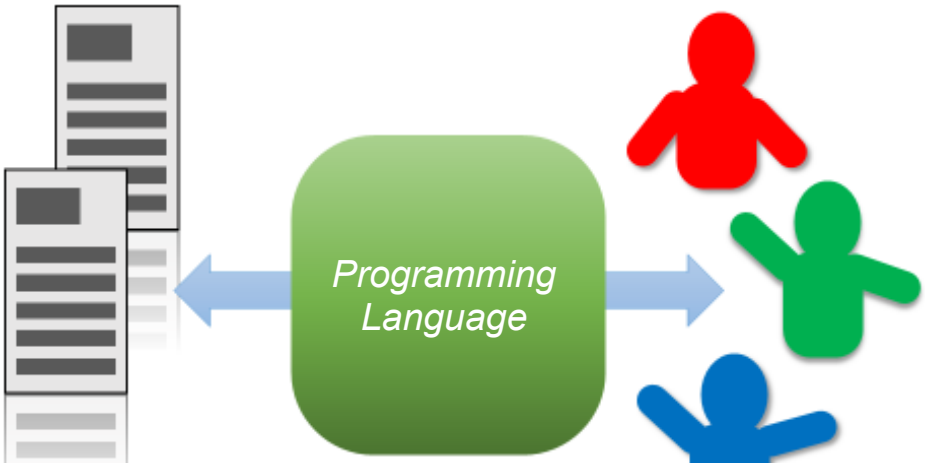
Programming Practice Python 2023

Column Edition



Hardware is stubborn.
Human beings are selfish.
These two are connected by programming languages.

Hajime Kita, Institute for Liberal Arts and Sciences
Yoshitaka Morimura, Institute for Information Management and Communication
Masako Okamoto, Center for Promotion of Excellence in Higher Education
Kyoto University

Version 2023/10/15

Table of Contents

[1] The text on the cover is a parody of line no. 48 of Tetsudo Shoka, Tokaido Edition, a railway song that begins with "Pulling out of Shimbashi with a whistle." Among all 66 lines leading up to Kobe, there are 9 lines that sing of Kyoto (including Yamashina)—lines 45 to 53. Tetsudo Shoka, Tokaido Edition, was written in 1900, but unfortunately Kyoto Imperial University, which was founded in 1897, does not appear in the lyrics.

0. Column: Starting from 0

0.1 Python starts at 0

In the range function, which is used to generate a range of numbers in Python, range(5) will generate 5 integers from 0 to 4 (0, 1, 2, 3, 4). Also, the first element in the list a = [0, 1, 2] can be referenced by typing a[0], where 0 is the index. This may feel unusual to users with no experience of other programming languages such as C.

0.2 Shouldn’t 1 be the beginning?

We start counting from 1 when we count objects in our daily lives. This method is convenient as the last number will be the total number of objects when you finish counting.

So why start from 0?

Suppose that there are locations to store data for Mr. 1 to Mr. 10 and these neighbors are assigned house numbers. The data is first stored in the address of Mr. 1, and if that address is *a*, then the addresses of Mr. 2 to Mr. 10 will be a+1 to a+9. The numerical value you add to your number and Mr. 1's address is off by one. If you accept the premise of Mr. 0 to Mr. 9, the address of n can be calculated neatly as a+n.

Also consider counting up to 10. The sequence is 1, 2, 3,…, 10, but the last 10 has a carry over and requires 2 digits. There are 10 numbers that can be handled by 1 digit, including 0, but while counting from 1 allows 0 to be left unused, 2 digits end up being required to represent 10 at the end.

For this reason, it is convenient for computers, which are machines that aid us in calculations, to start at 0. It has become customary to start with 0 in many programming languages, such as C. However, in the language Fortran, which was created in the early days of programming languages and is still used in scientific computing, the subscript of an array (a data format in which elements can be manipulated by subscripts like a list in Python) starts with 1.

Even in the world of mathematics, it seems that some mathematicians include 0 in the definition of natural numbers and others do not.

0.3 The take home:

There is a saying, “When in Rome, do as the Romans do.” It's a shortcut to resign yourself to this and get used to it, but you must use it with caution in areas of applications that have a culture of using subscripts starting with 1, such as mathematics.

1. Column: What Does Float Mean?

1.1 Floating-point numbers

The function that converts decimal points to data that can be interpreted in Python is float(). It's a word that implies to be gently buoyant, so the meaning isn't immediately obvious. In fact, the type of data that decimal numbers are handled as is called a floating-point number, so the first word of that term is the origin of this function name. However, even by calling this a "floating" "decimal point," beginners may not understand what it refers to. The opposite of a floating-point number is a fixed point number.

A fixed point number is a numerical expression with a fixed number of digits after the decimal point. For example, a fixed point number with 8 digits in the integer part and 4 digits after the decimal point can be expressed as ± 00000000.0000 where the 0's could each be any number from 0-9.

However, scientific computing uses very small absolute values, such as the Planck constant (about 6.62×10^{-34} Js) and very large numbers, such as Avogadro's number (about 6.02×10^{23} mol^{-1}). It is difficult for fixed point numbers to express such numerical values, and as shown in the notation example, it is effect to express this as the product of "a number (significand) in which the integer part is expressed as one digit" and "a power (exponent) of 10 (base)." Such an expression is also used in computers and is called a floating-point number. That is, it means a numerical value in which the actual position of the decimal point fluctuates unsteadily.

Since it is complex to express the exponent of the power of 10 with superscripts on the display of a computer (or scientific calculator), the numerical parts of the previous Planck constant and Avogadro's number are written as follows:

6.62E-34

6.02E23

When computers had low processing power, it was common to use 4 bytes to represent floating-point numbers (single precision floating-point numbers). However, this has few significant figures in the significand, and has problems in terms of practical accuracy. Therefore, 8 bytes are often used to express floating-point numbers with sufficient precision for practical use; 8-byte floating-point numbers are called a double precision floating-point number. Floating point arithmetic in C was implemented with double-precision arithmetic as the standard, but in the name of the data type at that time, "float" is used to mean single-precision and "double" is used to mean double-precision floating point (numbers). Despite "float" seeming like gibberish, some people may have an even harder time comprehending "double."

Double-precision floating-point numbers are commonly used in Python, but the function for converting strings to double-precision floating-point numbers is called "float."

1.2 The downsides of working with floating-point numbers in binary

In order to realize high-speed calculations, binary numbers (which have a base of 2) are widely used in computer hardware instead of decimal numbers (which have a base of 10). When dealing with integers, there is essentially no difference between decimal and binary numbers, but this is not the case for decimal numbers.

In base-10 fractions, 1/3 becomes 0.33333… and cannot be represented by a finite number of digits, which means that base-10 numbers are expressed as the sum of fractions to represent each digit, where 1/10, 1/100, 1/1000…, are each multiplied by a number from 0 to 9 then added together. Although there appear to be many people who feel uncomfortable when they learn this in elementary school, you will eventually learn to get used to it.

The sizes of each digit in binary written in base-10 would be 1/2, 1/4, 1/8, 1/16..., and the numerical value of a binary string is represented by the sum of these fractions after they are each separately multiplied by either 0 or 1. For this reason, 0.1, which is often expressed in decimal numbers, cannot be accurately represented in binary numbers. The discomfort experienced in elementary school arises handling base-10 numbers in our daily lives, which is in turn due to the use of computers.

In fact, if you try the following formula that evaluates whether a three-time addition of 0.1 in Python is equal to 0.3, then:

```
0.1 + 0.1 + 0.1 == 0.3
```

Its value will be False[1].

In scientific and technical calculations, it is not a significant problem that 0.1 is an exact value in most cases, as long as it is sufficiently accurate[1]. However, in accounting calculations that deal with money, while handling a considerable number of digits as the amount of money, the interest rate is often treated as a decimal number such as 0.01 (1%). For this reason, it becomes necessary to avoid not being able to accurately represent base-10 decimals. In order to use a computer for accounting, it is necessary to calculate base-10 numbers inside the computer; Python provides a module called “decimal” that operates using exact decimal numbers.

Reference

[1] 15. Floating Point Arithmetic, Its Problems and Limitations, Python Tutorial
https://docs.python.jp/3/tutorial/floatingpoint.html

[1] Since science and technology deal with actual physical existence, it is only necessary to express the achievable accuracy to a certain extent. Picture this as being just about as accurate as you would be if you were to make a rod with an exact length of 1 m.

2. Column: Newton's Method

In the first part of this course, you did calculations to find approximations of square roots. The approximate value x_i of the square root $\sqrt{a}$ of a number a can be updated using the formula (recurrence relation) below:

$$x_{i+1} = \frac{x_i + a/x_i}{2}$$

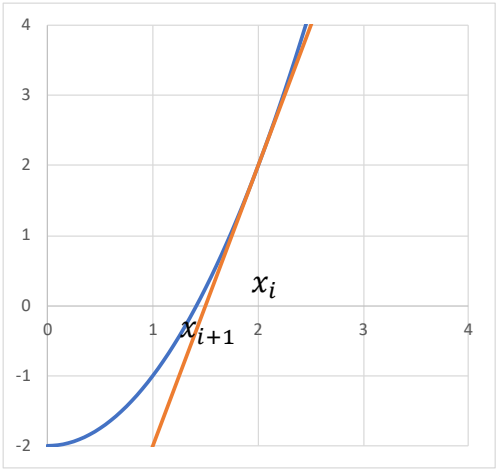
The formula is simple and converges quickly, but it is inconvenient to calculate by hand or using a calculator because it is necessary to record the reciprocal.

2.1 Newton's method

We gave an intuitive explanation about this method in the lesson materials; this is an equation for finding the square root of a:

$$f(x) = x^2 - a = 0$$

It is an algorithm that finds the solution to the above equation called Newton's method (the origin of the name is, naturally, Isaac Newton. It is also called the Newton-Raphson method).



In Newton's method, the tangent line of $f(x)$ at the point $(x_i, f(x_i))$ on $f(x)$ is drawn for the approximate value x_i of the equation $f(x) = 0$, and the point of intersection between it and the x axis is set as the next approximate value. The equation for the tangent line is

$$y - f(x_i) = f'(x_i)(x - x_i)$$

Meaning if you solve x as $y = 0$

$$x = x_i - \frac{f(x_i)}{f'(x_i)}$$

is derived. This is posited as x_{i+1}, so

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

is derived. To find the square root, apply $f(x) = x^2 - a$ to get the recurrence relation at the beginning.

$$x_{i+1} = x_i - \frac{x_i^2 - a}{2x_i} = \frac{x_i + a/x_i}{2}$$

2.2 Calculating the n-th root

Incidentally, to find the n-th root,

$$f(x) = x^n - a = 0$$

as per the above, by obtaining the recurrence relation

$$x_{i+1} = \frac{(n-1)x_i + a/x_i^{n-1}}{n}$$

is derived. This is the point obtained by internally dividing the approximate value x_i and another approximate value a/x_i^{n-1} calculated from it into $n-1:1$. Programming calculations with this formula is straight forward, so try attempting it yourselves.

2.3 Equal temperament

You may not be familiar with the n-th root, but the 12th root is used in musical scales. Sounds that are one octave apart in the scale have twice the frequency. In the scale called equal temperament, which is fundamentally used in music, one octave is divided into 12 semitones, but the pitches that differ by one semitone are configured so that the frequency ratio differs by the 12th root of 2 (about 1.059). In order for chords to resound beautifully, it is desirable that the pitches have an integer ratio, and the scale created in this way is called just intonation, but the frequency ratio between semitones varies, creating the problem of making it difficult to change key. Equal temperament is a tuning system that is easy to use, at the cost of allowing some turbidity in chords due to approximating the step size.

3. Column: Relative Accuracy

3.1 Numerical accuracy

In science, people have tended to pay attention to significant figures when calculating. This is because in the natural sciences, the measured values obtained by experiments and observations contain errors. For example, the fact that the radius of a circle is measured and expressed as a value of $r = 3.456\,\mathrm{m}$ means that up to the numerical value "6," which is three digits after the decimal point, is meaningful. That is, because the digits below it are rounded off, the measured value r is expressed as

$$3.4555\,\mathrm{m} \leq r < 3.4565\,\mathrm{m}$$

thus, the width of uncertainty is $3.4565\,\mathrm{m} - 3.4555\,\mathrm{m} = 0.001\,\mathrm{m}$, i.e., 1 mm.

If the same measurement method is used to measure the same range, there will be an error of about 1 mm.

On the other hand, the length of the outer circumference of this circle is given by $L = 2\pi r$, so calculating it gives

$$21.7115\mathrm{m} \leq L < 21.7178\,\mathrm{m}$$

and the measurement error is $0.001 \times 2\pi \sim 0.00628$. In this case, the absolute value of the error changes because it is multiplied by an accurate constant 2π, but the ratio to the value L does not change from 3.456 : 0.001. This is the reason why it is important to pay attention to the number of significant digits in science calculations. Instead of examining the ratio accurately in order to make calculations easier, we are thinking in terms of the number of digits.

3.2 Absolute and relative accuracy

In the calculation of numerical values using a computer, approximate calculation is performed using a sequence of numbers that converges to a true value. In this case as well, where to terminate the calculation depends on the required accuracy of the calculation.

For example, consider calculating an approximation of a using a sequence $a_1, a_2, \cdots \to a$ that converges to a true value. If the lower bound a_i^L and upper bound a_i^U of a_i are also given, the precision of a_i is

$$a_i^U - a_i^L$$

Here, if this value is less than 0.001, the calculation ends. That is,

$$a_i^U - a_i^L \leq 0.001$$

If so, the accuracy of the calculation is an absolute value of 0.001.

However, considering the calculation of the square root, for example, the square root of 10000 is 100, so the absolute precision of 0.001 is quite high, while the square root of 1/10000 is 1/100 = 0.01 in comparison. This means an accuracy of 0.001 is pretty undesirable.

Therefore, if a certain percentage (for example, 0.001) of accuracy is required for the value obtained by calculation, the determination condition for that is

$$a_i^U - a_i^L \leq a \times 0.001$$

and if you divide both sides by a

$$\frac{a_i^U - a_i^L}{a} \leq 0.001$$

it will be determined accordingly. However, since the true value a is not known, the calculated value a_i actually obtained is used

$$\frac{a_i^U - a_i^L}{a_i} \leq 0.001$$

which is what it will be determined as. Completing the calculation with relative accuracy in this way can be said to apply the idea of significant figures performed in scientific calculations to the approximate calculation of numbers.

4. Column: Parameters and Arguments

Consider an example of calling a function in Python.

```
# This is the function definition
def square(x):
    return x*x

# This is the caller
y = 2
y2 = square(y)
print(y2)
```

The function square() takes a parameter x, and the caller calls the function square() with the variable y as the argument. The parameters in the function definition are also called “formal arguments,” and the caller's arguments are also called “actual parameters.”

I retain the names parameter (or formal argument) and argument (or actual parameter), but beginners may be confused by the different names of the arguments/parameters on the side of the caller and one being called.

However, the same sort of ideas is used in the real world.

For example, consider ordering food in a restaurant. A conversation such as the following is often heard:

> **Waiter:** Welcome, can I take your order?
>
> **Customer B:** The pork cutlet set meal
>
> **Waiter:** The pork cutlet set meal? Understood.
>
> The waiter heads to the kitchen
>
> **Waiter:** Pork cutlet set meal for table 3
>
> **Chef:** Table 3, pork cutlet set meal - coming up

As a matter of fact, for restaurant employees (unless they are referring to very familiar customers), the customer's name doesn't matter. Therefore, they use the expression “table x” to identify customers. This can be called a parameter (or formal arguments) in the restaurant.

By using parameters, the restaurant can standardize the execution of work and improve efficiency. It doesn't matter how the customer is identified in the restaurant as long as it is well concealed[1].

Programming languages have been developed based on how easy it is for people to write instructions that operate computers. As a result, it is thought that a method similar to human behavior in the real world is used.

[1] Some people may not feel very good about being called “table 3” should they hear it. If you fly business class on an airplane, you may be addressed with your real name by the flight attendant (tickets are purchased with the passenger’s real name for operational reasons). This is a service that takes good care of the customer. How do you feel about it for college classes?

5. Column: Scope of Variables

The extent to which the variables used in a programming language are visible is called **“the scope of variables.”**

Consider the following in the real world.

When creating materials, a great deal of memos are generated along the way. Suppose you ask one of your subordinates to create part of the materials.

Superior A: Excuse me, Mr. B, could you create Figure 3 of this document?

Subordinate B: Certainly. Working at your workbench makes it easier to refer to materials, so can I work there?

Mr. A: That's fine. Go ahead and use it.

Mr. B completes Figure 3 while creating various memos.

Mr. B: Mr. A, Figure 3 is ready. I’ll get back to my original work now.

Mr. A: Thank you, that’s really helped me out.

Mr. B leaves.

Mr. A: He’s gone and left the work memos as they are, so now it’s difficult to tell them apart from the original material... That’s a problem.

So, what should Mr. B have done? If he had done the following, he wouldn’t have been a burden to Mr. A.

Throw away the work memos he made after the job is completed.

To do so, use paper especially meant for memos in order to tell Mr. A's material and the memos apart.

Alternatively, leave memos on a workbench without Mr. A’s materials.

5.1 Local variables = variables that get discarded

In programming languages, many jobs are “subcontracted” by calling functions. On this occasion, the variables used in the function corresponding to Mr. B's memos in the previous example are reserved separately as local variables and discarded at the end. Local variables are not visible from outside the function.

5.2 The scope of variables in Python

The scope of variables depends on the programming language. In Python,

- Variables (local variables) in the function cannot be seen from the outside.
- Global variables defined outside of a function can only be read from within the function.
- To write to a global variable from within a function, you must declare it with a global statement.
- The target variable of the 'for' statement (for example, i, in for statement '`for i in range(5):`') remains even after the 'for' statement ends.

6. Column: Random Numbers

6.1 Computers and random numbers

A computer performs exactly what is written in a program. Considering that a computer generates a series of numbers, the value of the series to be generated next can be accurately predicted from the program and the values of variables. On the other hand, a random number sequence is a sequence of "haphazard" numbers. Therefore, unless you introduce random elements from an external source, you cannot essentially generate a random number sequence within the program.

A computer-generated sequence of random numbers is a sequence that "appears random" and strictly should be called a "pseudorandom number sequence." The quality of the generated random number sequence is determined by the algorithm, but various tests will examine with what kind of meaning a good random number sequence is generated.

Random numbers are used in many areas in computer applications. Games, for one, are likely familiar to you, and they are often used in simulations within academic research. When generating a random number sequence, the question is whether or not the random number sequence can be reproduced. In a game, if you can reproduce the series, it ends up being less interesting. Therefore, a computer clock is used as the initial value for generating a random number series. Alternatively, in academic research, if the same series cannot be reproduced, work efficiency decreases and other people are unable to retest it. Therefore, we explicitly give an initial value to create a random number sequence.

6.2 Random numbers you will want to use

The random numbers that you will want to use when writing various programs are as follows.

- Random integers
  - Uniform random numbers which are random numbers that occur with the same probability in a certain range like a dice roll
  - Random numbers that follow a particular distribution, such as binomial or Poisson distributions
- Random floating-point numbers
  - Uniform random numbers
  - Random numbers that follow a specific distribution, typically a normal distribution, but random numbers that follow a variety of distributions are also required.

- Random order
  There are times when you want to randomly order a list, etc., or select a certain number randomly and without duplication, such as by lottery.

6.3 Using the random module

There is a “random” module that functions as a module for generating random numbers in Python. Random numbers are generated using a well-tested method called the Mersenne Twister. That said, it is recommended to use the more secure “secrets” module for security purposes.

6.3.1 Giving a “seed” for random numbers.

- random.seed()

A function that gives a seed to generate a random number sequence. The system clock is used as the seed if there is no argument. By giving an integer type argument, it will be used as a seed to initiate a random number sequence. For example, in the following examples, randrange(10), which generates random numbers from 0 to 9 by calling seed(), returns different values.

```
import random
random.seed()
random.randrange(10)
2
random.seed()
random.randrange(10)
3
```

On the other hand, explicitly giving an argument to seed() results in the same random numbers being returned.

```
import random
random.seed(1)
random.randrange(10)
2
random.seed(1)
random.randrange(10)
2
```

6.3.2 Creating an integer random number

- random.randrange(stop)

The argument is the same as the range() function, which generates an integer random number in the range generated by the range() function. For example, random.randrange(5) will generate a random number from 0 to 4.

- random.randint(a, b)

Generates an integer random number from a to b. Unlike the randrange() function, both a and b are included in the generated random numbers.

6.3.3 Creating a random order

- ramdom.sample(population, k)

Produce random order of size k from the population. The range() function and list can be used for the population. The result will be a randomly ordered list and the list given as population is unchanged.

```
import random
a = list(range(6))
a
[0, 1, 2, 3, 4, 5]
b = random.sample(a, 6)
b
[4, 0, 2, 5, 1, 3]
a
[0, 1, 2, 3, 4, 5]
```

6.3.4 Creating a real random number

- random.random()

Create a real random number greater than or equal to 0 and less than 1.

- random.gauss(mu, sigma)

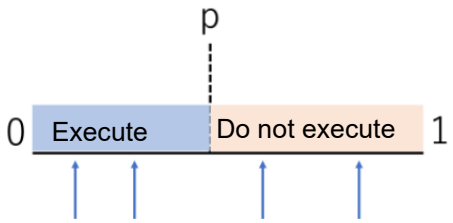
Generates random numbers that follow a normal distribution with mean mu and standard deviation sigma (also known as Gaussian distribution after the mathematician Gauss).

6.4 Executing with a certain probability

You may want to do something with a certain probability in the program (let's call it the value of a variable called p). On that occasion, program as follows.

if (p > random.random()):

The block you wish to execute



Random numbers generated by random.random()

6.5 Random number egneration with NumPy

NumPy, a library that supports advanced numerical calculations, can generate many random numbers at once, which improves calculation efficiency. Many functions are defined. Some of them are as below. Note that the name is different from the random module. In the example below, NumPy is customarily imported under the name np.

- numpy.random.seed: A function that gives a seed for random numbers
- numpy.permutation: By giving an array, it creates a new array that rearranges it.
- numpy.random.rand: Generates a specified number of random numbers that follow a continuous uniform distribution. If no argument is given, one float type random number is returned. If an argument is given, a random number of the ndarray type of that size will be returned.

```
import numpy as np
    np.random.rand()
    0.22963100707952988
    np.random.rand(2,4)
    array([[0.25512767, 0.92925799, 0.37337589, 0.76565738],
           [0.04984103, 0.77138769, 0.61715466, 0.0413605]])
```

- numpy.random.randint: Generates an integer random number. The means of giving arguments is somewhat complicated, so refer to the following example.

```
import numpy as np
// random integer from 0 and less than 5
np.random.randint(5)
2
```

```
// random integer from 0 (the first argument) and less than 4 (the
second argument)
np.random.randint(0,4)
3
// generate 2 times 4 ndarray of random integers from 0 and less than
4.
np.random.randint(0,4,(2,4))
array([[1, 1, 1, 2],
       [0, 3, 2, 3]])
// generate 2 times 4 ndarray of random integers less than 5
// the second argument is omitted, and array size is specified with
name 'size.'
np.random.randint(5,size=(2,4))
array([[3, 2, 4, 4],
       [3, 1, 0, 3]])
```

- numpy.random.randn: Generates random numbers that follow a standard normal distribution. The last n here stands for normal distribution.

7. Column: Recursion

7.1 Processing numerical formulas

In the formulas we use every day, we use parentheses () to control the order of calculations.

1×2 + 3×2 = (1+3)×2
When converting this way, the calculation of multiplying by 2 is done only once. Therefore, it is necessary to give priority to addition, hence we write the addition part in parentheses ().

Parentheses (brackets) enable various mathematical formulas to be written within them, and they also allow parentheses to be inside them. Whenever parentheses appear, we “shelve” the previous calculation and calculate the part inside the parenthesis. When further parentheses appear, we shelve the other parts again and calculate inside the parentheses first.

7.2 Relative pronouns

A sentence in English consists of a noun subject and a verb, with an object and a complement forming the structure. Adjectives modify nouns, but there are relative pronouns as a means to qualify them, and the subject and verb appear again in the part modified by relative pronouns, while the whole structure is regarded as a “relative pronoun clause.” As with the previous formula, the structure of the subject (noun) and verb appeared again in the relative pronoun clause. This makes it difficult to comprehend, but it may be possible to modify the nouns in this clause again with the relative pronoun clause[1].

I read the book.

I read the book that was referred in a book.

I read the book **that was referred in the book** that was referred in another book.

7.3 Recursion

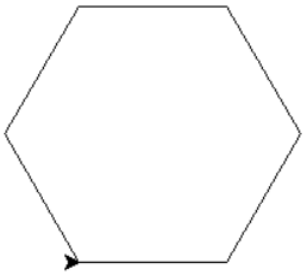
Both the above mathematical formulas and relative pronouns are characterized by the fact that the structures of the formulas and composition are nested, and it is required that the generation and interpretation be shelved and the same procedure be applied on the inside. The mechanism by which the same processing is applied to “parts” in a program is called recursion. A commonly used example is factorial calculation. The factorial of N can be expressed as 1 when N is 1, and N! = N * (N-1)! when N is greater than 1 (integer). If you write this programmatically, it will be as follows.

[1] A grammatical structure such as this is called phrase structure grammar in the field of linguistics. Proposed by Chomsky, it is also used for parsing on computers.

```
def frac(n):
    if n==1:
        return 1
    else:
        return n*frac(n-1)
```

Now think about figures; the following is a program that draws a hexagon with turtle graphics.

```
from turtle import *
for i in range(6):
    forward(100)
    left(60)
```



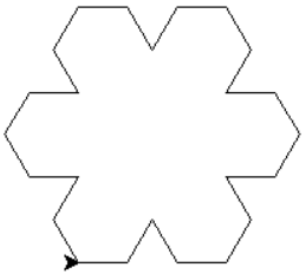
Now consider transforming the forward(100) part as follows.

- 1/3 length goes straight on
- Rotate 60 degrees to the left
- Go straight with a length of 1/3
- Rotate 120 degrees to the right
- Go straight with a length of 1/3
- Rotate 60 degrees to the left
- Go straight with a length of 1/3

Create and run a detour function in this way.

```
from turtle import *
def detour(L):
    LL = L/3
```

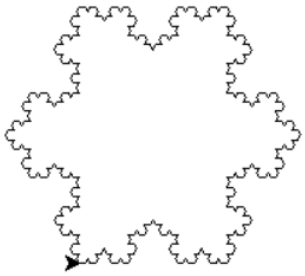
```
        forward(LL)
         left(60)
         forward(LL)
         right(120)
         forward(LL)
         left(60)
         forward(LL)

for i in range(6):
         detour(100)
         left(60)
```



Moreover, in the function detour, call your own detour instead of using the forward function. This is where recursion comes into play! However, if the length is 10 or less, it is drawn with forward without changes.

```
def detour(L):
         if (L < 10):
                  forward(L)
         else:
                 LL = L/3
                  detour(LL)
                  left(60)
                  detour(LL)
                  right(120)
                  detour(LL)
                  left(60)
                  detour(LL)
```

In figures like this, a part of it has similar to the figure itself, and they are called fractal figures. In the natural world, a structure such as this appears as a physical phenomenon like a snowflake, or it manifests in the shaping of life like a tree branch.



8. Column: GUI

8.1 What is a GUI?

GUI is an abbreviation for graphical user interface, which is an environment that hosts interaction between users and computers, which is mainly used in modern personal computers and smartphones. On the other hand, the environment for operating a computer by inputting characters such as conda prompt is called CUI, or character user interface. It can be said that IDLE's Python shell is eclectic.

8.2 Difficulty behind the scenes in software development for user friendliness

Tkinter is an environment for easily programming GUI applications with Python, but in actual programming, after coming to understand event-driven concepts and MVC architecture, etc., it is necessary to specify in detail the arrangement, display, operation, and so forth of widgets. When you actually write a program, you will notice that this entails a lot of work. GUI development appears to occupy a large part of the total workload when developing software for human use.

What's more, in the background of what you are programming directly, the window system realizes a huge amount of processing such as mouse and keyboard input and processing window overlap. In addition, tkinter provides an environment that can be easily programmed with Python by absorbing the differences between OSs such as Windows, macOS, and Linux.

A GUI using a touch panel such as a smartphone interprets various finger operations (gestures) to change the manner of operation, and the program finely adjusts the display, screen scrolling, entering and exiting menus, etc. making it easy for people to use. For example, when the scrolled screen reaches the edge on a smartphone, it does not simply stop, but produces an animation showing it bouncing off. Originally, the screen was not designed to bounce like this, but to make it easier for people to understand intuitively, the program is used to purposely implement what is likely to happen in the physical world.

8.3 Exclusion and inclusion

GUIs made it possible to operate computers intuitively and contributed to expanding their use. Even children who are not sufficiently capable of using a keyboard to input characters can operate a device to some extent with a GUI.

On the other hand, it is also important to understand that **some people are excluded** from using computers with GUIs. It is difficult for visually impaired people to use a GUI environment, and some elderly people find it difficult to double-click the mouse button. It may also be difficult for people who have problems with their fingers to operate smartphones that make heavy use of gestures.

GUI also makes it tricky to teach people how to operate devices (such as smartphones). It is difficult to convey the method of operation with only characters and text. The reason why there are so many pages in books and magazines that describe how to use a computer is that you cannot tell how to operate it without showing the screen, and it seems that many aspects would not be clear without the assistance of videos.

The opposite of "**exclude**" is "**include**." Recently, a design method called inclusive design is attracting attention. Computers, on the other hand, are expensive to develop software for, and many are developed for able-bodied people for commercial reasons, including to recoup the cost of development, but at the same time, by programming properly, more people can be included. If the system is designed with the MVC (model-view-control) architecture introduced in class, it becomes easier to configure the view and control, which are interfaces with the user, in accordance with the user, while maintaining the essential job of the computer as a model.

9. Column: Programming & Japanese

- The Never-ending Battle against Character Encoding

9.1 UTF-8 in Python

Many of the OSs and programming languages that run on computers have been developed in the English-speaking world, and using Japanese on computers, which uses far more kanji than the alphabet, is inevitably one step behind.

Nevertheless, Japanese is a language that has been available on computers throughout the world since the early days (!). However, the language causes confusion in “character encoding,” which is the mechanism for expressing Japanese characters on a computer. Microsoft has been selling software for the Japanese market since the early days, but the operating system MS-DOS, the predecessor of Windows, adopted the Shift-JIS character encoding for Japanese. After that, Unicode was developed for encoding that can be used internationally, including languages that use a large number of characters such as Chinese and Korean, and even Windows now uses this code internally. However, **Shift-JIS code is used for the file name and the character encoding of text files** in Windows in Japan in order to maintain compatibility with conventional files. Apple's Mac computers used to provide an OS using Shift-JIS encoding for the Japanese market, but now it uses Unicode. Unicode has also become the standard in Linux in recent years.

Python also had a problem handling character encoding in Python 2, but Python 3 adopts Unicode and uses utf-8 as the character encoding scheme, which is a method of handling Unicode in a computer. In the Windows environment, Python itself has the information that the shift-JIS code is used for the file name etc., and this converts the character encoding.

Often at the beginning of the Python source code you find a comment

```
# -*- encoding: utf-8 -*-
```

This is called a magic comment that shows the Python processing program that this source code is written in utf-8. Even if nothing is written, it is still treated as utf-8.

Also, before this comment, you can see

```
#!/usr/bin/python
```

This is called a shebang, and indicates a program that starts when a Python program is directly called in the shell of an OS such as Linux.

9.2 Character code error in the source code

If you write the same characters in the kanji code (so-called full-width characters) where you should originally write in ASCII code (so-called half-width alphanumeric characters) in the Python source code, a difficult-to-spot error will be produced. In Python, blocks are represented by

indentation, but if you insert a full-width space character instead of a half-width space character, the appearance will not change which makes the resulting error difficult to see.

9.3 Pay careful attention to the ¥ symbol

The backslash in ASCII code is assigned to the ¥ symbol in Japanese code so far. In UTF-8, a code is assigned separately for the yen symbol, but **in Windows, the backslash code (5C) is also displayed as a yen symbol** [1]. You must replace the backslash found in English documentation, etc. with the yen symbol.

On the other hand, if the yen symbol is entered on Mac, it will be entered as the UTF-8 yen symbol (A5) instead of the backslash code, and will not be the backslash (5C) that Python originally requires. This will result in an error, so enter it with a backslash.

9.4 Pay careful attention also to diacritics of KANA

In UTF-8, there are two ways to describe the diacritics of Kana characters. One is to treat “が”, for example, in one character code, the other way is to treat it by two characters “か” and “ ゛”, and to display the combined image on a screen. In using matplotlib on Mac, the latter expression may fail to show correct images. This problem is more serious in using such characters in file names.

[1] Even with Windows, depending on the development environment, it seems that the font displaying the backslash symbol ends up selected.

10. Column: Namespaces

10.1 Confusion with names

Namespace is a mysterious term, but refers to a logical space where names can be used with validity.

Before talking about namespaces on computers, we need to consider them in actual life. For example, let's say there's someone called Ichiro Tanaka. If you call this person by name, it's probably "Ichiro" at home. This is because you don't know exactly who the surname Tanaka would be referring to. At work, they would mainly be addressed as "Mr. Tanaka," but if there is someone with the same surname, they may be called "Ichiro" or by their full name "Ichiro Tanaka." We naturally devise a way to call the person by an identifiable name on the spot.

How about your phone number? You have to dial gradually longer numbers to call from a distance, such as by adding a local code, an area code, or an international code. The system is devised to assign numbers hierarchically so that short numbers can be identified if they are nearby.

10.2 Namespaces on computers

Computer programs access many objects (such as variables, functions and so on) by naming them. If you use a library (such as a Python module), then you have to use a lot of function names. If these are made accessible in a uniform space without any particular stratification, this will result in name conflicts that use the same name with different meanings. Therefore, it is necessary to limit the range (namespace) in which names can be used. For example, Python's math library "math" would be

```
import math
```

Also, pi defined here would be called as

```
math.pi
```

This is slightly troublesome.

Accordingly you get,

```
from math import *
```

which can then be used without adding "math.".

```
sin(0.5*pi)
```

It is convenient to use this instead of a scientific calculator because it can be calculated neatly. However, in math, the base of the natural logarithm is defined as the variable

```
e
```

Thus, if you inadvertently assign a value to a variable called e, the definition will collapse. (Unfortunately Python has no mechanism to safeguard the definition of constants).

This kind of confusion encountered is known as “namespace pollution.” It would be advisable to refrain from easy imports so as not to pollute namespaces.

11. Column: Documenting a Program

11.1 The program quickly becomes unable to cope

Even the person who wrote the computer program will no longer understand why they wrote it over time. Furthermore, in order for multiple people to continuously maintain and improve a program, a document explaining the program becomes necessary.

However, it takes a lot of effort to create a document explaining a program, and there are some things that only the person who has written the program can understand. The best opportunity to document a program is when writing it. Programmers are better off writing about a program when they understand the content, which will motivate them and improve work efficiency.

11.2 Documenting a program

One method of documenting a program is to read it and make it easier to understand, but there are two ways to do this.

One is to read the program itself and make it easy to understand. It is important to assign easy-to-understand variable names and function names, but it is also important to use **“coding conventions”** that use nomenclature, characters such as uppercase and lowercase letters, and terms in a unified manner. PEP8 is known as the coding conventions for Python[2]

- Another way is to annotate the program with comments. Concerning annotation for program as well as conding conventions, see [3].

11.3 docstring

Python introduces an interesting gimmick called 'docstring' which is related to program documentation. This defines a string (docstring) that explains the program at the beginning of modules, functions, and methods. The help() function creates a description using this docstring for modules, functions, etc. read by the import statement. What differs from annotation is that it is a string defined as a program, so it can be referenced by other programs.

11.4 Magic numbers

In Chapter 15 of the main text in which programming of tic-tac-toe is treated, we use the following definitions of the turn and game state as constants. After these definitions, we use these constants instead of the actual numbers.

```
OPEN = 0
FIRST = 1
SECOND = 2
DRAW = 3
```

If we don't use these constants, but use the actual numbers in the program, it makes reading the program difficult. Further, when we want to revise the program, we may face difficulties in distinguishing these numbers from those used for other meanings.

Actual numbers embedded in the program are called 'Magic Numbers.' Magic numbers make treatment of the program difficult, and we should define constants and use them as in the example above.

References

[2] PEP 8 -- Style Guide for Python Code, https://www.python.org/dev/peps/pep-0008/ (Accessed on 2/12/2020)

[3] Al Sweigart: Beyond the Basic Stuff with Python, No Starch Press (2020)

12. Column: Trigonometric Functions

12.1 Manufacturing and trigonometric functions

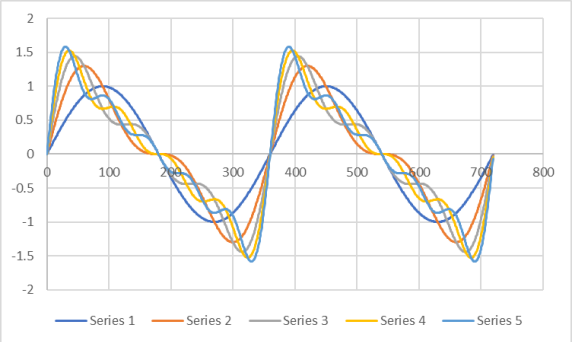
When learning trigonometric functions in high school, many people may be wondering what the point of learning them is. In this lesson, you made an analog clock using the graphics function of tkinter. In order to draw the hands of the clock, the time is re-read to the angle, and the X and Y coordinates for the position of the tip of the hand are required, so the trigonometric function is indispensable. There are likely more than a few people among you who are using trigonometric functions in a practical way for the first time.

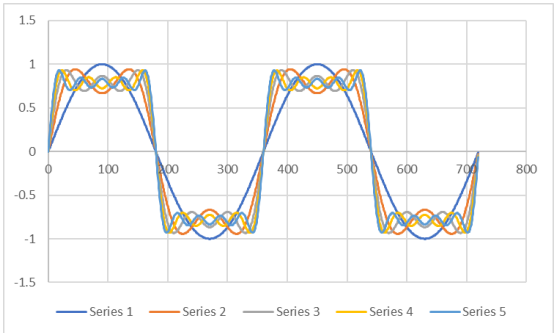
As in this example, trigonometric functions relate the angle of an inclined object to the length in the horizontal and vertical directions. Therefore, it is an essential function in manufacturing.

12.2 Trigonometric functions as waves

To give another example, I have shown an example in which the periodic function is represented by the sum of trigonometric functions. Sawtooth waves and rectangular (square) waves are respectively approximated as the sum of trigonometric functions as shown below[1]

- $f_1(x) = \frac{\sin(x)}{1} + \frac{\sin(2x)}{2} + \frac{\sin(3x)}{3} + \frac{\sin(4x)}{4} \ldots$
- $f_2(x) = \frac{\sin(x)}{1} + \frac{\sin(3x)}{3} + \frac{\sin(5x)}{5} + \frac{\sin(7x)}{7} \ldots$





The sum of the trigonometric functions that approximate a specific function such as this is called a Fourier series. It is known that periodic functions can be approximated by the sum (series) of the constant multiples of the sine function (sin) and cosine function (cos), which have a period of one-fraction of the period of the function (an integral multiple frequency).[2]The coefficient of each term

[1] For sawtooth and square waves with maximum and minimum values of 1, -1, this equation is multiplied by the coefficients 2/π and 4/π, respectively.

[2] In the above example, since the original function is an odd function (the graph is symmetric to the origin), only the sine function (sin) appears. Also, since you are trying to approximate a discontinuous function by the sum of trigonometric functions, which is a continuous function, the irrational here appears at the discontinuity point of the function.

of the Fourier series is obtained by multiplying the function to be expressed by the Fourier series by the trigonometric function of each term and integrating.

12.3 Listen to the difference in waveform as sound

Even if the pitch is the same, the timbre (tone color) will differ depending on the type of instrument. One of the reasons for this is that the sound produced by an instrument includes a sine wave (fundamental wave or fundamental tone) with a frequency that represents the pitch, and a sine wave (higher harmonic or harmonic overtone) with an integral multiple of that frequency. This is because the strings of string instruments and the tubes of wind instruments resonate not only with fundamental waves but also with harmonics.

At this point, let's actually listen to and compare the waves synthesized by the computer.

- sinwavw.wav is a sine wave with a frequency of 440Hz
- harmonics-saw.wav is a Fourier series approximation of a sawtooth wave up to the 7th higher harmonic.
- harmonics_sq.wav is a Fourier series approximation of a rectangular (square) wave up to the 7th higher harmonic.

These files are audio files in a format called a waveform (with a .wav extension). In Python, you can read and write files in wav format using a module called wave. The program used to generate the above data is the following makesoundfile.py (referring to the reference material[4]). Concerning programming of audio data in Python see also[5].

Program 12-1 (makesoundfile-en.py)

Row	Source code
1	`import␣numpy␣as␣np`
2	`import␣wave`
3	`import␣struct`
4	
5	`#␣Audio␣file␣name`
6	`fname␣=␣'harmonics-sq.wav'`
7	`wf␣=␣wave.open(fname,␣'w')`
8	`#␣Audio␣file␣parameters,␣1␣channel␣(monaural),`
9	`#␣1␣point␣audio␣data␣2␣byte␣(16␣bit)`
10	`#␣Sampling␣frequency␣44.1␣kHz`
11	`ch␣=␣1`
12	`width␣=␣2`
13	`samplerate␣=␣44100`
14	`wf.setnchannels(ch)`
15	`wf.setsampwidth(width)`
16	`wf.setframerate(samplerate)`

```
17
18  # Continuous for 10 seconds
19  time = 10
20  numsamples = time * samplerate
21
22  print( "Number of channels =", ch)
23  print( "Sample width (bytes) =", width)
24  print( "Sampling rate (Hz) =", samplerate)
25  print( "Number of samples =", numsamples)
26  print( "Recording time =", time)
27  print( "Output file name =", fname)
28
29  # Create signal data (with NumPy ndarray)
30  freq = 440                    # Set the frequency freq of the basic circumference to 440 Hz
31  x=np.linspace(0, time, numsamples+1) # 0 ≤ t ≤ time divided evenly into numsamples
32  h1=np.sin(2*np.pi*freq*x)          # Sine wave of the fundamental wave
33  h2=np.sin(2*np.pi*2*freq*x)        # Sine wave of the second higher harmonic
34  h3=np.sin(2*np.pi*3*freq*x)        # Sine wave of the third higher harmonic
35  h4=np.sin(2*np.pi*4*freq*x)        # Sine wave of the fourth higher harmonic
36  h5=np.sin(2*np.pi*5*freq*x)        # Sine wave of the fifth higher harmonic
37  h6=np.sin(2*np.pi*6*freq*x)        # Sine wave of the sixth higher harmonic
38  h7=np.sin(2*np.pi*7*freq*x)        # Sine wave of the seventh higher harmonic
39
40  # Synthesize waves by multiplying fundamental waves and harmonics by coefficients
41  y = h1 + 0*h2/2 + h3/3 + 0*h4/4 + h5/5 + 0*h6/6 + h7/7
42
43  y=np.rint(32767*y/max(abs(y)))       # Keep within the range of # [-32767, 32767]
44  y=y.astype(np.int16)                 # Convert to a 16-bit integer
45  y=y[0:numsamples]                    # Cut off at numsamples individual data
46
47  # Convert ndarray to bytes object
48  data=struct.pack("h" * numsamples, *y)
49
50  # Export data
51  wf.writeframes(data)
52  wf.close()
```

References

[4] 桂田 祐史：Python を使った WAVE ファイルの処理, http://nalab.mind.meiji.ac.jp/~mk/lecture/fourier-2018/python-sound/(accessed Dec 3, 2018, in Japanese)

[5] 青木直史：Python で始める音のプログラミング, オーム社 (2022, in Japanese)

13. Column: Reference & Duplication

What users should be aware of in programming languages is often to have a “reference” to the location of the actual information, as opposed to the information itself that the variable handles. In Python, all variables are treated uniformly as references to them, not to the entities (objects) they represent.

13.1 The reference and duplication of information; their advantages and disadvantages

There are advantages and disadvantages to referencing and duplicating information. Consider the following situation in the real world.

> **Mr. A**: Mr. B, could you give me the materials for the meeting we had the other day?
> **Mr. B**: They’re on that shelf, so please use them as you like.
>
> A few days later
>
> **Mr. B**: Mr. A, could you return the materials for the meeting we had the other day?
> **Mr. A**: You said to use them as I like, so I cut out the necessary parts for the report. Should I give you back the rest?

This is a pickle. Since Mr. A and Mr. B **shared the same materials**, the result of Mr. B's modifications to them affects Mr. A as well. In this situation, Mr. B should have made **a copy of the materials** before handing them over to Mr. A.

On the other hand, if Mr. A only wants to browse a part of the materials, it is troublesome to make a copy, and if Mr. B asks Mr. A to do some of his work, it is more effective to use the same materials.

On a computer, however, duplicating information is easier compared to using paper materials, but even so, when dealing with a large amount of information, the time required for duplication makes efficiency in carrying this out problematic. It is effective to look at the same information by reference when there is no potential harm, and to duplicate it when working independently.

13.2 Reference and duplication in Python

In Python, all variables are treated uniformly as references to actual data. On the other hand, with regard to data, numerical values and strings are treated as immutable data that does not allow

contents to be rewritten, so even those who are accustomed to other programming languages can write programs relatively naturally. However, since lists and other data are forms of mutable data that allow the contents to be rewritten, the following points should be noted.

- Rewriting the data given as an argument in a function call, etc.
  In the following example, the variable a is assigned integer type data and b is assigned a list. If a and b are given as arguments to the function f(), although the value of a has not been rewritten after executing f(), the 0th element of b has.

```
a = 1
b = [1,2,3]
def f(x, y):
    x = 0
    y[0] = 0
a, b
(1, [1, 2, 3])
f(a, b)
a, b
(1, [0, 2, 3])
```

- Rewriting global variables. In the example below, the global variable a is a list, but it is not declared global within the function. However, the elements of a are rewritable.

```
a = [1,2,3]
def f():
    a[0] = 0
a
[1, 2, 3]
f()
a
[0, 2, 3]
```

- Make duplicates of data such as lists. If you want to use duplication of list, you need explicitly make it as follows.

```
a = [0, 1, 2]
b = a.copy()
a[0] = 1
```

a

[1, 1, 2]

b

[0, 1, 2]

14. Column: Personification

In computer programming, you often come across terms with suffixes such as -er. This is because personifying expressions are suitable as they make requests to a computer that works automatically. In the explanation given to the class, the term “constructor” came up.

In this textbook, you will experience working with multiple turtles and then explain object-oriented programming in the chapter called Classes. In this section, I explained that **objects are like robots**. Some object-oriented textbooks explain that an object = a thing, but I believe it's hard to imagine that an object operates actively by calling a method. In reality, it is not uncommon to create objects with personified names.

The word computer seems to have originally meant the person in charge of calculation until the advent of mechanical calculators. Automation has replaced the work that humans were doing with machines.

As computers have become cheaper, the Internet of Things (IoT), in which computers are incorporated into everything and connected to networks, continues to gain attention. Until now, we have replaced human work with machines, but from now on, it may be easier to think that "things" can function like humans. It may become a reality that we desire various things to be personified. As usual, we use a lot of paper documents and books, but for people like me who are not good at tidying up, I want documents and books that can tidy themselves up on their own accord. Now, what do you imagine could be personified?

15. Column: Escaping

We write the source code and so forth of a program as a text file in a sequence of characters, but particular symbols are often used in a special sense. For example, we enclose string constants in "" as in

```
"string"
```

However, the problem emerges what should I do if I want to include " itself in the string? In other words, you need a means to “escape” from “interpretation as a command.” In Python, you can use either single quotes or double quotes to enclose a string, which is slightly convenient. For example, when you want to treat the entire "string" as a string, then

```
'"string"'
```

is the way to express it.

If you want to insert a symbol such as a new line in the string, then write ¥n. Writing that starts with ¥ is called an “escape sequence.” It really is a case of escaping, isn't it? Then, the way you write the character “¥” is to write two characters in a row as “¥¥.”[1]

In the format for converting numbers, etc. to strings (interpreted by the “format” method), “{}” has a special meaning as the conversion format. So then, you may wonder what to do when you want to treat “{” or “}” itself as a string. In this case, you simply write two in a row like “{{.”

When learning programming, you will often encounter things such as wanting to “escape” from “interpretation” in string notation. Checking how to escape is a small detail, but it's good to bear in mind. Confirming emergency exits is also a must in the world of programming.

[1] Often you’ll want your program to generate “other programs.” In order for the escape sequence to be output correctly in the generated program, it is necessary to prevent the escape sequence from being interpreted by the generated program.