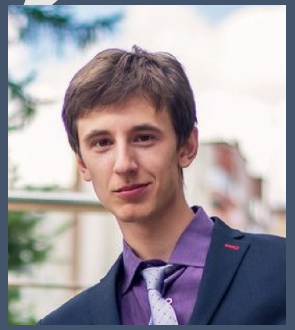
Crystallography and Crystal Chemistry
VIII International School-Conference of Young Scientists 2023

Tutorial 3: Introduction to Python and practice



Dr. **Anton** O. **Boev**

PhD in **Physics, Research Scientist**

Center for **Energy Science** and **Technology**

Skoltech, Moscow, Russian Federation

November 10th, 2023

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

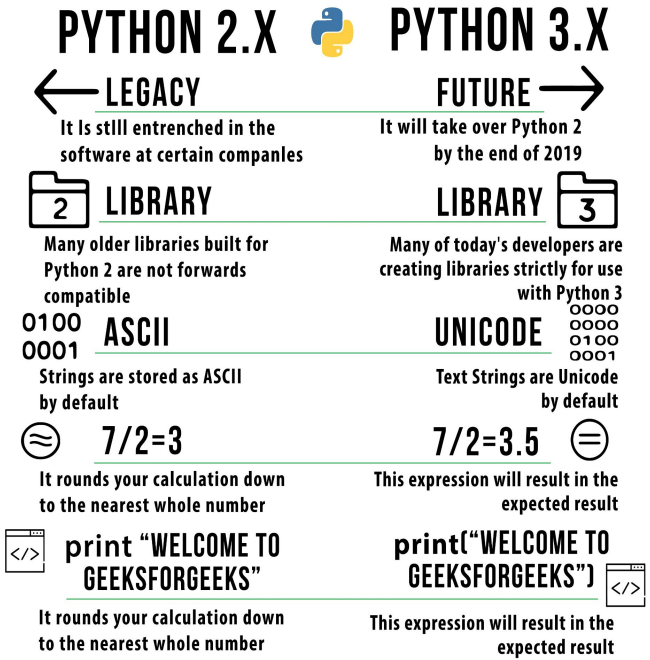
https://docs.python.org/

Python’s philosophy

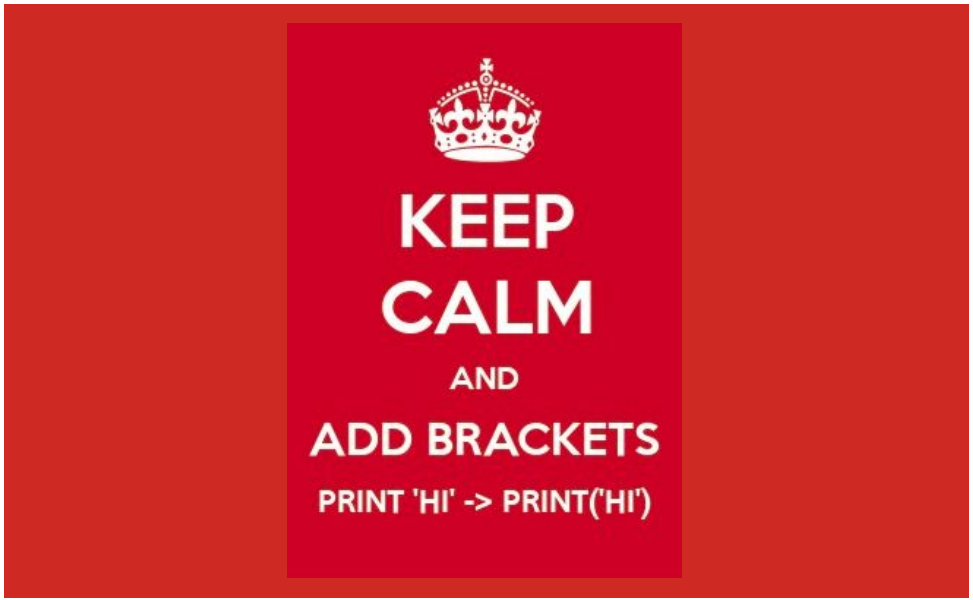
```
>>> import this
The Zen of Python, by Tim Peters

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!
>>>
```

What the version is actual?



We use version 3.6 for coding, but some other people's programs may be old and written with version 2.x.



Skoltech

Quick introduction into Python

Useful links for Python studying

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

We hope two links will be more than enough

P.S.

Now is 23:07, night before the school starts.

I’m exhausted...

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
```

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

string format

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

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

https://python.land/python-tutorial

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'
```

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

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

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

```
>>> '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'
```

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 or equal to
<	smaller than	==	is equal
>=	greater than or equal to	!=	is not equal

```
>>> 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
```

Lists

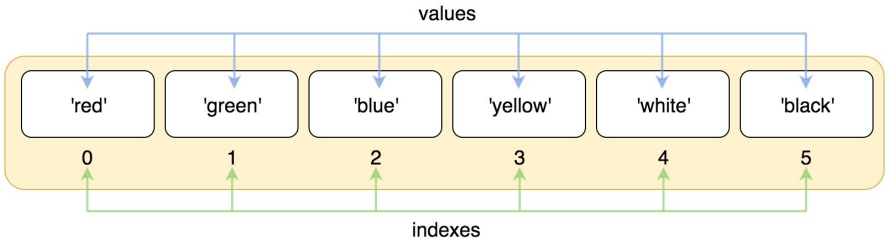
Lists are used to store multiple items in a single variable.

```
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")
```

Built-in Python list methods

Method	Application
append()	Add one or more items at the end of list
insert()	Insert an item at the defined index
extend()	Add all items of a list to the end of this list
remove()	Removes an item with given value from the list
pop()	Removes an item with given index from the list
clear()	Remove all items from the list
index()	Returns index of the item in list with given value
copy()	Returns a copy of the list to a target reference
count()	Returns number of items with specified value
sort()	Sort a list by default in ascending order; Can also sort a list in reverse order or sort by a callable key
reverse()	Reverses the order of a list

Dictionaries

Creating a Python Dictionary

```
>>> 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() ???`

Built-in Python dictionary methods

Method	What is does	Example
`clear()`	Remove all key/value pairs (empty the dictionary)	`phone_numbers.clear()`
`get(key)`	Get a single item with given key, with optional default value	`phone_numbers.get('Martha', 'Unknown person')`
`items()`	Returns a view object containing key-value pairs from the dictionary	phone_numbers.items()
`keys()`	Returns a view object with a list of all keys from the dictionary	`phone_numbers.keys()`
`values()`	Returns a view_object with a list of all values from the dictionary	`phone_numbers.values()`
`pop(key, default_value)`	Returns and removes the element with the specified key	`phone_numbers.pop('Martha')`
`popitem()`	Returns and removes the last inserted item (Python 3.7+) or a random item	`phone_numbers.popitem()`
`setdefault(key, value)`	Returns the value of specified key. If the key does not exist, it's inserted with the given value	`phone_numbers.setdefault('John Doe', 1234)`
`update(iterable)`	Add all pairs from given iterable, e.g. a dictionary	`phone_numbers.update({"Alina": 1234, "Alice", 2345})`

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
```

Functions

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

```
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil", "Refsnes")
```

Classes

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
print(p1.name)
print(p1.age)
```

Exercises

Don’t hesitate to test your Python skills;)

You can go through a small test.

Just click a link below
Please...

Good luck!

https://www.w3schools.com/python/exercise.asp?filename=exercise_syntax1

Thx!