Pronunciation of the integers with full use of the place value system

Thomas Colignatus
http://thomascool.eu
May 9 & 17 & June 28 2018

Abstract

Kids in kindergarten and Grade 1 live in a world of sounds so that the pronunciation of numbers is important. When they are learning to read and write, the co-ordination of (i) sounds, (ii) numerals and (iii) written words is important. We already have the place value system fully in the numerals but not yet in pronunciation and written words. The following provides for this. The definition should become an ISO standard, though the notebook with package is quite simple because of the nature of the issue. The notebook with package provides an implementation and transliteration for English, German, French, Dutch and Danish, while other languages might employ *Mathematica*'s IntegerName and WordTranslation without transliteration. Four levels in the curriculum are recognised for which routines are provided: (1) sounds, codified by words, (2) learning the numerals, (3) advanced: numerals in blocks of three digits, such that 123456 = {1 hundred, 2 ten, 3} thousand & {4 hundred, 5 ten, 6}, with the comma pronounced as "&" too, and (4) accomplished: 123 thousand 456 pronounced in above place value manner. The traditional pronunciation has level -4.

Keywords

mathematics education, place value system, pronunciation, Common Core, Mathematica, Wolfram language, programming, package

MSC2010

97M70 Mathematics education. Behavioral and social sciences

Cloud

This notebook with package (updated version) is also available at:

(1) http://community.wolfram.com/groups/-/m/t/1334793

(2) https://www.wolframcloud.com/objects/thomas-cool/MathEd/2018-05-09-Pronunciation-of-integers.nb

(3) for this notebook with package: https://zenodo.org/record/1244008 or DOI 10.5281/zenodo.1244008

(4) for the PDF: https://zenodo.org/record/1244063 or DOI 10.5281/zenodo.1244063

I have not seen this implementation of pronunciation elsewhere (except in Chinese though without &), so please refer to these locations so that others can find the full documentation.

Contents

Start (subsection for the initialisation cell with the package)

1. Introduction

1.1. Key example

A picture says more than a thousand words:

$$25 = 2 \times 10 + 5 = \text{two·ten \& five}$$

The default is Speak → True.

```
PlaceValue[25]
```

two·ten & five·

The connectives "&" and "·" are used to codify the sound, and differ from the operators "plus" and "group" (multi-plus, repeat, times), since + and × have commutation, association and distribution.

The center dot (·) is unpronounced. The trailing center dot is deliberate and indicates the ghost of the departed one. The strict use of the place value system is that 1 is actually 1 of 1 (also distinguishing numbers and digits). Normally we simplify, but we should be able to show how the system actually works.

```
PlaceValue[25, Speak → False, Simplify → False]
```

two·ten & five·one

The ampersand (&) will surprise native speakers of English but derives from quite some consideration. German ("und"), Dutch ("en") and Danish ("og") have the "&" connective and for good reason. See the discussion of the connectives below.

```
PlaceValueTable[25, Speak → False]
```

Integer	Place Value	PV Digits	Traditional
25	two·ten & five·	2 · 10 & 5	twenty five

```
PlaceValueTable[25, Language → "Danish", Speak → True]
```

Integer	Place Value	PV Digits	Traditional
25	to·ti & fem·	2 · 10 & 5	femogtyve

1.2. Kids live in a world of sounds

A number consists of sound, numeral, word (that records the sound). Kids in kindergarten and Grade 1 live in a world of sounds so that the pronunciation of numbers is important. When they are learning to read and write, the co-ordination of (i) sounds, (ii) numerals and (iii) written words (subvocalisation but still sounds) is important. We already have the place value system fully in the numerals but not yet in pronunciation and words.

The place value system puts digits in positions, with the digit being the *weight* and the position the *place value* (in our case a power of 10). The traditional pronunciation has these breaches upon the full use of the place value system:

- On order: nine-teen instead of ten & nine. (Rule: speak the highest place value first.)
- On weights: twenty instead of two·ten. (Rule: speak the weight of the place value.)

We do not have to assume that kids must learn *formal* multiplication (grouping) and the table and the powers of 10 before they can work with the numbers and the place value system. But it is another thing to hide the very structure that is the object of learning. It is better to use the full place value system, which will support the involved learning of arithmetic.

The US Common Core State Standards (2018) rightly wants for kindergarten: “Work with numbers 11-19 to gain foundations for place value: CCSS.Math.Content.K.NBT.A.1.” They do not say that they implement only a *partial* and not the *full* place value system.

PM. The West reads and writes from left to right, while the numbers came from a region in India that read and wrote from right to left. We can leave this phenomenon as it is because there is advantage in beginning the pronunciation with the highest place value with nonzero weight.

1.3. What this notebook with package provides

This notebook with package provides an implementation for English, German, French, Dutch and Danish. For these languages there is a transliteration in English. For other languages *Mathematica*'s **IntegerName** and **WordTranslation** can be used for an automated translation, see the examples for Italian and Spanish, but then without transliteration. Only needed are 18 terms, for sign, “&”, ten digits and the powers of 10 to a million (for our main application). The ampersand (&) is quite universal but will be pronounced differently, and the center dot (·) remains unpronounced.

The reason to have more languages is that this is an international issue, though fragmented over the languages. A windfall benefit is that the reader and listener may experience a little bit how it would be for kids to rely on sounds to learn about numbers and their structure.

Colignatus (2015b, 2018a) manually typed out conversion tables for English, German, French, Dutch and Danish. This close reading is important since natural languages (and their language committees) develop peculiarities, while there also can be conflicts between current pronunciation and the simple elaboration with 10, see German zehn | zig and Dutch tien | tig below. Adapting to another language than those checked ones may still cause such conflicts for the proposed full use of the place value system. The feature of transliteration started out for presentation (of the checked languages) but appears to be important as a gateway for quality control (of unchecked languages) as well.

The US Common Core (2018) has negative numbers only in Grade 6, and Holland and the UK only in Grade 7 (junior highschool). For us it remains useful to include the negative sign too.

1.4. A natural language as a dialect of mathematics

We tend to use the term “natural language” but we should not forget that scores of influential authors and committees have been working on the traditional pronunciation of the natural numbers. When there is scope for improvement then it can be discussed. This is not an issue of spelling reform but an issue of mathematics education. Arithmetic and number sense are not only about numerals and the operations but also about how you pronounce the numbers, especially for kids who cannot read or write yet, and how it percolates into later thought (that might be “subvocalised speaking”).

The suggestion is that schools indeed embrace this *full use of the place value system*, so that language at school is the *language of arithmetic*, and so that the common language on the numbers

used at home can be regarded as a *dialect* of this language of arithmetic. Kids can deal with such differences in language. It is 12 years after the implementation from kindergarten and up that the national judicial system must have worked out whether the pronunciation of arithmetic would also be relevant and acceptable for legal issues.

- Colignatus (2015a), *A child wants nice and no mean numbers*. (To adapt to the amendment of (2018a).)

1.5. The need for an ISO standard

The system proposed here is simple but still supports a *full use of the place value system* for education in kindergarten and elementary school. It must become an ISO standard, so that educators and textbook & software publishers but also researchers have stability of their environment. Even when schools would not implement the system (so fast), researchers and teacher trainers require a standard to correct their research findings for confounding by the natural languages. The definition of this standard is given here, and this notebook with package only provide an implementation:

- Colignatus (2015b, 2018a), *The need for a standard for the mathematical pronunciation of the natural numbers. Suggested principles of design. Implementation for English, German, French, Dutch and Danish.*
- The amendment in 2018a is the use of the connectives, see below.

1.6. Discussion on content and a technical report

The suggestion for a full place value system was motivated by education. The package implements this, and this notebook contains much of a technical discussion on how to use the package. However, the very reason to make the suggestion derives from education, and we should not forget about this very motivation. This notebook thus also looks at how the package might be applied for the Common Core State Standards (2018). We can identify 4 levels of number sense relevant for pronunciation and the full use of the place value system that are supported by 4 routines in the package. This discussion is not intended as a comment on the curriculum or as a suggestion how the curriculum might look like.

This present discussion on content does not discuss the fundamentals for the standard, and the reader is referred to (2015ab, 2018a) for the more involved discussion. This present notebook with package provides for a software implementation in *Mathematica,* and we concentrate on the latter.

Though the package contains the full system - with the definition that must become an ISO standard - it has practical limitations. The main purpose of this notebook is to set some first steps towards implementation and to circulate the idea. Programming on language is quite involved, and implementing change even more.

1.7. Transliteration for a quick impression

When you install *Mathematica* you select your **$Language** in the Preferences. You can change the Preferences for another language and then do a restart. A setting of $Language will also provide for such pronunciation of the digits and numerals, though in traditional fashion. The routines provided here should give access to the full place value system of your $Language, though with these warn-

ings: (i) except for the comment on quality control (see above), (ii) and perhaps except for the setting of some options, likely an adaptation of the connectives (see below). I did not test changing $Language from “English”.

At times it is an acceptable and faster way to transliterate words. Tourists are familiar with transliterations in their tourist guides. For example, French "deux" is pronounced in English as "duh". We use this transliteration now since it provides for a quick impression (without restart) how the full use of the place value system would look also in some other languages. This helps to identify that this is an international issue that is served by an ISO standard. Indeed, other languages are also included in the proposal Colignatus (2015b, 2018a) but it makes a difference to actually hear it pronounced.

The transliteration must be defined for the $Language in your installment of *Mathematica*. The package basically allows for different settings, yet has been developed while using $Language = "English".

1.8. Structure of the paper

Below, I briefly indicate some history and other researchers. The main section explains the working of the package. It will be most instructive though to start with examples that highlight the properties. The main body of the text will use English, French and German while there will be a bit more later about Dutch and Danish. The definitions of the routines are printed in the **Appendix**. The standard routines in *Mathematica* provide for a link to the current (traditional) pronunciation in the natural languages. The traditional (wrong) pronunciation and notation has received quite some support, but let us now look at the full use of the place value system.

2. Example in English

The following describes our implementation (and not the English language as it is). There could be a difference between the output that the package generates, intended for interactive learning, and texts that a textbook could show. For example, a textbook could have two·ten & five but the package may put in some more blanks.

2.1. Speaking and writing

There is a difference between speaking and writing. Speaking uses a pause. The present pause might be a bit long, but this length was pleasant for the transliteration of other languages. The routines tend to have default Speak → True, but you can turn it off. The routines show the words in the chosen language and not the transliterations, unless you explicitly ask for the transliterations.

```
PlaceValue[21]  (*default Speak → True *)
```

two·ten & ·one

The routines store what was spoken (or often what could have been spoken if not turned off).

```
LatestPronunciation[]
```

two·ten <silence msec="300"/> & ·one

2.2. Simplify

For the place value system, it is important to recognise that 1 is actualy 1 of 1 (with a conceptual difference between digits and single digit numbers). After a short while this meticulous accuracy becomes annoying whence we simplify. It is advisable to teach the place value system, but it is also acceptable to simplify "one hundred" into "hundred", and to pronounce 10 as "ten" and not as "0 hundred, 1 ten, 0 one". Nevertheless, for teaching it is important to show both the whole system and its simplification. The trailing dot appears to be a quite useful indicator of this simplification.

```
PlaceValue[1, Speak → False]
```

·one

```
PlaceValue[1, Speak → False, Simplify → False]
```

one·one

Above, there has been simplification of 21, indicated by a trailing dot.

```
PlaceValue[21, Speak → False, Simplify → False]
```

two·ten & one·one

2.3. Traditional pronunciation

PartialPlaceValue calls *Mathematica*'s IntegerName and gives the traditional pronunciation.

```
PartialPlaceValue[123]
```

one hundred twenty three

PlaceValue gives the full place value pronunciation.

```
PlaceValue[123]
```

·hundred & two·ten & three·

2.4. The connectives

The connectives "&" and "·" have an important role. They differ from the operators "plus" and "group" (multi-plus, repeat, times), since + and × have commutation, association and distribution.

- The ampersand (&) is the ghost of addition, but simply "and", and not the operator "plus" with all its properties. The ampersand should be pronounced, namely to separate the place value positions. It is already (often) pronounced in German, Dutch and Danish. Other languages better adopt this practice too. Remember that we are speaking about the language of arithmetic to be used in school and not about an integral language reform (that would evolve).
- The center dot (not pronounced) is the ghost of multiplication of the *weight* and the *place value*. Remember: 5 days 2 hamburgers is not the same as 2 days 5 hamburgers.

Kids in kindergarten and Grade 1 live in a world of sounds. Thus they should be provided with the &-separator of the place value positions, so that they have this anchor on which is what. For adults and native speakers of English it may seem superfluous. Indeed, I myself in (2015a, footnote 10, and 2015b before the amendment in 2018a) found the use of "&" "distractive". My proposal then was to use the center dot for "&" too: thus without the distinction and merely as an unpronounced connec-

tive, as 25 = two·ten·five. However, in reconsideration, the empirical observation is that the &-separator really is there. Its existence must be acknowledged instead of hidden from sight.

Namely, in natural language, putting two terms alongside, like in 2 km, means scalar multiplication. In multiplication as grouping, kids learn to use the times-symbol, but we do not write 2 × km. Later students will learn that 2 *a* is multiplication in general, also dropping the times-symbol. If they would have been trained by the pronunciation of the very numbers (and *a* in 2*a* would be a number, in this scenario like *a* = 25 = two·ten five, thus without the "&") then we create a conundrum: (1) within "*a* = 25 = two·ten five" the lack of an interfix means addition and (ii) outside of this, in 2 *a,* the lack of an interfix means multiplication ? We should not create conundrums. Thus 25 = two·ten & five. It should help understand 2 × 25. Perhaps adults might not speak the ampersand but for kindergarten and elementary school it is part of the system.

Indeed, in kindergarten and Grade 1 kids will tend to focus on the & as an important new symbol in their universe, but this is not "distractive" but only fortunate, because it will form a stepping stone for the later learning on addition, i.e. using plus. Eventually they would tend to focus on the figures in the numbers and not the connectives.

NB. The ampersand is remarkably large in common fonts (like upper case), and it is better displayed in a smaller font (like lower case). It appears that Speak cannot deal with a Stylebox in the strings that we are using. The solution was to create the language "Amplish", that has the True form with the smaller font for the ampersand, and that is transliterated in English in the default font. This worked fine, and thus all languages here have the smaller ampersand (which might cause that one has to adapt the package to work under a different $Language). The functionality of providing transliteration was useful here too. This only applies for the routines PlaceValue and PlaceValueRules. It is useful to know for the option settings. Check the visual difference:

```
PlaceValue[25, Language → "Amplish"] (*default*)
```

two·ten & five·

```
PlaceValue[25, Language → "English"]
```

two·ten & five·

2.5. Do not use ones, tens, hundreds

The Common Core (2018) has:

```
CCSSMathContent["Grade 1", "NBT", "B.2.C"]
```

Understand place value. CCSS.Math.Content.1.NBT.B.2.C
The numbers 10, 20, 30, 40, 50, 60, 70, 80, 90
refer to one, two, three, four, five, six, seven, eight, or nine tens (and 0 ones)

CCSS writes about "tens", and elsewhere also about "ones". CCSS likely can be so sloppy since common English has "twenty" instead of "two·tens" etcetera. If English had adopted such system with the plurals of the bases, with a multitude of s's, then the abuse might be noticed sooner.

For CCSS "two × three" should also be "twos × threes", because the two implies that the threes are plural, and the three implies that the twos are plural. It shows that one doesn't understand what a number is. Instead of "5 tens" one better says "5 of ten", since "of" means grouping (multiplication).

The use of “ones” and “tens” pre-dates the development of the place value system, and is not supported in pronunciation, since the numbers 11-19 are properly pronounced in place value manner. The proper expression for 20 is 2·10, and we speak the base, which is 10 and not “tens”. One should avoid speaking about “ones” and “tens”. These plurals arise in a context when teachers are groping for a way of expression, but those words are confusing for kids as if these words would really be defined. Likely this is similar to 1 car versus 2 cars, but a base has a different linguistic treatment. (We are not pronouncing telephone numbers by rattling of the digits either.)

Remember that a puzzle is only complete when all pieces are in the proper places. When something has a logical structure then people are bound to get at least one piece wrong and develop an emotional hangup on it, and each person or nation another piece. There is no other solution than to clarify the logical structure.

2.6. Myriad and lakh

The figure 10,000 tends to be pronounced commonly as “ten thousand”. Remarkably, there are no specific terms for 10^4 and 10^5 in all the languages in the package. Perhaps French may have a more common acceptance of the Greek myriad = 100^2. This is another breach of the place value system. Wikipedia (a portal and no source) observes the lack too, and mentions Greek myriad and Indian lakh. I have adopted these words for the languages mentioned in the packages too. It takes some adjustment to hear 500,000 being pronounced as 5 lakh, but such adjustment is the very purpose of the exercise. It is not an urgent issue for kindergarten though.

```
PlaceValue[500 000, Speak → False]
```

five·lakh

These names are used in the main method that pronounces each entry separately. When the blocks of 3 digits are pronounced then they are superfluous.

2.7. The sign

The proper pronunciation of the minus sign in numbers is “negative”, because “minus” is used for the binary operation. *Mathematica*’s IntegerName gives the number and uses “negative” indeed.

```
PartialPlaceValue[-10]
```

negative ten

The new routines allow us to show the structure of the place value system.

```
PlaceValue[-10, Speak → False, Simplify → False]
```

negative one·ten & zero·one

Unfortunately *Mathematica* uses Minus[*x*] for *-x,* and also pronounces it that way (it may pronounce the format in *Mathematica* rather than the number). Wikipedia (a portal and no source) regards “negative” as American English, but the better diagnosis is that British English is imprecise. Dutch can distinguish 7 minus 10 = min 3.

```
? Minus
```

$-x$ is the arithmetic negation of x. ≫

```
{Minus[10], -10}
```

$\{-10, -10\}$

```
Speak[%]
```

3. History, tradition, assumptions, advantages

3.1. History of proposals for change

Jane Austen (1775-1817) apparently still wrote “three and twenty” instead of “twenty-three”. The English speaking people are lucky that they managed the change on the order (major) though unlucky with the loss of “and” (minor).

Norway managed the change in the 1950s.

Fred Schuh (1875-1966) of TU Delft proposed this in the 1950s in Holland but he didn’t convince the minister of education. I am not aware of other Dutch authors who propose this nowadays.

In Germany there is a small movement with Lothar Gerritzen and Peter Morfeld to change their "ein und zwanzig" (perhaps no blanks) into something more useful: https://zwanzigeins.jetzt. Originally they thought about using “zwanzigeins” (apparently no blanks) but they are open to alternatives and hope to attain a more developed proposal at the end of 2018. Recently they provided support for this present research on getting an ISO standard, see https://zwanzigeins.jetzt/infos/normung-der-aussprache

In Denmark there are Lisser Rye Ejersbo and Morten Misfeldt (2015), at Aarhus. Ejersbo recommends the recent book of the 23rd ICMI conference, Bussi & Sun (eds) (2018). See http://vbn.aau.dk/da/publications/danish-number-names-and-number-concepts(7b79a70d-d42b-49dc-af1f-75775c9292f6)/export.html

China uses the place value system in the form of Level 2 discussed below. I am only kicking in open doors. It might be an option to translate a number first into Chinese and then transcribe back, see Uy (2003) . However, this present notebook with package gives more control and includes didactic notions. Chinese does not pronounce the connective “&” (though it is implied) and we should better use it. *Mathematica* already provides this routine.

```
Table[IntegerString[10^k, "TraditionalChineseDecimal"], {k, 0, 4}]
```

{一, 一〇, 一〇〇, 一〇〇〇, 一〇〇〇〇}

Fateman (2013) discusses computerised speaking of math, with a proper distinction between how we currently pronounce the numbers and how we “should” do this. It is much wider and deeper than my present purposes.

Mathematica provides routines IntegerName and WordTranslation but we cannot always use them, since (i) we must avoid the breaches, put terms into proper order and get rid of “twenty” and “ten thousand”, and (ii) we want to show the intermediate steps without simplification so that kids can

see what the place value system actually is.

3.2. Tradition is not without reason but still counterproductive

Traditional pronunciation and its didactics are not without reason.

Kids start with 0-10, in which 10 is a new name "ten" and not "1·ten" and then simplified. These kids would use grouping (multiplication) but have no developed vocabulary and command on this. When continuing with 11-20 ("eleven" is "one left over" after 10) and speaking from right to left (and perhaps writing so too), they write 19 and in the conventional dialect speak "nine-teen", which fits the order of their low numbers writing from right to left. Our reference to left and right is tricky since kids might still be struggling with which is what. "Twenty" again is a single notion, since kids might not have command of multiplication yet. In English kids make the switch to "speak the highest place value first" at 20. In German and Dutch they do so at 100. This focus might make sense in kindergarten and Grade 1, if we neglect the existence of other numbers, or if we neglect that the same kids in kindergarten and Grade 1 grow up and have to learn those other numbers too.

If kids in kindergarten and elementary really started out writing from right to left in general, also for words, from the argument that empirical research has shown that this would be easier for them, then one might have an argument that 13 best is pronounced from right to left. Likely though, the order survives from the Fibonacci's *Liber Abaci* of 1202. Not only the current order but also the traditional pronunciation is simply *imposed* upon them, with the only argument that school teaches tradition for the sake of tradition. A change may require retraining kindergarten teachers who may be weak on arithmetic anyway yet this is a one-time investment with major persistent benefits.

3.3. Assumptions

Full use of the place value system:

1. does not require kids to understand formal multiplication and have command of the table of 10 and later the powers of 10, but only adopts a pronunciation and way of writing of both numerals and words that support this later development rather than hindering it
2. makes the switch to "spreak the highest place value first" already at 10 (and not 20 or 100), which is precisely the moment when the phenomenon occurs for the first time, and which thus establishes the system without exception (and kids can be told about this rule)
3. recognises that "teen" and "ty" (indeed in the world of sounds) are useful indicators for value (compare: three, thirteen, thirty, third) but sees more profit in fully using the place value system so that value transpires from the structure in the pronunciation rather than from employing different sounds (since sounds can also sabotage and obscure the structure).

This approach makes the assumption that kids can acquire a structure. This is a weak assumption since kids can already acquire language and a sense of reality that are filled with all kinds of structures. If kids can learn something illogical then they should be able to learn something logical. Chinese kids learn the place value system on a regular basis. Our topic is not really an issue of empirical research but one of traditional thinking in the world of educators and researchers. Complications are: The latter world of education may have little command of mathematics and think that tradition already captures mathematics. *Sacrosanct* means: high respect for something that you do not grasp. Mathematicians dealing with arithmetic in school may think abstractly and have no

background in empirics, and focus on learning the numerals.

3.4. Main advantages

To teach the place value system, it is advisable *to actually use it*.

- Pronuncation (that kids start with) would be co-ordinated with the numerals, and later with the written words. (Number: pronunciation, numeral, word.)
- The workflow is into a single direction, and does not jump around.
- Merely speaking a number aloud can already solve calculation questions. (For many kids: thinking may be “subvocalised speaking”, as reading is subvocalised.)
- Current co-ordination failures (i.e. the breaches) often are not discussed so that kids simply do not know why they find issues complicated, and they are groping in the dark.
- There will be a sizeable savings in teaching and learning time, that can be spent on real issues.
- Research on mathematical skill and number sense is highly contaminated by this pronunciation issue. Much research draws invalid conclusions. Doing good research is quite impossible because you cannot simply experiment on kids by first training them on your own theory on the place value method. We neither can compare international results when there is such variety and invalidity. (In studies in Holland and Denmark, kids were observed to use English, and telling others to do so too, because they better understood the numbers in that way.)

4. A structure for the curriculum

4.1. Overview

This notebook with package is not intended as a course on the full place value system for kindergarten and elementary school. Teachers would tend to start with discussion in print for their first orientation. Colignatus (2015b, 2018a) is such an introduction, and provides tables for the pronunciation of the integers in the full place value system, for English, German, French, Dutch and Danish. This notebook with package supplements this material with the ability of sound (basically in $Language, but for practical purposes now in transliteration). We mimick the situations that kids would be in, at different levels of competence. The purpose of this Section is to indicate which aspect of the place value system is highlighted by what routine.

We will quote from: http://www.corestandards.org/Math/Content/NBT.

Below we start with counting but of course quickly meet with the transition from counting to addition. We focus on the pronunciation and leave this transition aside, though there is a clear link of course between the ampersand and plus. See Colignatus (2018b), *Tables for addition and subtraction with better use of the place value system.*

For the curriculum on the place value system we can recognise a structure with the following levels. Per level there is a supporting routine. We first discuss the levels, and later collect them into a switching routine. This switching routine can now be used for a quick overview of what the routines do. These examples use higher numbers merely to show the properties. Obviously kids start with 0-10 but such a number doesn’t highlight what the routine does.

Level 1. Sounds only (kids cannot read and write). Words record the sounds for us. Speaking them gives their world.

```
PronounceInteger[12, Level → 1, Speak → False]  (*overview without Speak*)
```

·ten & two·

Level 2. Learning the digits and numerals (already having the sounds for the digits).

```
PronounceInteger[1234, Level → 2, Speak → False, Print → False]
(*overview without Speak*)
```

1000 & 2 · 100 & 3 · 10 & 4

Also learning how to write the words.

```
PronounceInteger[1234, Level → 2, Speak → False] (*default*)
```

·thousand & two·hundred & three·ten & four·

1000 & 2 · 100 & 3 · 10 & 4

Level 3. Advanced: putting digits in blocks of three. (There are slightly special lists because we want to Speak “one hundred” instead of “one one-hundred”.)

```
PronounceInteger[123 456, Level → 3, Speak → False] (*overview without Speak*)
```

{{{1 · 100, & , 2 · 10 & 3}, 1000}, & , {4 · 100, & , 5 · 10 & 6}}

Level 4. Accomplished: using blocks of three digits (but pronounced in full place value).

```
PronounceInteger[123 456, Level → 4, Speak → False] (*overview without Speak*)
```

123 thousand, 456

```
LatestPronunciation[]
```

·hundred & two·ten & three· thousand, four·hundred & five·ten & six·

4.2. Level 1. Sounds

4.2.1. No reading and writing yet

Kids in kindergarten start with pronunciation. They cannot read and write yet. Their sounds are encoded in words but only the teacher would know how to write those words. The teacher must get used to the full place value system, and will first rely on written words. The advantage of using words is that the pronunciation becomes unambiguous for the particular language.

The main routine supports this situation by using words for pronunciation and transliteration. We produce the sounds that kids would hear and what they would copy in rehearsing, and we print the spoken words in the chosen language for the sake of the teacher. The routine speaks in $Language (here "English") but may use transliteration.

```
PlaceValue[#, Language → "French"] & /@ Range[0, 10]
```

{zero·, ·un, deux·, trois·, quatre·, cinq·, six·, sept·, huit·, neuf·, ·dix}

Rehearsing the sequence will cause kids to regard these words as a sequence, and the sequence can be related to other phenomena.

4.2.2. Common Core on 11-19 in sounds

```
CCSSMathContent["Kindergarten", "NBT", "A.1"]
```

Work with numbers 11–19 to gain foundations for place value. CCSS.Math.Content.K.NBT.A.1.
Compose and decompose numbers from 11 to 19 into ten ones and some further ones, e.g., by using objects or drawings, and record each composition or decomposition by a drawing or equation (such as 18 = 10 + 8); understand that these numbers are composed of ten ones and one, two, three, four, five, six, seven, eight, or nine ones.

Observe that CCSS speaks about “ten ones” instead of “ten of one”, and see the rejection of “ones”, “tens” and “hundreds” above. Kids can collect ten apples in a basket of ten. They can agree that there are *ten of 1 apple* and *1 basket of ten apples*. If kids can count, then they also should know what they are counting (fingers, cars, baskets), and thus it is proper to explicitly discuss with them what to take as unit of account. Taking the basket of ten as the base, they may accept that a number is spoken as “one of ten” but simplified to ten since this is the base.

Instead of rehearsing all numbers from 0 to 20, we speak a small selection now.

```
(PlaceValue /@ Range[8, 13]) // MatrixForm
```

$$\begin{pmatrix} \text{eight·} \\ \text{nine·} \\ \text{·ten} \\ \text{·ten \& ·one} \\ \text{·ten \& two·} \\ \text{·ten \& three·} \end{pmatrix}$$

The latter in French.

```
(PlaceValue[#, Language → "French"] & /@ Range[8, 13]) // MatrixForm
```

$$\begin{pmatrix} \text{huit·} \\ \text{neuf·} \\ \text{·dix} \\ \text{·dix \& ·un} \\ \text{·dix \& deux·} \\ \text{·dix \& trois·} \end{pmatrix}$$

If your $Language = “English” too, then you can verify that the latter French words were transliterated in English.

```
LatestPronunciation[]
```

·dees <silence msec="300"/> ayh trouwah·

4.2.3. A possible exercise still in speaking only

The following exercise is only in English and not in the other languages. The routine has the answer of the sum as input. In this case we select an outcome less than 20.

```
CCSSMathExercise[15, 12, Speak → True]
```

You know the numbers three· and ·ten & two·, and how to speak them. You now count three· up from ·ten & two·, and then you find the number ·ten & five·. You now learn to speak the number ·ten & five·.

The exercise is done by speaking the counts.

```
(PlaceValue /@ Range[12, 15]) // MatrixForm
```

$$\begin{pmatrix} \cdot\text{ten \& two}\cdot \\ \cdot\text{ten \& three}\cdot \\ \cdot\text{ten \& four}\cdot \\ \cdot\text{ten \& five}\cdot \end{pmatrix}$$

4.3. Level 2. Sounds and numerals, and then reading and writing

At the second level, kids learn to read and write the digits 0, ..., 9 and the numerals 0, ..., 9.

4.3.1. Showing the numeral and speaking it

The first step mimicks this learning process by showing the numeral and speaking the (transliterated) word. Running this routine often might cause that the association between the sound and the picture is transferred into human memory.

```
PlaceValueDigits[4, Language → "French", Print → False]
```

4

Of course, this does not yet give information about how this 4 relates to the other numbers. We are only discussing the link between the pronunciation (often already known) and the figure.

4.3.2. Showing the numeral, speaking it, and learning the word

In a second step kids also learn the word. The routine shows the numeral and official word, and speaks the (transliterated) word. Again, running the routine often might cause that the association is transferred into human memory. I do not want to imply that kids first must learn to write the word before they can continue with the numerals. This only intends to show that the routines can support such learning, assuming having or getting command of sounds.

```
result = PlaceValueDigits[4, Language → "French"] (*Print default*)
```

quatre·

4

The following statement collects the input, what has been printed and spoken, and the output.

```
{4, LatestPronunciation[Print], LatestPronunciation[], result}
```

{4, quatre·, kahtruh·, 4}

4.3.3. The digit / numeral by itself is not place value yet

Observe that digits do not have a trailing dot. Learning the digits differs from learning the numbers in a place value system. The digits could also be used in another system.

It depends upon didactics and empirical research whether it is useful at this stage to discuss that 4 can also be understood as 4·1 in full place value system. Likely this would be totally confusing. I advise against this, and I actually do not think that research on this would be so urgent.

```
result = PlaceValueDigits[4, Language → "French", Simplify → False]
```

quatre·un

4 · 1

The latter gives the new number denoted in the place value system (though not yet constructed at this stage), its unsimplified pronunciation, and its decomposition in the *weighted* place values. Do not use this here. It only becomes a useful insight when the system is already quite developed. I only mention it here to issue the warning not to use it here.

4.3.4. Introducing the symbol 10

Once the digits and numbers 0 ... 9 have been mastered (to a sufficient degree to move on) there is the crucial step of *creating* the positional system, moving into the numerals of 10 ... 20.

```
CCSSMathContent["Grade 1", "NBT", "B.2.A"]
```

Understand place value. CCSS.Math.Content.1.NBT.B.2.A
10 can be thought of as a bundle of ten ones — called a "ten."

CCSS seems confused between (i) learning what ten is and (ii) learning how to write it in numeral and word. CCSS again speaks about “ten ones” instead of “ten of one”.

Likely Grade 1 has already been counting from zero to ten because of the fingers. In that case the sound “ten”with its numerical meaning would have been introduced above already. (Remember CCSS.Math.Content.K.NBT.A.1 above !) The crux here is to relate the familiar sound to the new picture of 10 (writing is drawing small pictures). The teacher can explain that there are ten digits 0, ..., 9, and that we need a new symbol for ten, and that this new symbol is 10. This will not be the telephone number “one-zero” but a written “10” is pronounced as “ten”.

Running this routine often might cause that the association between the sound and the picture is transferred into human memory. (It is important to identify the moments and methods of learning.)

```
PlaceValueDigits[10]
```

·ten

10

```
PlaceValueDigits[10, Language → "French"]
```

·dix

10

4.3.5. Building up 11-19 as symbols

Having 10, it is easy to say that additional counts 1, ..., 9 will be recorded in the position of the 0. Again, kids have already learned aspects about the numbers 11, ..., 19 entirely in their world of sounds (see the subsection above), and the learning goal now is to read and write the figures. A major issue is to write (from left to right) first the 1 and then the additional count, The 1 is higher because it represents 10. The shortcut (pons asinorum) is that the count is recorded in the place of the 0.

```
CCSSMathContent["Grade 1", "NBT", "B.2.B"]
```

Understand place value. CCSS.Math.Content.1.NBT.B.2.B
The numbers from 11 to 19 are composed of a ten and one, two, three, four, five, six, seven, eight, or nine ones.

Basically we have vector addition.

```
{1, 0} + {0, count} (* 10 + count, decomposed by positions *)
```

{1, count}

It might become feasible to show the kids numerals in a question like this.

```
CCSSMathExercise[15, 12, "Grade" → "Grade 1", Speak → True, IntegerDigits → True]
```

You know the numbers 3 and 12, and how to write and speak them. You now count 3 up
from 12, and then you find the number 15. You now learn to write and speak the number 15.

The above was spoken in full place value manner. You might recognise the larger ampersand.

```
LatestPronunciation[]
```

You know the numbers three· and ·ten & two·, and how to write and speak them. You now count three· up from ·ten & two·, and then you find the number ·ten & five·. You now learn to write and speak the number ·ten & five·.

```
result = PlaceValueDigits[inp = 15, Language → "French"]
```

·dix & cinq·

10 & 5

We thus have the new number denoted in the place value system, its pronunciation, and its decomposition in the *weighted* place values.

```
{inp, LatestPronunciation[Print], result}
```

{15, ·dix & cinq·, 10 & 5}

4.3.6. Introducing 20

```
CCSSMathContent["Grade 1", "NBT", "B.2"]
```

Understand place value. CCSS.Math.Content.1.NBT.B.2
Understand that the two digits of a two−digit number represent
amounts of tens and ones. Understand the following as special cases: B.2.A, B.2.B, B.2.C

After 9, we have no longer digits available to record at the position zero, and we make the transition of the position.

```
{1, 0} + {1, 0} (* 10 + 10 decomposed by positions*)
```

{2, 0}

```
CCSSMathExercise[inp = 20, 19,
 "Grade" → "Grade 1", Speak → True, IntegerDigits → True]
```

You know the numbers 1 and 19, and how to write and speak them. You now count 1 up
from 19, and then you find the number 20. You now learn to write and speak the number 20.

From the rehearsing the sequences before, the kid likely already knows that 20 comes after 19. The present learning goal is how to record the sound using digits.

It depends upon empirics how much can be discussed about place value arithmetic. We have: 19 + 1

= 10 + 9 + 1 = 10 + 10 = 20. Thus we unpack 19, reorder, pack 9 + 1 and pack 10 + 10 .

```
result = PlaceValueDigits[inp, Language → "French"]
```

deux·dix

$2 \cdot 10$

We thus have the new number denoted in the place value system, its pronunciation, and its decomposition in the *weighted* place values.

```
{inp, LatestPronunciation[Print], result}
```

{20, deux·dix, $2 \cdot 10$}

4.3.7. Building up 21-29

```
CCSSMathContent["Grade 1", "NBT", "B.2"]
```

Understand place value. CCSS.Math.Content.1.NBT.B.2
Understand that the two digits of a two−digit number represent
amounts of tens and ones. Understand the following as special cases: B.2.A, B.2.B, B.2.C

See Colignatus (2018b) for tables of addition and subtraction with full use of place value.

```
{1, 2} + {1, 4}
```

{2, 6}

The excercise mentions counting but we are not looking at the shift from counting to addition here. Numerals are shown but the exercise is pronounced in full place value manner.

```
CCSSMathExercise[inp = 26, 12,
 "Grade" → "Grade 1", Speak → True, IntegerDigits → True]
```

You know the numbers 14 and 12, and how to write and speak them. You now count 14 up
from 12, and then you find the number 26. You now learn to write and speak the number 26.

```
LatestPronunciation[]
```

You know the numbers ·ten & four· and ·ten & two·, and how to write and
speak them. You now count ·ten & four· up from ·ten & two·, and then you find the
number two·ten & six·. You now learn to write and speak the number two·ten & six·.

```
result = PlaceValueDigits[inp, Language → "French"]
```

deux·dix & six·

$2 \cdot 10$ & 6

We thus have the new number denoted in the place value system, its pronunciation, and its decomposition in the *weighted* place values.

```
{inp, LatestPronunciation[Print], result}
```

{26, deux·dix & six·, $2 \cdot 10$ & 6}

4.3.8. The table of 10

```
CCSSMathContent["Grade 1", "NBT", "B.2.C"]
```

Understand place value. CCSS.Math.Content.1.NBT.B.2.C
The numbers 10, 20, 30, 40, 50, 60, 70, 80, 90
refer to one, two, three, four, five, six, seven, eight, or nine tens (and 0 ones)

See the comment above on avoiding “tens”, and how the “s” would pollute the table of ten.

```
PlaceValueTable[Range[10, 90, 10], Language → "French"]
```

Integer	Place Value	PV Digits	Traditional
10	·dix	10	dix
20	deux·dix	2 · 10	vingt
30	trois·dix	3 · 10	trente
40	quatre·dix	4 · 10	quarante
50	cinq·dix	5 · 10	cinquante
60	six·dix	6 · 10	soixante
70	sept·dix	7 · 10	soixante−dix
80	huit·dix	8 · 10	quatre−vingts
90	neuf·dix	9 · 10	quatre−vingt−dix

```
? PlaceValueTable
```

PlaceValueTable[{n}] gives a table of {n} with rows {n, pronunciation
in PlaceValue, PV Digits, traditional pronunciation}. Options are passed on to
PlaceValue (default Speak –> True) and PartialPlaceValue(default Speak –> False)
PlaceValueTable[n] gives the table with only n
PlaceValueTable[Row, n || {n}] gives the rows as lists only
PlaceValueTable[First, n] gives {n, PV, PV Digits}

4.3.9. Building up the positional system

We now repeat the above, in ever higher numbers, with the next place value transition at 10^2.

```
CCSSMathContent["Grade 1", "NBT", "A.1"]
```

Extend the counting sequence. CCSS.Math.Content.1.NBT.A.1
Count to 120, starting at any number less than 120. In this range,
read and write numerals and represent a number of objects with a written numeral.

```
{1, 0, 0} + {0, count10, count1}
```

{1, count10, count1}

The following is spoken in full place value manner.

```
CCSSMathExercise[120, 119, "Grade" → "Grade 1", Speak → True, IntegerDigits → True]
```

You know the numbers 1 and 119, and how to write and speak them. You now count 1 up
from 119, and then you find the number 120. You now learn to write and speak the number 120.

```
LatestPronunciation[]
```

You know the numbers ·one and ·hundred & ·ten & nine·, and how to write and speak
them. You now count ·one up from ·hundred & ·ten & nine·, and then you find the number
·hundred & two·ten. You now learn to write and speak the number ·hundred & two·ten.

```
result = PlaceValueDigits[inp = 120, Language → "French"]
```

·cent & deux·dix

100 & 2 · 10

We thus have the new number denoted in the place value system, its pronunciation, and its decomposition in the *weighted* place values.

```
{inp, LatestPronunciation[Print], result}
```

{120, ·cent & deux·dix, 100 & 2 · 10}

4.3.10. Grade 2 and 100

```
CCSSMathContent["Grade 2", "NBT", "A.1.A"]
```

Understand place value. CCSS.Math.Content.2.NBT.A.1.A
100 can be thought of as a bundle of ten tens — called a "hundred."

Surprisingly, CCSS requires kids of Grade 1 to count to 120 but CCSS also wants that the word “hundred” is mentioned only in Grade 2 ? Likely kids at that stage would also know that 100 is 10 of 10. My impression is: CCSS wants Grade 2 to grow aware that 100 features in the place value system precisely because it is a power of 10. My other impression is that it suffices to show the role of 10, 100, 1000 etcetera but discuss the power structure only later. Often you better use something first, so that the explanation can better be understood once you know what it is about. Dutch mathematician and teacher Gerrit Mannoury (1867-1956): “The meaning of words is their use.” (my transscription) - and this was later also seen by Ludwig Wittgenstein (1889-1951), and likely before by C.S. Peirce (1839-1914) though I do not have a quote. Do not wait too long with the explanation otherwise students develop awkward misunderstandings and habits. My third impression is that CCSS is pernicious on grouping. They avoid the term “grouping” (multiplication) but use “bundling”, as a lawyer might suggest that bundling doesn’t necessarily mean grouping, and hide definitions to make it seem serious. My suggestion is to do a proper table of 10, so that also 10 of 10 = 100 = hundred is included (and not limit this to 90 as in CCSS.Math.Content.1.NBT.B.2.C above). In the full place value, 100 is not decomposed in ten·ten. But we obviously want kids to know this. At some point it can be explained that using ten digits to construct numerals causes the use of the powers of ten. A square of 10^2 and a cube of 10^3 should not be hard to grasp. PM. Tradition simplifies for one 1 but curiously doesn’t simplify fully for one 100.

```
PlaceValueTable[100]
```

Integer	Place Value	PV Digits	Traditional
100	·hundred	100	one hundred

```
PlaceValueTable[100, Simplify → False]
```

Integer	Place Value	PV Digits	Traditional
100	one·hundred & zero·ten & zero·one	1 · 100 & 0 · 10 & 0 · 1	one hundred

4.3.11. Grade 2 and 100 - 1000

```
CCSSMathContent["Grade 2", "NBT", "A.1"]
```

Understand place value. CCSS.Math.Content.2.NBT.A.1
Understand that the three digits of a three−digit number represent amounts of hundreds, tens, and ones; e.g.,
706 equals 7 hundreds, 0 tens, and 6 ones. Understand the following as special cases: A.1.A, A.1.B

```
PlaceValueDigits[706, Language → "French"]
```

sept·cent & six·

7 · 100 & 6

```
PlaceValueDigits[706, Language → "French", Simplify → False]
```

sept·cent & zero·dix & six·un

7 · 100 & 0 · 10 & 6 · 1

```
CCSSMathContent["Grade 2", "NBT", "A.1.B"]
```

Understand place value. CCSS.Math.Content.2.NBT.A.1.B
The numbers 100, 200, 300, 400, 500, 600, 700, 800, 900 refer to
one, two, three, four, five, six, seven, eight, or nine hundreds (and 0 tens and 0 ones).

```
CCSSMathContent["Grade 2", "NBT", "A.2"]
```

Understand place value. CCSS.Math.Content.2.NBT.A.2
Count within 1000; skip−count by 5s, 10s, and 100s.

```
CCSSMathContent["Grade 2", "NBT", "A.3"]
```

Understand place value. CCSS.Math.Content.2.NBT.A.3
Read and write numbers to 1000 using base−ten numerals, number names, and expanded form.

From the Glossary: “Expanded form. A multi-digit number is expressed in expanded form when it is written as a sum of single-digit multiples of powers of ten. For example, 643 = 600 + 40 + 3.”

Our routines allow digging deeper.

```
PlaceValueDigits[643, Language → "French", And → "+"]
```

six·cent + quatre·dix + trois·

6 · 100 + 4 · 10 + 3

We already had this in “&” but CCSS only knows about “+“. My suggestion is that kids know about secret codes, and thus know about the distinction between *content* and *coding*. They should be able to understand that “&” and “·” are used to encode the pronunciation of the numbers and that “+“ and “×” are the operators with properties of commutation, association and distribution. If they don’t get it then it is the objective of education that they get it. Watching movies with secret codes is an acquired taste.

4.3.12. Grades 3 & 4 and 100 - 1000

Grade 3 has multiplication and Grade 4 is more aware of complexities.

```
CCSSMathContent["Grade 3", "OA", "D.9"]
```

Solve problems involving the four operations, and
identify and explain patterns in arithmetic. CCSS.Math.Content.3.OA.D.9
Identify arithmetic patterns (including patterns in the addition table or
multiplication table), and explain them using properties of operations.

```
CCSSMathContent["Grade 4", "NBT", "A.2"]
```

Generalize place value understanding for multi−digit whole numbers. CCSS.Math.Content.4.NBT.A.2
Read and write multi−digit whole numbers using base−ten numerals, number
names, and expanded form. Compare two multi−digit numbers based on meanings of
the digits in each place, using >, =, and < symbols to record the results of comparisons.

Students are now up to a discussion of the full place value system using their command of numbers and multiplication. This shows how the place value system works, though using $Language for the (single) StepMonitor and again using "+". We might show with and without simplification.

```
PlaceValueDigits[643, Language → "French", StepMonitor → True, And → "+"]
```

TheSum[times[6, 100], times[4, 10], times[3, 1]]

six·cent + quatre·dix + trois·

$6 \cdot 100 + 4 \cdot 10 + 3$

PM. The place value system speaks all positions separately, e.g. also 10^4 and 10^5. We write out all zeros (1000) and do not use exponents since we assume that kids haven't had exponents yet but they can count the number of zeros. The following obviously is not for 2nd Grade, but it shows how the routine works. The spoken and written version is actually the Chinese method, see Uy (2003), though with the added feature here of the ampersand infixes. We have freedom to choose the infix (but a StyleBox will not work).

```
PlaceValueDigits[1 115 124, And → ","]
```

·million , ·lakh , ·myriad , five·thousand , ·hundred , two·ten , four·

$1000000 \cdot 100000 \cdot 10000 \cdot 5 \cdot 1000 \cdot 100 \cdot 2 \cdot 10 \cdot 4$

4.4. Level 3. Blocks of three digits

I did not see the following in the CCSS but may have overlooked it.

The students learn higher numbers, and how to pronounce them by blocks of 3 digits. This convention is efficient - and avoids the need for special names for 10^4 and 10^5. The insertion of "and" between the blocks and their weights provides a useful resting pause.

```
result = PlaceValueBlock[654 121]
```

$\{\{\{6 \cdot 100,\ \&\ ,\ 5 \cdot 10\ \ \&\ \ 4\},\ 1000\},\ \&\ ,\ \{1 \cdot 100,\ \&\ ,\ 2 \cdot 10\ \ \&\ \ 1\}\}$

For larger values we might display without the ampersand (which is the form in the Abstract).

```
PlaceValueBlock[654 121, And → "", Speak → False]
```

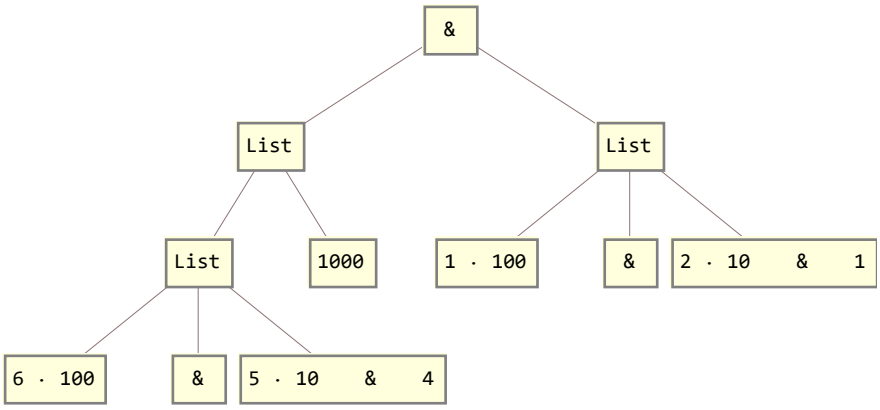
$\{\{\{6 \cdot 100,\ \ ,\ 5 \cdot 10\ \ \ 4\},\ 1000\},\ \ ,\ \{1 \cdot 100,\ \ ,\ 2 \cdot 10\ \ \ 1\}\}$

The layout of these lists has the following explanation: The routine PlaceValueBlock only uses $Language. The package was written with $Language = "English". It appears that Speak[100] is

“one hundred” and Speak[1·100] thus is “one one-hundred”. The “one one-hundred” and “one one-thousand” were so annoying, if not confusing, in combination with pronouncing the three digits and their place values, that I changed to “hundred” and “thousand” in speech.

There is a difference between what we want *Mathematica* to Speak (with the infixes) and what we want to Show now (at the head of the expression). The following is somewhat okay.

```
TreeForm[result /. List[x_, " & ", y_] :> (x ~" & "~ y)]
```



4.5. Level 4. Daily use

In the final phase kids are mastering or have mastered the place value system. The following routine adopts the convention of *Mathematica*’s IntegerName: absolute values less than 1000 are given as words (effectively PlaceValue), and higher values are combinations of blocks of digits and their bases in words (thousand, million, etc.). This allows high numbers. The block bases can be submitted to WordTranslation. Drawbacks: (i) The use of another language is relatively slow because of WordTranslation. (ii) Transliteration was included later and wasn’t designed for speed. PM. We separate the digit bases by a comma. This option for And can be adapted but is not passed on to PlaceValue.

```
PlaceValueIntegerName[99, Language → "French"]
```

neuf·dix & neuf·

```
PlaceValueIntegerName[1011, Language → "French"] (* above 999 *)
```

1 mille, 11

```
LatestPronunciation[]
```

·uhnn meel, ·dees <silence msec="300"/> ayh ·uhnn

```
PlaceValueIntegerName[1090011]
```

1 million, 90 thousand, 11

```
LatestPronunciation[]
```

·one million, nine·ten thousand, ·ten & ·one

```
PlaceValueIntegerName[1234451011, Language → "French", Speak → False]
```

1 billion, 234 million, 451 mille, 11

```
LatestPronunciation[]
```

·un billion, deux·cent & trois·dix & quatre· million, quatre·cent & cinq·dix & ·un mille, ·dix & ·un

Just to be sure: WordTranslation provided two possibilities, and the routine always takes the first.

```
WordTranslation["billion", "English" → "French"]
```

{billion, milliard}

The LatestPronunciation[] gives words in French without transliteration into English, because of the option Speak → False. Speaking it in English will be far removed from the actual pronunciation in French. If you want a transliteration then use Speak → True, or Level 1 again (that also drops the first "un"). In all cases transliteration is restricted to absolute numbers less than 10^7 (and thus "billion" is pronounced in English).

4.6. Level -4. Traditional pronunciation

Traditional pronunciation uses the place value system partially, with breaches of order and structure. Given the diagnosis in my booklet (2015a) *A child wants nice and no mean numbers*, it was tempting to use the name MeanNumbers for the tradition routine, but there is a difference between an expressive book title and programming, and thus I settled for PartialPlaceValue. It is at level -4. Conventionally it uses the blocks of 3 digits. The routine uses *Mathematica*'s IntegerName using DigitsWords and adopts its convention: For -1000 < *n* < 1000 only words are shown but for higher absolute values the blocks of digits are shown. Drawback: no transliteration.

```
PartialPlaceValue[321]
```

three hundred twenty one

```
PartialPlaceValue[654 321]
```

654 thousand 321

Speak stops when meeting a hyphen, and thus hyphens are replaced by blanks. Unfortunately for other languages, IntegerName concatenates the words. It is too involved to undo this.

```
PartialPlaceValue[21, Language → "German"]
```

einundzwanzig

```
StringReplace[%, "und" → " oond "] // Speak
```

When kids have classes on programming, writing code, it might be a nice suggestion to let them program the way how they pronounce the integers in the full place value system, and how their parents pronounce them in the dialect of traditional English.

4.7. One switching routine to rule them all

When you are new to these routines (like I was while writing them) then it may be easy to get lost for a moment. With some structure it is easier to see how they hang together. Therefor, PronounceInteger is a switching routine for selecting Level → *level,* with above levels 1, 2, 3, 4 and -4.

```
? PronounceInteger
```

PronounceInteger[n] pronounces n. Options are passed on to the routine that is called, except: if the chosen Language is equal to $Language then it is not passed on. It uses first the Level and then the Method setting in the options, default Level –> 1, Method –> PlaceValue. Option All –> {...} gives the methods in the MathEd`PronounceInteger` package. See the documentation per routine. If you want to specify the selection by Method –> routine name, select also Level –> 0. Perhaps easier to remember than the routine names, and taking advantage of the levels of understanding, you can also specify the Level –> lev:

```
chosen =Switch[lev,
 1, PlaceValue,
 2, PlaceValueDigits,
 3, PlaceValueBlock,
 4, PlaceValueIntegerName,
–4, PartialPlaceValue,
_, Null];
```

If none of these is used (say with Level –> 0) then the Method –> m option is used. If no method is in the list that All provided: $Failed

```
Options[PronounceInteger]
```

```
{Level → 1, Method → PlaceValue, Language → English, Speak → True,
  All → {PlaceValue, PlaceValueH, PlaceValueDigits, PlaceValueDigits$Language,
    PlaceValueBlock, PlaceValueIntegerName, PartialPlaceValue, PlaceValueTable}}
```

Who knows about PlaceValue or PlaceValueTable would tend to call them directly, but the following serves as an illustration. Observe the special role of the Language option. We may set the option to say “Italian”, and then PronounceInteger will call the routines with that setting. However, when the call uses Language → $Language, then the option is not passed on, and the option settings of the subroutines take precedence. PlaceValue has its own setting Language → “Amplish”, and thus we can still get the ampersand in smaller FontSize.

```
PronounceInteger[12]
```

```
·ten & two·
```

PlaceValueTable itself would not include the traditional pronunciation but this call via PronounceInteger inserts the option Speak → True. Note: “twelve” is “two left over after 10” and “ten & two” then is clearer.

```
PronounceInteger[12, Level → 0, Method → PlaceValueTable]
```

Integer	Place Value	PV Digits	Traditional
12	·ten & two·	10 & 2	twelve

5. Zig in German and tig in Dutch

5.1. Conflict of traditional and proper pronunciation

For English, traditional 19 = nine-teen and place value 90 = nine·ten (traditionally ninety) are not in conflict.

For German, traditional 19 = neun-zehn and place value 90 = neun·zehn (traditionally neunzig) are in conflict. Making a transition will cause much confusion. Dutch has the same problem.

5.2. A solution is to use zig in German and tig in Dutch for 10

Like English "-ty", German uses "-zig" at the end of 20, (30,) 40 through 90. Dutch has "-tig". Thus we may use *zig* for 10 in German and *tig* for 10 in Dutch. For me, this flash of insight in 2012 was a major step for accepting that there is scope for a full use of the place value system (in Holland). The changes in pronunciation are least with zig and tig.

June 2018 brought this new discovery for me: Van Bree (1987:259) explains that Dutch *-tig* derives from Proto-German or Gothic *tigus* ("multiple of ten") and Menninger (1958:139) explains this for German *-zig,* and translates *tigus* as "Zehnheit". For the English world perhaps a useful reference might be https://www.etymonline.com/word/-ty

Thus, what for me in 2012 was an insight using the existing regularity in names of numbers, actually derives from a structure that already existed and which is highlighted now. Potentially such structure was encouraged by the influence of Latin.

Germans or Dutch who have a problem with zig or tig - with the new phenomenon of "Zehnsucht" - and who still want to make full use of the place value system, could also consider borrowing "ten" from English, or use scientific "deca" (though with two syllables). Or Germans could use "tien" from Dutch, and the Dutch "zehn" from German. In Dutch "tig" already is used sometimes in the sense of "some unknown very big number" and the change to tig = 10 can still be made.

For the following, observe the usefulness of the trailing dot again, that helps identify the difference between "zig & neun" and "neun·zig". If you reading this, remember that the routines speak the numbers as well.

```
PlaceValueTable[{19, 90}, Language → "German"]
```

Integer	Place Value	PV Digits	Traditional
19	·zig & neun·	10 & 9	neunzehn
90	neun·zig	9 · 10	neunzig

```
PlaceValueTable[{19, 90}, Language → "Dutch"]
```

Integer	Place Value	PV Digits	Traditional
19	·tig & negen·	10 & 9	negentien
90	negen·tig	9 · 10	negentig

The German association "Zwanzig-eins" are not fundamentalists in terms of the full place value system.

```
PlaceValue[21, Speak → False, Simplify → False, Language → "German"]
```

zwei·zig & ein·ein

```
PlaceValue[21, Speak → False, Language → "German"]
```

zwei·zig & ·ein

5.3. Comparison with English

Remarkably, the changes for English could well be larger than for German. E.g. 60 changes from “sixty” to “six·ten”. An English speaking nation might prefer to change “ten” into “ty” and pronounce this as “t” or “tee” (like in Danish), which gives “six·ty” again. This would actually minimise the changes, even allowing for the special twenty, thirty, forty and fifty. English would also reintroduce the “and” again that German never abolished. Kids find English easier than the reversed order in German, Dutch and Danish, which only shows how important that order is, and which likely does not give evidence on the “and”. My impression is that English would rather stick to “ten” and accept the consequences. But, with the available routines, we may create an alternative “Englishty” and check how it sounds.

```
PlaceValue[11, Language → "Englishty", Show → True]
```

·tee <silence msec="300"/> & ·one

```
PlaceValueTable[{10, 11, 20, 21, 60, 61}, Language → "Englishty"]
```

Integer	Place Value	PV Digits	Traditional
10	·ty	10	ten
11	·ty & ·one	10 & 1	eleven
20	two·ty	2 · 10	twenty
21	two·ty & ·one	2 · 10 & 1	twenty one
60	six·ty	6 · 10	sixty
61	six·ty & ·one	6 · 10 & 1	sixty one

5.4. When 10 might be changed anyway

I was surprised to see that Finnish 10 is quite a long word. When adapting to the full use of the place value system anyway, one might imagine that there is a Finnish National Committee on the Pronunciation of 10 that chooses a shorter word.

```
PlaceValue[10, Language → "Finnish"]
```

PlaceValue: No list of rules found for language Finnish

PlaceValue: Initialisation with IntegerName and WordTranslation: please wait

PlaceValueRules: Warning: automated translation of Finnish and not carefully checked on conflicts and peculiarities

·kymmenen

5.5. Peculiarities in natural language

Current German makes a distinction between “eins” for 1 separately and “ein” for further use. Current German uses “sieben” for 7 separately, and uses “sieb” for combinations. Such deviations

happen in natural language but a systematic use has only one application. I took what currently are most frequent: ein and sieb. (Danish has “syv”.)

```
PlaceValue[77, Language → "German"]
```

sieb·zig & sieb·

The advantage of the routine with the pronunciation and transliteration now clearly shows, in that we do not just have the words (for writing essays) but also have an approximation how it would sound in rehearsing in class in (original) kindergarten (German word), to help judge whether the suggestion to adapt “ein” and “sieb” could be acceptable to consider.

```
PlaceValue[#, Language → "German"] & /@ Range[0, 10]
```

{null·, ·ein, zwei·, drei·, vier·, fuenf·, sechs·, sieb·, acht·, neun·, ·zig}

The present proposal clashes with some proposals at the “Zwanzigeins” movement on the numbers 0-20 that are important for education: zig for zehn, and thus also 11-19 using zig, and zwei·zig, to show the working of the place value system. Arbitrary in my implementation are: ein for eins, sieb for sieben, dreißig would also become drei·zig. See Colignatus (2015b, 2018a). Remember also my distinction between the math language in school and the dialect of current language outside of school. Earlier I suggested that the 21 movement would become stronger if it included the full place value system alongside their original views, as one of the issues to consider. It appears that this actually has happened, and the board of “Zwanzigeins” informed me that they support research in this direction: https://zwanzigeins.jetzt/infos/normung-der-aussprache

6. Translate, transliterate and quality control

6.1. Translate versus transliterate

To Speak in another language, the user can adapt the $Language in the Preferences of *Mathematica,* then restart. Both the numbers and words should be pronounced in the specified language, though in traditional manner.

There will be users who have *Mathematica* running under one specific language (e.g. $Language = “English”) and who still have interest in the pronunciation of the different languages. In that case, transliteration may help. The transliteration is provided in PlaceValueRules[“yourlanguage”] with the option “transliteratorlanguage” → {...}.

PlaceValue and PlaceValueRules can also call WordTranslation from English to automatically supply the words of another languague. The call PlaceValueRules[Set, “youlanguage”]. This does not provide for the transliteration. One is advised to save the resulting PlaceValueRules[“yourlanguage”] because the initialisation takes some moments. The words for 10^4 and 10^5 are generally missing and thus replaced by myriad and lakh.

6.2. Warning: automation does not guarantee quality

PlaceValueRules[Set, “youlanguage”] issues a **warning** when the submitted language isn't in the list PlaceValueRules["Checked"]. One may turn off the message or include a language in PlaceValueRules[“Checked”] provisionally, but it is better to actually check first (and then use AppendTo on

the list of checked languages). The package is conservative on giving the message but will repeat it on the tables, see below.

6.3. Checked in PlaceValueRules: English, German, French, Dutch and Danish

The package can be simple since we only require 2+10+6 = 18 transliterations per language, to pronounce the integers with absolute values less than 10^7.

The routine PlaceValue[*n*] takes the true words and transliterations from the PlaceValueRules["language"]. Implemented have only been English, German, French, Dutch and Danish, up to 10^7. They have been checked on name conflicts and language peculiars. Amplish and Englishty are variants of English that have a role in the exposition.

```
PlaceValueRules["Checked"]
```

{English, Amplish, Danish, Dutch, Englishty, French, German}

6.3.1. German

```
PlaceValue[9189, Speak → False, Language → "German"]
```

neun·tausend & ·hundert & acht·zig & neun·

```
PlaceValueRules["German"]
```

{True → {Negative → negativ, And → &, N → {null, ein, zwei, drei, vier, fuenf, sechs, sieb, acht, neun},
Power → {zig, hundert, tausend, myriad, lakh, million}}, English →
{Negative → nayguhteef, And → oond, N → {noul, ayen, tsvwaayie, dryie, fear, fuihnf, sex, sieb, akht, noin},
Power → {tzeeg, hoondert, tausand, myriad, lakh, meelyohn}}}

Above, we saw "zig" in German being transliterated by "tzeeg" in English, for the simple reason that the current setting of $Language is "English".

6.3.2. French

French can use *dix*.

```
PlaceValue[9189, Speak → False, Language → "French"]
```

neuf·mille & ·cent & huit·dix & neuf·

```
PlaceValueRules["French"]
```

{True → {Negative → negative, And → &, N → {zero, un, deux, trois, quatre, cinq, six, sept, huit, neuf},
Power → {dix, cent, mille, myriade, lakh, million}}, English → {Negative → nayguhteef,
And → ayh, N → {zeyrouh, uhnn, duh, trouwah, kahtruh, saink, sees, seht, weet, neuhf},
Power → {dees, sun, meel, meeri udd, lakh, meelyon}}}

6.3.3. Dutch

For Dutch, the design questions are alike those for German. The best solution would be to use "tig" for "tien".

```
PlaceValue[9189, Speak → False, Language → "Dutch"]
```

negen·duizend & ·honderd & acht·tig & negen·

```
PlaceValueRules["Dutch"] (*distinction between minus and min*)
```

{True → {Negative → min, And → &, N → {nul, een, twee, drie, vier, vijf, zes, zeven, acht, negen},
Power → {tig, honderd, duizend, myriad, lakh, miljoen}},
English → {Negative → min, And → n, N → {nuhl, ayn, tway, dree, fear, vaif, zes, zayven, ahgt, naigen},
Power → {tikgh, hon dairt, duuzand, myriad, lakh, milyoon}}}

6.3.4. Danish

Danish can use *ti.*

```
PlaceValue[9189, Speak → False, Language → "Danish"]
```

ni·tusind & ·hundrede & otte·ti & ni·

For Danish, I transliterated listening to Google Translate. I might not hear subtleties that native speakers would hear. The connective “and” is “og”, which reminds of German “auch” and Dutch “ook” meaning “also”. It is quite a diaspora.

```
PlaceValueRules["Danish"]
```

{True → {Negative → negativ, And → &,
N → {nul, en, to, tre, fire, fem, seks, syv, otte, ni}, Power → {ti, hundrede, tusind, myriade, lakh, million}},
English → {Negative → nayguhtew, And → oh, N → {nul, ayn, toh, trya, fear, fem, sex, suwe, ude, knee},
Power → {tee, hoonah, toosind, myriade, lakh, meelyohn}}}

6.4. Using non-implemented languages, e.g. Italian and Spanish

Please remember that there is no automated transliteration.

6.4.1. Italian

```
PlaceValueRules["Italian"] (*unknown*)
```

PlaceValueRules(Italian)

```
PlaceValue[134, Language → "Italian"]
```

PlaceValue: No list of rules found for language Italian

PlaceValue: Initialisation with IntegerName and WordTranslation: please wait

PlaceValueRules: Warning: automated translation of Italian and not carefully checked on conflicts and peculiarities

·cento e tre·dieci e quattro·

After this initialisation, the call is much faster. The user must provide a transliteration of “e” but “con” already provides a more Italian sound.

```
PlaceValue[134, Language → "Italian", And → "con"]
```

·cento con tre·dieci con quattro·

```
PlaceValueRules["Italian"]
```

{True → {Negative → negativo, And → e, N → {zero, uno, due, tre, quattro, cinque, sei, sette, otto, nove},
Power → {dieci, cento, cento, myriad, lakh, milione}},
English → {Negative → negativo, And → e, N → {zero, uno, due, tre, quattro, cinque, sei, sette, otto, nove},
Power → {dieci, cento, cento, myriad, lakh, milione}}}

6.4.2. Spanish

```
PlaceValueRules["Spanish"] (*unknown*)
```

PlaceValueRules(Spanish)

```
PlaceValue[134, Language → "Spanish"]
```

PlaceValue: No list of rules found for language Spanish

PlaceValue: Initialisation with IntegerName and WordTranslation: please wait

PlaceValueRules: Warning: automated translation of Spanish and not carefully checked on conflicts and peculiarities

·cien y tres·diez y cuatro·

After this initialisation, the call is much faster. A quick fix for pronunciation is perhaps this.

```
PlaceValue[134, Language → "Spanish", And → "e"]
```

·cien e tres·diez e cuatro·

```
PlaceValueRules["Spanish"]
```

{True → {Negative → negativo, And → y, N → {cero, uno, dos, tres, cuatro, cinco, seis, siete, ocho, nueve},
Power → {diez, cien, cien, myriad, lakh, millón}},
English → {Negative → negativo, And → y, N → {cero, uno, dos, tres, cuatro, cinco, seis, siete, ocho, nueve},
Power → {diez, cien, cien, myriad, lakh, millón}}}

6.5. A general comment

“Deux” in French has been transliterated here as “duh”. English has a closer sound in “bird” and “dirt”. I found no way to unpack this sound from “bird” and paste it into “d...h”.

```
res = WordData["bird", "PhoneticForm"]
```

b'ɜ·d

InputForm:

```
Style[b'ɜ·d, FontFamily -> Arial Unicode MS]
```

Submitting this to Speak generates a blank.

```
Speak["'ɜ·"]
```

```
SpokenString["'ɜ·"]
```

'ɜ·

```
Speak[%]
```

For transliteration it would be useful when such phonetic forms could also be submitted to Speak.

I don’t know anything about linguistics, and the following are some first discoveries.

In evolution, disregarding sign language, pronunciation came first, and writing was developed by encoding how someone pronounced something (beth = house with two rooms = B). In computer programming, the words and translations between languages were first, and the pronunciations came later. We take pronunciation as (conceptually) basic again since kids live in world of sound and only learn to read and write later on.

The International Phonetic Alphabet (IPA). Wikipedia (a portal and no source): https://en.wikipedia.org/wiki/International_Phonetic_Alphabet.

https://en.wikipedia.org/wiki/Orthography#Correspondence_with_pronunciation

See also: http://www.wolframalpha.com/input/?i=pronunciation+of+bird

7. Conversion tables

The routine PlaceValueTable allows the automatic generation of tables of conversion, and thus comes with a big warning. Automated translations must be scrutinised for flukes. However, creating such tables is quite useful to start with the required close reading. For these tables, the message that an automated translation hasn't been checked is a useful reminder at each call. Please do not turn this off, but do the checking and correcting, and then do an AppendTo[PlaceValueRules["Checked"], "yournewlanguage"]].

Standard conversion tables would be 0-10, 20-30, 30-40, etcetera. We can also consider other tables.

The table of 7 in English.

```
PlaceValueTable[Range[7, 70, 7]]
```

Integer	Place Value	PV Digits	Traditional
7	seven·	7	seven
14	·ten & four·	10 & 4	fourteen
21	two·ten & ·one	2 · 10 & 1	twenty one
28	two·ten & eight·	2 · 10 & 8	twenty eight
35	three·ten & five·	3 · 10 & 5	thirty five
42	four·ten & two·	4 · 10 & 2	forty two
49	four·ten & nine·	4 · 10 & 9	forty nine
56	five·ten & six·	5 · 10 & 6	fifty six
63	six·ten & three·	6 · 10 & 3	sixty three
70	seven·ten	7 · 10	seventy

The table of 7 in unchecked Spanish, installed above.

```
PlaceValueTable[Range[7, 70, 7], Language → "Spanish"]
```

PlaceValueRules: Warning: automated translation of Spanish and not carefully checked on conflicts and peculiarities

Integer	Place Value	PV Digits	Traditional
7	siete·	7	siete
14	·diez y cuatro·	10 & 4	catorce
21	dos·diez y ·uno	2 · 10 & 1	veintiuno
28	dos·diez y ocho·	2 · 10 & 8	veintiocho
35	tres·diez y cinco·	3 · 10 & 5	treinta y cinco
42	cuatro·diez y dos·	4 · 10 & 2	cuarenta y dos
49	cuatro·diez y nueve·	4 · 10 & 9	cuarenta y nueve
56	cinco·diez y seis·	5 · 10 & 6	cincuenta y seis
63	seis·diez y tres·	6 · 10 & 3	sesenta y tres
70	siete·diez	7 · 10	setenta

The package is conservative on the warning, and only generates it at installation and for these tables.

```
PlaceValue[1234, Language → "Spanish"]
```

·cien y dos·cien y tres·diez y cuatro·

The table of 7 in unchecked Swedish. The warning is issued twice: first at installation and secondly at creating the Table.

```
PlaceValueTable[Range[7, 70, 7], Language → "Swedish"]
```

PlaceValueRules: Warning: automated translation of Swedish and not carefully checked on conflicts and peculiarities

PlaceValue: No list of rules found for language Swedish

PlaceValue: Initialisation with IntegerName and WordTranslation: please wait

PlaceValueRules: Warning: automated translation of Swedish and not carefully checked on conflicts and peculiarities

Integer	Place Value	PV Digits	Traditional
7	sju·	7	sju
14	·tio och fyra·	10 & 4	fjorton
21	två·tio och ·ett	2 · 10 & 1	tjugoett
28	två·tio och åtta·	2 · 10 & 8	tjugoåtta
35	tre·tio och fem·	3 · 10 & 5	trettiofem
42	fyra·tio och två·	4 · 10 & 2	fyrtiotvå
49	fyra·tio och nio·	4 · 10 & 9	fyrtionio
56	fem·tio och sex·	5 · 10 & 6	femtiosex
63	sex·tio och tre·	6 · 10 & 3	sextiotre
70	sju·tio	7 · 10	sjuttio

8. ISO standard

8.1. ISO and NEN

Colignatus (2015b, 2018a) contains the definition for the full use of the place value system in a way that fits education. It should become an ISO standard, so that educators, researchs and textbook and software publishers have a persistent environment. This notebook with package implements this. The latter has some limitations on maximum place value million, and has some language particulars, but those are not material to the standard.

In Holland, NEN is the portal for International ISO: https://www.nen.nl/About-NEN.htm. They are no public utility so that the development of norms and standards must be paid for by users. They also have a crowd-funding option for such initiatives: https://nencrowd.nl. Before they allow their crowd-funding support, they want some guarantee that there is a community with an interest in the project. For this issue it would be researchers, the Ministry of Education and publishers. To my regret, I have met no interest yet.

8.2. Open source journals

Publication could be done in an open access journal, and the proper way to get open access journals is by the universities, see Gowers (2017) and my letter to VSNU, Colignatus (2016). Present

(paywall) journals tend to be specialised on approaches to education and psychology, and may require field testing on actual implementation, and apparently are not open to the particular approach of *redesign of mathematics education* that this notebook with package presents. Thus I observed that such journals can publish invalid research but will not publish or even report about this present proposal for improvement. Quis custodiet ipsos custodes ?

9. Conclusions

We already have the place value system in the numerals but not yet in pronunciation and written words. A unity of using the full place value system will work wonders in school. This is an issue of mathematics education, with the distinction between the language of arithmetic in school and its dialect of the common language outside of school. It is only a question whether the educational system would be willing to adapt.

Colignatus (2015b, 2018a) gives the definition for a full use of the place value system for kindergarten and elementary school. This is simple by the nature of the issue. My own change from (2015b) to (2018a) on the "&" connective shows that this simplicity still can be quite involved, but also that the key design features have been well-considered. This notebook with package implements this system. There must be some features in the package that are arbitrary. The transliteration is only provided for some quick indications and not the key part. The notebook with package looks meagre compared to the wealth and subtlety that already exist for the pronunciation of the natural languages and current conventions on pronouncing the numbers. The awkward ways of the past are being hardcoded now. Hopefully some of those resources can be used to support the full use of the place value system.

I have not seen this implementation of pronunciation elsewhere (except in Chinese though without the "and"), so please refer to these locations so that others can find the full documentation.

10. Appendix. The package

10.1. Key properties of PlaceValue

The PlaceValue routine (PronounceInteger with Level → 1) is the main routine. Key properties are:

- The Language → "language" option selects the language, and determines how integers are written and spoken.
- Implemented are English, German, French, Dutch and Danish, assuming that $Language (the transliterator) is English. Also given are Amplish (writes an ampersand in smaller FontSize) and Englishty (writing "ty" and speaking "tee" for 10).
- **Written output** is in the words of PlaceValueRules["language"], option True → {settings}.
- **Spoken output** is in the words of PlaceValueRules["language"], option "transliterator" → {settings}. These words are submitted to Speak. The transliterator that speaks this transliteration is $Language (often "English").

- For each language the user provides, in PlaceValueRules: (i) option True → {**words** for the minus sign, “&”, digits 0 ... 9 and values 10^1, ..., 10^6, in the relevant language}, and (ii) option “transliterator” → {**words** for the same input as above but now as a transliteration in terms of that transliterator language (commonly $Language)}. See for example PlaceValueRules[“German”].
- If PlaceValue is called for a language *lan* that is not in PlaceValueRules[] then it calls PlaceValueRules[Set, *lan*].
- SpeakTransliteration → "language" currently only works for $Language. Perhaps in the future it might be possible to have access to other languages without a restart.
- $Language has been set in the Preferences of *Mathematica.* The user can set to to another language and do a restart.

10.2. Overview

The package has:

```
? Cool`MathEd`PronounceInteger`*
```

▼ Cool`MathEd`PronounceInteger`

LatestPronunciation	PlaceValueDigits$Language	PronounceInteger
PartialPlaceValue	PlaceValueH	PronounceIntegerReadMe
PlaceValue	PlaceValueIntegerName	PronounceIntegers
PlaceValueBlock	PlaceValueRules	SpeakTransliteration
PlaceValueDigits	PlaceValueTable	

```
? PronounceIntegerReadMe
```

PronounceIntegerReadMe[] explains the package

```
PronounceIntegerReadMe[]
```

Mathematica already provides for (i) IntegerName[n, "language"], (ii) IntegerName[n, "DigitsWords"] for blocks of three digits, and (iii) WordTranslation for "negative", and (iv) a wealth of expression in IntegerString. These all encode tradition: with 56 = fifty−six instead of ten·five & six, except in IntegerString[n, "TraditionalChineseDecimal"]. Also, these all tend to hide the steps in the place value system that we want to show. Our routines are:

(1) PronounceInteger provides a switch between methods, and helps to focus on your preferred method

(2) PlaceValue (main routine): word based, so that the pronunciation is unambiguous for the particular language. It also allows transliteration (speaking foreign words with approximations in English)

(3) PlaceValueDigits: integer based, for when kids go from sounds to reading and writing numerals

(4)PlaceValueBlock: show in blocks of three digits (quite alike DigitsWords, but notice the differences).

(5) PlaceValueIntegerName builds around IntegerName DigitsWords for full place value speech

Likely, teaching is best done likely in order (2)−(5) so that first the individual steps are shown and then the block form. PM. A routine f stores what is spoken in LatestPronunciation[] and may display differently, see the documentation of f

The discussion in the notebook also uses - but not directly on pronunciation:

```
? Cool`MathEd`CCSSMath`*
```

▼ Cool`MathEd`CCSSMath`

CCSSMathContent	CCSSMathExercise

```
? CCSSMathExercise
```

CCSSMathExercise[n, ..., m, opts] passes the numbers on to a routine that recognises the
CCSSMathContent labels in opts, see Options[CCSSMathExercise] and CCSSMathContent.
When Show –> True then the CCSSMathContent on those labels is shown (default False)
CCSSMathExercice[n, m:10] for "Grade" –> "Kindergarten" or "Grade 1"
counts from m to n. Defaults Speak –> False, IntegerDigits –> False (if
True, the exercise question shows digits, but still speaks full place value)

10.3. Switching routine

The PronounceInteger routine and its options were already shown above.

```
PronounceInteger[12, Level → 0, Method → Random]
```

PronounceInteger: Unknown Level –> 0 and Method –> Random

$Failed

10.4. Level 1

```
? PlaceValue
```

PlaceValue[n] gives n in the full place value system, as text or in
Speak, and the latter with pauses for hearing and possible transliteration in
the system default language (e.g. $Language = English). n is in (–10^7, 10^7).
The routine is also called with PronounceInteger[n, Level –> 1]
Options with defaults are:
Language –> "Amplish" uses PlaceValueRules["Amplish"], see there
WordTranslation –> Automatic calls IntegerName and
WordTranslation when the language is not available in PlaceValueRules, which
takes some time for initialisation and does not provide for transliteration
SpeakTransliteration :> $Language (often "English"), see there
Speak –> True,
Length –> 300, an integer for the duration of pause between the positions,
Join –> "·" (unpronounced center dot) between the texts of the weight and place value
Simplify –> True eliminates terms with 0 weights and replaces 1 P by P, for P = 1, 10, ..., 10^6
Show –> False does not show the input for Speak, so that one reads the words
in the language itself and hears the transliteration (while reading affects hearing)
PronounceInteger[Options, "Reset"] resets the options to loading time

```
Options[PlaceValue]
```

{Join → ·, Language → Amplish, Length → 300, Show → False, Simplify → True,
Speak → True, WordTranslation → Automatic, SpeakTransliteration :→ $Language}

```
? SpeakTransliteration
```

SpeakTransliteration is an option of PlaceValue with default $Language. It selects the relevant setting from the PlaceValueRules. If the Language option is equal to the transliterator ($Language) then writing and speaking will be done from the True rule. When the option Language is set to another value, then the integers are written from True but spoken by the transliterator

```
? PlaceValueRules
```

PlaceValueRules["Language"] = {True –> t, "OtherLanguage" –> f}, with t = {Negative –> string, And –> "&", N –> {strings for 0...9}, Power –> {strings for 10, ..., 10^6}}, for the true way of writing and pronunciation (when this original language = $Language), and f = {... similar ...} for the transliteration into the actual $Language. See the examples for "English" (f is not required given $Language = "English") and "Dutch" (t <> f).
Negative –> how the sign of a negative number is written or pronounced
And–> what string is used to separate the product terms of weights and place values
N –> {names how the digits 0 ... 9 are written or pronounced}
Power –> {names how 10 ... 10^6 are written or pronounced}.
See SpeakTransliteration for checking on $Language.
Implemented are English, German, French, Dutch and Danish (with some guessing on transliteration). The language Amplish has True –> with an ampersand in a smaller font, and otherwise English. Englishty has ty for 10 pronounced as tee. The languages have (Greek) myriad for 10^4 and (Indian) lakh for 10^5
PlaceValueRules[] contains the list of languages for which there is transliteration
PlaceValueRules[Select, item, lan, translit:"English"] selects for lan the item (e.g. Power) from both True and translit, and gives a rule or list of rules (true –> transliteration) for replacement
PlaceValueRules[Set, "ToLanguage", FromLanguage:"English"] uses IntegerName for the digits and WordTranslation for "negative", "and" and "million" and imposes myriad and lakh. These routines may take time, and thus one is advised to save the result. There is no automation for the transliteration
PlaceValueRules[Check, "Language"] for another language than in PlaceValueRules["Checked"] will issue a warning that the place value pronunciation is generated by automation and hasn't been checked on conflicts. When you have done the checking then this list can be adapted manually or by AppendTo

```
PlaceValueRules[]
```

{Amplish, Danish, Dutch, Englishty, French, German, Finnish, Italian, Spanish, Swedish}

```
PlaceValueRules["Checked"]
```

{English, Amplish, Danish, Dutch, Englishty, French, German}

```
PlaceValueRules[Select, Power, "French"]
```

{dix → dees, cent → sun, mille → meel, myriade → meeri udd, lakh → lakh, million → meelyon}

10.5. Level 2

```
? PlaceValueDigits
```

PlaceValueDigits[n] pronounces n by PlaceValue[n] and displays by PlaceValueDigits$Language[n]
Options are passed on to said routines, and see their own options, like Simplify and Show
Own options are Speak –> True (default), Print –> True (default) prints
    the outcome of PlaceValue[n]. LatestPronunciation[Print] stores what is printed
The routine is also called with PronounceInteger[n, Level –> 2]

```
Options[PlaceValueDigits]
```

{Speak → True, Print → True}

```
? PlaceValueDigits$Language
```

PlaceValueDigits$Language[n] pronounces n by using digits for the place values rather than words
    (PlaceValue): n can be as large as Mathematica allows, but also relies on how $Language
    pronounces those integers. In English 2 100 is pronounced as "two one–hundred". Option
    Simplify –> True (default) for handling of 0 and 1 weights, StepMonitor –> False (default)
    whether to show an intermediate step, Length –> 300 (default) for the pause, Join –>
    "·" (default) for connection of terms in display (not spoken), Speak –> True (default).
The routine is also called as a subroutine by PlaceValueDigits[n],
    which is also called by PronounceInteger[n, Level –> 2]

```
Options[PlaceValueDigits$Language]
```

{And → &, Join → ·, Length → 300, Simplify → True, Speak → True, StepMonitor → False}

```
PlaceValueDigits$Language[22 324 443 224, And → ",", Speak → False]
```

2 · 10000000000 · 2 · 1000000000 · 3 · 100000000 · 2 · 10000000
· 4 · 1000000 · 4 · 100000 · 4 · 10000 · 3 · 1000 · 2 · 100 · 2 · 10 · 4

10.6. Level 3

```
? PlaceValueBlock
```

PlaceValueBlock[n] pronounces n in digit blocks of 3, and within those blocks in place value manner. The routine is also called with PronounceInteger[n, Level –> 3]
For example 123456 = {{1 hundred & 2 ten & 3}, thousand} & {4 hundred & 5 ten & 6}. This is like common usage except for the integers in 11–99. Option Length –> 300 (default) puts "&" in the middle of that pause. Option Power –> {...} by default replaces 100 by hundred, 1000 by thousand and 10^6 by million. (This is not done in PlaceValueDigits$Language, because there are more words, like for the digits, and including the awkward 10^4 and 10^5). The numbers 11–99 use the options of PlaceValueDigits$Language, namely to display in numbers and not words. If your $Language is not "English", then you might consider to adapt the settings manually. The following is an automation though:
PlaceValueBlock[SetOptions, "language"] sets the options to stated "language", using WordTranslation from the current language in the Options (default "English")
PlaceValueBlock[Options, "Reset"] resets to the Options at loading

```
Options[PlaceValueBlock]
```

```
{And → &, Join → ·, Language → Amplish, Length → 300,
  Power → {1 000 000 → million, 1000 → thousand, 100 → hundred}, Simplify → True, Speak → True}
```

```
PlaceValueBlock[123 456]
```

```
{{{1 · 100,  & , 2 · 10   &   3}, 1000},  & , {4 · 100,  & , 5 · 10   &   6}}
```

```
LatestPronunciation[]
```

```
{{{1 · hundred,  & , 2 · 10   &   3}, thousand},  & , {4 · hundred,  & , 5 · 10   &   6}}
```

10.7. Level 4

```
? PlaceValueIntegerName
```

PlaceValueIntegerName[n] lets IntegerName[n, "DigitsWords"] make the blocks of 3 digits, and then calls PlaceValue on these digits. IntegerName DigitsWords uses text for Abs[n] < 1000, and we follow this. The block bases of thousand, million etcetera are translated by WordTranslation. The text displayed has blocks of digits, with block bases separated by option And –> ", " (default). This option is not passed on to PlaceValue for speech of the blocks. The spoken text can be recovered from LatestPronunciation[]. Transliteration is called but will slow the execution
The routine is also called with PronounceInteger[n, Level –> 4]

```
Options[PlaceValueIntegerName]
```

```
{Language → English, Speak → True, And → , }
```

```
PlaceValueIntegerName[1 234 451 011]
```

```
1  billion,  234  million,  451  thousand,  11
```

```
LatestPronunciation[]
```

·one billion, two·hundred & three·ten & four·
million, four·hundred & five·ten & ·one thousand, ·ten & ·one

10.8. Level -4

```
? PartialPlaceValue
```

PartialPlaceValue[n] has the traditional pronunciation and words of the numbers 11–99, thus in opposite order of the place value system, when speaking the number from Left To Right, and morphing of words. It calls IntegerName with DigitsWords. If Abs[n] < 1000 then there are words only, and higher n get combinations of both digit blocks and words for digit block values (thousand, million). Option defaults are Language –> "English" and Speak –> True. Speak apparently fails on the hyphen and thus this replaced by a space. Speak[–10] says Minus 10 but IntegerName uses Negative 10. There is no transliteration.
The routine is also called with PronounceInteger[n, Level –> –4]

```
Options[PartialPlaceValue]
```

{Language → English, Speak → True}

```
PartialPlaceValue[1 234 451 011]
```

1 billion 234 million 451 thousand 11

```
LatestPronunciation[] (*in traditional manner*)
```

1 billion 234 million 451 thousand 11

11. Literature

Thomas Colignatus is the scientific name of Thomas Cool, econometrician and teacher of mathematics, Scheveningen, Holland.

Bree, C. van (1987), *Historische grammatica van het Nederlands,* Foris Publications, https://dbnl.org/tekst/bree001hist02_01/bree001hist02_01.pdf

Bussi, M. G. B. & Sun X. H. (eds) (2018), *Building the Foundation: Whole Numbers in the Primary Grades,*The 23rd ICMI Study, Springer

Colignatus, Th. (2015a), *A child wants nice and no mean numbers*,
(1) website: http://thomascool.eu/Papers/NiceNumbers/Index.html
(2) PDF on Zenodo: https://zenodo.org/record/291979
(3) Publisher for a print: https://www.mijnbestseller.nl/shop/index.php/catalog/product/view/id/118082/s/a-child-wants-nice-and-no-mean-numbers-79074-www-mijnbestseller-nl/

Colignatus, Th. (2015b, 2018a), *The need for a standard for the mathematical pronunciation of the natural numbers. Suggested principles of design. Implementation for English, German, French, Dutch and Danish*, https://doi.org/10.5281/zenodo.774866

Colignatus, Th. (2016), *Letter to VSNU and others on membership dues and open access publishing,*

https://boycottholland.wordpress.com/2016/10/12/letter-to-vsnu-and-others-on-membership-dues-and-open-access-publishing/

Colignatus, Th. (2018b), *Tables for addition and subtraction with better use of the place value system,* PDF https://doi.org/10.5281/zenodo.1241349 and notebook with package https://-doi.org/10.5281/zenodo.1241404 and http://community.wolfram.com/groups/-/m/t/1313408 (update May 5)

Common Core Standards (2018), *Number & Operations in Base Ten,* http://www.corestandards.org/-Math/Content/NBT/ with also *Understand place value*

Ejersbo, L. R., M. Misfeldt (2015), "The relationship between number names and number concepts", Paper presented at ICMI Sudy-23, Macau SAR, China. Included in Sun et al. (eds) (2015)

Fateman, R. (2013), *How can we speak math?,* https://people.eecs.berkeley.edu/~fateman/papers/s-peakmath.pdf

Gowers, T. (2017), *The end of an error?*, https://www.the-tls.co.uk/articles/public/the-end-of-an-error-peer-review/

Menninger, K. (1958), *Zahlwort und Ziffer: eine Kulturgeschichte der Zahl,* Volumes 1-2, Göttingen: Vandenhoeck und Ruprecht 1979

Sun, X., Kaur, B., & Novotna, J. (eds) (2015), "Conference proceedings of the ICMI study 23: Primary mathematics study on whole numbers", www.umac.mo/fed/ICMI23/doc/Proceedings_ICMI_S-TUDY_23_final.pdf

Uy, F.L. (2003), *The Chinese Numeration System and Place Value*, Teachting Children Mathematics, NCTM, p244-247

Wolfram Research, Inc. (2017), *Mathematica*, Version 11.2, http://www.wolfram.com