Co-funded by the
Erasmus+ Programme
of the European Union

Dragan Mašulović

Complex numbers

Novi Sad, December 2021.

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

Co-funded by the
Erasmus+ Programme
of the European Union

1 A refresher on complex numbers

The imaginaty unit. Let i denote a new "number" whose defining property is that

$$i^2 = -1.$$

It is clear that this new "number" is not an element of the reals $\mathbb{R}$ because a square of any real number such as 3.9 or -1.34234543 is greater than zero, with the exception of 0 whose square is 0. That's why this new object is called "imaginary": it is a product of our imagination!

Complex numbers. Nevertheless, if we adjoin i to the reals and pretend that everything is fine, it turns out that we can actually compute with such "quantities"! For example,

$$\begin{aligned}(2-3i)\cdot(i-3) &= 2i-6-3i^2+9i && \text{[just multiplying out]}\\ &= 2i-6+3+9i && \text{[because } i^2=-1\text{]}\\ &= 11i-3. && \text{[collecting like terms]}\end{aligned}$$

We see that new objects that we can algebraically manipulate are compound "numbers" consisting of a *real part*, the number that dangles "freely" in the sum, and the *imaginary part*, the number that multiplies the *imaginary unit* i:

$$z = \underset{\substack{\uparrow\\ \text{real}\\ \text{part}\\ \mathrm{Re}(z)}}{-3} + \underset{\substack{\uparrow\\ \text{imaginary}\\ \text{part}\\ \mathrm{Im}(z)}}{11}\ \underset{\substack{\uparrow\\ \text{imaginary}\\ \text{unit}}}{i}$$

The real part of a complex number z is denoted by $\mathrm{Re}(z)$, while its imaginary part is denoted by $\mathrm{Im}(z)$.

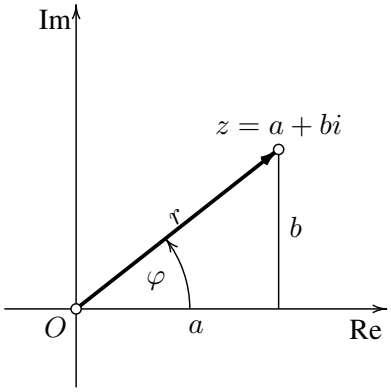
Therefore, for $z = -3+11i$ we have that $\mathrm{Re}(z) = -3$ and $\mathrm{Im}(z) = 11$. (*Note that* $\mathrm{Im}(z) = 11$*, just the number 11;* ***not*** $11i$***!***)

Trigonometric representation of complex numbers. Such compound "numbers" are actually called *complex numbers* (because their structure is not simple, but complex) and the set of all the complex numbers is

$$\mathbb{C} = \{a+bi : a, b \in \mathbb{R}\}.$$

Having two components, its real and its imaginary part, complex numbers can be represented graphically as follows:

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."



So, the real part of z is taken for its x-coordinate, while its imaginary part is taken for its y-coordinate. Given r and φ, elementary trigonometry tells us that

$$a = r\cos\varphi \quad \text{and} \quad b = \sin\varphi.$$

This is the *trigonometric representation of the complex number* $z = a + bi$.

The modulus and the argument (phase) of the complex number. The length of the arrow leading to z is *the modulus of* z:

$$r = |z| = \sqrt{a^2 + b^2}$$

while the angle φ is *the arguement* or *the phase* of z:

$$\varphi = \mathrm{Arg}(z) = \arctan(b/a), \text{ whenever } a \neq 0.$$

In case $a = 0$ we take

$$\varphi = \mathrm{Arg}(z) = \pi/2, \text{ in case } b > 0,$$

and

$$\varphi = \mathrm{Arg}(z) = -\pi/2, \text{ in case } b < 0.$$

We shall take the argument of a complex number to always belong to the interval $[-\pi/2, \pi/2]$. Moreover, we do not define the argument of $0 = 0 + 0i$ because the arrow pointing to 0 has no length, so there is no angle to measure.

Arithmetic of complex numbers. As we have seen, it is easy to compute the sum, difference and product of complex numbers, and the corresponding formulas are obvious:

$$\begin{aligned}(a + bi) + (c + di) &= (a + c) + (b + d)i\\ (a + bi) - (c + di) &= (a - c) + (b - d)i\\ (a + bi) \cdot (c + di) &= (ac - bd) + (ad + bc)i\end{aligned}$$

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

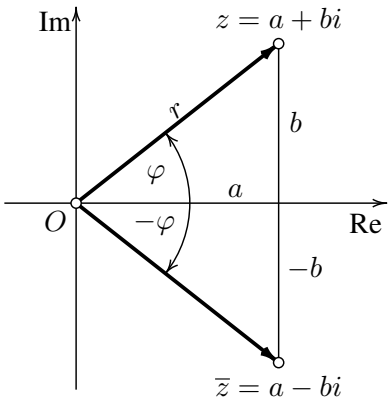
However, division of complex numbers is a bit more involved. Dividing a complex number by a real number is straightforward:

$$\frac{a+bi}{c} = \frac{a}{c} + \frac{b}{c}i.$$

In order to explain how two complex numbers can be divided let us introduce the new operation on complex numbers called the *conjugation*. For a complex number $z = a + bi$, its *conjugate* $\overline{z}$ is the complex number obtained by *chainging the sign of* ***the imaginary part only***:

$$z = a + bi \quad \rightsquigarrow \quad \overline{z} = a - bi.$$

This is how conjugation behaves graphically:



Therefore, we see that

$$\mathrm{Re}(\overline{z}) = \mathrm{Re}(z), \quad \mathrm{Im}(\overline{z}) = -\mathrm{Im}(z), \quad |\overline{z}| = |z|, \quad \mathrm{Arg}(\overline{z}) = -\mathrm{Arg}(z).$$

Moreover,

$$z \cdot \overline{z} = (a+bi)(a-bi) = a^2 - (bi)^2 = a^2 + b^2 = |z|^2,$$

whence follows that:

$$\frac{1}{z} = \frac{\overline{z}}{|z|^2}.$$

This now easily leads to the division formula for complex numbers:

$$\frac{w}{z} = \frac{w \cdot \overline{z}}{|z|^2}.$$

As is the case when dividing "usual" numbers, these formulas make sense only in case $z \neq 0$. Another way to put the division formula is:

$$\frac{a+bi}{c+di} = \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2}i, \quad \text{when } c+di \neq 0.$$

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

A final remark. The new "number" i was introduced so that $i^2 = -1$. *One should* ***never*** *write* $i = \sqrt{-1}$ because this is not entirely true, and may lead to confusion. Namely, is we take the view that $i = \sqrt{-1}$ then we can perform the following computation:

$$1 = \sqrt{(-1)\cdot(-1)} = \sqrt{-1}\cdot\sqrt{-1} = i\cdot i = -1.$$

Nonsense!

The square root of a complex number is a complicated operation. In short, we take the square root of a complex number to be a *set of values*:

$$\sqrt{z} = \{w \in \mathbb{C} : w^2 = z\}.$$

Therefore,

$$\sqrt{-1} = \{i, -i\}.$$

Hence, it is simply not true that $i = \sqrt{-1}$. The correct statement is:

$$i \in \sqrt{-1},$$

and this eliminates the above contradiction.

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

2 At the computer keyboard

As we have seen, complex numbers are unpleasant to compute with by hand. Fortunately, computing technology can help us here! In this section we are going to show how to use a computer algebra system called *SageMath* to help us get around. It is important to stress that SageMath is based on Pyhton and *free!*

Installing SageMath. In order to download and install SageMath, go to

`www.sagemath.org`

and follow the installation instructions. SageMath comes bundled with *Jupyter* – an interactive environment for experimenting with data and math. Once the installation is complete, search for *SageMath Notebook* in the list of available applications on your computer and start it. A short introduction to Jupyter can be found at the end of this manuscript.

Complex numbers in SageMath. SageMath is a *computer algebra system* which means that it is meant to help us do algebra on a computer. Let's try some simple stuff:

```
In [1]: 3+4
Out[1]: 7
```

Complex numbers in SageMath are written slightly differently than in mathematics – the only difference is that we *can use both I (the capital I) and i (the lowercase i) for the imaginary unit*, and that have to explicitly write the multiplication symbol $*$. Because the output subsystem of SageMath prefers I as the imaginary unit, we shall also opt for I as the imaginary unit while working in SageMath. Therefore, we shall write:

```
In [2]: (3 - 4*I)*(2 + I)
Out[2]: -5*I + 10
```

Of course, we can use variables to name complex numbers:

```
In [3]: z = 1 - 2*I
        w = I + 3
        z*(1 + w)
Out[3]: -7*I + 6
```

and we can easily compute the modulus, and the real and imaginary parts of a complex number:

```
In [4]: # the modulus of z, its real and imaginary part
        print(abs(z), z.real(), z.imag())

        sqrt(5) 1 -2
```

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

In order to compute the argument of a complex number we have to import a library called `cmath` – complex math. It provides the method called the `phase`:

```
In [5]:  import cmath
         cmath.phase(z)

Out[5]: -1.1071487177940904
```

Note that it suffices to import `cmath` (or any other library) only once during the session! Methods provided by the library will be available from that moment on. Here's how we can compute the conjugate of a complex number:

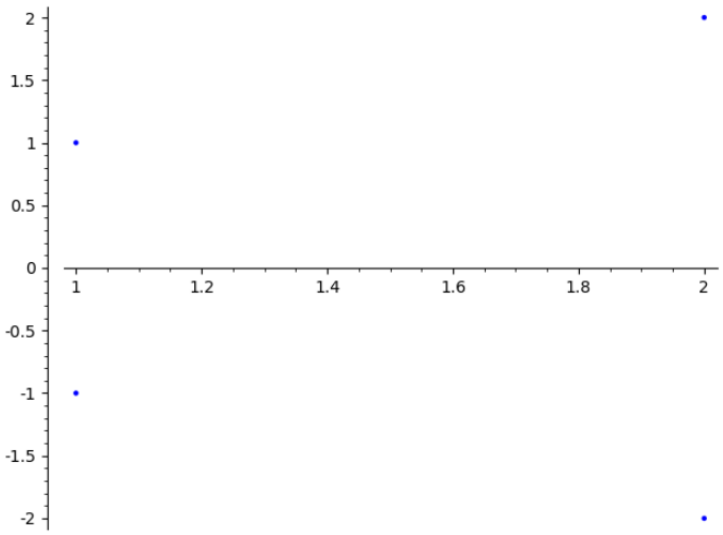
```
In [6]:  z.conjugate()

Out[6]: 2*I + 1
```

SageMath has a useful function `list_plot` which takes a list of points or complex numbers and displays a plot:

```
In [7]:  nums = [1 - I, 2 + 2*I, 1 + I, 2 - 2*I]
         list_plot(nums)
```

Out[7]:

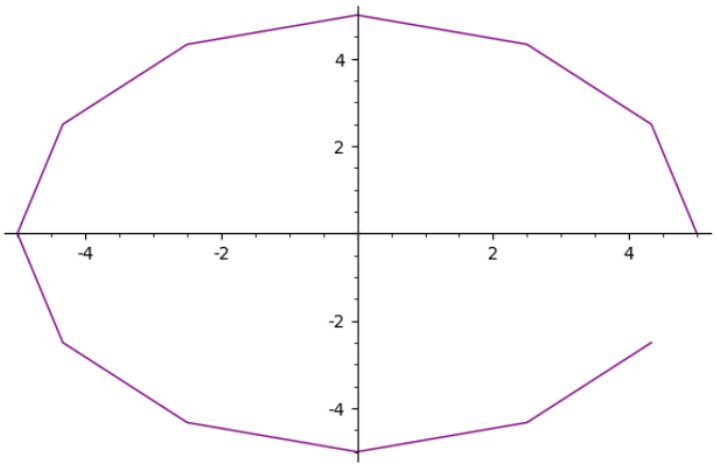


"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

We can join the points and change the color of the plot lines:

In [8]:
```
a = pi / 6
r = 5
nums = []
for k in range(12):
    nums.append(r*cos(k*a) + r*sin(k*a)*I)
list_plot(nums, plotjoined=True, color='purple')
```

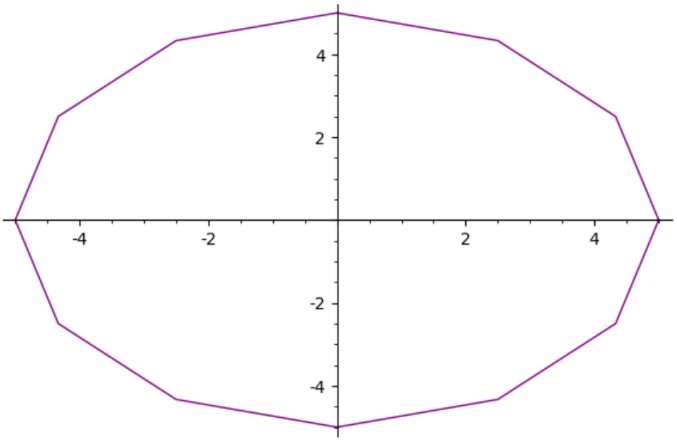
Out[8]:



In order to achieve a more pleasing output, let's just close the loop:

In [9]:
```
a = pi / 6
r = 5
nums = []
for k in range(12):
    nums.append(r*cos(k*a) + r*sin(k*a)*I)
# add a copy of the first element to the end
# of the list to close the loop
nums.append(nums[0])
list_plot(nums, plotjoined=True, color='purple')
```

Out[9]:



"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

3 Where did complex numbers come from?

We, humans, have discovered complex numbers while trying to understand how complicated equations are solved. Let us start with a simple equation like:

$$6x - 7 = 0.$$

This is a *linear equation (in one unknown)* and we know that it has a unique solution $x = \frac{7}{6}$. Let us now step up the game and try to solve this equation:

$$x^2 - 6x + 8 = 0.$$

This is a *quadratic equation (in one unknown)*. Using the well-known formula for the square of the binomial we get the following:

$$\begin{aligned} x^2 - 6x + 8 &= 0 \\ (x^2 - 6x + 9) - 1 &= 0 \\ (x - 3)^2 &= 1 \end{aligned}$$

whence we easily get that

$$x - 3 = 1 \text{ or } x - 3 = -1,$$

so we end up with two solutions:

$$x_1 = 4, \quad x_2 = 2.$$

Let's see how SageMath solves equations. To deal with equations, we first have to announce what are the variables that we are going to use to form equations. Since `x` is typically used as a letter denoting the unknown value, we shall tell that to SageMath like so:

```
In [1]: var('x')
Out[1]: x
```

We can now solve the two equations from above by calling the `solve` function, which takes the following form:

`solve(`*equation*`,` *unknown*`)`

Since SageMath is built on Python, equations are written using ==:

```
In [2]: solve(6*x - 7 == 0, x)
Out[2]: [x == (7/6)]

In [3]: solve(x^2 - 6*x + 8 == 0, x)
Out[3]: [x == 4, x == 2]
```

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

Now, let's try to solve $x^2 + 2x + 5 = 0$:

```
In [4]: solve(x^2 + 2*x + 5 == 0, x)
Out[4]: [x == (-2*I - 1), x == (2*I - 1)]
```

We see that the solutions are complex numbers! For quite some time the mathematicians were puzzled by the fact that solving euqations with real coefficients leads to solutions that are not real numbers, so we devote the following section to a short historical overview of complex numbers popping all over the plase while solving equations.

4 Exercises

1. Fill in the following table:

n	0	1	2	3	4	5	6	7	8	9	10	11	12
i^n	1	i											

   Can you provide a general formula for i^n that depends on the divisibility of n by 4?

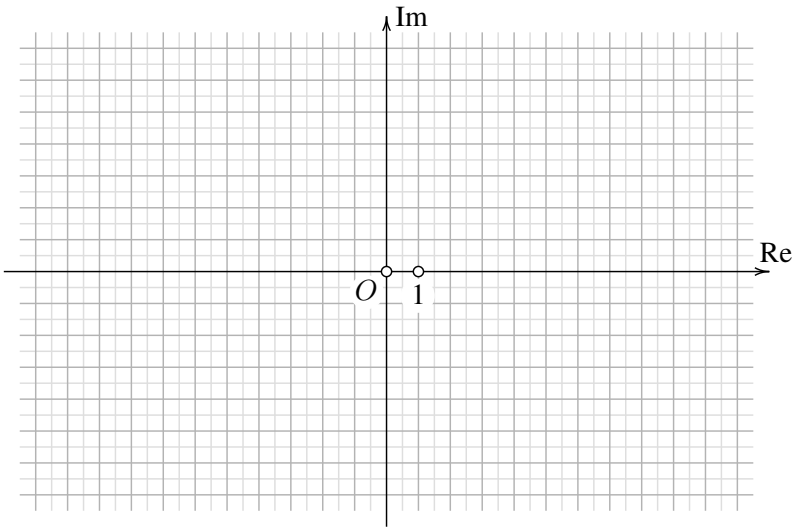
2. Compute the real and the imaginary part of the following complex numbers:

$$2 - 3i, \quad i + 3, \quad -2 - 5i, \quad 5 - (-3)i, \quad 7i - 3i^2.$$

3. Compute the conjugate of each complex numbers from the exercise above.

4. Graph the following complex numbers in the complex plane and for each of them compute the modulus and the argument:

$$a = 3 + 3i, \quad b = -4 + 4i, \quad c = -5, \quad d = 6i, \quad e = 4 - 4i.$$



"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

5. Compute by hand and using SageMath:

$$\frac{1-i}{1+i}, \quad \frac{1}{1+\frac{1}{i}}, \quad \frac{(5-3i)(7-2i)}{(3i-5)(2i+7)}$$

6. Using SageMath graph the following complex numbers in the complex plane and for each of them compute the modulus and the argument:

$$a = 1+2i, \quad b = -5+i, \quad e = -1-4i.$$

7. Using complex arithmetic in SageMath write a Python program that draws a regular 12-gon.

8. Solve the following equations by hand and in SageMath:

$$3x-5=7, \quad 1-2x+5x^2=0, \quad x^3+x=0.$$

References

[1] J. P. D'Angelo. An Introduction to Complex Analysis and Geometry. American Mathematical Society, 2010

[2] V. J. Katz. A History of Mathematics: An Introduction. 3rd Ed, Addison-Wesley, 2009

[3] A. Speight. Byte-Size Python. John Wiley & Sons, 2020

[4] P. Zimmermann et al. Computational Mathematics with SageMath. Volume 160 of Other Titles in Applied Mathematics, SIAM 2018

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

Co-funded by the
Erasmus+ Programme
of the European Union

Addendum: A short introduction to Jupyter

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

Co-funded by the
Erasmus+ Programme
of the European Union

1. Introduction to Jupyter

In this lecture we demonstrate:

1. what is Jupyter, how to navigate through Jupyter notebook and how to use Jupyter to evaluate Python expressions;
2. how to use variables in Jupyter and how to execute small Python programs; and
3. what are libraries and how to use functions provided by standard libraries.

1.1. Jupyter does arithmetic

Jupyter is an interactive environment in which you can enter some text (just as the text you are reading right now), evaluate arithmetic expressions, run Python programs, analyze data, represent data as tables and diagrams, and much, much more.

Each Jupyter *notebook* consists of *cells*, and each cell holds some text, a math expression or a Python program. For the moment we shall refrain from explaining how to enter text into Jupyter notebook cells. Instead, we shall focus on evauating expressions and running Python programs.

When you enter some expression or a Python program into a cell, the cell can be *evaluated* by clicking the **Run** button:

or pressing [CTRL]+[ENTER]

For example:

In [1]:

```
3 * 19
```

Out[1]:

57

In [2]:

```
(12 + 51) * 14
```

Out[2]:

882

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

Co-funded by the
Erasmus+ Programme
of the European Union

In [3]:

```
2**(5**3)
```

Out[3]:

```
42535295865117307932921825928971026432
```

Let us recall that two asterisks in Python stand for exponentiation.

1.2. Jupyter can execute Python commands

Sometimes it is convenient to name values, in particular in case we are talking about some complicated numbers such as π or some complicated expressions. We can later refer to those values by simply typing the we have given them. For example, the estimated population of the Earth on July 1st, 2019 is 7,714,576,923. Python command

```
WorldPopulation_2019 = 7714576923
```

will introduce a new variable `WorldPopulation_2019` into the system and assign to it the value `7714576923` . Let's recall: names of variables in Python *must* start with a letter and may contain letters, digits and the special symbol `_` (underscore).

In [4]:

```
WorldPopulation_2019 = 7714576923
```

After running this cell you get no response from the system. The Jupyter has simply memorized that the value of the variable `WorldPopulation_2019` is 7,714,576,923. It is estimated that 27.8% of world population lives in cities. Therefore, this many people live in cities:

In [5]:

```
WorldPopulation_2019 * 27.8 / 100
```

Out[5]:

```
2144652384.5939999
```

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

Let us recall: the word *percent* comes from Latin *pro centum* which means "in a hundred". This is why

$$47\% = \frac{47}{100} = \text{fourty-seven percent}.$$

For examples, there are 856 students in a scholl and 25% of them are A students. How many A students are there in the school?

Answer. There are 214 A students in the school because

$$856 \cdot 25\% = 856 \cdot \frac{25}{100} = 214.$$

There are 326 B students in the same school. What percentage does that make?

Answer. Assume this makes $x\%$ of all the students in the school. Then

$$856 \cdot x\% = 856 \cdot \frac{x}{100} = 326.$$

Solving for x gives:

$$x = \frac{326 \cdot 100}{856} \approx 38.08.$$

Therefore, approximately 38.08% of students in the school are B students.

Let us solve another problem.

Exercise. Mary wanted to buy a pair of trousers whose price was $6,799.99 but her Mom said that that was too much. Mary waited for the sales season and next time she went to the shop the trousers were 25% off. That, together with some lower lip trembling, persuaded Mary's Mom to buy her the trousers. At the cashier the were pleasantly surprised to learn that they were eligible for an additional 3% discount on already discounted trousers. What was the final price of the trousers?

In [6]:

```
price = 6799.99
discount1 = price * 25 / 100
new_price = price - discount1
discount2 = new_price * 3 / 100
new_price - discount2
```

Out[6]:

```
4946.992725
```

The first four instructions assign some values to some variables. The last line in the above cell contains an expression. Since we are in an *interactive environment*, expressions are evaluated immediately and the result of running the cell is the value of the last expression in the cell.

Instead, we could have written the code like this:

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

In [7]:

```
price = 6799.99
discount1 = price * 25 / 100
new_price = price - discount1
discount2 = new_price * 3 / 100
print("The price after all the discounts is $", new_price - discount2, sep="")
```

```
The price after all the discounts is $4946.992725
```

This time the `print` instruction *prints* the value of the expression togehter with some decorating text while the system *returns nothing as the result of running the cell* (there is no response of the system in the form of `Out[ ]:`).

While working with interactive environments it is convenient to accept the following manner: we use `print` only when we have to display several values at once or if we wish to provide an explanation, as we did in the last example.

1.3. Libraries

In modern programming languages, and Python is one of them, we can do incredible things simply because they all come with a myriad of *functions* that someone has already prepared for us. This makes our lives much easier: most things that an average (and many above-average) users need have already been provided. We just have to find them!

Since we are talking about thousands of functions, literaly, they have all been organized and packed into *libraries of functions*. For example, the library of maths functions is, unsurprisingly, called `math` . There we can find functions such as `sqrt` (which computes the *square root*), `sin` (which computes the *sine* of an angle) and `cos` (which computes the *cosine* of an angle), but also constants such as `pi` (which is, of course, an *approximation* of π).

For example, the following computes the circumference of a circle:

In [8]:

```
from math import pi
r = float(input("Enter radius: "))
obim = 2 * r * pi
print("The circumference is:", obim)
```

```
Enter radius: 10
The circumference is: 62.83185307179586
```

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."

The first line in this example demonstrates how to import a particular element from a library:

```
from math import pi
```

instructs the environment to locate the library `math` and import `pi` from it.

Here comes another example. We are going to write a Python program that computes the hypotenuse c of a right triangle given its legs a and b. Recall that, by Pythagoras, $c = \sqrt{a^2 + b^2}$.

We clearly need the function `sqrt` that computes the square root of a number. It's located in the library `math`, of course.

In [10]:

```
from math import sqrt
a = float(input("a = "))
b = float(input("b = "))
c = sqrt(a**2 + b**2)
print("c =", c)
```

```
a = 3
b = 4
c = 5.0
```

The last two examples were complete computer programs: we got the input from the user using the `input` command, we processed the data, and then displayed the results using `print`.

However, working in the interactive environment makes it possible for us to work with pieces of code that we can modify and execute as many times as we like. We can easily *experiment with data*, which is an important part of understanding *modern data analysis*. Instead of using the `input` command, we assign values to variables directly in the program, run the analysis, change the values assigned to variables, run the analysis again, and repeat the process as many times as needed.

This philosophy applied to the last example (computing the hypotenuse of a right triangle) gives the following code:

In [11]:

```
from math import sqrt
a, b = 3, 4
sqrt(a**2 + b**2)
```

Out[11]:

```
5.0
```

Assignment `a, b = 3, 4` means that `a` becomes 3, while `b` becomes 4. The output of running the cell is the value of the expression `sqrt(a**2 + b**2)`.

"The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein."