Euler Totient Function

Student Activity

7 8 **9** 10 11 12

TI-Nspire™ Activity Student 120 min

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: $\varphi(12) = 4$.

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.

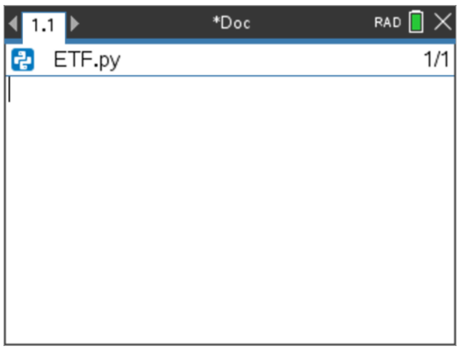
Writing a Program

Instructions:

Start a new document; insert a new Python program.

Add Python > **New**

Call the program: ETF



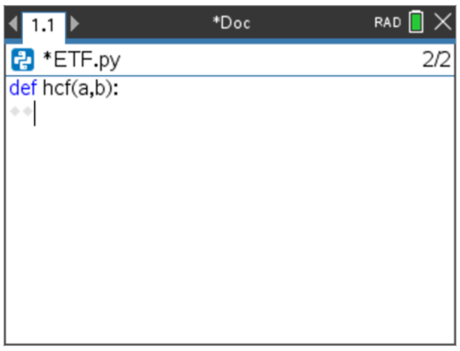
© Texas Instruments 2023. You may copy, communicate and modify this material for non-commercial educational purposes provided all acknowledgements associated with this material are maintained.

Author: P. Fox

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.

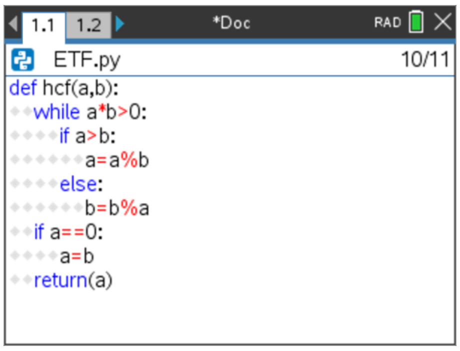


```
def hcf(a,b):
  
```

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.



```
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)
```

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.

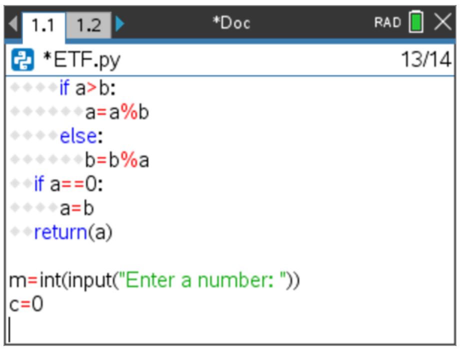
Start by requesting an input value and setting a counter equal to 0

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.



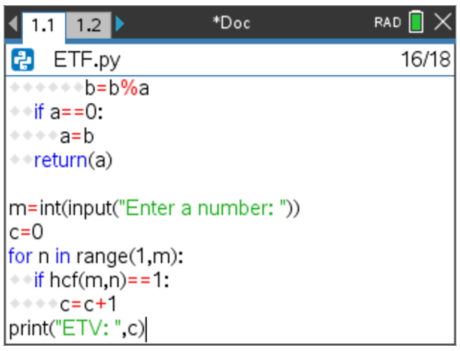
```
    if a>b:
      a=a%b
    else:
      b=b%a
  if a==0:
    a=b
  return(a)

m=int(input("Enter a number: "))
c=0
```

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', however the hcf(m,m)$\neq$1 so this last check would not change the counter value c.



```
      b=b%a
  if a==0:
    a=b
  return(a)

m=int(input("Enter a number: "))
c=0
for n in range(1,m):
  if hcf(m,n)==1:
    c=c+1
print("ETV: ",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.

© Texas Instruments 2023. You may copy, communicate and modify this material for non-commercial educational purposes provided all acknowledgements associated with this material are maintained.

Author: P. Fox

Question: 3.

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

Question: 4.

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

Question: 5.

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

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

Question: 6.

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

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

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.

Question: 8.

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

Question: 9.

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

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.

© Texas Instruments 2023. You may copy, communicate and modify this material for non-commercial educational purposes provided all acknowledgements associated with this material are maintained.

Author: P. Fox

Texas Instruments