2023

An introduction to Python programming Language for beginners

BABAK ZOLGHADR-ASLI

SESSION TWO | EXERCISES & CHEATSHEETS

UNIVERSITY OF QUEENSLAND & UNIVERSITY OF EXETER

Contents

I. Examples

```
>>> age = 29
>>> age
29
>>> print(age)
29
>>> type(age)
<class 'int'>
>>> id(age)
2027153065072
>>> age = 30
>>> id(age)
2027153065104
>>> age = 29
>>> id(age)
2027153065072
>>> height = 175.5
>>> height
175.5
>>> type(height)
<class 'float'>
>>> id(height)
```

```
2027192787760
>>> complex_number = 2 + 3j
>>> complex_number
(2+3j)
>>> type(complex_number)
<class 'complex'>
>>> complex_number.real
2.0
>>> complex_number.imag
3.0
>>> complex_number.conjugate()
(2-3j)
>>> complex.conjugate(complex_number)
(2-3j)
>>> abs(-3.12)
3.12
>>> abs(-12)
12
>>> abs(complex_number)
3.605551275463989
>>> round(3.7)
4
>>> round(2.7976, ndigits=3)
2.798
>>> round(1.45332, 2)
1.45
>>> bool_obj_1 = True
>>> bool_obj_2 = False
>>> type(bool_obj_1)
```

```
<class 'bool'>
>>> type(bool_obj_2)
<class 'bool'>
>>> True + 2
3
>>> False * 5
0
>>> 10/True
10.0
>>> 'a single quoted string'
'a single quoted string'
>>> "a double quoted string"
'a double quoted string'
>>> ''' 'a triple quoted string', 'another triple quoted string' '''
" 'a triple quoted string', 'another triple quoted string' "
>>> string = '''Now
you can go to the next line
or use 'quotes' '''
>>> print(string)
Now
you can go to the next line
or use 'quotes'
>>> text_2 = "Python's name is inspired by a TV show."
>>> text_3 = '"Elementary my dear Watson" is one of the most famous phrases
that was never said by Sherlock Holmes!'
>>> text_4 = '''One of Shakespeare's famous quotes is "to be, or not to be,
that is the question."'''
>>> print('This string object contains a single quote (i.e., \').')
This string object contains a single quote (i.e., ').
```

```
>>> print('skip the backslash \\.')
skip the backslash \.
>>> print('print \'')
print '
>>> print('jump to next line\nto print this!')
jump to next line
to print this!
>>> print('a\tb\tc\t you know the rest ...')
a     b     c      you know the rest ...
>>> name = 'Babak'
>>> last_name = 'Zolghadr-Asli'
>>> full_name = name + ' ' + last_name
>>> print(full_name)
Babak Zolghadr-Asli
>>> type(full_name)
<class 'str'>
>>> '*'*10
'**********'
>>> string = 'Python'
>>> string[0]
'P'
>>> string[-1]
'n'
>>> string[-6]
'P'
>>> string[5]
'n'
>>> string = 'slicing a string sequence'
>>> string[10:16]
```

```
'string'
>>> string[17:]
'sequence'
>>> string[:-18]
'slicing'
>>> string[10:16:2]
'srn'
>>> len('Babak')
5
>>> name = 'tom'
>>> name.capitalize()
'Tom'
>>> str.capitalize('pYThon IS THE BEST!')
'Python is the best!'
>>> chess_champion = 'mAgNUS caRlsEn'
>>> chess_champion.title()
'Magnus Carlsen'
>>> text = 'PyThOn 4 eVer.'
>>> text.upper()
'PYTHON 4 EVER.'
>>> text.lower()
'python 4 ever.'
>>> test = '11100111001010'
>>> str.count(test,'0')
6
>>> test.count('0')
6
>>> text = "Where's Waldo?"
>>> text.index('Waldo')
```

```
8
>>> text.find('Waldo')
8
>>> text.find('Martin Handford')
-1
>>> text = ' club  '
>>> text.strip()
'club'
>>> text = ' Hi Babak   '
>>> text.strip(' ka')
'Hi Bab'
>>> separator = ' '
>>> separator.join('abcd')
'a b c d'
>>> 'I   Love coding'.split()
['I', 'Love', 'coding']
>>> '02345'.replace('0','1')
'12345'
>>> '123'.isdigit()
True
>>> '1 2'.isdigit()
False
>>> 'abc'.isalpha()
True
>>> 'a.b'.isalpha()
False
>>> list_1 = [1, 2, 3, 4]
>>> list_1
[1, 2, 3, 4]
```

```
>>> list_2 = ['Lists', 'are', 'useful', 'tools']
>>> list_2
['Lists', 'are', 'useful', 'tools']
>>> list_3 = [4.2, 3.14]
>>> list_3
[4.2, 3.14]
>>> list_4 = [list_1, list_3]
>>> list_4
[[1, 2, 3, 4], [4.2, 3.14]]
>>> list_5 = ['it is a mess!', 2.718, 3, [1, '4', 'S']]
>>> list_5
['it is a mess!', 2.718, 3, [1, '4', 'S']]
>>> list_1 = [1, 2, 3, 4]
>>> list_1[0]
1
>>> list_1[:2]
[1, 2]
>>> list_1[-2]
3
>>> list_1 = [1, 2, 3, 4]
>>> list_1 + [5, 6]
[1, 2, 3, 4, 5, 6]
>>> list_1 * 2
[1, 2, 3, 4, 1, 2, 3, 4]
>>> list_1 = [1, 2, 3, 4]
>>> list_1.append(5)
>>> list_1
[1, 2, 3, 4, 5]
>>> list_1.append('S')
```

```
>>> list_1
[1, 2, 3, 4, 5, 'S']
>>> list_obj = [1, 2, 3, 4]
>>> list_obj.extend([5, 6])
>>> list_obj
[1, 2, 3, 4, 5, 6]
>>> list_obj = [1, 2, 3, 4, 5]
>>> list_obj.insert(1, 6)
>>> list_obj
[1, 6, 2, 3, 4, 5]
>>> list_obj = [1, 2, 3, 4]
>>> list_obj.pop()
4
>>> list_obj
[1, 2, 3]
>>> list_obj.pop(0)
1
>>> list_obj
[2, 3]
>>> list_obj.clear()
>>> list_obj
[]
>>> list_obj = [1, 1, 2, 5, 4, 5]
>>> list_obj.remove(5)
>>> list_obj
[1, 1, 2, 4, 5]
>>> list_obj = [1, 2, 3, 4]
>>> len(list_obj)
4
```

```
>>> list_obj = [1, 2, 3, 4, 1, 1]
>>> list_obj.count(1)
3
>>> list_obj = [1, 1, 2, 5, 4, 5]
>>> list_obj.index(1)
0
>>> list_obj = ['a', 'l', 'i', 'o', 'v']
>>> list_obj.reverse()
>>> list_obj
['v', 'o', 'i', 'l', 'a']
>>> list_obj = ['a', 'l', 'i', 'o', 'v']
>>> list_obj[::-1]
['v', 'o', 'i', 'l', 'a']
>>> sum([1, 5.6, 2])
8.6
>>> sum((3+1j, 2, 5.6, 2-3j))
(12.6-2j)
>>> l = [1, 5.6, 2]
>>> avr = sum(l)/len(l)
>>> avr
2.8666666666666667
>>> list_obj = [1, 4, 1, 5, 2, 3]
>>> list_obj.sort()
>>> list_obj
[1, 1, 2, 3, 4, 5]
>>> list_obj = ['B', 'a', 'b', 'a', 'k']
>>> list_obj.sort()
>>> list_obj
['B', 'a', 'a', 'b', 'k']
```

```
>>> list_obj.sort(reverse = True)
>>> list_obj
['k', 'b', 'a', 'a', 'B']
>>> list_obj = [1, -1, 2, 5, 6]
>>> sorted(list_obj)
[-1, 1, 2, 5, 6]
>>> sorted(list_obj, reverse = True)
[6, 5, 2, 1, -1]
>>> sorted('hi')
['h', 'i']
>>> a = [1, 2, 3, 4]
>>> id(a)
2010918606784
>>> a.append(5)
>>> a
[1, 2, 3, 4, 5]
>>> id(a)
2010918606784
>>> b=a
>>> b
[1, 2, 3, 4, 5]
>>> id(b)
2010918606784
>>> a
[1, 2, 3, 4, 5]
>>> list_obj = [1, 2, 3, 4, 5]
>>> list_copy = list_obj.copy()
>>> id(list_obj)
2010950888128
```

```
>>> id(list_copy)
2010918565056
>>> list_copy.pop()
5
>>> list_copy
[1, 2, 3, 4]
>>> list_obj
[1, 2, 3, 4, 5]
>>> tuple_1 = (1, 2, 3, 4)
>>> tuple_1
(1, 2, 3, 4)
>>> tuple_2 = ('Lists', 'are', 'useful', 'tools')
>>> tuple_2
('Lists', 'are', 'useful', 'tools')
>>> tuple_3 = (4.2, 3.14)
>>> tuple_3
(4.2, 3.14)
>>> tuple_4 = (tuple_1, tuple_3)
>>> tuple_4
((1, 2, 3, 4), (4.2, 3.14))
>>> tuple_5 = ('it is a mess!', 2.718, 3, (1, '4', 'S'))
>>> tuple_5
('it is a mess!', 2.718, 3, (1, '4', 'S'))
>>> tuple_obj = (1, 2, 3, 4)
>>> tuple_obj[0]
1
>>> tuple_obj[:2]
(1, 2)
>>> tuple_obj + (1, 2)
```

```
(1, 2, 3, 4, 1, 2)
>>> tuple_obj * 2
(1, 2, 3, 4, 1, 2, 3, 4)
>>> tuple_obj = (1, 2, 3, 4)
>>> len(tuple_obj)
4
>>> tuple_obj = (1, 2, 3, 4)
>>> tuple_obj.index(2)
1
>>> tuple_obj = (1, 1, 2, 3)
>>> tuple_obj.count(1)
2
>>> tuple_obj.count(4)
0
>>> set_1 = {1, 2, 3, 4}
>>> set_1
{1, 2, 3, 4}
>>> set_2 = {'Lists', 'are', 'useful', 'tools'}
>>> set_3 = {4.2, 3.14}
>>> set_4 = {'it is a mess!', 2.718, 3, (1, '4', 'S')}
>>> set_2
{'tools', 'useful', 'Lists', 'are'}
>>> set_3
{3.14, 4.2}
>>> set_4
{2.718, 3, (1, '4', 'S'), 'it is a mess!'}
>>> {1, 3, 3, 1, 2}
{1, 2, 3}
>>> set_obj = {1, 5.5, 'sets'}
```

```
>>> len(set_obj )
3
>>> set_1 = {1, 2}
>>> set_1
{1, 2}
>>> set_2 = {2, 1}
>>> set_2
{1, 2}
>>> set_obj = {1, 2, 3}
>>> id(set_obj)
2010950685568
>>> set_obj.add(4)
>>> set_obj
{1, 2, 3, 4}
>>> id(set_obj)
2010950685568
>>> set_obj.remove(4)
>>> set_obj
{1, 2, 3}
>>> id(set_obj)
2010950685568
>>> set_obj.discard(1)
>>> set_obj
{2, 3}
>>> set_obj.discard(5)
>>> set_obj.pop()
2
>>> set_obj
{3}
```

```
>>> set_obj.clear()
>>> set_obj
set()
>>> set_1 = {1, 2, 3, 4}
>>> set_2 = {3, 4, 5, 6}
>>> set_1.union(set_2)
{1, 2, 3, 4, 5, 6}
>>> set_1
{1, 2, 3, 4}
s>>> et.union(set_1, set_2)
{1, 2, 3, 4, 5, 6}
>>> set_1.intersection(set_2)
{3, 4}
>>> set.intersection(set_1, set_2)
{3, 4}
>>> set.update(set_1, set_2)
>>> set_1
{1, 2, 3, 4, 5, 6}
>>> set_1 = {1, 2, 3, 4}
>>> set_1.update(set_2)
>>> set_1
{1, 2, 3, 4, 5, 6}
>>> set_1 = {1, 2, 3, 4}
>>> set_2 = {3, 4, 5, 6}
>>> set_1.intersection_update(set_2)
>>> set_1
{3, 4}
>>> set_1 = {1, 2, 3, 4}
>>> set_1.difference(set_2)
```

```
{1, 2}
>>> set_1.difference_update(set_2)
>>> set_1
{1, 2}
>>> set_1 = {1, 2, 3, 4}
>>> set_1 - set_2
{1, 2}
>>> set_obj = {1, 2, 3}
>>> set_duplicate = set_obj.copy()
>>> frzenset_1 = frozenset({1, 2, 3})
>>> frzenset_1
frozenset({1, 2, 3})
>>> dict_obj = {'apple':1, 'orange':2, 'banana':5}
>>> dict_obj
{'apple': 1, 'orange': 2, 'banana': 5}
>>> dict_obj = {1:50, 1.2:22, ('a', ('a',)):12, frozenset([1,2]):'1'}
>>> dict_obj
{1: 50, 1.2: 22, ('a', ('a',)): 12, frozenset({1, 2}): '1'}
>>> dict_obj = {'apple':1, 'orange':2, 'banana':5}
>>> len(dict_obj)
3
>>> dict_obj = {'apple':1, 'orange':2, 'banana':5}
>>> dict_obj['apple']
1
>>> dict_obj = {'apple':1, 'orange':2, 'banana':5}
>>> dict_obj['melon'] = 10
>>> dict_obj
{'apple': 1, 'orange': 2, 'banana': 5, 'melon': 10}
>>> dict_obj['apple'] = 5
```

```
>>> dict_obj
{'apple': 5, 'orange': 2, 'banana': 5, 'melon': 10}
>>> dict_obj = {'apple':1, 'orange':2, 'bannana':5}
>>> dict_obj.get('apple')
1
>>> dict_obj = {'apple':1, 'orange':2, 'banana':5}
>>> dict_obj.keys()
>>> dict_keys(['apple', 'orange', 'banana'])
>>> dict_obj.values()
>>> dict_values([1, 2, 5])
>>> dict_obj.items()
>>> dict_items([('apple', 1), ('orange', 2), ('banana', 5)])
>>> dict_obj.update([('blueberry', 4), ('banana', 2)])
>>> dict_obj
{'apple': 1, 'orange': 2, 'banana': 2, 'blueberry': 4}
>>> dict_obj.pop('orange')
2
>>> dict_obj
{'apple': 1, 'banana': 2, 'blueberry': 4}
>>> dict_obj.popitem()
('blueberry', 4)
>>> dict_obj
{'apple': 1, 'banana': 2}
>>> dict_obj.clear()
>>> dict_obj
{}
>>> dict_duplicate = dict_obj.copy()
>>> list_obj = [1, 2, 3]
>>> tuple_obj = tuple(list_obj)
```

```
>>> set_obj = set(list_obj)
>>> str_obj = 'Ready to be converted!'
>>> list(str_obj)
['R', 'e', 'a', 'd', 'y', ' ', 't', 'o', ' ', 'b', 'e', ' ', 'c', 'o', 'n',
'v', 'e', 'r', 't', 'e', 'd', '!']
>>> 1 + 5
6
>>> 7.8 + 9.9
17.7
>>> (2j) + (5+1j)
(5+3j)
>>> -(2+3j)
(-2-3j)
>>> 2 * 5
10
>>> 6 / 1.4
4.285714285714286
>>> 5//3
1
>>> 4%3
1
>>> 4**3
64
>>> 4**.5
2.0
>>> a = 2
>>> a += 7
>>> a
9
```

```
>>> a *= 2
>>> a
18
>>> a /= 3
>>> a
6.0
>>> a **= 2
>>> a
36.0
>>> a -= 6
>>> a
30.0
>>> a //= 4
>>> a
7.0
>>> a %= 3
>>> a
1.0
>>> # comment
>>> a = 2 #comment
```

II. Exercises

1. Guess what would be printed on the terminal. (2 min)

A.

```
>>> 1 / False
```

B.

```
>>> True / (False + True)
```

C.

```
>>> True / False + True
```


2. Use subscription operator to select the sub-sequence 'Python' in each case. (2 min)

A. `text_2 = 'Have you seen Monty Python?'`

B. `text_3 = 'P--y--t--h--o--n'`

3. Guess what would be printed in the terminal without executing the code. (2 min)

```
>>> text = 'so by learning this you can work with strings.'

>>> text[0] + text[6] + text[11] + text[24] + text[-5:-2]
```

4. Guess what would be printed in the terminal without executing the code. (1 min)

```
>>> text = 'drawkcab daer uoy nac'

>>> text[::-1]
```

5. An engineer has been monitoring the performance of a water resources system over a specific time frame. In case the performance was acceptable in a given month, the engineer would mark it with the letter 'S', otherwise the letter 'F' would be used to label the said period. The results were archived as the following string:

```
>>> system_performance = 'SSSFFSSFSSFS'
```

A. How long was the system under monitor?

B. In which time step the system failed first?

C. How many times the system performed successful?

D. How many time the system failed?

E. How many times the system bounced back to the success status after a failure event?

F. What if the next time frame is a success; Update the string.

G. You have been asked to replace 'S' with 1 and 'F' with 0. Recreate the string in which the system performance is logged.

(5min)

6. Guess what would be printed in the terminal in each case. (2 min)

A.

```
>>> a = [1, 2, 3]
>>> b = a
>>> c = b
>>> c += [4]
>>> c
>>> b
>>> a
```

B.

```
>>> a = 1
>>> b = a
>>> c = b
>>> c = 2
>>> c
>>> b
>>> a
```

7. Guess what would be printed in the terminal in each case. (2 min)

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

>>> list_obj.count(1)

>>> list_obj[0].count(1)
```

8. We intend to write a script that return the nth largest value in a given list sequence. Say, the list is `[1, 5, 10, 11, 8, 12, 14, 13, 9, 2, 7, 3]` and we want to see what is the second greatest value and 5th smallest value. (2 min)

9. Can you find a way, implementing what you have learn thus far to return the unique elements in a sequence? (2 min)

```
>>> tuple_obj = (1, 1, 5, 6, 6, 8, 8)

>>> list_obj = [1, 1, 5, 6, 6, 8, 8]

>>> str_obj = 'babak'
```

10. Guess what would be printed in terminal. (1 min)


```
>>> (8+j) * 2
```



```
>>> (8+1j) * 2
```

III. Recap.

A *cheatsheet* for numeric-oriented data type's methods and attributes.

Method/Attribute	Description
`.real`	The attribute that stores the real part of a `complex` object.
`.imag`	The attribute that stores the imaginary part of a `complex` object.
`.conjugate()`	Returns the reflection symmetry of a `complex` object with respect to the real axis.

A *cheatsheet* for arithmetic operators in Python.

Arithmetic operator	Description
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	True division
`//`	Floor division
`%`	Remainder or modulo
`**`	Exponentiation

A *cheatsheet* for assignment operators in Python.

Assignment operator	Description
`=`	Assignment operator
`+=`	Augmented addition assignment
`-=`	Augmented subtraction assignment
`*=`	Augmented multiplication assignment
`/=`	Augmented true division assignment
`//=`	Augmented floor division assignment
`%=`	Augmented remainder assignment
`**=`	Augmented exponentiation assignment

A *cheatsheet* for `str` objects.

Method/Attribute	Description
`+`	Merges two given `str` objects.
`*`	If multiplied by an `int`, it would duplicate the `str` for the specified time.
`.capitalize()`	Returns the capitalized version of the `str` object.
`.title()`	Return a `str` object where each word in the original sequence is title-cased.
`.upper()`	Return a copy of the `str` object converted to uppercase letters.
`.lower()`	Return a copy of the `str` object converted to lowercase letters.
`.count()`	Return the number of times a specified character appears in a `str` object.
`.index()`	Return the index in which a certain sub-sequence first appear in a `str` object.
`.find()`	Return the index in which a certain sub-sequence first appear in a `str` object.
`.strip()`	Return a copy of the `str` object with whitespaces removed from the object.
`.join()`	Merge the items in an iterable with a specified character in between them.

`.split()`	Break down a `str` object based on a specified separator.
`.replace()`	Return a `str` object with a specific sub-string is replaced by another.
`.isdigit()`	Determines whether if all the characters in a `str` object are digits.
`.isalpha()`	Determines whether if all the characters in a `str` object are letters.

A *cheatsheet* for `list` objects.

Method/Attribute	Description
`.append()`	Appends an element to the end of a `list` object.
`.extend()`	Concatenates the elements of an iterable to the end of a `list` object.
`.insert()`	Adds an element in a specific position within a `list` object.
`.clear()`	Removes all elements from a `list` object.
`.pop()`	Drops an element with a specified index from a `list` object.
`.remove()`	Removes a specified value from a `list` object.
`.count()`	Return the number of times a specified character appears in a `list` object.
`.index()`	Returns the index that a certain element first appear in a `list` object.
`.reverse()`	Reverses the order in which the elements are arranged in a `list` object.
`.sort()`	Sorts the elements in a `list` object.
`.copy()`	Returns a shallow copy of a `list` object.

A *cheatsheet* for `tuple` objects.

Method/Attribute	Description
`.count()`	Return the number of times a specified character appears in a `tuple` object.
`.index()`	Returns the index that a certain element first appear in a `tuple` object.

A *cheatsheet* for `set` objects.

Method/Attribute	Description
-	Return the difference of two given `set` objects.
`.add()`	Appends an element to a `set` object.
`.remove()`	Removes an element from a `set` object.
`.discard()`	Removes an element from a `set` object.
`.pop()`	Removes an alternatively selected element from a `set` object.
`.clear()`	Removes all elements from a `set` object.
`.union()`	Returns the union of two given `set` objects.
`.intersection()`	Returns the intersection of two given `set` objects.
`.update()`	An in-place version of the `.union()` method.
`.intersection_update()`	An in-place version of the `.intersection()` method.
`.difference()`	Return the difference of two given `set` objects.
`.difference_update()`	An in-place version of the `.difference()` method.
`.copy()`	Returns a shallow copy of a `set` object.

A *cheatsheet* for `frozenset` objects.

Method/Attribute	Description
-	Return the difference of two given `frozenset` objects.
`.union()`	Returns the union of two given `frozenset` objects.

`.intersection()`	Returns the intersection of two given `frozenset` objects.
`.difference()`	Return the difference of two given `frozenset` objects.
`.copy()`	Returns a shallow copy of a `frozenset` object.

A *cheatsheet* for `dict` objects.

Method/Attribute	Description
`.get()`	Returns the value for a specified key in a `dict` object.
`.keys()`	Returns the keys of a `dict` object.
`.values()`	Returns the values of a `dict` object.
`.items()`	Returns the items stored in a `dict` object.
`.update()`	Append an item to a `dict` object.
`.pop()`	Removes a specified key from a `dict` object.
`.popitem()`	Removes a specified item from a `dict` object.
`.clear()`	Removes all elements from a `dict` object.

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

Built-in functions	Description
`id()`	Returns the identity of an object in Python.
`print()`	Prints the values passed to the function in the terminal.
`len()`	Returns the number of items in a container.
`abs()`	Returns the absolute value of numeric data type.
`round()`	Returns number rounded version of numeric values.
`reversed()`	Returns the reversed version of an iterable object.
`sum()`	Returns the summation of all elements in an iterable object.
`sorted()`	Returns a new sorted `list` object from the items in iterable object.
`type()`	Returns the type of an object in Python.

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

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

AWARDS & HONORS

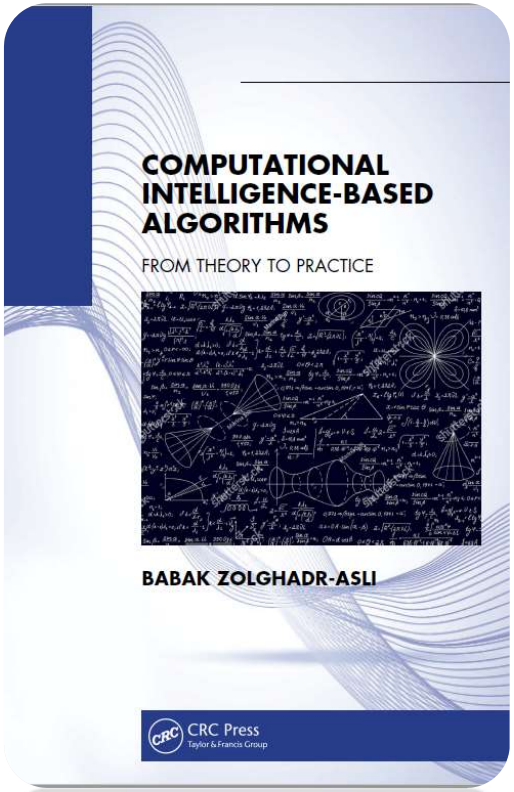
Outstanding researcher award in "the 26th Research Festival", University of Tehran (2017); Outstanding student award in "the 8th International Festival and Exhibition", University of Tehran (2018); Outstanding M.Sc. thesis award in "the 5th National Festival of Environment", Tehran Iran (2018); Winner 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



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