Journal of Mathematical Problems, Equations and Statistics

E-ISSN: 2709-9407
P-ISSN: 2709-9393
JMPES 2024; 5(2): 59-65

www.mathematicaljournal.com
Received: 02-08-2024
Accepted: 03-09-2024

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

Madhav Dhakal
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

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

Some application of Vedic techniques in python programming

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

DOI: https://doi.org/10.22271/math.2024.v5.i2a.151

Abstract
Vedic mathematics is credited to Jagat guru Shankaracharya Sri Bharati Krishna Tirthaji Maharaja, a scholar of Mathematics, Sanskrit, and Philosophy. It explores the interconnection of ancient wisdom with modern computational system through this paper and investigates the potential applications of Vedic mathematics in the domain of computer science and information technology. This paper addresses those Vedic techniques which connect in to technology-based Python programming for enhancing the computational efficiency in IT and computer science applications. The main objective of this paper briefly presents the application of Vedic mathematics techniques in Python programming with some examples.

Keywords: Vedic mathematics, python, applications, algorithm design

1. Introduction
Veda is Sanskrit word which common definition is knowledge or fountain of head. Vedic mathematics is derived from the ancient text, particularly from the 'Atharva Veda'. It deals the branches of modern mathematics which are today aware of [1]. Bharati Krishna Tirthaji Maharaja (1884 – 1960) is a founder of Vedic mathematics who discovered 16 sutras and 13 sub – sutras between the period of 1911 to 1918 and involved these sutras are widely applied in the modern system of mathematics to ease up any complicated mathematical problems [1].
Veda is the earliest layer of ancient culture and the oldest sacred texts of Hinduism in the world. They are primarily composed in Sanskrit language. On the other hand, Vedic mathematics is a logic and system of mathematics that based on the principles and techniques found in ancient Vedic texts with a basic idea and rule. These sutras are concise formulas that simplify complex calculations and problem – solving processes and covers various areas of mathematics [2].
Vedic mathematics with its roots provides an interesting approach that can be advantageous in modern computing applications by offering an edge of time and space complexities over conventional methods. It can inspire novel algorithmic approaches and enhance computational thinking by providing alternative methods to traditional algorithms [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 Information Technology applications where precise calculations are necessary to avoid errors, 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 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 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]. The convergence of Vedic mathematics with 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 foundational principle of Vedic mathematics investigates their potential applications and benefits in the context of information technology [6].

2. Python Programming

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 [9].

3. Some Vedic principles and its application in Python Programming

3.1 Nikhilam Navatascaramam Dasatah Sutra

Meaning: All from 9 and the Last from 10

In algebraic form

Let a and b be two-digit numbers, then the product is obtained by the formula

$a \times b = 100 (a + b_1) + a_1b_1$, where $a_1 = (a – base)$ and $b_1 = (b – base)$

$= 10^2 (a + b_1) + a_1b_1$, where $a_1 = (a – base)$ and $b_1 = (b – base)$

Or,

$a \times b = 10^2 (b + a_1) + a_1b_1$, where $a_1 = (a – base)$ and $b_1 = (b – base)$

In general, if a and b are n digit numbers, the product is obtained by the formula

$a \times b = 10^n (a + b_1) + a_1b_1$, where $a_1 = (a – base)$ and $b_1 = (b – base)$

Case I

Example: Multiplication of 9998 and 9985

9998	- 2
9985	-15
9983	30

Final answer is $9983 \times 10000 + 30 = 99830030$

Case II:

When the number of digits is different. In this case we use the formula [7]

Example: Multiplication of 999997 and 995 (different digit number)

Choose integer m = 1000000 (near base of 999997), and n = 1000 (near base of 995).

To find k, using formula $k = \frac{m}{n} = 1000$, where $m > n$

Let a = 999997, b = 995, Then using formula

$\because$ k (b – n) = 1000 (995 – 1000) = 1000 x – 5 = -5000]

999997	- 3
995	- 5
994997	15

[Add 999997 and - 5000 = 994997]

Final answer is 994997 × 1000 + 15 = 994997015 (near base of smaller number is considered)
In algebraic form
Here, a > b, then the product of a and b is
$a \times b = n [a + k (b – n)] + a_1b_1$, where a_1 = (a – base) and b_1 = (b – base)
Alternatively,

999997	-3
995	-999005
992	2997015

(Base is 1000000)

Final answer is 992 × 1000000 + 2997015 = 994997015

Computational method (Applying Python Programming)

Algorithm of Nikhilam sutra in case I and II

Input: Two numbers xxx and yyy.

Identify the base as 10n10^n10n where nnn is the number of digits in the larger of xxx or yyy.

- Compute the deficit of xxx as xdeficit=base−xx_{\text{deficit}} = \text{base} - xxdeficit=base−x.
- Compute the deficit of yyy as ydeficit=base−yy_{\text{deficit}} = \text{base} - yydeficit=base−y.
- Multiply the deficits to get the right part: right_part=xdeficit×ydeficit\text{right_part} = x_{\text{deficit}} \times y_{\text{deficit}} right_part=xdeficit×ydeficit.
- Cross-subtract to get the left part: left_part=x−ydeficit\text{left_part} = x - y_{\text{deficit}} left_part=x−ydeficit or y−xdeficity - x_{\text{deficit}} y−xdeficit.
- Combine the parts to get the result: result=left_part×base+right_part\text{result} = \text{left_part} \times \text{base} + \text{right_part} result=left_part×base+right_part.
- Output the result

```
Enter the first number: 9998
Enter the second number: 9985
Identified base: 10000
Deficit of 9998 from base 10000: 2
Deficit of 9985 from base 10000: 15
Multiplication of deficits: 2 * 15 = 30
Cross-subtracting: (9985 - 2) = 9983
Combining parts: 9983 * 10000 + 30 = 99830030
The product of 9998 and 9985is: 99830030
Enter the first number: 999997
Enter the second number: 995
Identified base: 1000000
Deficit of 999997 from base 1000000: 3
Deficit of 995 from base 1000000: 999005
Multiplication of deficits: 3 * 999005 = 2997015
Cross-subtracting: (995 - 3) = 992
Combining parts: 992 * 1000000 + 2997015 = 994997015
The product of 999997 and 995 using Nikhilam method is: 994997015
```

3.2 Antyayordasakepi Sutra

Meaning: Last Totaling Ten

In algebraic form

Let $a = a_1 a_2$, and $b = b_1 b_2$ be two-digit numbers. Where $a_1 = b_1$ and $a_2 + b_2 = 10$. a, b ϵ N. Then the product of two numbers a and b is obtained by the formula

$a \times b = 100\, a_1 (a_1 + 1) + a_2 b_2$.

In general, let $a = a_1 a_2 a_3 \ldots a_n$ and $b = b_1 b_2 b_3 \ldots b_n$, be n digit numbers. Where $a_1 = b_1$, $a_2 = b_2$, $\ldots a_{n-1} = b_{n-1}$ and $a_n + b_n = 10$. a, b $\in N$. Then product of two numbers a and b is obtained by the formula

$a \times b = 100\, a_1 a_2 \ldots a_{n-1} (a_1 a_2 \ldots a_{n-1} + 1) + a_n b_n$

Example: Multiply 857 and 853 (n = 3)

85	7
85	3
85 × 86	21

(Last 7 + 3 = 10)

(∵ Using Ekadhikena Purvena sutra)

The result is 7310 × 100 + 21 = 731021

Computational method (Applying Python Programming)

Algorithm of Antyayordasake'pi sutra

Input Two numbers x and y

Output: Final result of multiplication

Step-1:

- Extract the last digits of x and y
- Extract the leading digits of x and y
- Ensure last_x + last_y == 10.
- Ensure leading_x == leading_y.

Step-2: If Criteria Not Met

If (last_x + last_y! = 10) OR (leading_x! = leading_y):

Return "Criteria is not support."

Step-3: Calculate Products:

- Compute the product of the leading digit by the next higher number.
- Compute the product of the last digits.

Step-4: Combine Results

- Combine the results by placing the product of the leading digits in the hundreds place and adding the product of the last digits.

3.3 Shesanyankena Caramena Sutra

Meaning: The Remainders by the Last Digit

Example: Divide 2 by 7

$\frac{2 \times 10}{7}$ = 6 (remainder) 6 × 7 = 42 2 (last term)

$\frac{6 \times 10}{7}$ = 4 (remainder) 4 × 7 = 28 8 (last term)

$\frac{4 \times 10}{7}$ = 5 (remainder) 5 × 7 = 35 5 (last term)

$\frac{5 \times 10}{7}$ = 1 (remainder) 1 × 7 = 07 7 (last term)

$\frac{1 \times 10}{7}$ = 3 (remainder) 3 × 7 = 21 1 (last term)

$\frac{3 \times 10}{7}$ = 2 (remainder) 2 × 7 = 14 4 (last term)

$\frac{2 \times 10}{7}$ = 6 (remainder) 6 × 7 = 42 2 (last term)

$\therefore \frac{2}{7}$ = 0. 2857142…

This formula can be applied only in this case, that is $(\frac{1}{7}, \frac{2}{7}, \frac{3}{7}, \ldots, \frac{6}{7})$.

Computational method (Applying 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.

```
Enter the dividend: 2
Enter the divisor: 7

The result of dividing 2 by 7 is: 0.(285714)

Calculation Steps:
Integer part: 0
Step 1: 20 divided by 7 gives quotient digit = 2 with new remainder = 6
Step 2: 60 divided by 7 gives quotient digit = 8 with new remainder = 4
Step 3: 40 divided by 7 gives quotient digit = 5 with new remainder = 5
Step 4: 50 divided by 7 gives quotient digit = 7 with new remainder = 1
Step 5: 10 divided by 7 gives quotient digit = 1 with new remainder = 3
Step 6: 30 divided by 7 gives quotient digit = 4 with new remainder = 2
Decimal part: (285714)
Result: 0.(285714)
```

3.4 Puranapuranabhyam Sutra

Meaning: By the Completion or Non – Completion

Example: Solve the cubic equation $x^3 – 2x^2 - 5x + 6 = 0$

We have, $(x – 1)^3 = x^3 – 3x^2 + 3x – 1$

Rearrange the given equation by completing cube

Or, $x^3 – 3x^2 + 3x – 1 + x^2 – 8x + 7 = 0$

Or, $(x - 1)^3 + (x – 1)(x – 7) = 0$
Or, $(x – 1)(x^2 – x – 6) = 0$
Or, $(x – 1)(x + 2)(x – 3) = 0$
$\therefore x = -2, 1$, and 3.

Computational method (Applying Python programming)

```
Original equation: x^3 - 2x^2 - 5x + 6 = 0

Rearranging the equation to complete the cube:
Expanded cubic term: (x - 1)^3 = x^3 - 3x^2 + 3x - 1
Rearrange: x^3 - 2x^2 - 5x + 6 = (x - 1)^3 + (x^2 - 8x + 7)
Simplify the equation: (x - 1)^3 + (x^2 - 8x + 7) = x**3 - 2*x**2 - 5*x + 6 = 0
Factoring the equation: (x - 1) (x^2 - x - 6) = 0
Factoring the quadratic part: x^2 - x - 6 = (x + 2)(x - 3)
Final factorization: (x - 1)(x + 2)(x - 3) = 0

Solutions: x = [-2, 1, 3]
```

This sutra also can be applied to conic sections in co – ordinate geometry.

3.5 Lopanasthapanabhyam Sutra
Meaning: By alternate elimination and retention
This formula can be applied to find the factors of harder polynomials and also to find the equation of the pair of lines and so on.

Example: Find the pair of lines of this equation $2x^2 + 2y^2 + 5xy + 2x – 5y - 12 = 0$

Given polynomial is $2x^2 + 2y^2 + 5xy + 2x – 5y – 12$

By given sutra, put $y = 0$

$2x^2 + 2x - 12 = (x + 3)(2x – 4)$ … (1)

Put $x = 0$

$2y^2 – 5y – 12 = (y - 4)(2y + 3)$ … (2)

Fill the gaps from equation (1) and (2), we obtain the required factors $(x + 2y + 3)(2x + y – 4)$ … (3)

Factors of given polynomial is $(x + 2y + 3)(2x + y – 4)$

Hence, the pair of lines of this equation is $(x + 2y + 3)(2x + y – 4) = 0$

Python code:

```python
import sympy as sp

# Define the variables
x, y = sp.symbols('x y')

# Define the polynomial
polynomial = 2*x**2+5*x*y+2*y**2+2*x-5*y-12

# Step 1: Set y = 0 and factorize
step1_polynomial = polynomial.subs(y, 0)
step1_factors = sp.factor(step1_polynomial)

# Step 2: Set x = 0 and factorize
step2_polynomial = polynomial.subs(x, 0)
step2_factors = sp.factor(step2_polynomial)

# Factorize the polynomial
required_factors = sp.factor(polynomial)

# Display the results
print(f"Original Polynomial: {polynomial}")
```

```
# Display the results
print(f"Original Polynomial: {polynomial}")
print(f"Step 1 Polynomial (y=0): {step1_polynomial}")
print(f"Step 1 Factors: {step1_factors}")
print(f"Step 2 Polynomial (x=0): {step2_polynomial}")
print(f"Step 2 Factors: {step2_factors}")
print(f"Required Factors: {required_factors}")
```

Computational method (Applying Python programming)

```
Original Polynomial: 2*x**2 + 5*x*y + 2*x + 2*y**2 - 5*y - 12
Step 1 Polynomial (y=0): 2*x**2 + 2*x - 12
Step 1 Factors: 2*(x - 2)*(x + 3)
Step 2 Polynomial (x=0): 2*y**2 - 5*y - 12
Step 2 Factors: (y - 4)*(2*y + 3)
Required Factors: (x + 2*y + 3)*(2*x + y - 4)
```

4. Conclusion

According to the proposed method, Vedic sutras are applied in technology-based program Python for reducing speed, power consumption and increasing efficiency. This paper shows Vedic mathematics techniques help to develop the new algorithms for improving all areas. The exploration of the application of Vedic mathematics in the field of IT reveals of captivating synergy between ancient wisdom and modern innovation. There is an interconnection between Vedic mathematics and Python program for reducing time consumption of heavy and complex calculations. Future possible research direction of this paper is to explore those Vedic sutras and its application in many other IT programming languages and mathematical software.

5. References

1. Tirtha BK. Vedic mathematics. India: Motilal Banarsidass Pvt. Ltd.; c1965.
2. Solanki V. A review paper on Vedic mathematics. Int J Innov Res Eng Manag. 2021;8(6):160-163.
3. Bajaj J, Jajodia B. Squaring technique using Vedic mathematics. In: Proceedings of the International Conference on Women Researchers in Electronics and Computing; c2021. p. 597-606.
4. Anil AR. A survey on implementation of Vedic mathematics sutras in information technology; c2021. Available from: https://www.instavm.org
5. Jain S, Jagtap VS. Vedic mathematics in computer: A survey. Int J Comput Sci Inf Technol. 2014;5(6):7458-7459.
6. Mathur RP. Application of Vedic mathematics in computer science. Int J Sci Res. 2024;13(2):990-993.
7. Kumar R, *et al*. Proof of Nikhilam sutra for multiplication by applying modular arithmetic. [Internet]; c2023. Available from: https://researchgate.net
8. Kuhlman D. A Python book: Beginning Python, advanced Python, and Python exercises; c2013. Available from: http://www.opensource.org/licences/mit-licence.php
9. What is Python? Executive summary. [Internet]. 2024 Jun 6. Available from: https://www.python.org
10. Kapoor SK. Vedic mathematics for all. India: Lotus Press Publishers and Distributors; c2015.
11. Thakur RS. Advanced Vedic mathematics. India: Rupa Publications; c2019.
12. Kumar CS. Vedic theorems based on Vedic mathematics sutras; c2024. Available from: https://www.researchgate.net
13. [Last accessed 2024 May 31]. Available from: https://www.researchgate.net