$p(10) =$

1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
0+0+0+0+0+0+0 0+0+0+0+0+0
+0+0+0+0+0+0+ +0+0+0+0+0+0+

Predicting Primes

0+0+0+0+0+0+0 0+0+0+0+0+0
+0+0+0+0+0+0+ +0+0+0+0+0+0+

By: Hans **Z**immer

0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0

= 29

Predicting Primes

By: **H**ans **Z**immer

I. Introduction

Hello dearest reader, and welcome to my essay!

Let me start with a slightly mischievous proclamation: what if I told you that there exists a *single* formula, that can give you any prime you want! Just input 1 into the formula and you get the first prime, input 2 and you get the second prime, input 3 and you get the third.

You might not believe me. After all, prime numbers are famously unpredictable. However, lets try to spot a pattern…

$$2, 3, 5, 7, 11, 13, 17 \ldots$$

Hmm… as it turns out, there's no **apparent** pattern here. But I assure you, that by the end of this essay, we'll have built such a formula by ourselves. So, let's begin, shall we?

II. What does it mean to "generate" a prime?

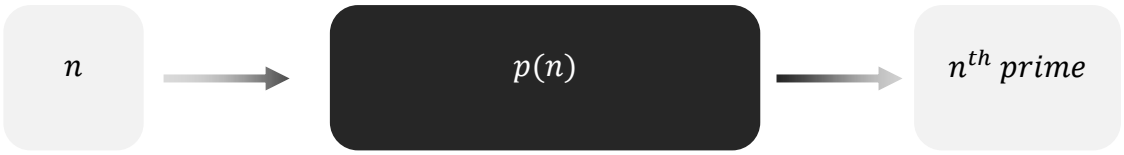
We can go through every number, check its prime factors, keep a counter, and increment that counter if the number is a prime until the counter is equal to n, and we can display the n^{th} prime, where n is prime the user wanted. Here's the code to do exactly that in Python:

```
n = int(input("Enter which prime you want: ")) + 1
count = 0
num = 0

while count < n:
    num += 1
    for i in range(2, num):
        if num % i == 0:
            break
    else:
        count += 1

print(num)
```

What we want is a function that takes in a number n and gives us the n^{th} prime. What if we turn Math into our own programming language?



III. Turing into the "Primes" Sherlock Holmes

We know, a number is prime if it has exactly two divisors. So, we can just check the number of divisors a number has to determine if it is prime or not.

Instead of using $\left\lfloor \left\lfloor \frac{k}{j} \right\rfloor - \frac{k}{j} \right\rfloor = \begin{cases} 0, & \text{if } j \text{ divides } k \\ -1, & \text{if } j \text{ divides } k \end{cases}$;

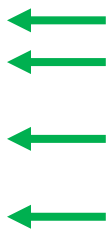
But instead of trying to check factors directly, let's try a novel approach. Let us consider the expression:

$$\frac{(j-1)!+1}{j}$$

Expression 1

Let us look at its result for a few values of j:

j	$\frac{(j-1)!+1}{j}$
1	2
2	1
3	1
4	1.75
5	5
6	20.16666667
7	103
8	630.125
9	4480.111111
10	36288.1



We can see that, our expression is only an integer when j is prime! *(except when j = 1)*! So, we have found a **prime detector**. We just need to convert this into a **prime generator**.

Formally, from Wilson's Theorem (see sources for proof), **Expression 1** proved to be an integer when j is 1 or prime; and not an integer, when j is composite, for all j. So,

$$\frac{(j-1)!+1}{j} = \begin{cases} \in \mathbb{Z}, & \text{iff } j \text{ is } prime \\ \notin \mathbb{Z}, & \text{if } j \text{ is } composite \end{cases}$$

Definition of Expression 1

Let's modify our function to give us 1 (if prime) and 0 (if not prime). This will make it much easier to work with. Let's add an **integer detector** on top of our prime detector.

On a seemingly unrelated note, let's take a look at the graph of cosine of πx.

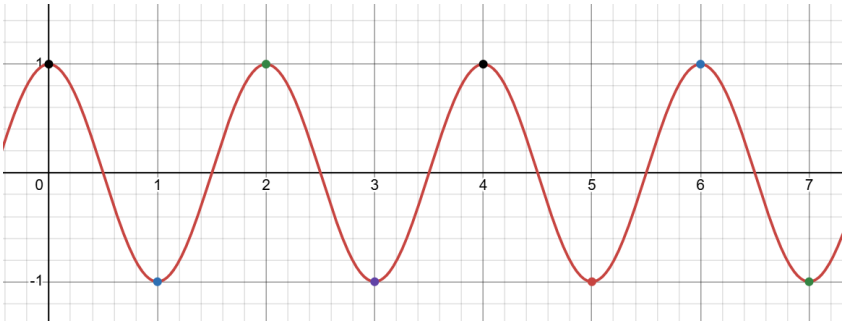


Fig 1: Graph of $\cos(\pi x)$

We can see this function, **spits out ± 1, when x is an integer**, and spits out a |number| less than $|\pm 1|$, when x is not an integer. So, if we square this, the possible numbers we can get will be between [0,1].

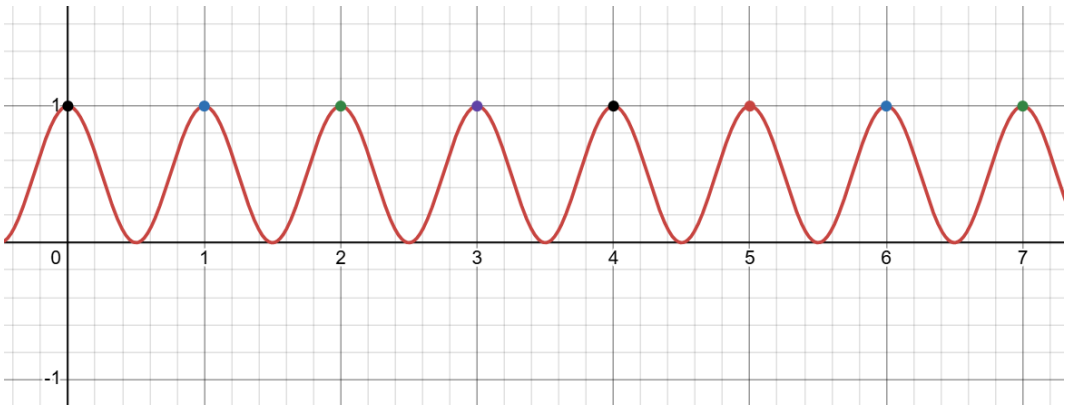


Fig 2: Graph of $\cos^2(\pi x)$

Now, if we floor this, we will get 0, if the number is 0 or ≤ 1;
And we will get 1, if and only if the number is an integer.

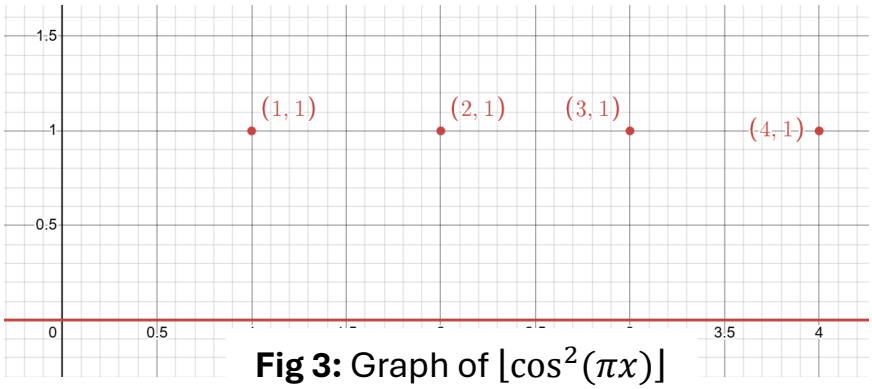


Fig 3: Graph of $\lfloor \cos^2(\pi x) \rfloor$

Therefore, we have found an integer detector!
Here's what we've done so far, compiled.

$$\left\lfloor \cos\left(\pi \, \frac{(j-1)! + 1}{j}\right)^2 \right\rfloor$$

Hence combining the prime detector and the integer detector, we get this expression that will give us 1, if and only if j is prime, otherwise it will give us 0.

We can sum this where j goes from 1 up through i, every time j is a prime, the expression contributes +1 to the sum. For example, taking i to be 10...

$$\sum_{j=1}^{10} \left\lfloor \cos\left(\pi \, \frac{(j-1)! + 1}{j}\right)^2 \right\rfloor = 5 = 4 + 1$$

i.e. This gives us the **(number of primes $\leq$ 10) + 1.**

In general, if we sum up to i, we get the (# primes $\leq i$) + 1.

What we have found is basically the inverse function of our goal of finding the n^{th} prime number. The idea now, is that if we want to know the 4^{th} prime, then we can just ask:

- "Is the number of primes up through 1 less than 4?" **# YES**
- "Is the number of primes up through 2 less than 4?" **# YES**
- …
- "Is the number of primes up through 6 less than 4?" **# YES**
- "Is the number of primes up through 7 less than 4?" **# NO**

The answer will be **NO** when that number, itself is the 4^{th} prime. Here we can clearly see that 7 is the 4^{th} prime.

We can loop through the sum taking different values of $i = \{1,2,3, \dots 6,7,8,9, \dots\}$ and perform the $\leq$ to give us a **YES / NO** answer, and stop once we get our first **NO.** This will tell us when we hit the n^{th} prime.

Note:

- For the sake of simplicity…
- I will represent $\sum_{j=1}^{i} \left\lfloor \cos\left(\pi \frac{(j-1)!+1}{j}\right)^2 \right\rfloor$
- as **(# of primes $\leq i+1$);** *(# means number)*

So, coming back to our strategy of checking until we get our first **NO** to get the n^{th} prime. How can we implement this "**loop and inequality check**" into our formula?

Well, let's take a look at another expression, $\left(\frac{x}{4+1}\right)^{\frac{1}{x}}$ and look at its graph. Note that $x \in \mathbb{N}$.

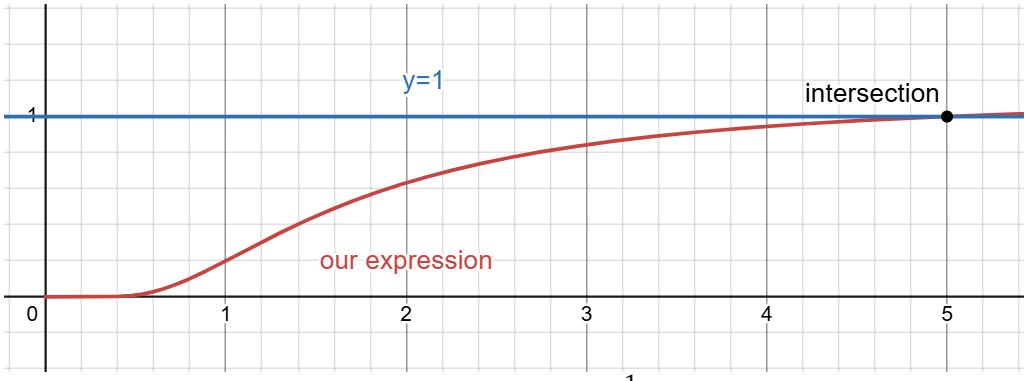


Fig 3: Graph of $(\frac{x}{4+1})^{\frac{1}{x}}$ and $y = 1$

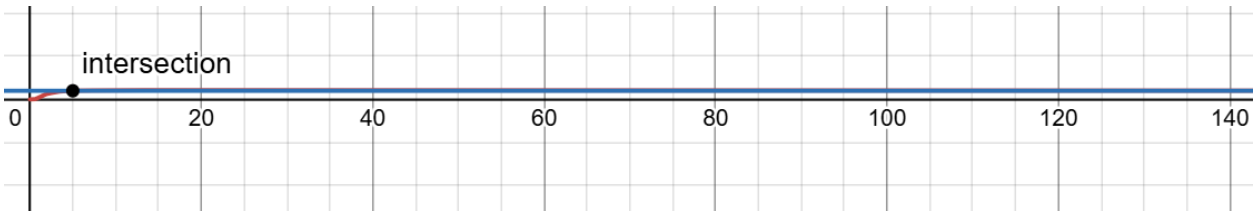


Fig 4: Graph of $(\frac{x}{4+1})^{\frac{1}{x}}$ zoomed out, there is only **one** intersection and its output always lies in the range [0,1]

Therefore, from the graph, $\left(\frac{x}{4+1}\right)^{\frac{1}{x}} = \begin{cases} \geq 1, \text{if } x > 4 \\ < 1, \text{if } x \leq 4 \end{cases}$ $x \in \mathbb{N}$

Which means that $\left\lfloor \left(\frac{x}{4+1}\right)^{\frac{1}{x}} \right\rfloor = \begin{cases} 1, \text{if } x > 4 \\ 0, \text{if } x \leq 4 \end{cases}$ $x \in \mathbb{N}$, see graph below

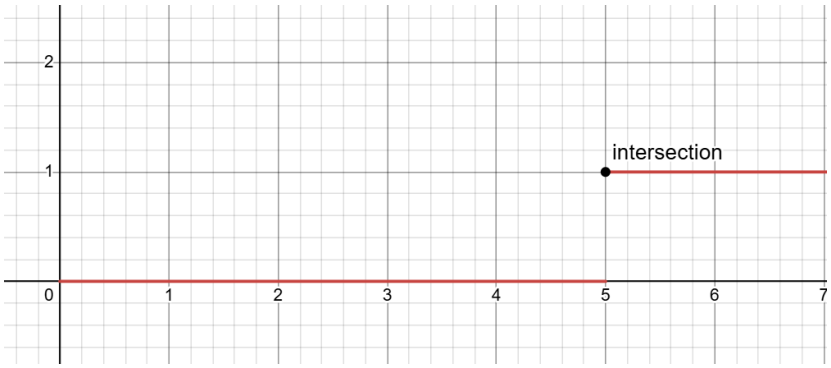


Fig 5:

Graph of $\left\lfloor (\frac{x}{4+1})^{\frac{1}{x}} \right\rfloor$

Inputting our # of primes counter into the expression…

We get, $\left\lfloor \left(\frac{n}{[\#\text{ of primes} \leq 10] + 1} \right)^{\frac{1}{n}} \right\rfloor = \begin{cases} 1, \text{if } n > (num \text{ of primes} \leq 10) \\ 0, \text{if } n \leq (num \text{ of primes} \leq 10) \end{cases}$

Since **(number of primes ≤ 10) + 1** is just an example, we can replace 10 with i, to generalize it.

So, $\left\lfloor \left(\frac{n}{[\#\text{ of primes} \leq i] + 1} \right)^{\frac{1}{n}} \right\rfloor = \begin{cases} 1, \text{if } n > (num \text{ of primes} \leq i) \\ 0, \text{if } n \leq (num \text{ of primes} \leq i) \end{cases}$

If you realize what we just did, you will notice that we have implemented an inequality without even using < or >!!

Further, our last expression is the same as:

$$\left\lfloor \left(\frac{n}{[\#\text{ of } primes \leq i] + 1} \right)^{\frac{1}{n}} \right\rfloor = \begin{cases} 1, \text{if } n^{th}\ prime > i \\ 0, \text{if } n^{th}\ prime \leq i \end{cases}$$

1. Since n **>** 4 (n = 5,6,…) , precisely when (n^{th} prime) > 10 (5^{th} prime = 11 > 10)
2. **OR** n > (# of primes ≤ 10) precisely when (n^{th} prime) > 10

Making i the subject, we get;

$$\left\lfloor \left(\frac{n}{[\#\text{ of } primes \leq i] + 1} \right)^{\frac{1}{n}} \right\rfloor = \begin{cases} 1, \text{if } i < n^{th}\ prime \\ 0, \text{if } i \geq n^{th}\ prime \end{cases}$$

The final step is to sum all of this where i goes from 1 to 2^n.

We can understand this through an example. Let us try to find the 4th prime using our formula.

$$\sum_{i=1}^{2^n} \left\lfloor \left(\frac{n}{(\#\text{ of primes } \leq i) + 1} \right)^{\frac{1}{n}} \right\rfloor$$

$$\sum_{i=1}^{16} \left\lfloor \left(\frac{4}{(\#\text{ of primes } \leq i) + 1} \right)^{\frac{1}{4}} \right\rfloor$$

Thus, for each value of $i <$ the 4th prime we get a +1. Hence the sum is 1 + 1 + 1 + 1 + 1 + 1 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 6 = **7** – 1 = 4th prime - 1

We do the sum till 2^n to go as high as necessary to ensure we get all the +1's possible. From **Bertrand's Postulate:** For every integer $m \geq 2$, there exists a prime p such that $m < p < 2m$. From these examples: (See sources aswell)

2 < **3** < 4
4 < **5** < 8
8 < **11** < 16
16 < **17** < 32

So, (# of primes $\leq 2^n$) $\geq n$, which implies that the n^{th} prime $\leq 2^n$. Accordingly, our expression is 1 less than the n^{th} prime, lets add 1 to our formula (see example above) and replace # of primes $\leq i$ + 1 with the original expression to get...

$$p(n) = 1 + \sum_{i=1}^{2^n} \left\lfloor \left(\frac{n}{\sum_{j=1}^{i} \left\lfloor \cos\left(\pi \frac{(j-1)! + 1}{j}\right)^2 \right\rfloor + 1} \right)^{\frac{1}{n}} \right\rfloor$$

What an absolute gargoyle! and in practice too, it is excruciatingly inefficient and tedious: A factorial within a sum **within** another sum.

What was the point of all this then, if it's so inefficient and not worth using?

Well.... Sorry, I kind of tricked you guys... The point **never *was*** to calculate the n^{th} prime. The point was to show that with some basic arithmetic, algebra, trigonometry, and a bit of number theory, you can quite **literally** engineer what ever you want using Mathematics itself, as the programming language!

A programming language that is so expressive that it can even find the n^{th} prime number if programmed to do so.

Arithmetic, Algebra, Trigonometry, and a pinch of Number Theory that have no intrinsic connection to primes, or divisibility come together to form a prime checker, an integer checker, and compute the number of primes up through i, and finally find the n^{th} prime number.

This goes to show the power that Mathematics wields when applied correctly, and shows that it is rightly called "The Language of the Universe".

Thank you all for coming on this incredible journey with me, together we have quite literally **built** a computer from scratch. We have implemented all of its features **only** using Mathematics, and nothing else.

Oh! And one last thing, this formula was discovered by **C.P. Willans** from the University of Birmingham.

If you wish to do so, try building an even & odd detector **and** a NOR and NAND gate (functions that take in n inputs of either 0 or 1: **NOR** – gives 1 if all n inputs are off, otherwise 0; **NAND** – gives 1 if at least one input is 0, otherwise 0) – just using Mathematics. If you have found the NOR and NAND functions, then congratulations. You can build a complete computer using Mathematics. :)

- **Miscellaneous:**

  To show it in action, I shall calculate the 10$^{\text{th}}$ prime number:

p(10)=1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1 +0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0 +0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0 +0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0 = 29

- **Sources:**

I. The Mathematical Gazette – "On formulae for the n^{th} prime number" – Birmingham University Press, Birmingham University – **C.P. Willans**

II. https://empslocal.ex.ac.uk/people/staff/rjchapma/courses/nt13/Wilson.pdf - **Wilson's Theorem & Its Proof**

III. https://math.stanford.edu/~ksound/Math155Spr12/Bertrand.pdf - **Bernard's Postulate & Its Proof**