFS, BFS
- **Advanced Techniques**: Multithreading, Databases, APIs, Decorators
- **Optimization**: Heap, Priority Queue, Generators

📢 **What's Next?**

- Want **practice questions** on these topics?
- Need **interview-style problems** to test your skills?

Just let me know, and I'll prepare them for you! 🚀🔥

You said:
I DONT WANT THIS ADVACED JUS WHAT NEEDED FOR CODING A PROGRAM FOR CODING EXAMINATION FOR INTERVIEW

ChatGPT said:

Got it! Here's what you need:

🔹 Essential Python Concepts for Coding Exams & Interviews

✅ 1. Input & Output

python
CopyEdit

```
name = input("Enter your name: ")  # Taking input
print("Hello,", name)  # Output
```

✅ 2. Data Types & Type Conversion

python
CopyEdit

```
num = int("10")  # Convert string to integer
flt = float("3.14")  # Convert string to float
txt = str(100)  # Convert number to string
```

✅ 3. If-Else Conditions

python

CopyEdit

```
num = int(input("Enter a number: "))
if num > 0:
    print("Positive")
elif num < 0:
    print("Negative")
else:
    print("Zero")
```

✅ 4. Loops (for, while)

python
CopyEdit

```
# For Loop
for i in range(1, 6):
    print(i)

# While Loop
n = 5
while n > 0:
    print(n)
    n -= 1  # Decrement
```

✅ 5. Functions

python
CopyEdit

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

print(add(5, 3))  # Output: 8
```

✅ 6. Lists (Dynamic Arrays in Python)

python
CopyEdit

```
nums = [1, 2, 3, 4, 5]
nums.append(6)  # Add element
nums.remove(3)  # Remove element
```

```
print(nums[0])   # Access first element
print(len(nums)) # Length of list
```

✅ 7. Tuples (Immutable List)

python
CopyEdit

```
tup = (10, 20, 30)
print(tup[1])  # Access second element
```

✅ 8. Dictionary (Key-Value Pair)

python
CopyEdit

```
person = {"name": "Alice", "age": 25}
print(person["name"])  # Output: Alice
person["city"] = "New York"  # Add new key-value pair
```

✅ 9. String Operations

python
CopyEdit

```
text = "hello world"
print(text.upper())  # HELLO WORLD
print(text.replace("hello", "hi"))  # hi world
print(text.split())  # ['hello', 'world']
```

✅ 10. Sorting a List

python
CopyEdit

```
nums = [5, 2, 9, 1]
nums.sort()
print(nums)  # Output: [1, 2, 5, 9]
```

✅ 11. Searching in a List

python
CopyEdit

```
nums = [10, 20, 30, 40]
```

```
print(30 in nums)  # Output: True
```

✅ 12. List Comprehension

python
CopyEdit

```
nums = [x for x in range(10) if x % 2 == 0]
print(nums)  # Output: [0, 2, 4, 6, 8]
```

✅ 13. Exception Handling

python
CopyEdit

```
try:
    num = int(input("Enter a number: "))
    print(10 / num)
except ZeroDivisionError:
    print("Cannot divide by zero!")
except ValueError:
    print("Invalid input! Please enter a number.")
```

✅ 14. File Handling (Reading & Writing)

python
CopyEdit

```
# Writing to a file
with open("test.txt", "w") as file:
    file.write("Hello, world!")

# Reading from a file
with open("test.txt", "r") as file:
    print(file.read())
```

✅ 15. Basic Recursion (Factorial Example)

python
CopyEdit

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
```

```
    return n * factorial(n - 1)

print(factorial(5))  # Output: 120
```

💡 **This is all you need for coding exams & interviews!** Do you want **practice problems** based on these concepts? 🚀

4o