

**Crystallography and Crystal Chemistry
X International School-Conference of
Young Scientists 2025**

Tutorial 3: Introduction to Python and practice

Dr. Anton Boev, PhD Arseniy Burov

PhD Maria Solovieva, PhD Daniil Chernyshov

PhD Nikita Davydov, MSc Ilya Kraev

Skoltech Center for
Energy Science
and Technology
Energy

November, 2025

What is Python?

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming.



Designed by Guido van Rossum

First appeared February 20, 1991;
30 years ago

<https://docs.python.org/>



Python's philosophy

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Quick introduction into Python

Useful links for Python studying

Links to study python:

- <https://www.w3schools.com/python>
- <https://python.land/python-tutorial>

Any other option, which you find the best.

It is up to your decision.



Essentials. Numbers

Integer. The Python integer is a non-fractional number, like 1, 2, 45, -1, -2, and -100. It's one of the three types of numbers Python supports natively, the others being floating point numbers and complex numbers.

```
>>> a = 156
>>> a
156
```

Float. Just put the floating point to get this number type

```
>>> a = 123.5
>>> a
123.5
```

Converting to an integer/float

```
>>> a = 123.5
>>> int(a)
123
```

```
>>> int(a)
150
```

```
>>> float(a)
150.0
```

```
>>> a = '150'
```

string format

<https://python.land/python-tutorial>

Essentials. Strings

A string in Python is a sequence of characters

```
>>> a = 'jdghkjdaghdklghdls'  
>>> a  
'jdghkjdaghdklghdls'
```

Converting to a string and some operations

```
>>> a = 568  
>>> str(a)  
'568'
```

```
>>> my_age = 27  
>>> f'My age is {my_age}'  
'My age is 27'
```

```
>>> 'Hello world'.split()  
['Hello', 'world']
```

```
>>> mystring = 'Hello world'  
>>> mystring[::-1]  
'dlrow olleH'
```

```
>>> 'Hello world'.replace('world', 'dudes')  
'Hello dudes'
```

```
>>> 'a' + 'b'  
'ab'  
>>> 'ab'*4  
'abababab'  
>>> 'a' - 'b'  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

```
>>> a = 'pupok'  
>>> a[0]  
'p'  
>>> a[1]  
'u'  
>>> a[2]  
'p'  
>>> a[3]  
'o'  
>>> a[-1]  
'k'
```

Essentials. Booleans

A boolean is the simplest data type; it's either `True` or `False`.

In Python, we use booleans in combination with conditional statements to control the flow of a program:

```
>>> door_is_locked = False
>>> if door_is_locked:
...     print("Mum, open the door!")
... else:
...     print("Let's go inside")
...
Let's go inside
>>> _
```

>	greater than
<	smaller than
>=	greater than or equal to

```
>>> 2 > 1
True
>>> 2 < 1
False
>>> 2 < 3 < 4 < 5 < 6
True
>>> 2 < 3 > 2
True
>>> 3 <= 3
True
>>> 3 >= 2
True
>>> 2 == 2
True
>>> 4 != 5
True
```

Essentials. Lists

Lists are used to store multiple items in a single variable.

```
mylist = ["apple", "banana", "cherry"]
```

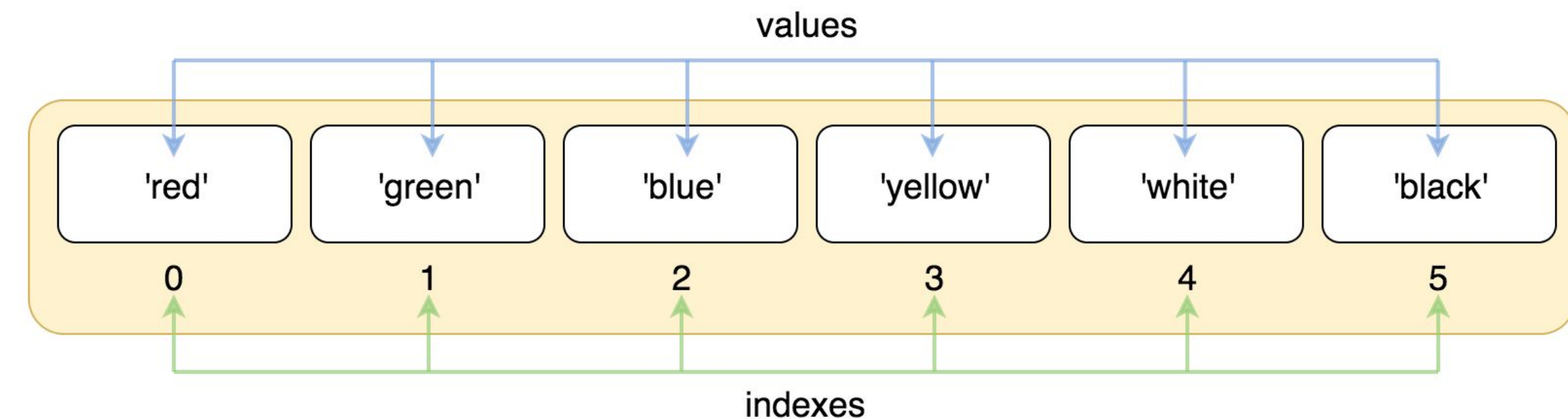
List items can be of any data type

```
list1 = ["apple", "banana", "cherry"]
```

```
list2 = [1, 5, 7, 9, 3]
```

```
list3 = [True, False, False]
```

List items are indexed and you can access them by referring to the index number. **Note:** The first item has index 0.



Using the `append()` method to append an item

```
thislist = ["apple", "banana", "cherry"]
```

```
thislist.append("orange")
```

Essentials. Dictionaries

```
>>> phone_numbers = { 'Jack': '070-02222748',  
                      'Pete': '010-2488634' }  
>>> my_empty_dict = { }  
>>> phone_numbers['Jack']  
'070-02222748'
```

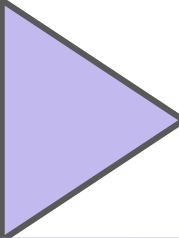
```
>>> phone_numbers['Eric'] = '06-10101010'  
>>> del(phone_numbers['Jack'])  
>>> phone_numbers  
{'Pete': '010-2488634', 'Eric': '06-10101010'}
```

Check if a key exists in a Python dictionary

```
>>> 'Jack' in phone_numbers  
True  
>>> 'Jack' not in phone_numbers  
False
```

Some built-in dictionary methods return a view object, offering a window on your dictionary's **key** and **values**.

```
phone_numbers.keys() ???  
phone_numbers.values() ???
```



Python Conditions and If Statements

Python supports the usual logical conditions from mathematics:

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```

```
a = 200
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```

Python For Loop and While Loop

```
>>> for letter in 'Hello':  
...     print(letter)  
...  
H  
e  
l  
l  
o
```

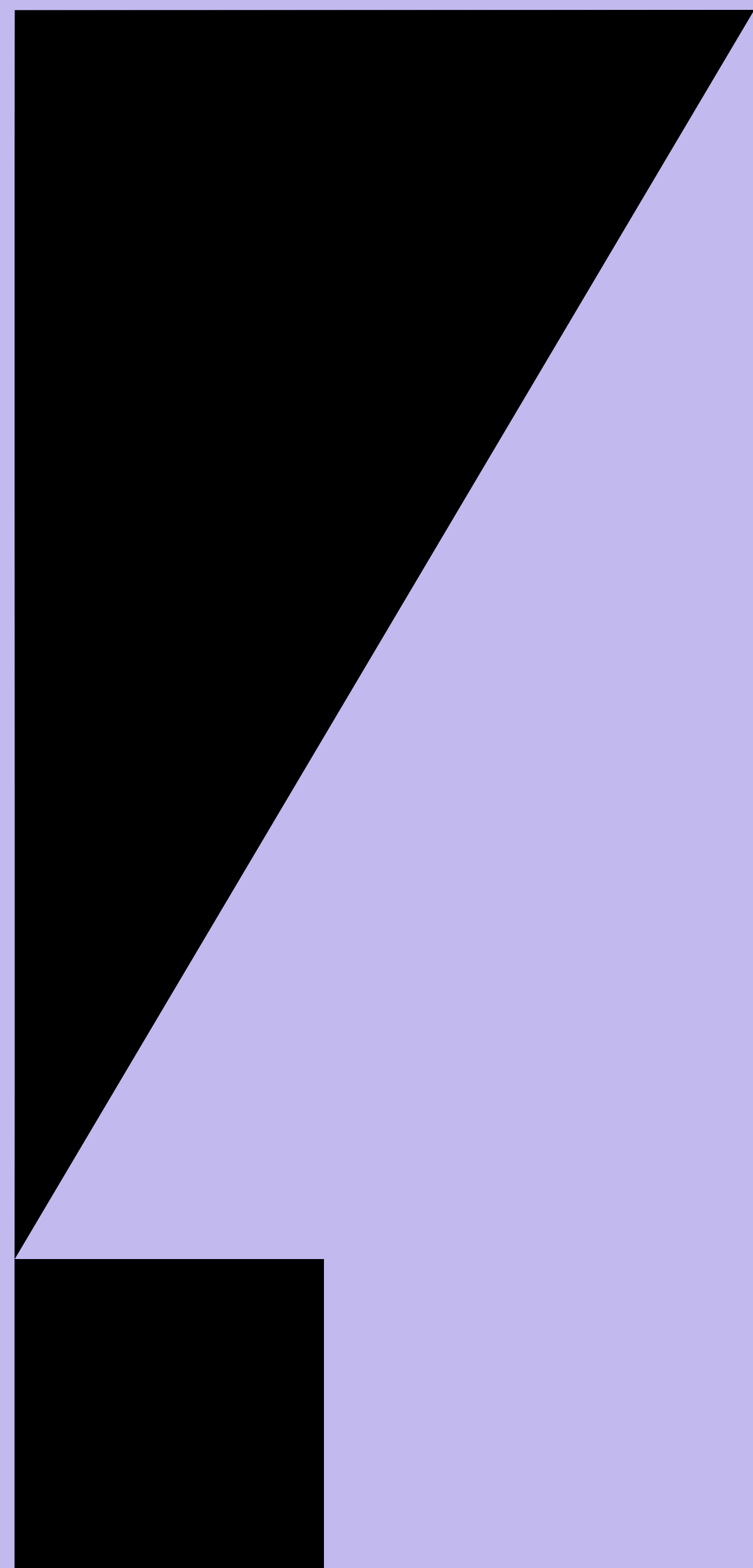
```
>>> mylist = [1, 'a', 'Hello']  
>>> for item in mylist:  
...     print(item)  
...  
1  
a  
Hello
```

```
>>> i = 1  
>>> while i <= 4:  
...     print(i)  
...     i = i + 1  
...  
1  
2  
3  
4
```

T

h

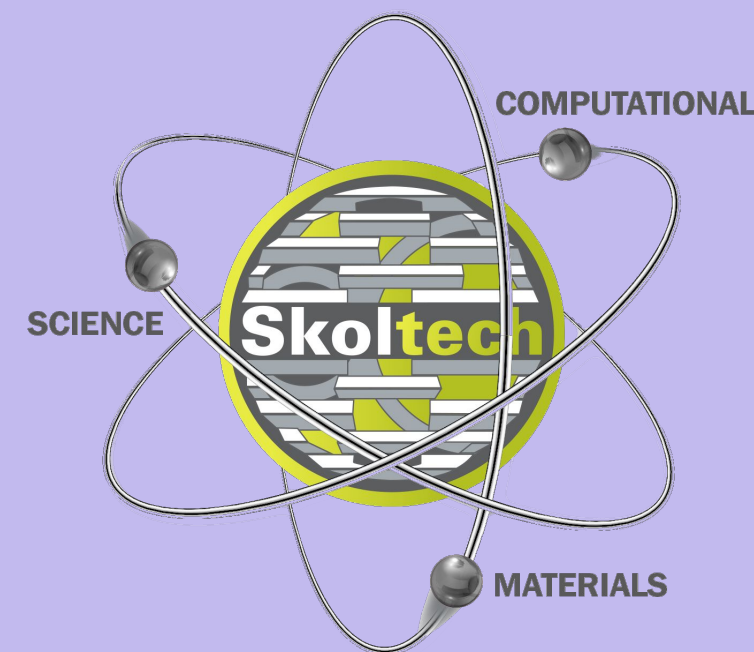
x



Acknowledgement

Skoltech
Energy

Center for
Energy Science
and Technology

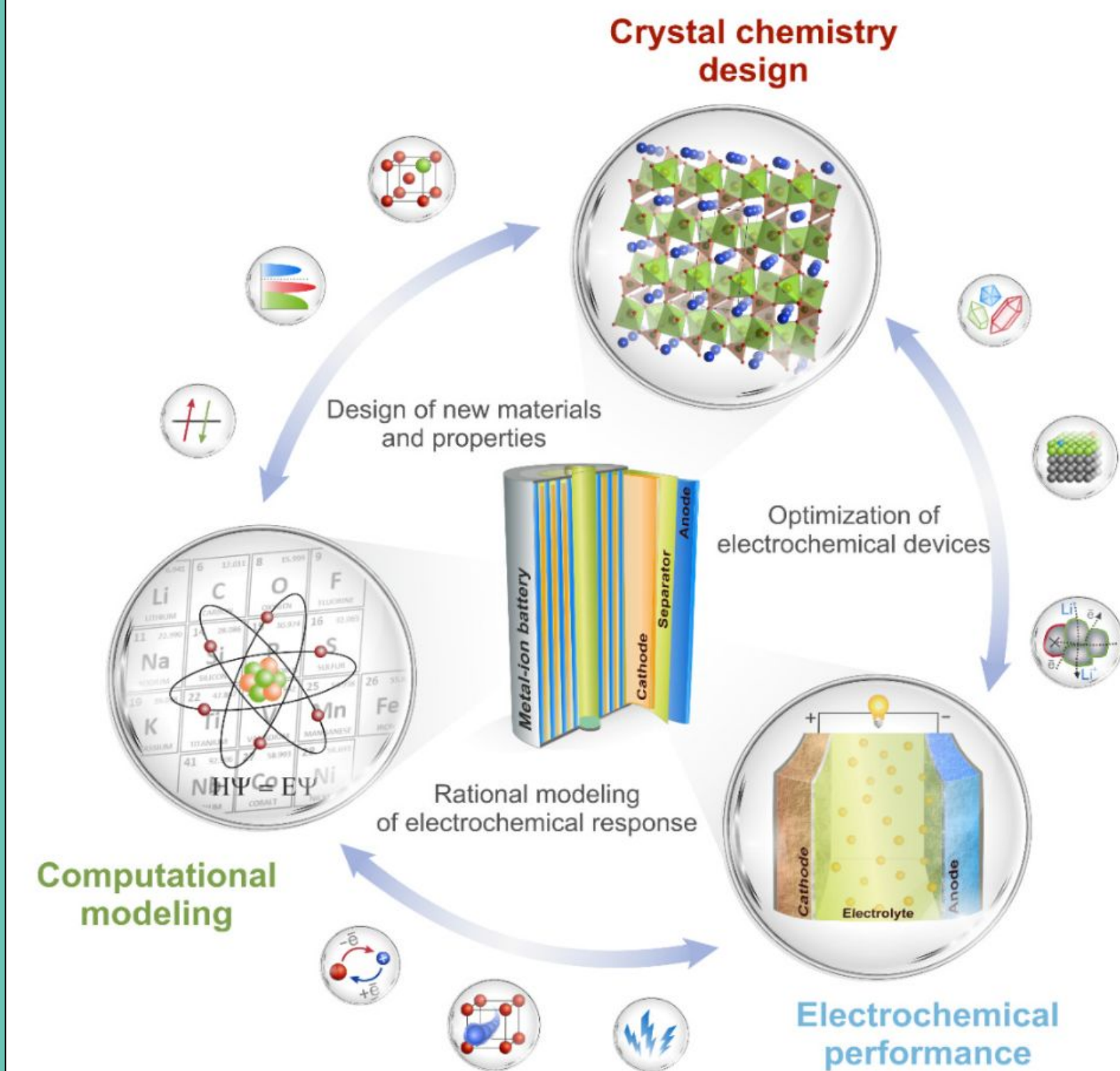


Russian Science
Foundation

Our group

Storion Research Lab

Center for Energy Science and Technology at Skoltech, Moscow



★ [MatSolver](#) - a web-service for predicting materials properties.