81 Python Projects

Author: PRITHAM PRAJWIN V

Co - Author: The USER

Abstract

This collection represents a transformative journey through 81 structured Python projects, designed to bridge the gap between abstract mathematical theory and tangible technical mastery. By organizing these challenges into six distinct categories, the roadmap cultivates a diverse spectrum of analytical thinking ranging from the elegance of number theory to the precision of algorithmic modeling.

Moving beyond isolated coding exercises, this curriculum is built on the philosophy of "progression over completion." Each project serves as a purposeful milestone, inviting the learner to translate complex logic into clear, elegant code while developing a natural intuition for problem-solving.

Ultimately, this journey is as much about personal growth as it is about programming. It offers a disciplined rhythm of challenges intended to refine raw curiosity into professional-grade technical clarity. By the final project, the learner will have evolved from following simple formulas to architecting sophisticated mathematical reasoning, proving that consistent, dedicated effort is the true catalyst for confidence in the digital age.

Introduction

This document structures 81 Python projects into a 9-week roadmap. Each week corresponds to a specific category of problems, shifting the learner from basic numerical logic to deeper algorithmic reasoning. The intention is not merely completion, but progression: a steady expansion of intuition, technique, and mathematical clarity.

The descriptions that follow focus on the essential formulas, core logic, or foundational algorithms behind each project, maintaining a concise thesis-style format.

List's View

**Section 1:
Number Theory & Sequences**

1. Prime Number Checker
2. Factorial Calculation
3. Fibonacci Sequence Generator
4. Perfect Number Checker
5. Armstrong Number Checker
6. Find All Divisors
7. Leap Year Checker
8. Decimal to Binary Converter
9. Binary to Decimal Converter
10. Sum of Digits
11. Reverse a Number
12. Palindromic Number Checker
13. Abundant Number Checker
14. Sum of Primes to N
15. Check for Happy Number
16. Friendly Numbers Pair Checker

**Section 2:
String & Text Manipulation**

17. Palindrome String Checker
18. Anagram Checker
19. First Non-Repeating Character
20. Count Vowels & Consonants
21. Caesar Cipher Encoder/Decoder
22. Run-Length Encoding (RLE)
23. Reverse Words in Sentence
24. Pangram Checker
25. Longest Palindromic Substring
26. Isomorphic Strings Checker
27. Manual Substring Search
28. Longest Word in List
29. Remove Punctuation
30. Simple Pig Latin Translator
31. Reverse Only Vowels
32. Title Case Consistency Checker

Section 3: Lists, Arrays & Data Structures

33. Find Duplicates
34. Unique Elements Filter
35. Second Largest Number
36. Rotate List
37. Check Sorted
38. Merge Two Sorted Lists
39. List Intersection
40. Matrix Addition
41. Matrix Transpose
42. Spiral Matrix Output
43. Flatten Nested List
44. Find Missing Number
45. Calculate Median
46. Simple Queue Implementation
47. Simple Stack Implementation
48. Find Majority Element
49. Continuous Subarray Sum

**Section 4:
Mathematical Algorithms & Geometry**

50. GCD (Euclidean Algorithm)
51. LCM of Two Numbers
52. Power of a Number
53. Collatz Conjecture Steps
54. Square Root Approximation
55. Triangle Area (Heron's Formula)
56. Quadratic Equation Solver
57. Pythagorean Triples Finder
58. Pascal's Triangle Generator
59. Simple Stock Profit Calculator
60. Compound Interest Calculator
61. Slope of a Line
62. Check Collinearity
63. Midpoint of Line Segment
64. Simple Harmonic Motion
65. Volume of a Sphere

**Section 5:
Intermediate Mathematicl Algorithmic Projects**

66. Prime Factorization
67. Generate Arithmetic Progression
68. Generate Geometric Progression
69. LCM of a List
70. Sieve of Eratosthenes
71. Matrix Multiplication
72. Determinant of 2×2 / 3×3 Matrix
73. Evaluate Polynomial (Horner's Method)
74. Numerical Integration (Trapezoidal Rule)
75. Numerical Differentiation
76. Solve Linear Equations (2×2 or 3×3)
77. Combinations & Permutations
78. Approximate Root of Cubic Equation
79. Vector Operations
80. Distance from Point to Line
81. Evaluate Continued Fractions

Section 1: Core Number Theory & Sequences (16)

1. **Prime Number Checker**: $n > 1$ is prime iff $\forall k \in \{2, \ldots, \lfloor\sqrt{n}\rfloor\},\ n \not\equiv 0 \pmod{k}$.
2. **Factorial Calculation**: Compute $n! = n(n-1)(n-2)\cdots 2\cdot 1$ (iterative or recursive).
3. **Fibonacci Sequence Generator**: $F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$.
4. **Perfect Number Checker**: n is perfect iff $\sum_{d|n,\, d<n} d = n$.
5. **Armstrong Number Checker**: n is Armstrong iff $n = \sum_{i=1}^{m} d_i^m$ where m is digit-count.
6. **Find All Divisors**: Return $\{d \mid d \mid n\}$; iterate $d \leq \sqrt{n}$ and add d and n/d.
7. **Leap Year Checker**: Gregorian rule: year divisible by 4 and not by 100, unless divisible by 400.
8. **Decimal to Binary Converter**: Repeated division by 2: collect remainders $n \to n//2$.
9. **Binary to Decimal Converter**: Value $V = \sum_{i=0}^{m-1} b_i 2^i$ for bits b_i.
10. **Sum of Digits**: $S(n) = \sum_i d_i$ for decimal digits d_i of n.
11. **Reverse a Number**: Reconstruct via $r = 0$; while $n > 0$: $r = 10r + (n \bmod 10)$; $n \mathrel{//}= 10$.
12. **Palindromic Number Checker**: n palindromic iff its digit sequence equals its reverse.
13. **Abundant Number Checker**: n is abundant if $\sum_{d|n,\, d<n} d > n$.
14. **Sum of Primes up to N**: $\sum_{p \leq N} p$ where primes p found via sieve or primality tests.
15. **Check for Happy Number**: Iterate $n \mapsto \sum(\text{digits})^2$; n is happy iff sequence reaches 1 (detect loops).
16. **Friendly Numbers Pair Checker**: Compare abundance index $\sigma(n)/n$; numbers are friendly if indices equal.

Section 2: String & Text Manipulation (16)

1. **Palindrome String Checker**: Normalize (lowercase, remove spaces/punct), check $s = s_{\text{rev}}$.
2. **Anagram Checker**: s, t are anagrams iff $\text{sorted}(s) = \text{sorted}(t)$ or char-counts equal.
3. **First Non-Repeating Character**: Use frequency map $f(c)$; return first c with $f(c) = 1$ scanning left-to-right.
4. **Count Vowels & Consonants**: Classify characters; counts: $V = \#\{c \in s : c \in \text{Vowels}\}$, C similarly.
5. **Caesar Cipher Encoder/Decoder**: Shift letters by k: $c \mapsto (c - k) \bmod 26$ (encode/decode by sign).
6. **Run-Length Encoding (RLE)**: Replace runs by symbol+count, e.g. $AAAB \to A3B1$ (compress contiguous equal chars).
7. **Reverse Words in Sentence**: Tokenize by whitespace and reverse token order: $w_1 w_2 \ldots w_n \mapsto w_n \ldots w_1$.
8. **Pangram Checker**: s pangram iff $\forall a \in \text{alphabet},\ a \in \text{letters}(s)$.
9. **Longest Palindromic Substring**: For each center expand: check palindromes with odd/even centers; take maximum length.
10. **Isomorphic Strings Checker**: Exists bijection f on chars s.t. $f(s[i]) = t[i]$ for all i (one-to-one mapping).
11. **Manual Substring Search**: Implement naive sliding window: check $s[i : i+m] == p$ for pattern p length m, or KMP for efficiency.
12. **Longest Word in List**: Return $\arg\max_{w \in \text{list}} |w|$.
13. **Remove Punctuation**: Filter characters by class: keep $\{c : c \in \text{alnum} \vee c = \text{space}\}$.
14. **Simple Pig Latin Translator**: For word starting with consonant cluster C: $C + rest \to rest + C + \texttt{ay}$; vowels handled differently.
15. **Reverse Only Vowels**: Two-pointer swap of vowels indices, leaving consonants in place.
16. **Title Case Consistency Checker**: Verify words start with uppercase and subsequent letters lowercase (exceptions can be specified).

Section 3: Lists, Arrays & Data Structures (17)

1. **Find Duplicates**: Use frequency map $f(x)$; duplications are $\{x : f(x) > 1\}$.
2. **Unique Elements Filter**: Use set or ordered-preserving map: result = first occurrence of each element.
3. **Second Largest Number (No Sorting)**: Track max1 and max2 in one pass: update accordingly.
4. **Rotate List by k**: For length n: new index $i \mapsto (i + k) \bmod n$ (right rotation); use reversal trick or slicing.
5. **Check Sorted**: Verify $\forall i,\ a_i \le a_{i+1}$.
6. **Merge Two Sorted Lists**: Two-pointer merge producing sorted output: $O(n + m)$ time.
7. **List Intersection**: Compute set intersection or merge-style intersection if lists sorted.
8. **Matrix Addition**: Given $A, B \in \mathbb{R}^{n\times m}$, $C_{ij} = A_{ij} + B_{ij}$.
9. **Matrix Transpose**: $A^\top$ where $(A^\top)_{ij} = A_{ji}$.
10. **Spiral Matrix Output**: Traverse layers: top row, right col, bottom row reversed, left col reversed — repeat shrinking bounds.
11. **Flatten Nested List**: Recursively or iteratively concatenate sublists into 1-D list.
12. **Find Missing Number (1..N)**: Using sum formula: missing $= \frac{N(N+1)}{2} - \sum$ given or xor method.
13. **Calculate Median**: Sort or use selection algorithm; median = middle value or average of two middles.
14. **Simple Queue Implementation**: FIFO operations: `enqueue(x)`, `dequeue()` with pointer/index or deque.
15. **Simple Stack Implementation**: LIFO: `push(x)`, `pop()` using list append/pop.
16. **Find Majority Element**: Use Boyer–Moore majority vote: candidate with final verification.
17. **Continuous Subarray Sum**: Use the sliding window technique for nonnegative arrays; apply prefix sums with a hash-map for general arrays to find a subarray that sums to the target.

Section 4: Mathematical Algorithms & Geometry (16)

1. **GCD (Euclidean Algorithm)**: $\gcd(a,b) = \gcd(b, a \bmod b)$ until remainder 0.
2. **LCM of Two Numbers**: $\operatorname{lcm}(a,b) = \dfrac{|ab|}{\gcd(a,b)}$.
3. **Power of a Number (x^y)**: Fast exponentiation via binary exponentiation: exponentiate in $O(\log y)$.
4. **Collatz Conjecture Steps**: Iterate $n \mapsto n/2$ if even, $3n+1$ if odd; count steps to reach 1 (empirical exploration).
5. **Square Root Approximation (Babylonian)**: Iteration $x_{k+1} = \frac{1}{2}\left(x_k + \dfrac{S}{x_k}\right)$ converges to $\sqrt{S}$.
6. **Triangle Area (Heron)**: For sides a, b, c, $s = \frac{a+b+c}{2}$, area $A = \sqrt{s(s-a)(s-b)(s-c)}$.
7. **Quadratic Equation Solver**: Roots $x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$; consider sign (D).
8. **Pythagorean Triples Finder**: Generate integers (m,n) with $m > n$: $a = m^2 - n^2,\ b = 2mn,\ c = m^2 + n^2$ for primitive triples.
9. **Pascal's Triangle Generator**: Row n entries $\binom{n}{k} = \dfrac{n!}{k!(n-k)!}$.
10. **Simple Stock Profit Calculator**: Max profit $= \max_{j>i}(price_j - price_i)$ computed in one pass tracking min price.
11. **Compound Interest Calculator**: $A = P\left(1 + \dfrac{r}{n}\right)^{nt}$ for principal P, rate r, n compounding per year, t years.
12. **Slope of a Line**: For points $(x_1, y_1), (x_2, y_2)$, slope $m = \dfrac{y_2 - y_1}{x_2 - x_1}$ (if denominator $\neq 0$).
13. **Check Collinearity**: Three points collinear iff area determinant zero: $\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = 0$.
14. **Midpoint of Line Segment**: Midpoint $M = \left(\dfrac{x_1 + x_2}{2}, \dfrac{y_1 + y_2}{2}\right)$.
15. **Simple Harmonic Motion (SHM)**: $x(t) = A\sin(\omega t + \phi)$ with angular freq. $\omega = \sqrt{k/m}$ for mass-spring.
16. **Volume of a Sphere**: Volume $V = \frac{4}{3}\pi r^3$, surface area $S = 4\pi r^2$.

Section 5: Intermediate Mathematical Algorithm Projects (16)

1. **Prime Factorization**: Repeatedly divide by primes $p \leq \sqrt{n}$; represent $n = \prod p_i^{\alpha_i}$.

2. **Generate Arithmetic Progression**: $a_n = a + (n-1)d$, sum $S_n = \frac{n}{2}(2a + (n-1)d)$.

3. **Generate Geometric Progression**: $a_n = ar^{n-1}$,
$$\text{sum } S_n = a\frac{r^n - 1}{r - 1} \text{ for } r \neq 1.$$

4. **LCM of a List**: Fold pairwise: $\operatorname{lcm}(a_1, \ldots, a_k) = \operatorname{lcm}(\ldots(\operatorname{lcm}(a_1, a_2), a_3)\ldots)$ via gcd formula.

5. **Sieve of Eratosthenes**: Mark multiples of each prime starting from 2 to generate primes $\leq N$ in $O(N \log\log N)$ time.

6. **Matrix Multiplication**: For $C = AB$, $C_{ij} = \sum_k A_{ik}B_{kj}$; complexity $O(n^3)$ naive.

7. **Determinant of** 2×2 / 3×3: 2×2: $\det\begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$; 3×3 via Sarrus or expansion.

8. **Evaluate Polynomial (Horner's Method)**: $P(x) = a_0 + x(a_1 + x(a_2 + \cdots))$ reduces multiplications to $O(n)$.

9. **Numerical Integration (Trapezoidal Rule)**:
Approximate $\int_a^b f(x)\,dx \approx \sum_{i=0}^{n-1} \frac{h}{2}(f(x_i) + f(x_{i+1}))$ with $h = (b-a)/n$.

10. **Numerical Differentiation**: Forward diff $f'(x) \approx \dfrac{f(x+h) - f(x)}{h}$;
central diff $\dfrac{f(x+h) - f(x-h)}{2h}$ for better accuracy.

11. **Solve Linear Equations (2×2 or 3×3)**: Use Gaussian elimination or Cramer's rule: for 2×2, $x = \dfrac{\det(B_x)}{\det(A)}$ when $\det(A) \neq 0$.

12. **Combinations & Permutations**: $nCr = \frac{n!}{r!(n-r)!}$, $nPr = \frac{n!}{(n-r)!}$; compute iteratively to avoid overflow.

13. **Approximate Root of Cubic (Newton's Method)**: Iterate $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$ for cubic $f(x)$ until tolerance.

14. **Vector Operations**: Magnitude $\|\mathbf{v}\| = \sqrt{\sum v_i^2}$, dot product $\mathbf{u} \cdot \mathbf{v} = \sum u_i v_i$, angle $\theta = \cos^{-1}\left(\frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}\right)$.

15. **Distance from Point to Line (2D)**: For line $Ax + By + C = 0$ and point (x_0, y_0), distance $d = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$.

16. **Evaluate Continued Fractions**: Compute iteratively using nested reciprocals: $a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \cdots}}$ truncated to desired depth.

END