Loops

Prof.Dr. Bahadır AKTUĞ
BME362 Introduction To Python

**Compiled from sources given in the references.*

Loops

- The loops in Python can be formed directly over the datatypes(lists, tuples, dictionaries, strings) or with “while/for” structures.
- When the number of repetition is known beforehand, “for” is with a loop variable or direct looping over datatypes is preferred.
- When the loop is to be executed with a specific condition, then “while” loop structure is preferred.
- Both “for” and “while” can be used in a nested structure.
- There are also other control commands for both loop commands:
  - continue
  - pass
  - break
- If a loop is formed over a sequence and the elements of the sequence are also sequences, then multiple loop variables can be used.

Loops

Directly with sequences:

Lists, tuples, sets, dictionaries and strings can be used to form loops.

```
>>> colors= set([“Yellow",“Blue",“Green"])
>>> for color in colors:
...         print(color)
...
Yellow
Blue
Green
```

Loops

Directly with sequences:

Lists, tuples, sets, dictionaries and strings can be used to form loops.

```
>>> classname = “EEE105"
>>> for letter in classname:
...    print(letter)
...
E
E
E
1
0
5
```

Some useful functions for loops

range()

- range() function produces a “range” object which consists of integers within a specified interval.
- This object can then be transformed into another datatype (list, tuple etc.).
- The general form of the command is given below:

```
range([start], end[, increment])
```

- The parameters of the "range" command have to be integers.
- Since version 3, the range command in Python returns an “iterator” object.

Some useful functions for loops

range() examples:

```
>>> for i in range(5):
...         print(i)
...
0 1 2 3 4
```

```
>>> list(range(0,10,3))
[0, 3, 6, 9]
```

```
>>> list(range(5,10))
[5, 6, 7, 8, 9]
```

Some useful functions for loops

enumerate()

- "enumerate" command produces an ordered index list for the elements of a sequence.
- "enumerate" command actually produces an “enumerate” object consisting of integers within the specified interval.
- “enumerate” object can be converted to any sequence type (lists, tuples etc.). The general form of the command is:

- enumerate(sequence [, start=0])

- The parameters of "enumerate" command must be integers.

Some useful functions for loops

enumerate() examples:

```
>>> choices = ['döner','adana','iskender','mantı']
>>> list(enumerate(choices))
[(0, 'döner'), (1, 'adana'), (2, 'iskender'), (3, 'mantı')]

>>> for index, item in enumerate(choices, start = 1):
...     print(index, item)
...
1 döner
2 adana
3 iskender
4 mantı
```

Some useful functions for loops

zip()

- "zip" function takes multiple sequences as input and glues them pairwise in an ordered manner.
- "zip" command produces a zip object within the dimensions of the input sequences.
- This object can be converted into another sequence (list, tuple etc.).
- The general form of the command is:

```
zip(a, b [, c, d, …])
```

Some useful functions for loops

zip() examples

```
>>> zip(range(5), range(1,20,2))
[(0, 1), (1, 3), (2, 5), (3, 7), (4, 9)]

>>> colors = ['red', 'green', 'blue']
>>> vals = [55, 89, 144, 233]
>>> for col, val in zip(colors, vals):
...         print(col, val)
('red', 55)
('green', 89)
('blue', 144)
```

Loops

for:

When a fixed number of loops is desired, the “for” command can be used as follows:

```
>>> for i in range(5):
...     print(i)
...
0
1
2
3
4
```

Loops

while:

- When the termination of a loop depends on a condition, “while” structure is preferred.
- There must be a “boolean” expression which evaluates to either “true” or “false” after "while“

```
>>> n = 3
>>> i = 0
>>> while i < n:
...   print(i)
...   i += 1
...
0
1
2
```

Commands within Loops

break

- “break” is necessary to terminate a loop (either while or for) at a specific point.
- When there are nested loops, the innermost loop is terminated.

```
>>> for letter in “EEE105":
...     if letter == ‘1':
...         break
...     print(letter)
...
E
E
E
```

Commands within Loops

continue

- When it is needed to return to the loop command and continue the loop with the next value, “continue” command is used (both for while and for).
- When the “continue” is used, the loop continues with next element (if any) and skips over the rest of loop block.

```
>>> for letter in “EEE105":
...     if letter == ‘1':
...         continue
...     print(letter)
...
E
E
E
0
5
```

Commands within Loops

pass

- When it is need to fill in a command block which does not do anything, “pass” command can be used as a placeholder.

```
>>> for letter in “EEE105":
...         if letter == ‘0':
...             pass
...         else:
...             print(letter)
...
E
E
E
1
5
```

References

1 Wentworth, P., Elkner, J., Downey, A.B., Meyers, C. (2014). How to Think Like a Computer Scientist: Learning with Python (3nd edition).
2 Pilgrim, M. (2014). Dive into Python 3 by. Free online version: DiveIntoPython3.org ISBN: 978-1430224150.
3 Summerfield, M. (2014) Programming in Python 3 2nd ed (PIP3) : - Addison Wesley ISBN: 0-321-68056-1.
4 Summerfield, M. (2014) Programming in Python 3 2nd ed (PIP3) : - Addison Wesley ISBN: 0-321-68056-1.
5 Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, 2001-, http://www.scipy.org/.
6 Millman, K.J., Aivazis, M. (2011). Python for Scientists and Engineers, Computing in Science & Engineering, 13, 9-12.
7 John D. Hunter (2007). Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95.
8 Travis E. Oliphant (2007). Python for Scientific Computing, Computing in Science & Engineering, 9, 10-20.
9 Goodrich, M.T., Tamassia, R., Goldwasser, M.H. (2013). Data Structures and Algorithms in Python, Wiley.
10 http://www.diveintopython.net/
11 https://docs.python.org/3/tutorial/
12 http://www.python-course.eu
13 https://developers.google.com/edu/python/
14 http://learnpythonthehardway.org/book/