DA	Data Science and Artificial Intelligence

Probability and Statistics: Counting (permutation and combinations), probability axioms, Sample space, events, independent events, mutually exclusive events, marginal, conditional and joint probability, Bayes Theorem, conditional expectation and variance, mean, median, mode and standard deviation, correlation, and covariance, random variables, discrete random variables and probability mass functions, uniform, Bernoulli, binomial distribution, Continuous random variables and probability distribution function, uniform, exponential, Poisson, normal, standard normal, t-distribution, chi-squared distributions, cumulative distribution function, Conditional PDF, Central limit theorem, confidence interval, z-test, t-test, chi-squared test.

Linear Algebra: Vector space, subspaces, linear dependence and independence of vectors, matrices, projection matrix, orthogonal matrix, idempotent matrix, partition matrix and their properties, quadratic forms, systems of linear equations and solutions; Gaussian elimination, eigenvalues and eigenvectors, determinant, rank, nullity, projections, LU decomposition, singular value decomposition.

Calculus and Optimization: Functions of a single variable, limit, continuity and differentiability, Taylor series, maxima and minima, optimization involving a single variable.

Programming, Data Structures and Algorithms: Programming in Python, basic data structures: stacks, queues, linked lists, trees, hash tables; Search algorithms: linear search and binary search, basic sorting algorithms: selection sort, bubble sort and insertion sort; divide and conquer: mergesort, quicksort; introduction to graph theory; basic graph algorithms: traversals and shortest path.

Database Management and Warehousing: ER-model, relational model: relational algebra, tuple calculus, SQL, integrity constraints, normal form, file organization, indexing, data types, data transformation such as normalization, discretization, sampling, compression; data warehouse modelling: schema for multidimensional data models, concept hierarchies, measures: categorization and computations.

Machine Learning: (i) Supervised Learning: regression and classification problems, simple linear regression, multiple linear regression, ridge regression, logistic regression, k-nearest neighbour, naive Bayes classifier, linear discriminant analysis, support vector machine, decision trees, bias-variance trade-off, cross-validation methods such as leave-one-out (LOO) cross-validation, k-folds cross-validation, multi-layer perceptron, feed-forward neural network; (ii) Unsupervised Learning: clustering algorithms, k-means/k-medoid, hierarchical clustering, top-down, bottom-up: single-linkage, multiple-linkage, dimensionality reduction, principal component analysis.

AI: Search: informed, uninformed, adversarial; logic, propositional, predicate; reasoning under uncertainty topics — conditional independence representation, exact inference through variable elimination, and approximate inference through sampling.

CS Computer Science and Information Technology

Section 1: Engineering Mathematics

Discrete Mathematics: Propositional and first order logic. Sets, relations, functions, partial orders and lattices. Monoids, Groups. Graphs: connectivity, matching, colouring. Combinatorics: counting, recurrence relations, generating functions.

Linear Algebra: Matrices, determinants, system of linear equations, eigenvalues and eigenvectors, LU decomposition.

Calculus: Limits, continuity and differentiability, Maxima and minima, Mean value theorem, Integration.

Probability and Statistics: Random variables, Uniform, normal, exponential, Poisson and binomial distributions. Mean, median, mode and standard deviation. Conditional probability and Bayes theorem.

Section 2: Digital Logic

Boolean algebra. Combinational and sequential circuits. Minimization. Number representations and computer arithmetic (fixed and floating point).

Section 3: Computer Organization and Architecture

Machine instructions and addressing modes. ALU, data-path and control unit. Instruction pipelining, pipeline hazards. Memory hierarchy: cache, main memory and secondary storage; I/O interface (interrupt and DMA mode).

Section 4: Programming and Data Structures

Programming in C. Recursion. Arrays, stacks, queues, linked lists, trees, binary search trees, binary heaps, graphs.

Section 5: Algorithms

Searching, sorting, hashing. Asymptotic worst case time and space complexity. Algorithm design techniques: greedy, dynamic programming and divide-and-conquer. Graph traversals, minimum spanning trees, shortest paths.

Section 6: Theory of Computation

Regular expressions and finite automata. Context-free grammars and push-down automata. Regular and context-free languages, pumping lemma. Turing machines and undecidability.

Section 7: Compiler Design

Lexical analysis, parsing, syntax-directed translation. Runtime environments. Intermediate code generation. Local optimization, Data flow analyses: constant propagation, liveness analysis, common sub expression elimination.

Section 8: Operating System

System calls, processes, threads, inter-process communication, concurrency and synchronization. Deadlock. CPU and I/O scheduling. Memory management and virtual memory. File systems.

Section 9: Databases

ER-model. Relational model: relational algebra, tuple calculus, SQL. Integrity constraints, normal forms. File organization, indexing (e.g., B and B+ trees). Transactions and concurrency control.

Section 10: Computer Networks

Concept of layering: OSI and TCP/IP Protocol Stacks; Basics of packet, circuit and virtual circuit-switching; Data link layer: framing, error detection, Medium Access Control, Ethernet bridging; Routing protocols: shortest path, flooding, distance vector and link state routing; Fragmentation and IP addressing, IPv4, CIDR notation, Basics of IP support protocols (ARP, DHCP, ICMP), Network Address Translation (NAT); Transport layer: flow control and congestion control, UDP, TCP, sockets; Application layer protocols: DNS, SMTP, HTTP, FTP, Email.

JAIDEV EDUCATION SOCIETY'S
J D COLLEGE OF ENGINEERING AND MANAGEMENT
KATOL ROAD, NAGPUR
Website: www.jdcoem.ac.in E-mail: info@jdcoem.ac.in
(An Autonomous Institute, with NAAC "A" Grade)
Affiliated to DBATU, RTMNU & MSBTE Mumbai
Education to Eternity

Examination Section

VISION

To win the trust of all stakeholders in conducting the assessment and evaluation.

MISSION

1. To frame and adopt procedure for various works involved in accountability.
2. To utilize the resources as per expertise of individual and maintaining good quality and standard of assessment work.
3. To ensure that the student participate in assessment process enthusiastically.

OFFICE OF CONTROLLER OF EXAMINATIONS

Time Table of B. Tech VII-Sem Winter 2025 (Regular) - Theory Examination

Time: 09.30 AM to 12.30 PM **Date: 25-09-2025**

Branch / Day & Date of Exam	Wednesday 29-10-2025	Saturday 01-11-2025	Thursday 06-11-2025	Monday 10-11-2025	Thursday 13-11-2025	Monday 17-11-2025
Computer Science & Engineering	Cyber Security & Cryptography	Data Science	Elective IV **SW / BDAT**	Elective V **NLP / AIWCC**	Research Methodology	OE-III **PHP**
Information Technology	Artificial Intelligence & Cognitive Robotics	Data Science	Elective V **ACV**	Elective IV **NLP**	Research Methodology	OE-III **CCSM**
Artificial Intelligence	Cyber Security & Privacy	Elective -V **KMML / GAFLS**	Elective -IV **SSDS / HC**	Natural Language Processing	Research Methodology	OE-III **CSS**
CSE (Data Science)	Bigdata Analytics	Embedded System & IoT Computing	Professional Elective – IV **BC / CI**	Professional Elective – V **WSEC**	Research Methodology	OE-III **E-Commerce**
Electronics & Telecommunication	Digital Communication	Professional Elective III **IoT & WSN**	Professional Elective IV **M COM**	Professional Elective V **ML**	IPR (Intellectual Property Rights)	OE-III **Sensors & Actuators**
Electrical Engineering	Switch gear and protection	High Voltage Engineering	Elective V **FACTS / UEET**	——	Intellectual Property Rights	OE-III **WHP**
Mechanical Engineering	Elective -III **CADMA**	Elective IV **RAC / WCM**	——	——	Intellectual property rights	OE-III **PGSP**
Civil Engineering	Engineering Economics, Estimating and Costing	Professional Elective III **CM**	Professional Elective IV **SCE / SHWM**	Professional Elective V **EIA & LCA / ADCS**	IPR (Intellectual Property Rights)	OE-III **Smart City**

Copy to:

1. Hon'ble Directors, JES
2. All Dean's / H.O.D.'s / SH's for neccessary action.

Dy. Controller of Examinations **Controller of Examinations** **Principal**

Controller of Examinations
JDCOEM, Nagpur

JAIDEV EDUCATION SOCIETY'S
J D COLLEGE OF ENGINEERING AND MANAGEMENT
KATOL ROAD, NAGPUR
Website: www.jdcoem.ac.in E-mail: info@jdcoem.ac.in
(An Autonomous Institute, with NAAC "A" Grade)
Affiliated to DBATU, RTMNU & MSBTE Mumbai
Education to Eternity

Examination Section

VISION

To win the trust of all stakeholders in conducting the assessment and evaluation.

MISSION

1. To frame and adopt procedure for various works involved in accountability.
2. To utilize the resources as per expertise of individual and maintaining good quality and standard of assessment work.
3. To ensure that the student participate in assessment process enthusiastically.

OFFICE OF CONTROLLER OF EXAMINATIONS

Time Table of B. Tech VII-Sem Winter 2025 (Regular) - Practical Examination

Time: 10.00 AM to 04.00 PM **Date: 25-09-2025**

Branch / Day & Date of Exam	Monday	Tuesday	Wednesday
	24-11-2025	25-11-2025	26-11-2025
Computer Science & Engineering	Data Science using R	Cyber Security & Cryptography	Project-I
Information Technology	Data Science using R	Middleware Technologies	Project-I
Artificial Intelligence	Natural Language Processing	Cyber Security & Privacy	Project phase-I
CSE (Data Science)	Bigdata Analytics	ES & IOT	Project Phase I
Electronics & Telecommunication	Digital Communication	Basic Electronic Simulation	Major Project-I
Electrical Engineering	Switch gear and protection	Computer Applications in Electrical Engineering	Project-I
Mechanical Engineering	Mechanical Engineering	Project Phase-I	____
Civil Engineering	Engineering Economics, Estimation and Costing	Project-I	____

Copy to:

1. Hon'ble Directors, JES
2. All Dean's / H.O.D.'s / SH's for neccessary action.

CONTROLLER OF EXAMINATIONS JDCOEM

Dy. Controller of Examinations Controller of Examinations Principal

Controller of Examinations
JDCOEM, Nagpur

JAIDEV EDUCATION SOCIETY'S
J D COLLEGE OF ENGINEERING AND MANAGEMENT
KATOL ROAD, NAGPUR
Website: www.jdcoem.ac.in E-mail: info@jdcoem.ac.in
(An Autonomous Institute, with NAAC "A" Grade)
Affiliated to DBATU, RTMNU & MSBTE Mumbai
Department of Information Technology
2025-26

Education to Eternity

Vision

To be recognized as a centre of excellence in the field of Information Technology where inquisitive minds of students are fostered, leading to skills professionals for satisfying the needs of society.

Mission

1. Apply knowledge of engineering fundamentals & cutting-edge technology to identify and implement innovative solutions for engineering problems and issue in society at large.
2. Build strong interpersonal skills and will engage in life long learning to enhance their career positions, both as team members and leaders.

All Students/Faculties of B.Tech VII Semester Information Technology Department are hereby informed that the schedule for the "Mid Semester Examination-II" are as follows:

Theory Examination

Sr. No.	Date	Day	Subject Name	Subject Code	Time
1	29.09.2025	Monday	Data Science	IT7T001	10.20 AM – 11.20 AM
2	30.09.2025	Tuesday	Artificial Intelligence & Cognitive Robotics	IT7T002	10.20 AM – 11.20 AM
3	01.10.2025	Wednesday	Professional Elective-IV	IT7TE04	10.20 AM – 11.20 AM
4	03.10.2025	Friday	Professional Elective -V	IT7TE05	10.20 AM – 11.20 AM
5	04.10.2025	Saturday	Open Elective	IT7O003	10.20 AM – 11.20 AM
6	06.10.2025	Monday	Research Methodology	IT7T005	10.20 AM – 11.20 AM

Internal Practical Examination

Batch	Lab Subject	Subject Code	Date	Time	Lab/Class Room Number
A1	Data Science using R Lab	IT7L001	07.10.2025	10:20 AM – 12:20 PM	VS 103
A2	Middleware Technologies Lab	IT7L002	07.10.2025	10:20 AM – 12:20 PM	VS 114
A1	Data Science using R Lab	IT7L001	08.10.2025	10:20 AM – 12:20 PM	VS 103
A2	Middleware Technologies Lab	IT7L002	08.10.2025	10:20 AM – 12:20 PM	VS 114
A1, A2	Project	IT7P003	09.10.2025	3:15 PM – 4:15 PM	VS 009

Exam Incharge

for. Academic Incharge

25/4/25
HOD, IT

Deputy Dean Academics

Dean Academic

Principal

To
The Director
Defence Lab, Jodhpur

Date: 8th January , 2026

Sub:- **Request for grant of leave availed during Mid-II and Final End Semester exams in College.**

Respected Sir,

I am Akash Ashok Ramanni, 6 month Paid Intern, working in STT/RDS group of the lab at DRALA Complex. I have availed leave from 26th September 2025 to 07th October 2025 for Mid-II Semester Theory Exam and from 22nd October 2025 to 26th November 2025 for Final End Semester Exams in college which is in Nagpur,Maharashtra. During this period, I was not able to attend the mandatory 15 days in office for the month of October and November 2025.The break-up of leave availed is as under:

Duration	Purpose of Leave
26/09/2025 to 27/09/2025	Travelling through train from Jodhpur to Nagpur, Maharashtra.
29/09/2025 to 06/10/2025	Attending Mid-II Semester Exams.
06/10/2025 to 07/10/2025	Travelling through train from Nagpur, Maharashtra to Jodhpur.
22/10/2025 to 23/10/2025	Travelling through train from Jodhpur to Nagpur, Maharashtra.
29/10/2025 to 25/11/2025	Attended End semester Theory and Practical exams.
25/11/2025 to 26/11/2025	Travelling through train from Nagpur, Maharashtra to Jodhpur.

Kindly grant me leave for the aforesaid period.
I will be very thankful to you.

Yours Sincerely

Akash Ashok Ramanni
6 month Paid Intern

Recommended by:

Sh. M L Meena, Sc 'F'
DRDO,DRALA Complex,Jodhpur

Through HRD

Real Python Pocket Reference

Visit realpython.com to turbocharge your Python learning with in-depth tutorials, real-world examples, and expert guidance.

Getting Started

Follow these guides to kickstart your Python journey:

- realpython.com/what-can-i-do-with-python
- realpython.com/installing-python
- realpython.com/python-first-steps

Start the Interactive Shell

```
$ python
```

Quit the Interactive Shell

```
>>> exit()
```

Run a Script

```
$ python my_script.py
```

Run a Script in Interactive Mode

```
$ python -i my_script.py
```

Learn More on realpython.com/search:

`interpreter` · `run a script` · `command line`

Comments

- Always add a space after the #
- Use comments to explain "why" of your code

Write Comments

```
# This is a comment
# print("This code will not run.")
print("This will run.")  # Comments are ignored by Python
```

Learn More on realpython.com/search:

`comment` · `documentation`

Data Types

- Python is dynamically typed
- Use `None` to represent missing or optional values
- Use `type()` to check object type
- Check for a specific type with `isinstance()`
- `issubclass()` checks if a class is a subclass

Type Investigation

```
type(42)                # <class 'int'>
type(3.14)              # <class 'float'>
type("Hello")           # <class 'str'>
type(True)              # <class 'bool'>
type(None)              # <class 'NoneType'>

isinstance(3.14, float)  # True
issubclass(int, object)  # True - everything inherits from object
```

Type Conversion

```
int("42")               # 42
float("3.14")           # 3.14
str(42)                 # "42"
bool(1)                 # True
list("abc")             # ["a", "b", "c"]
```

Learn More on realpython.com/search:

`data types` · `type checking` · `isinstance` · `issubclass`

Variables & Assignment

- Variables are created when first assigned
- Use descriptive variable names
- Follow `snake_case` convention

Basic Assignment

```
name = "Leo"            # String
age = 7                 # Integer
height = 5.6            # Float
is_cat = True           # Boolean
flaws = None            # None type
```

Parallel & Chained Assignments

```
x, y = 10, 20           # Assign multiple values
a = b = c = 0           # Give same value to multiple variables
```

Augmented Assignments

```
counter += 1
numbers += [4, 5]
permissions |= write
```

Learn More on realpython.com/search:

`variables` · `assignment operator` · `walrus operator`

Strings

- It's recommended to use double-quotes for strings
- Use `"\n"` to create a line break in a string
- To write a backslash in a normal string, write `"\\"`

Creating Strings

```
single = 'Hello'
double = "World"
multi = """Multiple
line string"""
```

String Operations

```
greeting = "me" + "ow!"  # "meow!"
repeat = "Meow!" * 3     # "Meow!Meow!Meow!"
length = len("Python")   # 6
```

String Methods

```
"a".upper()                     # "A"
"A".lower()                     # "a"
" a ".strip()                   # "a"
"abc".replace("bc", "ha")       # "aha"
"a b".split()                   # ["a", "b"]
"-".join(["a", "b"])            # "a-b"
```

String Indexing & Slicing

```
text = "Python"
text[0]      # "P" (first)
text[-1]     # "n" (last)
text[1:4]    # "yth" (slice)
text[:3]     # "Pyt" (from start)
text[3:]     # "hon" (to end)
text[::2]    # "Pto" (every 2nd)
text[::-1]   # "nohtyP" (reverse)
```

String Formatting

```
# f-strings
name = "Aubrey"
age = 2
f"Hello, {name}!"               # "Hello, Aubrey!"
f"{name} is {age} years old"    # "Aubrey is 2 years old"
f"Debug: {age=}"                # "Debug: age=2"

# Format method
template = "Hello, {name}! You're {age}."
template.format(name="Aubrey", age=2)    # "Hello, Aubrey! You're 2."
```

Raw Strings

```
# Normal string with an escaped tab
"This is:\tCool."       # "This is:    Cool."

# Raw string with escape sequences
r"This is:\tCool."      # "This is:\tCool."
```

Learn More on realpython.com/search:

`strings` · `string methods` · `slice notation` · `raw strings`

Numbers & Math

Arithmetic Operators

```
10 + 3    # 13
10 - 3    # 7
10 * 3    # 30
10 / 3    # 3.3333333333333335
10 // 3   # 3
10 % 3    # 1
2 ** 3    # 8
```

Useful Functions

```
abs(-5)              # 5
round(3.7)           # 4
round(3.14159, 2)    # 3.14
min(3, 1, 2)         # 1
max(3, 1, 2)         # 3
sum([1, 2, 3])       # 6
```

Learn More on realpython.com/search:

`math` · `operators` · `built in functions`

Conditionals

- Python uses indentation for code blocks
- Use 4 spaces per indentation level

If-Elif-Else

```
if age < 13:
    category = "child"
elif age < 20:
    category = "teenager"
else:
    category = "adult"
```

Comparison Operators

```
x == y    # Equal to
x != y    # Not equal to
x < y     # Less than
x <= y    # Less than or equal
x > y     # Greater than
x >= y    # Greater than or equal
```

Logical Operators

```
if age >= 18 and has_car:
    print("Roadtrip!")

if is_weekend or is_holiday:
    print("No work today.")

if not is_raining:
    print("You can go outside.")
```

Learn More on realpython.com/search:

`conditional statements` · `operators` · `truthy falsy`

Loops

- `range(5)` generates 0 through 4
- Use `enumerate()` to get index and value
- `break` exits the loop, `continue` skips to next
- Be careful with `while` to not create an infinite loop

For Loops

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

# Loop through collection
fruits = ["apple", "banana"]
for fruit in fruits:
    print(fruit)

# With enumerate for index
for i, fruit in enumerate(fruits):
    print(f"{i}: {fruit}")
```

While Loops

```
while True:
    user_input = input("Enter 'quit' to exit: ")
    if user_input == "quit":
        break
    print(f"You entered: {user_input}")
```

Loop Control

```
for i in range(10):
    if i == 3:
        continue  # Skip this iteration
    if i == 7:
        break     # Exit loop
    print(i)
```

Learn More on realpython.com/search:

`for loop` · `while loop` · `enumerate` · `control flow`

Functions

- Define functions with `def`
- Always use `()` to call a function
- Add `return` to send values back
- Create anonymous functions with the `lambda` keyword

Defining Functions

```
def greet():
    return "Hello!"

def greet_person(name):
    return f"Hello, {name}!"

def add(x, y=10):    # Default parameter
    return x + y
```

Calling Functions

```
greet()                  # "Hello!"
greet_person("Bartosz")  # "Hello, Bartosz"
add(5, 3)                # 8
add(7)                   # 17
```

Return Values

```
def get_min_max(numbers):
    return min(numbers), max(numbers)

minimum, maximum = get_min_max([1, 5, 3])
```

Useful Built-in Functions

```
callable()  # Checks if an object can be called as a function
dir()       # Lists attributes and methods
globals()   # Get a dictionary of the current global symbol table
hash()      # Get the hash value
id()        # Get the unique identifier
locals()    # Get a dictionary of the current local symbol table
repr()      # Get a string representation for debugging
```

Lambda Functions

```
square = lambda x: x**2
result = square(5)  # 25

# With map and filter
numbers = [1, 2, 3, 4]
squared = list(map(lambda x: x**2, numbers))
evens = list(filter(lambda x: x % 2 == 0, numbers))
```

Learn More on realpython.com/search:

`define functions` · `return multiple values` · `lambda`

Classes

- Classes are blueprints for objects
- You can create multiple instances of one class
- You commonly use classes to encapsulate data
- Inside a class, you provide methods for interacting with the data
- `.__init__()` is the constructor method
- `self` refers to the instance

Defining Classes

```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def bark(self):
        return f"{self.name} says Woof!"

# Create instance
my_dog = Dog("Frieda", 3)
print(my_dog.bark())  # Frieda says Woof!
```

Class Attributes & Methods

```
class Cat:
    species = "Felis catus"   # Class attribute

    def __init__(self, name):
        self.name = name      # Instance attribute

    def meow(self):
        return f"{self.name} says Meow!"

    @classmethod
    def create_kitten(cls, name):
        return cls(f"Baby {name}")
```

Inheritance

```
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        pass

class Dog(Animal):
    def speak(self):
        return f"{self.name} barks!"
```

Learn More on realpython.com/search:

`object oriented programming` · `classes`

Exceptions

- When Python runs and encounters an error, it creates an exception
- Use specific exception types when possible
- `else` runs if no exception occurred
- `finally` always runs, even after errors

Try-Except

```
try:
    number = int(input("Enter a number: "))
    result = 10 / number
except ValueError:
    print("That's not a valid number!")
except ZeroDivisionError:
    print("Cannot divide by zero!")
else:
    print(f"Result: {result}")
finally:
    print("Calculation attempted")
```

Common Exceptions

```
ValueError          # Invalid value
TypeError           # Wrong type
IndexError          # List index out of range
KeyError            # Dict key not found
FileNotFoundError   # File doesn't exist
```

Raising Exceptions

```
def validate_age(age):
    if age < 0:
        raise ValueError("Age cannot be negative")
    return age
```

Learn More on realpython.com/search:

`exceptions` · `errors` · `debugging`

Collections

- A collection is any container data structure that stores multiple items
- If an object is a collection, then you can loop through it
- Strings are collections, too
- Use `len()` to get the size of a collection
- You can check if an item is in a collection with the `in` keyword
- Some collections may look similar, but each data structure solves specific needs

Lists

```
# Creating lists
empty = []
nums = [5]
mixed = [1, "two", 3.0, True]

# List methods
nums.append("x")          # Add to end
nums.insert(0, "y")       # Insert at index 0
nums.extend(["z", 5])     # Extend with iterable
nums.remove("x")          # Remove first "x"
last = nums.pop()         # Pop returns last element

# List indexing and checks
fruits = ["banana", "apple", "orange"]
fruits[0]                 # "banana"
fruits[-1]                # "orange"
"apple" in fruits         # True
len(fruits)               # 3
```

Tuples

```
# Creating tuples
point = (3, 4)
single = (1,)    # Note the comma!
empty = ()

# Basic tuple unpacking
point = (3, 4)
x, y = point
x                 # 3
y                 # 4

# Extended unpacking
first, *rest = (1, 2, 3, 4)
first             # 1
rest              # [2, 3, 4]
```

Sets

```
# Creating Sets
a = {1, 2, 3}
b = set([3, 4, 4, 5])

# Set Operations
a | b             # {1, 2, 3, 4, 5}
a & b             # {3}
a - b             # {1, 2}
a ^ b             # {1, 2, 4, 5}
```

Dictionaries

```
# Creating Dictionaries
empty = {}
pet = {"name": "Leo", "age": 42}

# Dictionary Operations
pet["sound"] = "Purr!"   # Add key and value
pet["age"] = 7           # Update value
age = pet.get("age", 0)  # Get with default
del pet["sound"]         # Delete key
pet.pop("age")           # Remove and return

# Dictionary Methods
pet = {"name": "Frieda", "sound": "Bark!"}
pet.keys()          # dict_keys(['name', 'sound'])
pet.values()        # dict_values(['Frieda', 'Bark!'])
pet.items()         # dict_items([('name', 'Frieda'), ('sound', 'Bark!')])
```

Learn More on realpython.com/search:

`list` · `tuple` · `set` · `dictionary` · `indexing` · `unpacking`

Comprehensions

- You can think of comprehensions as condensed `for` loops
- Comprehensions are faster than equivalent loops

List Comprehensions

```
# Basic
squares = [x**2 for x in range(10)]

# With condition
evens = [x for x in range(20) if x % 2 == 0]

# Nested
matrix = [[i*j for j in range(3)] for i in range(3)]
```

Other Comprehensions

```
# Dictionary comprehension
word_lengths = {word: len(word) for word in ["hello", "world"]}

# Set comprehension
unique_lengths = {len(word) for word in ["who", "what", "why"]}

# Generator expression
sum_squares = sum(x**2 for x in range(1000))
```

Learn More on realpython.com/search:

`comprehensions` · `data structures` · `generators`

File I/O

File Operations

```
# Read an entire file
with open("file.txt", mode="r", encoding="utf-8") as file:
    content = file.read()

# Read a file line by line
with open("file.txt", mode="r", encoding="utf-8") as file:
    for line in file:
        print(line.strip())

# Write a file
with open("output.txt", mode="w", encoding="utf-8") as file:
    file.write("Hello, World!\n")

# Append to a File
with open("log.txt", mode="a", encoding="utf-8") as file:
    file.write("New log entry\n")
```

Learn More on realpython.com/search:

`files` · `context manager` · `pathlib`

Imports & Modules

- Prefer explicit imports over `import *`
- Use aliases for long module names
- Group imports: standard library, third-party libraries, user-defined modules

Import Styles

```
# Import entire module
import math
result = math.sqrt(16)

# Import specific function
from math import sqrt
result = sqrt(16)

# Import with alias
import numpy as np
array = np.array([1, 2, 3])

# Import all (not recommended)
from math import *
```

Package Imports

```
# Import from package
import package.module
from package import module
from package.subpackage import module

# Import specific items
from package.module import function, Class
from package.module import name as alias
```

Learn More on realpython.com/search:

`import` · `modules` · `packages`

Virtual Environments

- Virtual Environments are often called “venv”
- Use venvs to isolate project packages from the system-wide Python packages

Create Virtual Environment

```
$ python -m venv .venv
```

Activate Virtual Environment (Windows)

```
PS> .venv\Scripts\activate
```

Activate Virtual Environment (Linux & macOS)

```
$ source .venv/bin/activate
```

Deactivate Virtual Environment

```
(.venv) $ deactivate
```

Learn More on realpython.com/search:

`virtual environment` · `venv`

Packages

- The official third-party package repository is the Python Package Index (PyPI)

Install Packages

```
$ python -m pip install requests
```

Save Requirements & Install from File

```
$ python -m pip freeze > requirements.txt
$ python -m pip install -r requirements.txt
```

Related Tutorials

- Installing Python Packages
- Requirements Files in Python Projects

Miscellaneous

Truthy	Falsy
`-42`	`0`
`3.14`	`0.0`
`"John"`	`""`
`[1, 2, 3]`	`[]`
`("apple", "banana")`	`()`
`{"key": None}`	`{}`
	`None`

Pythonic Constructs

```
# Swap variables
a, b = b, a

# Flatten a list of lists
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
flat = [item for sublist in matrix for item in sublist]

# Remove duplicates
unique_unordered = list(set(my_list))

# Remove duplicates, preserve order
unique = list(dict.fromkeys(my_list))

# Count occurrences
from collections import Counter

counts = Counter(my_list)
```

Learn More on realpython.com/search:

`counter` · `tricks`