Problems

Multiples of 3 and 5 If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3; 5; 6 and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000.

```
import time

def timing_function(some_function, arg = 1000):

    """
    Outputs the time a function takes
    to execute.
    """

    def wrapper():
        t1 = time.time()
        print(some_function(arg))
        t2 = time.time()
        return "Time it took to run the function: " + str((t2 - t1)) + " seconds"
    return wrapper()

def euler_01(n):
    accum = 0
    for i in range(n):
        if 0 in (i %3, i% 5):
            accum +=i
    return accum

def main():
    print(timing_function(euler_01))
    return None

main()

>>> 233168
Time it took to run the function: 0.05571794509887695 seconds
```

Even Fibonacci numbers Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:

$$1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.

```
def euler_02(ub):
    a = 0
    b = 1
    accum = 0
    while a < ub:
        if a % 2 == 0:
            accum += a
        a, b = b, a + b
    return accum

def main():
    print(euler_02(4000000))

main()

>>> 4613732
```

Largest prime factor The prime factors of 13195 are $5, 7, 13$ and 29.

What is the largest prime factor of the number 600851475143 ?

```
import time
import math
from arithmetic import (divisors, isPrime)

def timing_function(some_function, arg =  600851475143):
    """
    Outputs the time a function takes
    to execute.
    """
    def wrapper():
        t1 = time.time()
        print(some_function(arg))
        t2 = time.time()
        return "Time it took to run the function: " + str((t2 - t1)) + " seconds"
    return wrapper()

def euler_03(n):
    ls = divisors(n)[::-1]
    for i in ls:
        if isPrime(i): return i

def main():
    print(timing_function(euler_03))

main()

>>> 6857
Time it took to run the function: 0.1721358299255371 seconds
```

Largest palindrome product A palindromic number reads the same both ways. The largest palindrome made from the product of two 2-digit numbers is $9009 = 91 \times 99$.

Find the largest palindrome made from the product of two 3-digit numbers.

```
import time

def timing_function(some_function):
    """
    Outputs the time a function takes
    to execute.
    """
    def wrapper():
        t1 = time.time()
        print(some_function())
        t2 = time.time()
        return "Time it took to run the function: " + str((t2 - t1)) + " seconds"
    return wrapper()

def euler_04():
    maximum = -1
    for a in range(999,99, -1):
        for b in range(999,99, -1):
            p = a * b
            s = str(p)
            if s == s[::-1]:
                if p > maximum:
                    maximum = p
    return maximum

def main():
    print(timing_function(euler_04))

main()
```

```
>>> 906609
Time it took to run the function: 0.4687800407409668 seconds
```

Smallest multiple 2520 is the smallest number that can be divided by each of the numbers from 1 to 10 without any remainder.

What is the smallest positive number that is evenly divisible by all of the numbers from 1 to 20?

```
import time

def timing_function(some_function):
    """
    Outputs the time a function takes
    to execute.
    """
    def wrapper():
        t1 = time.time()
        print(some_function())
        t2 = time.time()
        return "Time it took to run the function: " + str((t2 - t1)) + " seconds"
    return wrapper()


def euler_05():
    def prop(n):
        for d in range(1, 21):
            if n % d > 0:
                return False
        return True
    n = 4*9*5*7*11*13*17*19
    while not prop(n):
        n += n
    return n


def main():
    print(timing_function(euler_05))

main()

>>>
232792560
Time it took to run the function: 0.030534744262695312 seconds
```

Sum square difference

The sum of the squares of the first ten natural numbers is,

$$1^2 + 2^2 + \cdots + 10^2 = 385$$

The square of the sum of the first ten natural numbers is,

$$(1 + 2 + \cdots + 10)^2 = 55^2 = 3025$$

Hence the difference between the sum of the squares of the first ten natural numbers and the square of the sum is $3025 - 385 = 2640$.

Find the difference between the sum of the squares of the first one hundred natural numbers and the square of the sum.

```
from timing import timing_function

def euler_06(n = 100):
    accum = 0
    accum2 = 0
    for i in range(1, n + 1):
```

```
        accum += i
        accum2 += i**2
    return accum**2 - accum2

def main():
    print(timing_function(euler_06))

main()

>>>
25164150
Time it took to run the function: 0.0718851089477539 seconds
```

10001st prime By listing the first six prime numbers: 2, 3, 5, 7, 11, and 13, we can see that the 6th prime is 13. What is the $10,001$st prime number?

```
from timing import timing_function
from arithmetic import isPrime

def euler_07(n = 10001):
    if n == 1:
        return 2
    count = 2
    i = 5
    prime = 3
    while count < n:
        if isPrime(i):
            prime = i
            count += 1
        i += 2
    return prime

def main():
    print(timing_function(euler_07))

main()

>>>
104743
Time it took to run the function: 0.27366018295288086 seconds
```

Largest product in a series
The four adjacent digits in the 1000-digit number that have the greatest product are $9 \times 9 \times 8 \times 9 = 5832$. Find the thirteen adjacent digits in the 1000-digit number that have the greatest product. What is the value of this product?

```
7316717653133062491922511967442657474235534919493496983520312774506326239578318016984801869478851843858615607891129494954595017379583319528532088055111254069874715852386305071569329096329522744304355766896489504452445231617318564030987111217223831136222989342338030813533627661428280644486645238749303589072962904915604407723907138105158593079608667017242712188399879790879227492190169972088809377665727333001053367881220235421809751254540594752243525849077116705560136048395864467063244157221553975369781797784617406495514929086256932197846862248283972241375657056057490261407972968652414535100474821663704844031998900088952434506585412275886668811642717147992444292823086346567481391912316282458617866458359124566529476545682848912883142607690042242190226710556263211111093705442175069416589604080719840385096245544436298123098787992724428490918884580156166097919133875499200524063689912560717606058861164671094050775410022569831552000559357297257163626956188267042825248360082325753042075296345
```

```
from timing import timing_function
from lists import product
```

```
def max_product(ls, n, start = 1):
    """
    Returns the greatest product  of n consecutive numbers in the list ls

    When the length of ls is less than n, returns a 'start' value (default: 1) value.
    This function is intended specifically for use with positive numeric values and may
    reject non-numeric types.
    """
    while (len(ls) >= n):
        start = max(product(ls[0:n]), start)
        return max_product(ls[1:], n, start)
    return start

def euler_08():
    ls = list()
    with open("euler_08.txt") as f:
        line = f.readline().rstrip()
        while line:
            ls += list(map(int, list(line)))
            line = f.readline().rstrip()
        return max_product(ls, 13)

def main():
    print(timing_function(euler_08))

main()

>>>
23514624000
Time it took to run the function: 0.06601810455322266 seconds
```

Special Pythagorean triplet

A Pythagorean triplet is a set of three natural numbers, $a < b < c$, for which,

$$a^2 + b^2 = c^2$$

For example, $3^2 + 4^2 = 9 + 16 = 25 = 5^2$.

There exists exactly one Pythagorean triplet for which $a + b + c = 1000$. Find the product abc.

```
from timing import timing_function

def euler_09():
    for a in range(1, 1001):
        for b in range(a, 1001):
            c = 1000 - a - b
            if b < c:
                if a**2 + b**2 == c**2:
                    return a*b*c
            else:
                break

def main():
    print(timing_function(euler_09))

main()

>>>
31875000
Time it took to run the function: 0.12091183662414551 seconds
```

Summation of primes

The sum of the primes below 10 is $2 + 3 + 5 + 7 = 17$. Find the sum of all the primes below two million.

```
from timing import timing_function
from arithmetic import isPrime

def euler_10():
    accum = 2
    for n in range(3, 2000000, 2):
        if isPrime(n):
            accum += n
    return accum

def main():
    print(timing_function(euler_10))

main()

>>>
142913828922
Time it took to run the function: 13.390980243682861 seconds
```

Exercise 11: Fuel Efficiency(13 Lines) In the United States, fuel efficiency for vehicles is normally expressed in miles-pergallon (MPG). In Canada, fuel efficiency is normally expressed in liters-per-hundred kilometers (L/100 km). Use your research skills to determine how to convert from MPGto L/100 km. Then create a program that reads a value from the user in American units and displays the equivalent fuel efficiency in Canadian units.

Exercise 12: Distance Between Two Points on Earth (27 Lines) The surface of the Earth is curved, and the distance between degrees of longitude varies with latitude. As a result, finding the distance between two points on the surface of the Earth is more complicated than simply using the Pythagorean theorem. Let $(t1, g1)$ and $(t2, g2)$ be the latitude and longitude of two points on the Earth's surface. The distance between these points, following the surface of the Earth, in kilometers is:

$$\text{distance} = 6371.01 \times \arccos(\sin(t_1) \times \sin(t_2) + \cos(t_1) \times \cos(t_2) \times \cos(g_1 - g_2))$$

The value 6371.01 in the previous equation wasn't selected at random. It is the average radius of the Earth in kilometers.

Create a program that allows the user to enter the latitude and longitude of two points on the Earth in degrees. Your program should display the distance between the points, following the surface of the earth, in kilometers.

Hint: Python's trigonometric functions operate in radians. As a result, you will need to convert the user's input from degrees to radians before computing the distance with the formula discussed previously. The math module contains a function named radians which converts from degrees to radians.