Euler Totient Function

Teachers Teaching with Technology™
Professional Development from Texas Instruments

Teacher Notes and Answers

7 8 **9** 10 11 12

TI-Nspire™ | Activity | Student | 120 min

Euler Totient Function

Teacher Notes:

A PowerPoint slide show is provided with this activity as an introductory presentation for students to watch and help them understand how the algorithm works. The slides work progressively through the number from 1 to n, capturing any numbers that are co-prime with the original number n.

There is no calculator command for the Euler Totient function, however there is a short cut approach using the prime factorisation of a number. Once students have completed their program, they use the prime factor approach and compare it to their program.

This coding activity also introduces the notion of a 'function' and 'sub-routine'. The Euler Totient function relies upon the 'highest common factor'. Students can use their HCF routine from the previous activity, or import the 'math' module and use the calculator's built-in HCF routine. The activity is written to utilise the previous activity to support the concept of 'chunking' (educational neuroscience)

TI-Codes Lessons:

Unit 1 – Skill Builder 1
⇩
Unit 4 – Skill Builder 1

Commands:

- input
- for (range)
- if
- print
- int (number types)
- def function
- while
- % (modular arithmetic)

Introduction

The Euler Totient Function for a whole number '*n*' counts the quantity of numbers that are co-prime up to the number *n*. To help understand this definition, consider the number 12.

We need to check which numbers: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12} have a factor in common with 12, these numbers are discarded leaving us with the numbers that are co-prime. This is summarised in the table below.

Whole Numbers < n	1	2	3	4	5	6	7	8	9	10	11	12
Highest Common Factor	1	2	3	4	1	6	1	4	3	2	1	12

There are 4 numbers where the highest common factor is 1, these numbers are co-prime with 12: {1, 5, 7, 11}. The Euler Totient function for 12 is therefore equal to 4, this can be written as: $\varphi(12) = 4$.

Author: P. Fox

Here is another example for the number 9.

Whole Numbers < n	1	2	3	4	5	6	7	8	9
Highest Common Factor	1	1	3	1	1	3	1	1	9

The Euler Totient function for 9 is therefore equal to 6, this can be written as: $\varphi(9) = 6$.

Question: 1.

Create some pseudo-code for the Euler Totient function.

Answer: (Sample)

Request input
Reset Counter = 0
Loop from 1 to n
  If HCF(n,1) = 1 Then < increase counter >
End Loop

Writing a Program

Instructions:

Start a new document; insert a new Python program.

Add Python > New

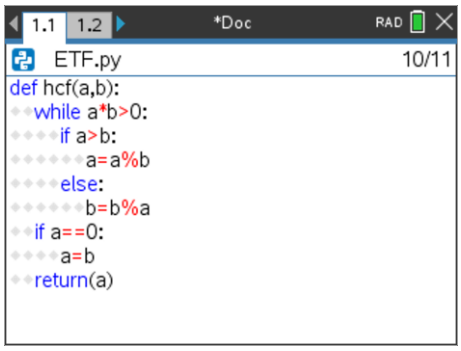
Call the program: ETF

The Euler Totient Function counts the quantity of numbers that are co-prime up to the specified number. When two numbers are co-prime their highest common factor is one, it therefore makes sense to use Euclid's algorithm to check the highest common factor. To do this efficiently, Euclid's algorithm can be defined as a function.

Built-ins > Functions > def function()

The function requires two parameters, the two numbers for which the highest common factor will be returned.

Euclid's algorithm for the Highest Common Factor can now be deployed through this function as per the activity on Euclid's algorithm. The only difference here is at the end of the function, 'return(a)', this is the value returned once the function has been called.



Note:

When the program runs, nothing happens with the function until it is called from the program.

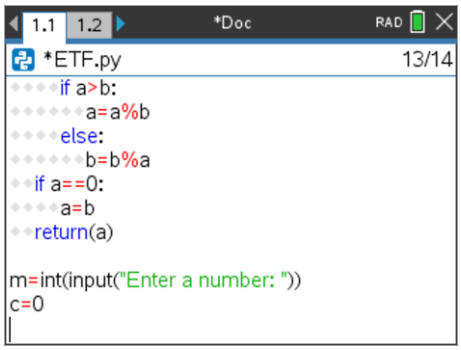
Author: P. Fox

The program needs to count the quantity of numbers that are co-prime with the selected (input) number. If two numbers are co-prime, their highest common factor is 1.

Start by requesting an input value and setting a counter equal to 0

m = int(input("Enter a number: ")

c = 0



Note:

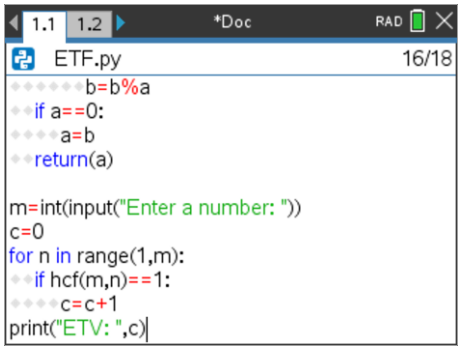
Variables 'a' and 'b' are used in the function, so it is best to avoid using them anywhere else in the program. Longer, more meaningful variable names can be used but keep them brief to avoid long lines of code.

The last part of the program is to scan the numbers from 1 to the designated value (m)* for numbers that are co-prime.

Each time one of these numbers is found, the counter increments by 1.

Note:

The loop will halt at 'm – 1', the hcf(m,m)≠1 so this last check would not change the counter value c.



Question: 2.

Check that your program produces the same results for the two worked examples, then try several others (by hand) and compare results.

Answer: The program returns the correct values for all numbers.

Question: 3.

Explore the Euler Totient function for prime numbers, what do you notice?

Answer: The Euler Totient function for a prime number 'n', returns the value n – 1.

Question: 4.

Determine the fraction: $\frac{n}{\varphi(n)}$ for the following values of n: 30, 60 and 90, comment on the results.

Answer: $\frac{30}{\varphi(30)} = \frac{30}{8} = 3.75$, $\frac{60}{\varphi(60)} = \frac{60}{16} = 3.75$ and $\frac{90}{\varphi(90)} = \frac{90}{24} = 3.75$

Other values for n with the same fraction (ratio) include: 120, 150, 180, 240, 270, 300 but 210 and 330 have very different results.

Teacher Notes: Students may sample a selection of multiples of 30 (as above) and jump to a conclusion too quickly if they miss 210 and 330. The clue lies in the prime factorisation of the multiples of 30.
30 = 2 x 3 x 5; 60 = 2^2 x 3 x 5; 90 = 2 x 3^2 x 5; 120 = 2^3 x 3 x 5; 150 = 2 x 3 x 5^2; 180 = 2^2 x 3^2 x 5; however, 210 = 2 x 3 x 5 x 7 which results in prime factorisation involving 4 prime factors, specifically a departure from: 2^a x 3^b x 5^c. Students should establish that for the Euler Totient function, the bases that are important not the exponents.

Author: P. Fox

Question: 5.

The number 100 can be expressed as: $2^2 \times 5^2$. Compare the Euler Totient value for 100 with the following calculation:

$$100 \times \left(1-\frac{1}{2}\right)\left(1-\frac{1}{5}\right)$$

Answer: $\varphi(100) = 40$. 100 x ½ x 4/5 = 40. The results are the same.

Question: 6.

The number 1125 can be expressed as: $3^2 \times 5^3$. Compare the Euler Totient value for 1125 with the following calculation:

$$1125 \times \left(1-\frac{1}{3}\right)\left(1-\frac{1}{5}\right)$$

Answer: $\varphi(1125) = 600$. 1125 x 2/3 x 4/5 = 600. The results are the same!

Question: 7.

Use the previous to questions to explore the prime factorisation approach to the Euler Totient function with the Euler Totient value determined by your program.

Answer: Results will vary, depending on the values that students chose, however in each case the answers will be the same.

Question: 8.

How does the prime factorisation approach to calculating the Euler Totient function explain your results to Question 4?

Answer: The prime factorisation for 30, 60 and 90 are of the form: $2^a \times 3^b \times 5^c$. The prime factorisation approach for calculating the Euler Totient function can be considered as two parts, the first part being the original number, the second, a combination of the prime factors (bases only). By dividing out the original number, we are only left with a calculation involving the prime factors, ignoring duplicity.

Question: 9.

Why does the 'short cut' approach to the Euler Totient function work?

Answer: Each prime factor removes the corresponding fraction of the remaining numbers.
Example, if 2 is a prime factor of some number 'n', then ½ of the numbers up to (and including n) will have a factor in common (2). Similarly, if 3 is a prime factor of 'n', then 1/3 of the remaining numbers will also have a factor in common with 'n'. The co-primes will be the complement of these calculations.

Author: P. Fox

Investigation

Re-write the Euler Totient function program to determine the Euler Totient function for a range of numbers, graph the results and explore any patterns.

Note: Use the ti-system import module to share data from the program with the TI-Nspire document.

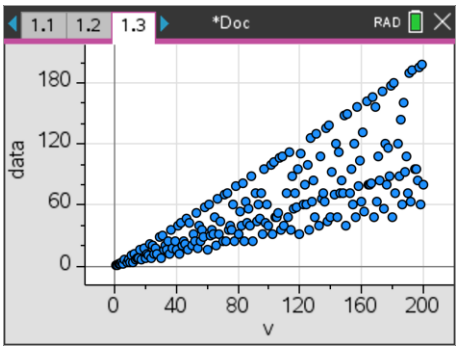
Answer: A sample of the Euler Totient program for a range of numbers is shown opposite. The program generates all the Euler Totient values from lower to upper and stores them in a list called: 'data'.

ETFL.py

```
from ti_system import *
#Euclids Algorithm for HCF
def hcf(a,b):
    while a*b>0:
        if a>b:
            a=a%b
        else:
            b=b%a
    if a==0:
        a=b
    return(a)

lower=int(input("Enter lowest number: "))
upper=int(input("Enter highest number: "))
data=[]
ns=[]
for n in range(lower,upper+1):
    print(n)
    ns.append(n)
for n in range(lower,upper+1):
    c=0
    for j in range(1,n+1):
        if hcf(n,j)==1:
            c=c+1
    data.append(c)
print(data)
store_list("etvs",data)
store_list("number",ns)
```

To generate a scatterplot, the whole numbers from 'lower' to 'upper' would need to be stored in a list, furthermore, the TI-System module would need to be installed so the variables from the Python shell can be shared to the TI-Nspire document variables.



There is a clear line of data above which there are no points. The points along this line are the prime numbers. $\varphi(p) = p - 1$, where p is prime.

There is also a 'line' of points assembled around $y = \frac{1}{2}(x-2)$. This collection of points corresponds to prime factorisations of the form: 2 x p, where p is prime.

With the exception of $\varphi(1) = 1$ and $\varphi(2) = 1$, $\varphi(n)$ is even. Why? This can be seen from the prime factorisation calculation method for the Euler Totient function. Consider *n* as even and then *n* as odd.

Another set of points of particular interest are those at the bottom of the graph:

1, 2, 3, 4, 6, 8, 10, 12, 14, 18, 20, 24, 30, 36, 42, 48, 60

The highly composite numbers are a subset of these numbers.

Number:	2	3	4	6	8	10	12	14
Euler Totient:	1	2	2	2	4	4	4	6
Prime Factorisation:	2	3	2^2	2×3	2^3	2×5	$2^2\times3$	2×7
Number:	18	20	24	30	36	42	48	60
Euler Totient:	6	8	8	8	12	12	16	16
Prime Factorisation:	2×3^2	$2^2\times5$	$2^3\times3$	$2\times3\times5$	$2^2\times3^2$	$2\times3\times7$	$2^4\times3$	$2^2\times3\times5$

Expressing each calculation using the prime factorisation method helps show why these numbers fall along the bottom of the graph.

Author: P. Fox