E-ISSN: 2707-6644
P-ISSN: 2707-6636
www.computersciencejournals.com/ijcpdm
IJCPDM 2024; 5(2): 34-40
Received: 02-07-2024
Accepted: 06-08-2024

Moti Prasad Giri
Graduate School of Science and Technology, Faculty of Science and Technology, Mid-West University, Birendranagar, Surkhet, Nepal

Dinesh Panthi
Department of Mathematics, Valmeeki Campus, Nepal Sanskrit University, Kathmandu, Nepal

Kanhaiya Jha
School of Science, Kathmandu University, Dhulikhel, Kavre, Nepal

Madhav Dhakal
Graduate School of Science and Technology, Faculty of Science and Technology, Mid-West University, Birendranagar, Surkhet, Nepal

Corresponding Author:
Moti Prasad Giri
Graduate School of Science and Technology, Faculty of Science and Technology, Mid-West University, Birendranagar, Surkhet, Nepal

Implementation of Vedic principles in python programming

Moti Prasad Giri, Dinesh Panthi, Kanhaiya Jha and Madhav Dhakal

DOI: https://doi.org/10.33545/27076636.2024.v5.i2a.102

Abstract
Vedic mathematics is ancient mathematics originating from Vedas offers distinct approaches to solve mathematical problems for calculations. The convergence of ancient wisdom and modern computation has been explored through this paper which investigates the potential applications of Vedic mathematics in the domain of computer science and Information Technology (IT). This paper addresses those Vedic principles in Python programming for enhancing the computational efficiency in IT and Computer Science applications. The main objective of this paper presents the implementation of various Vedic principles in Python programming, with some examples. Vedic principles often emphasize clarity, structure, and systematic approaches, which can be quite relevant in modern mathematics and software development. So, to reduce the complexity, we need to select appropriate Vedic sutras and then implement them appropriately.

Keywords: Vedic mathematics, techniques, python, applications, python code

Introduction
Veda is a Sanskrit word which means knowledge or sacred scriptures. Vedas are not the text on Mathematics but maintain a lot of mathematical concepts. Vedic mathematics is derived from the ancient Indian text, particularly from the “Atharva Veda”. It deals the branches of modern mathematics which we are today aware of. Vedic Mathematics is credited to Jagat Guru, Shankaracharya Sri Bharati Krishna Tirthaji Maharaj (1884-1960) a scholar of Mathematics, Sanskrit & Philosophy. He discovered 16 sutras (aphorisms) and 13 sub-sutras (corollaries) between the periods 1911 to 1918 from the Atharva Veda, involved sutras that can be applied in the modern system of mathematics to ease up any complicated mathematical problems. These sutras and subasutras collectively were later named as Vedic Mathematics [1-2]. The Ganit Sutras' also known as 'Sulabh Sutras' is the simple system of Mathematics. These Sutras can be applied to cover almost every branch of Mathematics. These formulae can be applied even to complex problems involving a large number of mathematical operations saves a lot of time and efforts in solving these problems. Applications of the sutras can be applied in different branches of mathematics. These techniques make mathematical calculation fast, easy, great confidence, and booster for students [3].

The applications of Vedic sutras in Information Technology can help several benefits, such as, improve the computation time significantly, reducing the power consumption in the large-scale IT infrastructures where energy efficiency is a priority. This is essential in IT applications where precise calculations are necessary to avoid errors and ensures reliable results, and improves the operational efficiency as compared with the traditional methods. This makes it a compelling area of exploration for optimizing computational tasks in modern technology environments [4]. The conventional mathematics algorithms are simplified and also optimized by using Vedic sutras. The few areas of modern mathematics can be applied in efficiently. In today's world of rapidly advancing technology and increasing demand for computational power, the efficiency and speed of calculations are paramount. Traditional methods of computation can indeed be time-consuming and resource-intensive, leading to higher power consumption and potentially slower processing speeds. Incorporating Vedic Mathematics principles into the implementation of digital systems thus not only addresses the current demands for faster and more efficient computation but also contributes to sustainable and environmentally responsible technology solutions. This makes it a valuable approach for modern information technology infrastructures seeking to optimize performance while minimizing energy consumption [5].

We are all well aware of constant evolution of educational methodologies and significance of diverse approaches to teaching and learning. The convergence of Vedic mathematics with modern information Technology (IT) presents a unique opportunity to harness the efficiency and elegance of ancient mathematical techniques for enhancing efficiency and innovation in modern computational process. This paper explores the foundational principles of Vedic mathematics investigates their potential applications and benefits in the context of Information Technology [6]. Vedic mathematics is not limited to specific mathematics domains but is applicable across various branches of mathematics, including arithmetic, algebra, geometry, trigonometry, and calculus. Its versatility and adaptability make it a valuable tool for solving diverse mathematical problems. By incorporating Vedic principles to modern mathematical education and practice, individuals can enhance their computational skills, cultivate mathematical intuition, and appreciate the beauty of mathematical concepts and techniques [7]. Vedic sutras, originating from ancient texts, are founded upon natural principles intrinsic to human thought processes. The application of Vedic algorithms across diverse branches of engineering and sciences, particularly computing, emphasizing their effectiveness and performance analysis based on delay metrics. By examining theoretical foundations and practical implementations, this paper shows how Vedic principles optimize computational efficiency through insights derived from natural cognitive patterns [8].

Python Programming in Vedic mathematics

Guido van Rossum is a Dutch programmer, renowned for creating the Python programming language. He held the title of "benevolent dictator for life" (BDFL) within the Python community until he decided to step down from that role on July 12, 2018. His contributions to the world of programming are immense, and Python continues to be one of the most popular and versatile programming languages used today.

Python is widely used general purpose, high- level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation (PSF). Python's emphasis on readability, concise syntax, and developer productivity has indeed contributed to its widespread adoption and popularity. Its design philosophy, often summarized as "Readability counts" or "There should be one-- and preferably only one - obvious way to do it," has resonated well with programmers across various domains. Python's simplicity and versatility make it a go-to choice for tasks ranging from web development to data analysis, machine learning, and more.

Python's characteristics as an interpreted, object-oriented, high-level language with dynamic semantics make it incredibly versatile and suitable for various programming tasks. Its built-in data structures, combined with dynamic typing and binding, indeed make it attractive for rapid development and scripting tasks. The language's emphasis on readability and simplicity not only aids in faster development but also reduces the cost of program maintenance, as you mentioned. Furthermore, Python's support for modules and packages encourages modular programming and code reuse, promoting good software engineering practices. The fact that Python and its extensive standard library are freely available for all major platforms further enhances its accessibility and widespread usage. Python's fast edit-test-debug cycle is indeed a significant factor contributing to increased productivity and programmer satisfaction. The absence of a compilation step means that developers can quickly make changes to their code and see the results immediately, which accelerates the development process. Python's error handling mechanism, based on exceptions, also simplifies debugging by providing clear error messages and stack traces when issues arise. The availability of a source-level debugger further enhances the debugging experience, allowing developers to inspect variables, evaluate expressions, set breakpoints, and step through code, all within the Python environment itself. Python is technology-based program which helps in Vedic mathematics [10].

***Source*:** https://en.wikipedia.org

Fig 1: The designer of Python, Guido van Rossum

Some Vedic principles and its implementation with Python Programming

Here are some examples of Vedic principles

3.1 Urdhva-Tiryagbhyam Sutra

Meaning: Vertically and Crosswise

Mathematical algorithm of Urdhva-Tiryagbhyam sutra

Let $a = a_1\ a_2\ a_3$ and $b = b_1\ b_2\ b_3$ be two positive integers. Then the product of two number is denoted by P and defined by the formula

$P = (a_1b_1)\ (a_1b_2 + a_2b_1)\ (a_1b_3 + a_3b_1 + a_2b_2)\ (a_2b_3 + a_3b_2)\ (a_3b_3)$

Let $a_1b_1 = c_1d_1$, $a_1b_2 + a_2b_1 = c_2d_2$, $a_1b_3 + a_3b_1 + a_2b_2 = c_3d_3$, $a_2b_3 + a_3b_2 = c_4d_4$, and $a_3b_3 = c_5d_5$. Then the above formula reduces as

$= (c_1d_1)\ (c_2d_2)\ (c_3d_3)\ (c_4d_4)\ (c_5d_5)$

$c_1\ d_1\ d_2\ d_3\ d_4\ d_5$ (taking second digit from the first term including first digit)

$+\ c_2\ c_3\ c_4\ c_5$ (taking first digit from the second term and place from second term)

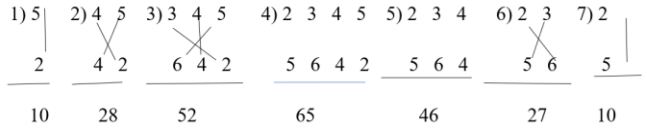
$= c_1\ (d_1 + c_2)\ (d_2 + c_3)\ (d_3 + c_4)\ (d_4 + c_5)\ (d_5)$

$$\therefore P = c_1 \prod_{i=1}^{5} d_i + \prod_{j=2}^{5} c_j$$

In general,
Let a = a_1 a_2 a_3… a_n and b = b_1 b_2 b_3 …bn be two n digit positive integers. Then the product of two number is denoted by P and defined by the formula

P = c_1 $\prod_{i=1}^{2n-1} d_i + \prod_{j=2}^{2n-1} c_j$

Example: Multiply 2345 x 5642 (n = 4)



= (10) (27) (46) (65) (52) (28) (10)

$$= \begin{array}{r} 10765280 \\ +\ 246521 \\ \hline 13230490 \end{array}$$

$\therefore$ 2345 $\times$ 5642 = 13230490

Computational method (Proposed design using in Python)
Algorithm of Urdhva-Tiryagbhyam sutra

1. Input: Positive Integers N1 and N2.
2. Initialize Variables:
   - Convert N1 and N2 to strings for easy iteration.
   - Calculate the lengths L1 and L2 of inputs.
   - Initialize a list intermediate_results with zeros to store intermediate multiplication results.
3. Calculate Intermediate Results:
   - Iterate through each digit of N1 and N2.
   - Multiply the current digits from N1 and N2.
   - Accumulate the result in intermediate_results at positions corresponding to the sum of the current indices of digits.
4. Handle Carry and Combine Results:
   - Call the add_with_carry function with intermediate_results.
   - This function iterates through the list of intermediate results:
   - Adds the current number with the carry.
   - Appends the last digit of the total to the final result list.
   - Updates the carry for the next iteration.
   - Adds any remaining carry to the result.
   - Returns the final result list and a dictionary containing intermediate results for each step.
5. Print Steps and Final Result:
   - Display the steps of the multiplication process along with the final result.
   - Iterate through the dictionary of intermediate results.
   - For each step, print the result of the multiplication, the total sum, the carry, and the partial product.
   - If it's the last step, print the final result with the carry.
6. Output: Displays the final product of N1 and N2

```
Enter the first number: 2345
Enter the second number: 5642
Step 1: Result: 10, Total Sum: 10, Carry: 1 => [0]
Step 2: Result: 28, Total Sum: 29, Carry: 2 => [90]
Step 3: Result: 52, Total Sum: 54, Carry: 5 => [490]
Step 4: Result: 65, Total Sum: 70, Carry: 7 => [0490]
Step 5: Result: 46, Total Sum: 53, Carry: 5 => [30490]
Step 6: Result: 27, Total Sum: 32, Carry: 3 => [230490]
Step 7: Result: 10, Total Sum: 13, Carry: 1 => [13230490]


The product of 2345 and 5642 is 13230490.
```

Fig 2: Computing Urdhva-Tiryagbhyam sutra proposed design in Python

2 Dhwajanka sutra
Meaning: Flag digit Method or Urdhva Tiriyak Method
Example: Divide 35785 by 125
Flag digit is 5

Remainders 11 11 6

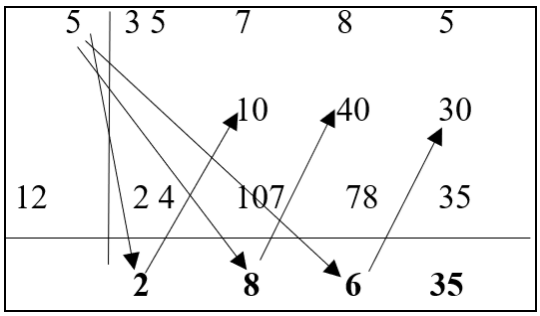


Fig 3: Process for division by using Dhwajanka sutra

Quotient = 286, and Remainder = 35
Computational method (Proposed design using Python programming):
Algorithm of Dhwajanka sutra

Input
Two numbers dividend and divisor.

Output
The final quotient and remainder

Step-1: Determine the Base
- Choose a base that is a power of 10, with the same number of digits as the divisor.
- Base = 10^ (number of digits in divisor)

Step-2: Calculate the Deviation
- Calculate the deviation of the divisor from the base.

- Deviation = base-divisor

Step-3: Initialize Quotient and Remainder
- Set quotient and remainder to zero.

Step-4: Iterate Over Each Digit of the Dividend:
- For each digit d in the dividend (from left to right):
  - Form a new number by bringing down the next digit to the current remainder: current = remainder * 10 + d
  - Determine the next digit of the quotient: q_digit = current // divisor
  - Update the quotient: quotient = quotient * 10 + q_digit
  - Calculate the new remainder: remainder = current % divisor

```
def dhwajanka_sutra_division(dividend, divisor):
    # Determine the base (10 raised to the length of the divisor)
    base = 10 ** len(str(divisor))
    # Calculate deviation (base - divisor)
    deviation = base - divisor

    print(f"Step 1: Identify base closest to the divisor: {base}")
    print(f"Step 2: Express divisor as base minus deviation: {divisor} = {base} - ({deviation})")

    quotient = 0
    remainder = 0

    for digit in str(dividend):
        # Bring down the next digit of the dividend
        current = remainder * 10 + int(digit)

        # Determine the next digit of the quotient
        q_digit = current // divisor
        quotient = quotient * 10 + q_digit
```

```
Enter the dividend: 35785
Enter the divisor: 125
Step 1: Identify base closest to the divisor: 1000
Step 2: Express divisor as base minus deviation: 125 = 1000 - (875)
Current quotient: 0
Current remainder: 3
Current quotient: 0
Current remainder: 35
Current quotient: 2
Current remainder: 107
Current quotient: 28
Current remainder: 78
Current quotient: 286
Current remainder: 35
The result of 35785 ÷ 125 using the Dhwajanka Sutra method is:
Quotient: 286, Remainder: 35
```

Fig 4: Computing Dhwajanka sutra proposed design in Python

3.3 Paravartya Yojayet Sutra

Meaning: Transpose and Adjust

Example: Divide 25687 by 125

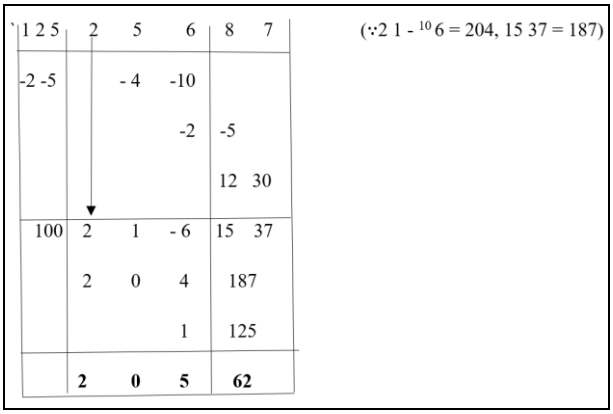


Fig 5: Process for division by using Paravartya Yojayet sutra

Quotient = 205 and Remainder = 62

Computational method (Proposed design using Python programming)

Algorithm of Paravartya Yojayet sutra

Paravartya_yojayet_division (dividend, divisor):

Step 1: Identify the base and complement
- Base = 10^length (divisor)
- Complement = base-divisor

Step 2: Convert dividend to string for digit manipulation
- dividend_str = str (dividend)
- n = length (dividend_str)

Step 3: Initialize quotient and carry
- quotient = ""
- carry = 0

Step 4: Perform division step-by-step
- for i = 0 ton -1 do
- current_digit = int (dividend_str[i])
- new_value = current_digit + carry
- quotient_digit = new_value // divisor
- carry = (new_value % divisor) * 10

- append quotient_digit to quotient
- print step details (optional)

Step 5: Calculate remainder

- remainder = carry // 10

Step 6: Return quotient and remainder

- return quotient, remainder

```
# Get user input
dividend = int(input("Enter the dividend: "))
divisor = int(input("Enter the divisor: "))

# Perform division using Paravartya Yojayet method
quotient, remainder = paravartya_yojayet_division(dividend, divisor)

# Output the result
print(f"The quotient of {dividend} divided by {divisor} using Paravartya Yojayet method is: {quotient}"
print(f"The remainder is: {remainder}")
```

```
Enter the dividend: 25687
Enter the divisor: 125
Identified base: 1000
Complement of divisor 125 with respect to base 1000: 875
Step 1: Current digit: 2, New value: 2, Quotient digit: 0, Carry: 20
Step 2: Current digit: 5, New value: 25, Quotient digit: 0, Carry: 250
Step 3: Current digit: 6, New value: 256, Quotient digit: 2, Carry: 60
Step 4: Current digit: 8, New value: 68, Quotient digit: 0, Carry: 680
Step 5: Current digit: 7, New value: 687, Quotient digit: 5, Carry: 620
The quotient of 25687 divided by 125 using Paravartya Yojayet method is: 00205
The remainder is: 62
```

Fig 6: Computing Paravartya Yojayet sutra proposed design in Python

3.4 Shesanyankena Caramena

Meaning: The Remainders by the Last Digit

Divide x by 7, where $0 < x < 7$ and $x \in N$.

Example: Divide $\frac{5}{7}$

$\frac{5 \times 10}{7}$ = 1 (remainder)

$1 \times 7 = 07$ 7 (last term)

$\frac{1 \times 10}{7}$ = 3 (remainder)

$3 \times 7 = 21$ 1 (last term)

$\frac{3 \times 10}{7}$ = 2 (remainder)

$2 \times 7 = 14$ 4 (last term)

$\frac{2 \times 10}{7}$ = 6 (remainder)

$6 \times 7 = 42$ 2 (last term)

$\frac{6 \times 10}{7}$ = 4 (remainder)

$4 \times 7 = 28$ 8 (last term)

$\frac{4 \times 10}{7}$ = 5 (remainder)

$5 \times 7 = 35$ 5 (last term)

$\frac{5 X 10}{7}$ = 1 (remainder)

1 x 7 = 07 7 (last term)

$\therefore \frac{5}{7}$ = 0. 7142857… which is an irrational number.

Computational method: (Proposed design using Python programming)

Algorithm of Shesanyankena Caramena sutra

1. **Input**
   - Dividend, Divisor
2. **Calculate**
   - quotient = dividend // divisor
   - remainder = dividend % divisor
3. **Initialize**
   - decimal_digits = []
   - seen_remainders = {}
4. **Record Integer Part**
   - Add quotient to steps.
5. **Process Decimal Part**
   - Remainder *= 10
   - step_count = 1
   - While remainder! = 0:
   - If remainder is in seen_remainders:

- Insert parentheses in decimal_digits.
- Break the loop.
- Compute Quotient Digit:
- quotient_digit = remainder // divisor
- Append quotient_digit to decimal_digits.
- Update Remainder:
- new_remainder = remainder % divisor
- remainder = new_remainder * 10
- Track Remainders:
- Add remainder and its position to seen_remainders.
- Increment step_count

6. Format Result

- Combine integer part and decimal digits.
- If no decimal digits, append .0.

7. Output

- Return result and steps list.

```
dividend = int(input("Enter the dividend: "))
divisor = int(input("Enter the divisor: "))

# Validate divisor
if divisor == 0:
    print("Error: Division by zero is not allowed.")
    return

# Calculate and display the result
result, steps = divide_with_steps(dividend, divisor)
print(f"\nThe result of dividing {dividend} by {divisor} is: {result}")
print("\nCalculation Steps:")
for step in steps:
    print(step)
```

```
Enter the dividend: 5
Enter the divisor: 7

The result of dividing 5 by 7 is: 0.(714285)

Calculation Steps:
Integer part: 0
Step 1: 50 divided by 7 gives quotient digit = 7 with new remainder = 1
Step 2: 10 divided by 7 gives quotient digit = 1 with new remainder = 3
Step 3: 30 divided by 7 gives quotient digit = 4 with new remainder = 2
Step 4: 20 divided by 7 gives quotient digit = 2 with new remainder = 6
Step 5: 60 divided by 7 gives quotient digit = 8 with new remainder = 4
Step 6: 40 divided by 7 gives quotient digit = 5 with new remainder = 5
Decimal part: (714285)
Result: 0.(714285)
```

4. Conclusion

This paper shows various Vedic sutras that are implemented in Python program for reducing speed and power consumption and increasing efficiency. It appears that the Vedic mathematics sutras help to develop the new algorithms for improving all areas. The exploration of the application of Vedic mathematics in the field of Information Technology reveals of captivating synergy between ancient wisdom and modern innovation. Vedic mathematics with its emphasis on mental calculation, symmetry, and pattern recognition offers a unique perspective on mathematical problem-solving that complements the established methods of computer science and information technology. This paper shows the implementation of Vedic mathematics principles in Python programming for reducing the time consumption of heavy and complex calculations.

The potential applications of Vedic techniques in to information technology offers a new approach to nurturing creative and adaptable problem solvers, and interdisciplinary collaboration between mathematicians and computer scientists can lead to the emergence of hybrid algorithms that push the boundaries of computational efficiency. The integration of Vedic principles in information technology embodies the dynamic interplay between tradition and progress.

References

1. Bharati Krishna Tirtha. Vedic mathematics. Delhi: Motilal Banarsidass Publications Private Limited; 1965.
2. Devi S. Application of Vedic mathematics in algebra. International Research Journal on Advanced Science Hub. 2020;2(11):61-63.
3. Anil AR. A survey on implementation of Vedic mathematics sutras in information technology. 2021 [cited 2024 Oct 19]. Available from: https://www.instavm.org
4. Jain S, Jagtap VS. Vedic mathematics in computer: A survey. International Journal of Computer Science and Information Technologies. 2014;5(6):7458-59.
5. Mathur RP. Application of Vedic mathematics in computer science. International Journal of Science and Research. 2024;13(2):990-93.
6. Kumar CS. Applications of Vedic mathematics for machine learning. 2024 [cited 2024 Oct 19]. Available from: https://www.researchgate.net
7. Khainar *et al*. Vedic mathematics-The cosmic software for implementation of fast algorithms. 2022 [cited 2024 Oct 19]. Available from: https://www.sinhgad.edu

8. Kuhlman D. A Python book: Beginning Python, advanced Python, and Python exercises. 2013 [cited 2024 Oct 19]. Available from: http://www.opensource.org/licences/mit-licence.php
9. Hart WE, Watson JP, Woodruff DL. Pyomo: Modelling and solving mathematical programs in Python. Mathematical Programming Computation. 2011;3:219-60.
10. What is Python? Executive summary. 2024 Jun 6 [cited 2024 Oct 19]. Available from: https://www.python.org
11. Thakur RS. Advanced Vedic mathematics. India: Rupa Publications; 2019.
12. Kumar CS. Vedic theorems based on Vedic mathematics sutras. 2024 May 31 [cited 2024 Oct 19]. Available from: https://www.researchgate.net
13. Thapliyal H, Srinivas MB. VLSI implementation of RSA encryption system using ancient Indian Vedic mathematics. 2017 [cited 2024 Oct 19]. Available from: https://www.researchgate.net
14. Bhatia D. Vedic mathematics made easy. India: Jaico Publishing; 2021.
15. Moti Pradasad Giri *et al*. Some application of Vedic techniques in Python programming. Journal of Mathematical Problems, Equations and Statistics, 2024, 5(2). DOI: https://doi.org/10.22271/math.2024.v5.i2a.151
16. Sher Singh Raikhola *et al*. A thematic analysis on Vedic mathematics and its importance. Open Access Library Journal, 2020, 7(08). Available from: https://doi.org/10.4236/oalib.1106665