Ordinal Numbers are a Little Off

Matt Draisey <matt@draisey.ca>

June 15, 2022

Cardinal and Ordinal Numbers

Put some apples on a table. A cardinal number answers the question of "how many?". There are *seven* apples — "seven" or "7" is a cardinal number. Now line them up left to right an pick the centre apple. An ordinal number answers the question "where is it". The centre apple is the *fourth* from the left — "fourth" or "4th" is an ordinal number. You can't ask the question "how many?" of the centre apple, *per se*, as it is just a single apple; there is no *fourness* associated with it. Rather, it is an apple that has three other apples to its left in the sequence of apples from left to right. If we count all the apples starting from the left and stopping at the centre apple then, including the centre apple in our count, we have four apples. In this sense, for every ordinal number there is an associated cardinal number. But we could have just as well counted the apples to the left of our centre apple and that would have answered the question "where is it?" with a count of three; this is just as precise and just as well defined as our inclusive count.

Natural Numbers and Arithmetic

For natural numbers (the numbers upon which we define addition, subtraction, multiplication & division) we don't really differentiate between cardinals and ordinals. How the natural numbers are to be used is part of the context. But in everyday usage we mostly associate arithmetic with cardinal numbers. An ordinal number is associated with the corresponding cardinal and thence to a natural number in an arithmetic calculation. Now zero is an essential tool in any algebraic expression of an arithmetic problem and zero terms do appear in arithmetic involving ordinals numbers. The trick to computing with ordinals is to only use factors which are the difference of ordinals — to use the ordinal "fourth" via natural number "4" we have to subtract from it the ordinal "first" via the natural number "1". The difference between two ordinal numbers is uniquely defined by the arithmetic difference of the corresponding natural numbers, independent of how we associate ordinal and cardinal numbers. Consider the subsequence of ordinal numbers 4th, 5th, 6th and 7th; the difference between the 7th and 4th ordinal must always be the natural number 3. If we say the 7th apple is associated with six apples to its left and 4th apple is associated with three apples to its left then the difference of six and three is still just the natural number 3.

It is interesting to note that if we count the steps between the 4th and 7th it is not an inclusive count; we must omit either the beginning or the ending ordinal in our count to get the correct number of steps as being three. This is easier to see if we consider the steps as being between ordinals: from *4th to 5th*, from *5th to 6th* and from *6th to 7th*; hence three steps in all. The most common sequence of ordinal numbers to deal with includes the first ordinal. There are six steps in the sequence 1st, …, 7th. In arithmetical calculations we almost always need to count the number of steps and so the n^{th} ordinal, in its traditional sense, would appear as a factor $(n - 1)$. All those *minus one* terms are a nuisance which a *zero-founded* system of ordinal numbers — such as we will introduce by the end of the document — would avoid. But first we must see some examples of traditional ordinal numbers in use, to get a sense of this.

Working Examples

Queues

Consider a queue for a bank teller. It takes a teller three minutes to deal with a customer. The head of a queue is identified by an ordinal as the first in the queue. The customer at the head of a queue doesn't have to wait at all. The fourth in the queue has to wait nine minutes for the three people ahead. The n^{th} person in the queue needs to wait $(n - 1) \times 3$ minutes.

Days of an Event

Consider a seven day conference starting on the fifteenth. The n^{th} day of the conference will be $(n - 1)$ days after the starting date; i.e. on the $(15 + n - 1)^{\text{th}}$ day of the month. Likewise, the nineteenth will be the fifth day of the conference: 15^{th} is the first day, the 19^{th} is four days after the 15^{th} and so is the fifth day of the conference. All these minus-ones and plus-ones are inconvenient.

Cyclical Phases of the Moon

The moon continuously evolves through phases in a month long cycle; a cycle that conventionally begins and ends with a new moon. Much as we define four cardinal points in a compass we define four *quarters of the moon*: the new moon, the first quarter moon (a waxing half-moon), the full moon and the third-quarter moon (a waning half-moon) — the second quarter (full) and fourth quarter (new) are never referred to as such. Note that it is conventional to start the cycle with the new moon (the fourth quarter moon) — this is a common pattern where the ordinal for the last in the cycle is elided with the *zeroth* ordinal and so begins the next cycle — twelve o'clock as a synonym for zero o'clock is the same.

For all other ways of expressing the phase of the moon — e.g. as a percentage — we drop the use of ordinal numbers.

Sequential Numbering of Days in the Gregorian Calendar

Let's look at the Gregorian and Julian calendars. Strictly speaking these will be the *proleptic* Gregorian calendar and a *proleptic* Julian calendar that may be applied to dates before the calendar was in use. In the proleptic Julian calendar from which the Gregorian calendar was derived, there is an extra day (February 29th) at the end of each four year period starting from March 1st of the year zero. The Gregorian calendar corrects this period every one hundred years and then again every four hundred years. We can use this periodicity to derive a formula to count the number of days from March 1st of the year zero of the proleptic Gregorian calendar. Using this formula we can determine the number of days between any two dates or reckon the day of the week for any given date.

Let's express a Gregorian calendar date as year y, month m and day d. In order to use the formula we need to number our months starting with March and define the year to start on March 1st. We will do the conversion from a modern year and month with a bit of Python code

```
if m >= 3:   # March through December
    m -= 2       # decrement month by two
else:        # January or February
    m += 10      # increment month by ten
    y -= 1       # "borrow from the next column" and decrement year by one
```

Now $m = 1$ for March, $m = 7, 8, 9 \,\&\, 10$ for September, October, November & December in accordance with their latin etymology, $m = 11$ for January and $m = 12$ for February. This reckoning of the year rolls over from one year to the next at the end of February to the beginning of March. Because we want the formula to express its value as an ordinal with March 1st of the year 0 being day 1 we need an initial term of 1 in our expression. The formula to number days is

$$1 + \underbrace{365y}_{\substack{\text{number of}\\ \text{days in}\\ \text{years past}\\ \text{excluding}\\ \text{February}\\ \text{29th's}}} + \underbrace{\left\lfloor \frac{y}{4} \right\rfloor - \left\lfloor \frac{y}{100} \right\rfloor + \left\lfloor \frac{y}{400} \right\rfloor}_{\substack{\text{number of February 29th's}\\ \text{in years past}\\ \textit{n.b. the Gregorian correction to the}\\ \textit{Julian calendar adds the terms in}\\ \textit{100 and 400}}} + \underbrace{30(m-1)}_{\substack{\text{in this year the}\\ \text{number of days}\\ \text{in prior months}\\ \text{excluding 31st's}\\ \text{of the month}}} + \underbrace{\left\lfloor \frac{3(m-1)}{5} + \frac{1}{2} \right\rfloor}_{\substack{\text{in this year the number}\\ \text{of prior 31st's of the}\\ \text{month}}} + \underbrace{(d-1)}_{\substack{\text{number of}\\ \text{days earlier}\\ \text{in this}\\ \text{month}}}$$

This expression uses *flooring* (sometimes called *rounding-down*) brackets ($\lfloor\cdot\rfloor$) where the *floor* is the largest integer less than or equal to the bracketed value. Adding a half inside a flooring bracket returns a *rounded* result that is the closest integer to the given value. Expressing a division inside a flooring bracket is a convenient way to show the result of taking a dividend and a divider, computing the quotient and remainder, and then throwing the remainder away. Python has an explicit *floored division operator* (//) with which we may code this as

$$1 + \overbrace{\texttt{365*y}} + \overbrace{\texttt{y//4 - y//100 + y//400}}^{\lfloor\frac{y}{4}\rfloor - \lfloor\frac{y}{100}\rfloor + \lfloor\frac{y}{400}\rfloor} + \overbrace{\texttt{30*(m-1)}} + \overbrace{\texttt{(6*(m-1) + 5)//10}}^{\left\lfloor\frac{3(m-1)}{5}+\frac{1}{2}\right\rfloor} + \overbrace{\texttt{(d-1)}}$$

Combining this with the snippet above gives us a complete function definition

```
def day_number_from_0000_03_01(y, m, d):
    if m >= 3:
        m -= 2
    else:
        m += 10
        y -= 1
    return 1 + 365*y + y//4 - y//100 + y//400 +\
           30*(m-1) + (6*(m-1) + 5)//10 + (d-1)
```

Plugging in $y = 0$ for year 0, $m = 3 \mapsto m = 1$ for March and $d = 1$ for the first of the month we get zeroes in all the terms but for the initial 1. See how we say "day one" and the "first day" interchangeably. The association between the cardinal and ordinal is ingrained into our everyday usage.

Now the formula has completely independent terms in year y, month m and day d because the added February 29th occurs at the end of a four year period and therefore needs no special handling. But the period starts on March 1st of the nonexistent year zero — the first *"Year of our Lord"* is the year 1AD and the year before it is the year 1BC with no year in between. For February 29th to be at the end of every fourth year it should have occurred on the day before March 1st of the years 1, 5, 9 and so on. But leap years are always divisible by 4. The framers of the calendar didn't treat year y like an ordinary ordinal number. The formula does treat m and d as ordinals. Every time we use month m or day d in the formula in order to count something we need to subtract 1 from it — traditional ordinals starting at 1 are fundamentally at odds with simple counting formulas and are always off by one. Only y, which isn't really an ordinal in this context, appears in the formula without being decremented before use.

Consider how we might optimize the expression

```
1 + 365*y + y//4 - y//100 + y//400 + 30*(m-1) + (6*(m-1) + 5)//10 + (d-1)
```

The 1's at either end of the expression cancel each other leading to simplified code. Expressions in ordinals often have cancelling terms of 1 which simplifies the calculation but serves to obfuscate the coding. Tweaking how we assign a number to months in the intermediate value for m further obfuscates the expression in the following Python function definition:

```
def day_number_from_0000_03_01(y, m, d):
    if m >= 3:
        m -= 3
    else:
        m += 9
        y -= 1
    return 365*y + y//4 - y//100 + y//400 + 30*m + (6*m + 5)//10 + d
```

This bit of code is difficult to interpret owing to an inconsistent treatment of ordinal numbers. By convention month and day are always represented by ordinals — in modern usage, very few other numeric variables are. And this particular arithmetic problem, the sequential numbering of days, is notorious for being implemented incorrectly. The full formula as stated at first isn't that hard — that years have to be treated differently than months and days is easy enough to remember — but the more efficient optimized code we finally developed is a trap for the unwary.

Centuries, Decades and Years?

The nineteen-hundreds are the years 1900 through 1999. We also call this the twentieth century. Or is the 20th century defined as the years 1901 through 2000 so as to agree with the first century being the years 1 through 100? The nineties are the years 1990 through 1999 which may or may not be the tenth decade of the twentieth century — it does, at least, overlap by nine years. If we care about precision we prefer terms like the nineteen-hundreds as opposed to the twentieth century.

To avoid year 2K bugs many existing computer databases had an additional two decimal digits fitted into the old records. To maintain the alignment of existing fields these digits would be wedged where they fit rather than where they logically belonged. The implicit value of 19 for the first two digits of the year was replaced with an explicit 19. By the time year 2000 rolled around there was space in the database to properly encode dates. This association of number 19 with the twentieth century and number 20 with the twenty-first is something we take for granted.

Ages

When we are born we enter our first year of life, but our age is not 1. A ten year old thing is in its eleventh year of existence. Using ordinals to refer to age doesn't occur in normal usage.

Ordinal Temporal Hours Superseded by Mechanical Clocks and Cardinal Equinoctial Hours

As another example let's look at time of day. The word "*hour*" originally referred to one-twelfth of a day or a night. Hours were denoted by simple ordinal numbers and were rarely subdivided. The first hour of the day would begin at sunrise and the twelfth would end at sunset with the night hours following a similar cycle. As the length of time from sunrise to sunset varies throughout the year so does the length of the daylight hour. Midday, at the end of the sixth daylight hour and the beginning of the seventh, and midnight, likewise delineated, are the only hours that we would consider to be *fixed* according to our modern sensibility.

The *equinoctial hour*, one twenty-fourth of a midday to midday period, was meant to be an unvarying measure of time and a *minute* one-sixtieth of an equinoctial hour. The adoption of the equinoctial hour was driven by a need for greater precision in timekeeping. The first minute of the first hour begins at midday, at twelve o'clock (zero o'clock). Now consider what happens when the clock reaches one hour after midday — it is much more sensible and convenient to denote zero hours and fifty-nine minutes on the clock (0:59 or 12:59) be immediately followed by one o'clock (1:00) than have the sixtieth minute of the first hour (1^{st}:60^{th}) be immediately followed by the first minute of the second hour (2^{nd}:1^{st}). The conflation of twelve o'clock and zero o'clock is simple by comparison.

Other Units of Measurement

As opposed to time or age, where ordinals are hard interpret but not totally alien, using ordinals to refer to distances or weights simply never occurs. The things measured are analogous to time but ordinals are almost never used to denote continuous quantities, even though we use discrete units in practice. Consider adding weights to balance scale one ounce at a time and stopping when you reach the seventh ounce of the second pound — this should be the same as telling you to stop at 1 lb. 6 oz. And telling you to weigh one pound by stopping to add weights to the balance at the second pound is nominally the same as telling you to stop adding weights to the balance at the first ounce of the second pound — this is utterly perverse.

Subdivision into smaller units is tricky when ordinals start at "*first*". Measuring off to the first inch of the second foot is the same as measuring off one foot in length. Counting the number of inches or the number of feet and inches is relatively straightforward and doesn't require any mental gymnastics. The units differ in size but the addition of them is not conceptually difficult. Likewise, measuring a distance by pacing it off or counting units sequentially is how we must often measure distances, yet ordinals numbers are never used in these situations because having to start at the "*first*" unit before we move at all leads to all kinds of trouble, and mixing units of different sizes leads to the craziness we have already mentioned.

Ordinal Numbers Themselves

Even the words we use for ordinal numbers don't really capture how they can built from smaller strides. Instead an ordinal is named according to its associated cardinal. The thirtieth ordinal number is the tenth ordinal in the third stride of ten ordinal numbers. The thirty-first ordinal number is the first ordinal in the fourth stride of ten. The words we use to denote ordinal numbers come in units of tens, hundreds and thousands yet the ordinals themselves are just as awkward to describe in these divisions as are lengths, weights and times.

Other Examples

Multidimensional Arrays in Computer Memory

Consider how a two dimensional array of double-precision floating-point numbers are laid out in contiguous memory. Each `double` takes 8 bytes of memory. In Pascal we could declare a variable of type "`array [i0..i1, j0..j1] of double`". The multidimensional array type is just a shorthand way of saying "`array [i0..i1] of array [j0..j1] of double`". The index for the first dimension can be any integer in the inclusive range `i0..i1` and so can take one of $\texttt{ilen} = \texttt{i1} - \texttt{i0} + 1$ different values. Likewise for $\texttt{jlen} = \texttt{j1} - \texttt{j0} + 1$. The total size of the array will be $\texttt{ilen} \times \texttt{jlen} \times 8$ bytes. We will use a postfix *address-of* operator "↓" so that the notation "`a`↓" will get the base address of the whole array "`a`" in memory, the notation "`a[i]`↓" will get the base address of a subarray of type "`array [j0..j1] of double`" consuming $\texttt{jlen} \times 8$ bytes of memory and the notation "`a[i, j]`↓" or "`a[i][j]`↓" will get the address of an individual entry of 8 bytes.

Within each "`array [j0..j1] of double`" subarray getting to the individual element indexed by `j` is simply a matter of hopping over any preceding `double`'s. There are $\texttt{j} - \texttt{j0}$ elements which precede `j` within the subarray so getting to the element requires multiplying $\texttt{j} - \texttt{j0}$ by 8 to get the count of bytes to hop over. Likewise finding the base address of `a[i]` is a matter of hopping over the $\texttt{i} - \texttt{i0}$ preceding "`array [j0..j1] of double`" subarrays. We combine the two calculations to find the address of an element indexed by `i` and `j`.

$$\texttt{a[i]}{\downarrow} = \texttt{a}{\downarrow} \; + \; \underbrace{(\texttt{i} - \texttt{i0})}_{\text{number of subarrays before this}} \times \underbrace{(\texttt{jlen}{\times}8)}_{\text{size of a subarray in bytes}}$$

$$\begin{aligned}\texttt{a[i, j]}{\downarrow} = \texttt{a[i][j]}{\downarrow} &= \texttt{a[i]}{\downarrow} \; + \; \underbrace{(\texttt{j} - \texttt{j0})}_{\text{number of }\texttt{double}\text{'s before this}} \times \underbrace{8}_{\text{size of a }\texttt{double}\text{ in bytes}} \\ &= \texttt{a}{\downarrow} \; + \; (\texttt{i} - \texttt{i0}) \times (\texttt{jlen}{\times}8) \; + \; (\texttt{j} - \texttt{j0}) \times 8 \\ &= \texttt{a}{\downarrow} \; + \; \big((\texttt{i} - \texttt{i0}) \times \texttt{jlen} \; + \; (\texttt{j} - \texttt{j0})\big) \times 8\end{aligned}$$

The C language requires array indices to start with zero. An array with the same memory layout would be declared to be of type "`double[ilen][jlen]`". We understand that the `ilen` number of components are indexed with the inclusive range $0, ..., (\texttt{ilen} - 1)$. Indeed, the association between a number n and its set of predecessors $\{0, ..., (n-1)\}$ is a common pattern in programming. We use the C-style prefix *address-of* operator "`&`" so that "`a`" by itself is the base address of the array in memory and the other terms "`&a[i]`" and "`&a[i][j]`" are as you would expect from the previous demonstration.

$$\texttt{\&a[i]} = \texttt{a} + \texttt{i} \times (\texttt{jlen}\times 8)$$

$$\begin{aligned}\texttt{\&a[i][j]} &= \texttt{\&a[i]} + \texttt{j} \times 8 \\ &= \texttt{a} + \texttt{i} \times (\texttt{jlen}\times 8) + \texttt{j} \times 8 \\ &= \texttt{a} + \big(\texttt{i} \times \texttt{jlen} + \texttt{j}\big) \times 8\end{aligned}$$

Always indexing from zero simplifies array address calculations in machine code and speeds up all array computations. Most modern computer languages follow C's conventions and always start their arrays at index zero even when performance considerations aren't an issue. Python is an interpreted language which follows the C convention for indexing — this style of indexing is easier to reason about and makes it more likely you will produce logically correct code.

Computer programmers quickly learn how hard it is to write correct code involving ordinal numbers. The first high-level programming languages like Fortran and COBOL indexed arrays starting at *one* thinking this would make life easier for coders. Later languages index their arrays starting at *zero* — most programmers like to pretend ordinal numbers don't exist at all — converting ordinals as needed at input time (subtracting 1) and output time (adding 1) and keeping the internal logic pristine.

Inclusive-Exclusive Number Ranges in Modern Computer Languages

Look at how Python tales a slice of a single dimensional array `a[i0:i1]`; this is a list of all the elements `a[i0]`, `a[i0+1]`, ..., `a[i1-1]`; it expresses an inclusive bound at the bottom and an exclusive bound at the top of the range. The indices alone can be generated by the expression `range(i0, i1)`. This is different to the inclusive ranges that Pascal (an earlier language) used to express array types. The number of elements in the Python range is just $\texttt{i1} - \texttt{i0}$.

The slice `a[:i]` is a synonym for `a[0:i]` and `range(i)` is a synonym for `range(0, i)`, both contain exactly `i` elements. The first such slice is just the empty slice `a[:0]` of zero elements, the second is the slice `a[:1]` of one element, the third is the slice `a[:2]` of two elements and so on.

As sets of natural numbers the expression `range(i0, i1)` can be considered the predecessors of `i1` less the predecessors of `i1`. For natural numbers $\texttt{i0} < \texttt{i1} < \texttt{i2} < \texttt{i3}$ the union of `range(i0, i1)` and `range(i2, i3)` is the same as `range(i0, i3)` taking away `range(i1, i2)`.

There is an elegence here, a logical structure just beyond our grasp. It should be familiar but it isn't. The historical definition of ordinal numbers is to blame.

Can We Imagine using a New Style of Ordinals?

In the arithmetic problems, as we have demonstrated, a count of predecessors of an ordinal is the key to effective calculation. This suggests a *natural* correspondence between ordinals and cardinals which is at odds with the word formation laws which create the word “fourth” from the word “four” while giving it a meaning, via an inclusive count, that requires it to have only three predecessors.

The words “first”, “second”, and “third” have unassailable meanings — the words “three” and “third” are clearly cognates — word formation laws then give us “four” and “fourth”, “five” and “fifth” — these strongly prejudice how we choose to count using ordinal numbers. Our prejudice for decimal numbers is similar; counting in tens, hundreds and thousands is a cultural phenomenon which is hard to shake off despite there being a rich history of other number bases being used, something which can be surprising for us to learn in later life. Likewise, cardinal and ordinal numbers are part of the natural language we have known since infancy and it is hard to imagine that the association between “one” and “first”, “four” and “fourth”, “twenty-seven” and “twenty-seventh” or the association between any cardinal and its corresponding ordinal could be arbitrary and based on the history of numbering systems rather than being fundamental and unassailable.

Even were we to recognize that counting and indexing have separate demands upon a numbering system we tend to assume that a modern logical reconstruction of cardinal and ordinal numbers would necessarily recapitulate the numbers as we know them. But this is not so. We need to recognize that the word formation laws by which we refer to ordinal numbers are a very old mistake, based on an reasonable but suboptimal association between the cardinal “one” and the ordinal “first”.

Before the invention of the cardinal number “zero” this was inevitable. In our indexing of apples the leftmost apple has zero apples to its left but, using a traditional inclusive count, we would number this as apple number one. By the start of the thirteenth century, when Fibonacci introduced the cardinal number “zero” to Europe, the greater part of the world’s population now had a better way to name ordinal numbers, which associates the initial ordinal with the number zero. However, no such general reform of ordinal numbers has taken place. Instead a piecemeal replacement of ordinal numbers with *specialized numbers* has invaded and complicated our lives.

The Modern Mathematical and Logical Construction of Ordinals

A Zero-Founded System

To logicians, mathematicians, philosophers, or anyone trying to grasp the infinite, the modern definition of ordinal numbers is a recursive one starting at *zero*. Each succeeding ordinal number is defined in terms of its predecessors. This construction is necessary for a logically consistent treatment of infinite ordinals but, as we have seen, it greatly simplifies our treatment of ordinals in the finite realm.

Set theory is a logical framework with a concept of containment but none of number. We can use set theory to define the ordinal number 0 as the empty set $\{\}$; then ordinal number 1 is defined as the set of its predecessors $\{0\}$, ordinal number 2 as the set of predecessors $\{0,\ 1\}$ and so on. The set of all such ordinal numbers is called ω and can be considered an ordinal number in its own right (it is the set of its predecessors) and, as such, is the first infinite ordinal. For our purposes, we limit our construction to finite ordinals and finite sets only. Recursively defining arithmetic on these *numbers* is intricate but straightforward. In this construction cardinal numbers, corresponding to the number of elements in a set, are just ordinal numbers used in a different context.

Word Formation and the Need for New-Style Zero-Founded Words

So the first, second and third ordinals should be associated with the cardinals *zero*, *one* and *two*. If "*zeroth*" were to be the first ordinal then, redefining existing words, "*first*" would need be the second ordinal, "*second*" would need be the third ordinal, "*fourth*" would need be the fifth ordinal and … there's a problem here. To avoid confusion we need some new words for modern *new-style* ordinal numbers, words different from the existing *old-style* ordinals so as not to utterly confuse ourselves.

We will propose word formation laws for new ordinals that are simple as possible. The columns shown are of old-style ordinals, the number of predecessors (a cardinal) and new-style ordinals respectively.

first	0	zeroër	eleventh	10	tenner	twenty-first	20	twentier
second	1	oner	twelfth	11	elevenner	twenty-second	21	twenty-oner
third	2	twoër	thirteenth	12	twelver	twenty-third	22	twenty-twoër
fourth	3	threër	fourteenth	13	thirteener	twenty-fourth	23	twenty-threër
fifth	4	fourer	fifteenth	14	fourteener	twenty-fifth	24	twenty-fourer
sixth	5	fiver	sixteenth	15	fifteener	twenty-sixth	25	twenty-fiver
seventh	6	sixer	seventeenth	16	sixteener	twenty-seventh	26	twenty-sixer
eighth	7	sevenner	eighteenth	17	seventeener	twenty-eighth	27	twenty-sevenner
ninth	8	eighter	nineteenth	18	eighteener	twenty-ninth	28	twenty-eighter
tenth	9	niner	twentieth	19	nineteener	thirtieth	29	twenty-niner

Here we form an ordinal by adding an "*-er*" on the end of its corresponding cardinal. So we can write 0er, 1er, 2er, etc. We reserve the old words to represent fractions: *one third*, *five sixteenths*, *one thirty-second* etc. Note that speakers of other languages have exactly the same problems as we do in English with inappropriate word formation rules. Old-style ordinals are an international problem.

1st	0	0er	6th	5	5er	11th	10	10er	16th	15	15er	21st	20	20er	26th	25	25er
2nd	1	1er	7th	6	6er	12th	11	11er	17th	16	16er	22nd	21	21er	27th	26	26er
3rd	2	2er	8th	7	7er	13th	12	12er	18th	17	17er	23rd	22	22er	28th	27	27er
4th	3	3er	9th	8	8er	14th	13	13er	19th	18	18er	24th	23	23er	29th	28	28er
5th	4	4er	10th	9	9er	15th	14	14er	20th	19	19er	25th	24	24er	30th	29	29er

Computer scientists, physicists and others in the hard sciences don't see word usage as being terribly important even though they are happy to use ordinals in their modern sense. They don't really care that the first law of thermodynamics comes after the zeroth law. The zeroth law is a twentieth century addition to the existing three laws of thermodynamics developed in the nineteenth century. But even the existence of the word "zeroth" is unfortunate as it requires an inconsistency somewhere else. The cure to this inconsistency is to not just banish "zeroth" but also "first", "second", "third", "tenth" and "twentieth". Or, to put it another way

> They [computer scientists, physicists, …] don't really care that the oner law of thermodynamics comes after the zeroër law. The zeroër law is a nineteener century addition to the existing three laws of thermodynamics developed in the eighteener century.

Expressions like "*the twentier century*" for the years 2000 through 2099 could make life so much easier for the rest of us. It is time for a general reform of ordinal numbers to take us through the twentier century.

The Final Word

As we have seen, old-style ordinal numbers are gradually being replaced by sequence numbers beginning with zero, numbers which are modern ordinals in all but name. This transition has been going on for hundreds of years — we need to speed that up. In this case the words "*first*" and "*second*", "*tenth*" and "*twentieth*", "*hundredth*" and "*thousandth*" are a hindrance to understanding because of inappropriate word formation laws which associate the natural number 10 with the word "*tenth*" which by history and precedent has only nine predecessors despite logical consistency and arithmetic convenience demanding it have ten.

Our suggested word-formation rule of appending an "*-er*" suffix to associate a modern ordinal with its proper cardinal is functional but dull — there may be other more aesthetically pleasing word-formation rules which we have not dreamt of.

Imperial units are history in most parts of the world. As are non-decimal forms of currency. Children aren't taught these archaic forms. Traditional ordinal numbers deserve to die a similar fate.