2023

# An introduction to Python programming Language for beginners

BABAK ZOLGHADR-ASLI

SESSION FOUR | EXERCISES & CHEATSHEETS

UNIVERSITY OF QUEENSLAND & UNIVERSITY OF EXETER

## Contents

# I. Examples

```
>>> type({1, 6})
set
>>> type([1, 5, 6])
list
>>> type(5.16)
float
>>> type({2:5, 9:8})
dict
>>> type(4)
int
>>> type(4+9j)
complex
>>> type('hi')
str
>>> type((7,8))
tuple
>>> type(frozenset({1, 4, 5}))
frozenset
>>> a = 123
>>> id(a)
```

```
94521824373056
>>> a = [1, 2, 3]
>>> b = a
>>> id(a), id(b)
(139986420852928, 139986420852928)
>>> b += [4]
>>> b
[1, 2, 3, 4]
>>> a
[1, 2, 3, 4]
>>> id(a), id(b)
(139986420852928, 139986420852928)
>>> len({1, 2, 5})
3
>>> a = [1, (6, 5), {5:6, 6:7}]
>>> len(a)
3
>>> len(a[-1])
2
>>> len(a[1])
2
>>> len('BABAK')
5
>>> a = frozenset({1, 5, 6})
>>> len(a)
3
>>> abs(1)
1
>>> abs(-1)
```

```
1
>>> abs(-3.14)
3.14
>>> abs(3+4j)
5.0
>>> obj = [1, -3, 5]
>>> result=list()
>>> for i in obj:
  result.append(abs(i))
>>> result
[1, 3, 5]
>>> obj = {1, -5, 6}
>>> result = set()
>>> for i in obj:
  result.add(abs(i))
>>> result
{1, 5, 6}
>>> obj = [1, -3, 5]
>>> result = [abs(i) for i in obj]
>>> result
[1, 3, 5]
>>> obj = {1, -5, 6}
>>> result = {abs(i) for i in obj}
>>> result
{1, 5, 6}
>>> round(5.12)
5
>>> round(5.7)
6
```

```
>>> round(3.1415, ndigits=2)
3.14
>>> round(3.1415, 2)
3.14
>>> sum([1, 5, 6])
12
>>> sum({1, 5, 6})
12
>>> sum((1, 5, 6))
12
>>> obj = [1, 5, 6]
>>> sum_val = 0
>>> for i in obj:
  sum_val += i
>>> sum_val
12
>>> obj = [1, [5, 6]]
>>> sum_val = 0
>>> for i in obj:
  x = type(i)
  if x == list or x == set or x == frozenset or x == tuple:
    y = 0
    for j in i:
      y += j
    sum_val += y
  else:
    sum_val += i
>>> sum_val
12
```

```
>>> range(5)
range(0, 5)
>>> list(range(5))
[0, 1, 2, 3, 4]
>>> range(2, 5)
range(2, 5)
>>> list(range(2, 5))
[2, 3, 4]
>>> range(1,5,2)
range(1, 5, 2)
l>>> ist(range(1, 5, 2))
[1, 3]
>>> list(range(-5,-2,1))
[-5, -4, -3]
>>> list(range(-2,-5,-1))
[-2, -3, -4]
>>> list(range(5,2))
[]
>>> list_obj = [1, 2, 3]
>>> tuple_obj = (1, 2, 3)
>>> str_obj = '123'
>>> reversed(list_obj)
<list_reverseiterator at 0x7f5120ceb1d0>
>>> type(reversed(list_obj))
list_reverseiterator
>>> for i in reversed(list_obj):
  print(i)
3
2
```

```
1
>>> list(reversed(list_obj))
[3, 2, 1]
>>> list_obj[::-1]
[3, 2, 1]
>>> list_obj
[1, 2, 3]
>>> reversed(tuple_obj)
<reversed at 0x7f5120ceb090>
>>> type(reversed(tuple_obj))
reversed
>>> tuple(reversed(tuple_obj))
(3, 2, 1)
>>> reversed(str_obj)
<reversed at 0x7f5120ccb790>
>>> type(reversed(str_obj))
reversed
>>> str(reversed(str_obj)) # it doesn't work like that!
'<reversed object at 0x7f5120d06050>'
>>> ''.join(reversed(str_obj))
'321'
>>> '.'.join(reversed(str_obj))
'3.2.1'
>>> str_obj[::-1]
'321'
>>> rev_str = str()
>>> for i in range(len(str_obj)-1,-1,-1):
  rev_str += str_obj[i]
>>> rev_str
```

```
'321'
>>> list_obj = [4, 2, 7]
>>> tuple_obj = (4, 2, 7)
>>> str_obj = '427'
>>> set_obj = {4, 2, 7}
>>> dict_obj = {4:9, 2:6, 7:5}
>>> sorted(list_obj)
[2, 4, 7]
>>> sorted(list_obj, reverse=True)
[7, 4, 2]
>>> sorted(list_obj)[::-1]
[7, 4, 2]
>>> # using list methods: sort()
>>> list_obj.sort()
>>> list_obj
>>> # Note that this one is IN PLACE
[2, 4, 7]
>>> list_obj.sort(reverse=True)
>>> list_obj
[7, 4, 2]
sorted(tuple_obj)
# Note that it would alwys return lists
[2, 4, 7]
>>> sorted(str_obj)
['2', '4', '7']
>>> ''.join(sorted(str_obj)[::-1])
'742'
>>> sorted(set_obj)
[2, 4, 7]
```

```
>>> sorted(dict_obj)

>>> # it works on the keys!

[2, 4, 7]

>>> keys_sorted = sorted(dict_obj)

>>> result = list()

>>> for key in keys_sorted:

  result.append(dict_obj[key])

>>> result

[6, 9, 5]

>>> list_obj = [1, 2, 3]

>>> for i in enumerate(list_obj):

  print(i)

(0, 1)

(1, 2)

(2, 3)

>>> for index, item in enumerate(list_obj):

  print(index, item)

0 1

1 2

2 3

>>> for i, j in enumerate(list_obj):

  print(i, j)

0 1

1 2

2 3

>>> list_a = [1, 2, 3]

>>> list_b = [4, 5, 6]

>>> zip(list_a, list_b)

<zip at 0x7f5120cab640>
```

```
>>> for i in zip(list_a, list_b):
  print(i)
(1, 4)
(2, 5)
(3, 6)
>>> for i, j in zip(list_a, list_b):
  print(i, j)
1 4
2 5
3 6
>>> isinstance(2, int)
True
>>> isinstance(5.12, float)
True
>>> isinstance(2, (complex, float, int))
True
>>> print('Hi')
Hi
>>> print('did you', 'miss', "Me?")
did you miss Me?
>>> a = 5
>>> print('the value is', a)
the value is 5
>>> int_obj = -3
>>> float_obj = 4.13341
>>> str_obj = 'Python'
>>> print('integer number: %d \nfloat number: %f'%(-int_obj, float_obj))
integer number: 3
float number: 4.133410
```

```
>>> print("I'm learning %s."%str_obj)
I'm learning Python.
>>> print('%.2f'%float_obj)
4.13
>>> print('%.1E'%float_obj)
4.1E+00
>>> x = 2
>>> y = 5
>>> print('x = %g; y = %g; \nx*y = %g'%(x,y,x*y))
x = 2; y = 5;
x*y = 10
>>> x = 6
>>> y = 10
>>> print('x = %g; y = %g; \nx*y = %g'%(x,y,x*y))
x = 6; y = 10;
x*y = 60
>>> language = 'Python'
>>> session = 4
>>> temp = 29.12
>>> print(f'''We're learning {language}
and currently we're on session {session}.''')
We're learning Python
and currently we're on session 4.
>>> print(f'The outside temperature is {temp} C.')
The outside temperature is 29.12 C.
>>> help(abs)
Help on built-in function abs in module builtins:

abs(x, /)
```

```
    Return the absolute value of the argument.

>>> input_data = input('Say something: ')

Say something: Hi

>>> input_data

'Hi'

>>> threshold = 20

>>> inp = input('Enter a number: ')

>>> dig_inp = float(inp)

>>> if threshold < dig_inp:

  print(f'greater than {threshold}')

else:

  print(f'less than or equal to {threshold}')

Enter a number: 21

greater than 20

>>> iterable_obj = [True, True, False, True]

>>> any(iterable_obj)

True

>>> all(iterable_obj)

False

>>> iterable_obj = [1, 5, -8, 12, 13]

>>> max(iterable_obj)

13

>>> min(iterable_obj)

-8

>>> dict_obj = {'apple':3, 'orange': 2, 'banana':5}

>>> max(dict_obj)

'orange'

>>> min(dict_obj)

'apple'
```

```
>>> pow(2, 3)
8
>>> 2**3
8
>>> divmod(5, 2)
(2, 1)
>>> 5//2, 5%2
(2, 1)
>>> map_obj = map(abs, [1, -1, 2, 5, 6])
>>> map_obj
<map at 0x7fd571ee5a90>
>>> for i in map_obj:
      print(i)
1
1
2
5
6
>>> filter_obj = filter(any, [(True, True, False), {False, True}, [False, False, False]])
>>> filter_obj
<filter at 0x7f5120d5e490>
>>> for i in filter_obj:
      print(i)
(True, True, False)
{False, True}
>>> list(filter_obj)
[(True, True, False), {False, True}]
>>> eval('3 * 9')
```

```
27
>>> eval('abs(-9) + 2')
11
>>> a = 11
>>> eval('a+2')
13
>>> b = eval('2.4')
>>> type(b)
float
>>> eval('min([11, 14])')
11
>>> eval('[i for i in range(5)]')
[0, 1, 2, 3, 4]
>>> s = '2.4'
>>> print(type(s), type(eval(s)))
<class 'str'> <class 'float'>
>>> exec('a = 5')
>>> exec('a += 2')
>>> exec('''for i in range(5):
      print(i)''')
0
1
2
3
4
>>> exec('a=3*2')
a
6
>>> def square(number):
      return number ** 2
>>> square(2)
4
>>> square(number=2)
4
>>> def unique_member(list_1, list_2):
      return [i for i in list_1 if i not in list_2]
>>> unique_member([1,2,3], [2,5,6])
>>> def unique_member_2(list_1, list_2):
  unique_values = set(list_1) - set(list_2)
  return list(unique_values)
>>> unique_member_2([1,2,3], [2,5,6])
[1, 3]
>>> def decimal_number_breaker():
      number = float(input('enter a number:'))
      inetegr_part = int(number)
      fraction_part = abs(number - inetegr_part)
      return inetegr_part, fraction_part
>>> a = input()
>>> a, type(a)
3.14
>>> ('3.14', str)
>>> int(3.14)
3
>>> int(3.9)
3
>>> decimal_number_breaker()
enter a number:3.1415
(3, 0.14150000000000018)
```

```
>>> decimal_number_breaker()
enter a number:-3.14
(-3, 0.14000000000000012)
>>> full, digit = decimal_number_breaker()
enter a number:3.1415
>>> full, digit
(3, 0.14150000000000018)
>>> def idle_func_none():
      return None
>>> idle_func_none()
>>> def idle_func_pass():
      pass
>>> idle_func_pass()
>>> def object_info(obj):
      print('object type: %s; object identity: %d'%(type(obj), id(obj)))
>>> object_info(full)
object type: <class 'int'>; object identity: 94394821343808
>>> list_a = [1, 2, 3]
>>> list_b = [2, 5]
>>> unique_member(list_a, list_b)
[1, 3]
>>> unique_member(list_b, list_a)
>>> unique_member(list_1 = list_a, list_2 = list_b)
[1, 3]
>>> unique_member(list_2 = list_b, list_1 = list_a)
[1, 3]
>>> def room_temp_reporter(temp):
      print('Room temperature is %.2f C.'%temp)
>>> room_temp_reporter(25)
```

```
Room temperature is 25.00 C.

>>> def room_temp_reporter(temp = 25):

      print('Room temperature is %.2f C.'%temp)

>>> room_temp_reporter()

Room temperature is 25.00 C.

>>> room_temp_reporter(32)

Room temperature is 32.00 C.

>>> def test_func(a = 2, b):

      pass

>>> test_func(2)

>>> test_func(2, b = 3)

>>> test_func(2, 3)

>>> def maximum_2(a, b):

      if a > b:

            return a

      else:

            return b

>>> maximum_2(4, -10)

4

>>> def arbitrary_argument_test(*test):

      return test

>>> arbitrary_argument_test(1, 5, 6, 7)

(1, 5, 6, 7)

d>>> ef arbitrary_keyword_argument_test(**test):

      return test

>>> arbitrary_keyword_argument_test(a = 1, b = 2, c = 3)

{'a': 1, 'b': 2, 'c': 3}

>>> def maximum(*args):

      max_val = args[0]
      for arg in args:
            if arg > max_val:
                  max_val = arg
      return max_val
>>> maximum(3, 1, 12)
12
>>> maximum([1, 2, 4])
[1, 2, 4]
>>> maximum(*[1, 2, 4])
4
>>> def square(a):
      '''Returns the square of a numeric value
a: numeric values, must be a int, float or complex object'''
      value = a ** 2 # this value would be returned
      return a ** 2
>>> square(2)
4
>>> help(square)
Help on function square in module __main__:

>>> square(a)
    Returns the square of a numeric value
    a: numeric values, must be a int, float or complex object
>>> square.__doc__
'Returns the square of a numeric value\na: numeric values, must be a int,
float or complex object'
>>> print(square.__doc__)
Returns the square of a numeric value
a: numeric values, must be a int, float or complex object
```

```
>>> def a_demonstration(a = 0):
      print(a)
      a = 1
>>> a_demonstration()
0
>>> a_demonstration()
0
>>> def a_technical_trap(a = []):
      print(a)
      a.append(1)
>>> a_technical_trap()
[]
>>> a_technical_trap()
[1]
>>> a_technical_trap()
[1, 1]
>>> help(a_technical_trap)
Help on function a_technical_trap in module __main__:

>>> a_technical_trap(a=[1, 1, 1])
>>> a_technical_trap(a=[1,2,3])
[1, 2, 3]
>>> a_technical_trap()
[1, 1, 1]
>>> a = 'Which object am I?'
>>> def switcher():
      a = 'Am I here?'
>>> a
'Which object am I?'
```

```
>>> switcher()

>>> a

'Which object am I?'

>>> def switcher():

      a = 'Am I here?'

      print(a)

>>> a

'Which object am I?'

>>> switcher()

Am I here?

>>> a

'Which object am I?'

>>> def switcher():

      global a

      a = 'Am I here?'

>>> a

'Which object am I?'

>>> switcher()

>>> a

'Am I here?'

>>> lambda x:x**2

<function __main__.<lambda>>

>>> lambda x:x%2==0

<function __main__.<lambda>>

>>> lambda x, y: x+y

<function __main__.<lambda>>

>>> lambda x, y: (x**2, y+3)

<function __main__.<lambda>>

>>> f = lambda x:x**3 + 2*x
>>> f(5)

135

>>> g = lambda x, y: (x**3 - 2*y, y**2)

>>> g(1, 5)

(-9, 25)

>>> f = lambda x=2:x*2

>>> f()

4

>>> f(3)

6

>>> list(map(lambda x:x**2, range(5)))

[0, 1, 4, 9, 16]

>>> list(filter(lambda x:x>0, [-1, 2, 5, -6]))

[2, 5]

>>> g = lambda x:5*x**2

>>> [g(i) for i in range(5)]

[0, 5, 20, 45, 80]

>>> import math

>>> import math as mt

>>> math.factorial(5)

120

>>> math.pi

3.141592653589793

from math import factorial

from math import *

>>> factorial(5)

120

>>> pi

3.141592653589793
```

THE UNIVERSITY
OF QUEENSLAND
AUSTRALIA

An introduction to **Python** programming Language for *beginners* *by* Babak Zolghadr-Asli

UNIVERSITY OF
EXETER

# II. Exercises

1. Try to write a function that replicates the `all()` built-in function, that is returns `True` if all elements of an iterable objects are `True`. (5 min)

2. Try to write a function that replicates the `any()` built-in function, that is returns `True` if any elements of an iterable objects are `True`. (5 min)

3. Write down two functions that would evaluate the values of dict object, and returns the item (i.e., paired key, value stored in tuple object) of the item with the highest and lowest value, respectively. Tets your program on the following dict object. (5 min)

```
>>> dict_obj = {'apple':3, 'orange': 2, 'banana':5}
```

4. Write a Polynomial function, which takes the coefficients as a sequential object and the variable value. (5 min)

5. Guess what happens here; This is called "Recursive Function" (5 min)

```
def recursive_function(m):

  if(m > 0):

    result = m + recursive_function(m - 1)

    print(result)

  else:

    result = 0

  return result


recursive_function(5)
```

# III. Recap.

A *cheatsheet* for some of the most important print format specifications Python.

| Built-in functions | Description |
|---|---|
| `%d` | Signed integer decimal. |
| `%e` | Compact scientific notation with e in the exponent |
| `%E` | Compact scientific notation with E in the exponent |
| `%f` or `%F` | Floating point decimal format. |
| `%s` | String objects |
| `%g` | Perform as `%e` if exponent is less than -4 or not less than precision otherwise perform as `%f` |
| `%G` | Perform as `%E` if exponent is less than -4 or not less than precision otherwise perform as `%f` |

A *cheatsheet* for some of the Python's built-in functions.

| Built-in functions | Description |
|---|---|
| `help()` | Prints available documentations on an object on the terminal. |
| `input()` | Accepts an input data from user in form of a `str` object. |
| `all()` | Returns `True` if all the elements in an iterable object are `True`. |
| `any()` | Returns `True` if any elements of an iterable objects is `True`. |
| `max()` | Returns the *maximum* value in an iterable object. |
| `min()` | Returns the *minimum* value in an iterable object. |
| `pow()` | Takes two numeric arguments are passed to the function and returns the first argument raised to the second argument. |
| `divmod()` | Takes two numeric arguments and returns the quotient and remainder of their division as a `tuple` object. |
| `map()` | Takes an iterable object and applies a function on the elements. Returns the results in form of an iterator objects. |
| `filter()` | Constructs an iterator from those elements of iterable for which function returns `True`. |
| `eval()` | Takes a `str` object and evaluates it as a Python expression. |
| `exec()` | Takes a `str` object and executes it as a Python expression. |
| `open()` | Opens a file and return a corresponding `file` object. |

A *cheatsheet* for some of the most important exceptions in Python.

| Built-in exception | Description |
|---|---|
| `BaseException` | General class of exceptions in Python |
| `ArithmeticError` | General subclass for arithmetic exceptions. |
| `ZeroDivisionError` | Raises by dividing a numeric object by zero. |
| `FloatingPointError` | Raises when operation on a floating point object fails. |
| `OverflowError` | Raises as the arithmetic operation gets overwhelming to be handled. |
| `MemoryError` | Raises when the system would run out of memory. |
| `LookupError` | Raises when the index of an object cannot be found. |
| `IndexError` | Raises when the index of a sequence object cannot be found. |
| `KeyError` | Raises when the index of a mapping object cannot be found. |

| | |
|---|---|
| `AttributeError` | Raises as an attribute cannot be found for an object. |
| `KeyboardInterrupt` | Raises when the user interrupts the interpreter manually. |
| `ImportError` | Raises when a package, library, or module cannot be found. |
| `ModuleNotFoundError` | Raises when a package, library, or module cannot be found. |
| `NameError` | Raises when the interpreter cannot find an identifier. |
| `RecursionError` | Raises as the interpreter exceeds the maximum recursion depth. |
| `SyntaxError` | Raises as an invalid Python syntax is passed to the interpreter. |
| `TypeError` | Raises as an incorrect or unsupported operation is applied on an object. |
| `ValueError` | Raises as arguments with inappropriate value is passed to a function. |
| `SystemError` | Raises as an internal factor interrupts the program flow. |

THE UNIVERSITY
OF QUEENSLAND
AUSTRALIA

An introduction to **Python** programming Language for *beginners* *by* Babak Zolghadr-Asli

UNIVERSITY OF
EXETER

# BABAK ZOLGHADR-ASLI

## QUEX-JOINT PH.D. CANDIDATE

## CONTACT

@babak_zolghadr

babakzolghadrasli.wordpress.co

@babakzolghadrasli

## EMAILS

b.zolghadrasli@uq.net.au
bz267@exeter.ac.uk

## RESEARCH AREA

o Water resources planning and management
o Climate change
o Sustainable development
o Decision-Making paradigms
o Deep Uncertainty
o Optimization
o Machine Learning
o Data Mining

## AWARDS & HONORS

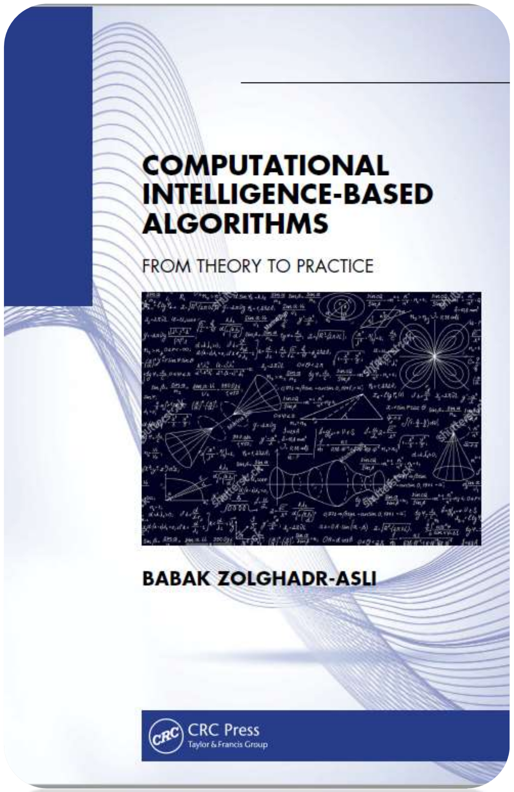
Outstanding researcher award in "the 26th Research Festival", University of Tehran (2017); Outstanding student award in "the 8th International Festival and Exhibition", University of Tehran (2018); Outstanding M.Sc. thesis award in "the 5th National Festival of Environment", Tehran Iran (2018); Winner of the "Prof. Alireaz Sepaskhah" 1st Scientific Award in water engineering [Shiraz University] (2019); Excellent Reviewer, Journal of Hydro Science & Marine Engineering (2020).

## SELECTED PUBLICATION

1. Zolghadr-Asli, B., Naghdyzadegan Jahromi, M., Wan, X., Enayati, M., Naghdizadegan Jahromi, M., Tahmasebi Nasab, M., Pourghasemi, H.R., & Tiefenbacher, J.P. (2023). "Uncovering the Depletion Patterns of Inland Water Bodies via Remote Sensing, Data Mining, and Statistical Analysis." Water, 15(8), 1508.
2. Zolghadr-Asli, B. (2023). "No-free-lunch-theorem: A page taken from the computational intelligence for water resources planning and management." Environmental Science and Pollution Research, DOI: 10.1007/s11356-023-26300-1.
3. Zolghadr-Asli, B. (2023). "Computational intelligence-based optimization algorithms: From theory to practice," CRC Press, (Typesetting and finalizing the publisher requirements).

FOR A FULL LIST VISIT: HERE

SCAN ME



COMPUTATIONAL INTELLIGENCE-BASED ALGORITHMS

FROM THEORY TO PRACTICE

BABAK ZOLGHADR-ASLI

CRC Press
Taylor & Francis Group

Coming out soon .... HOPEFULLY!!!!

## Chapter 9: Harmony Search Algorithm

SCAN ME

QUEX INSTITUTE

INTERNATIONAL SYMPOSIUM

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

University of Exeter

# Stay in touch

@babak_zolghadr

babakzolghadrasli.wordpress.com

@babakzolghadrasli

b.zolghadrasli@uq.net.au
bz267@exeter.ac.uk

SCAN ME

QUEX INSTITUTE
INTERNATIONAL SYMPOSIUM

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

UNIVERSITY OF EXETER
LUCEM SEQUIMUR

University of Exeter