



# Python Programming

## Lists

Mariusz Dzieńkowski

Institute of Computer Science
Lublin University of Technology
m.dzienkowski@pollub.pl

# List basic

- A list is a type that stores a sequential collection of elements. numbers.
- Many other programming languages uses a type called an array to store a sequence of data.
- An array has a fixed size. A Python list's size is flexible.
- Elements in a list can be accessed through an index operator.
- A list can contain the elements of the same type or mixed types.
- The elements in a list are separated by commas and are enclosed by a pair of brackets **[]**.

# Creating lists

- `list1 = list()` – create an empty list
- `list2 = list([1,3,4])` – create a list with elements 1, 3, 4
- `list3 = list(["red", "blue", "green")` – create a list with strings
- `list4 = list(range(2,5))` – create a list with elements 2, 3, 4
- `list5 = list("alpha")` – create a list with characters a, l, p, h, a
- `list6 = []` – create an empty list
- `list6 = [5,6,7]` – create an empty list
- `list6 = ["yellow","black"]`
- `list6 = [2, "three"]`

# Functions for lists

- Built-in functions used with lists: `len, max, min, sum, shuffle`

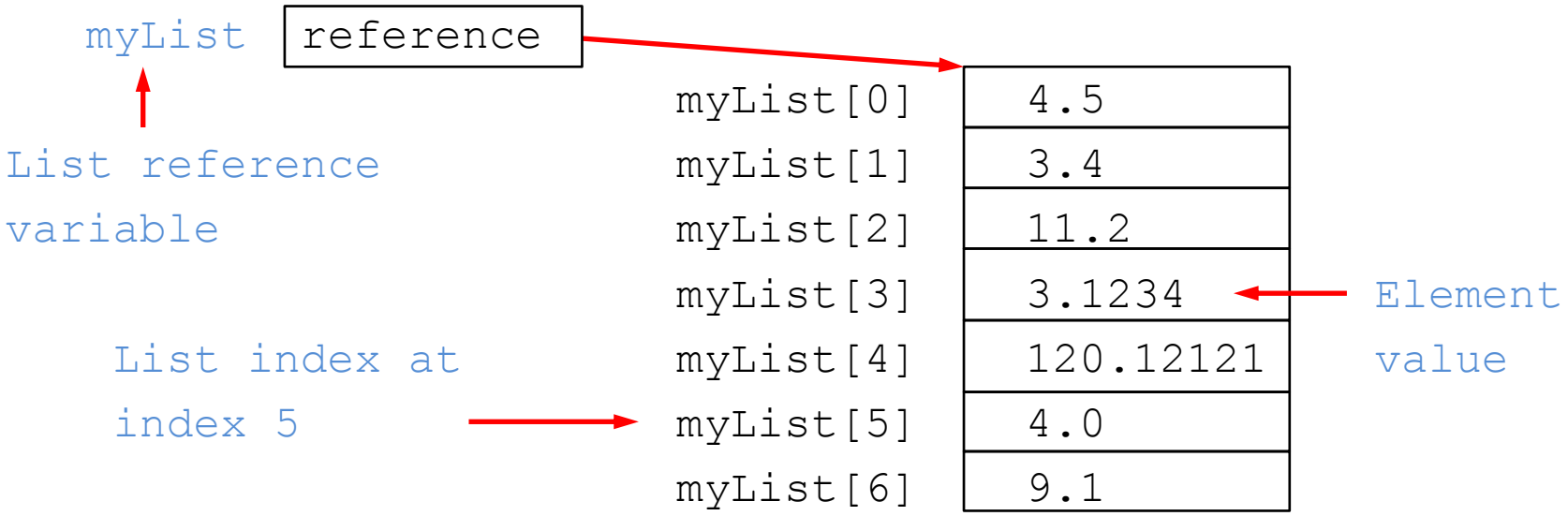
```
>>> list1 = [1, 3, 5, 0, 12]
>>> import random
>>> radnom.shuffle(list1)
>>> list1
[5, 3, 1, 12, 0]
```

- Invoking `radnom.shuffle(list1)` randomly shuffles the elements in list1.

# Index operator []

- List indexes are 0 based – they range from `0` to `len(myList)-1`.

```
myList = [4.5, 3.4, 11.2, 3.1234, 120.12121,4.0, 9.1]
```




- The `myList` has 7 elements with indexes from 0 to 9.

# Negative numbers as indexes

- Negative numbers as indexes are used to reference positions relative to the end of the list.
- The actual position is obtained by adding the length of the list with the negative index.

```
>>> list1 = [1, 3, 5, 0, 12, 45]
>>> list[-1]
45
>>> list1[-3]
0
```

# List slicing

- The index operator allows you to select an element at the specified index.
- The slicing operator returns a slice of the list using the syntax `list[start : end]`.
- The slice is a sublist from index `start` to index `end-1`.

```
>>> list1 = [1, 3, 5, 0, 12, 45]
>>> list[2 : 5]
[5, 0, 12]
```

- Negative index in slicing:

```
>>> list1 = [1, 3, 5, 0, 12, 45]
>>> list[1 : -3]
[3, 5]
>>> list[-4 : -2] #list1[-4 +len(list1):
[5, 0]
```

# Concatenation and repetition operators

- The concatenation operator (+) is used to join two lists
- The repetition operator (*) is used to replicate elements in a list.

```
>>> list1 = [4, 5]
>>> list2 = [1, 3]
>>> list3 = list1 + list2
[4, 5, 1, 3]
>>> list4 = 3 * list1
>>> list4
[4, 5, 4, 5, 4, 5]
```

- Determination if an element is in a list by using `in` or `not in`

```
>>> list1 = [4, 5, 9, 3, 8, 1, 2]
>>> 8 in list1
True
```

# Traversing elements

- The elements in a Python list are iterable.
- The for loop enables to traverse the list sequentially without using an index variable.

```
for u in myList:
    print(u)
```

- To traverse the list in different order or change elements in it is possible to use an index variable.

```
for i in range(0, len(myList),2):
    print(myList[i])
```

# Comparing lists

- The comparison operators >, >=, <, <=, ==, !=
- For comparison the two lists must contain the same type of elements.

```
>>> list1 = [”red”, ”green”, ”blue”]
>>> list2 = [”green”, ”blue”, ”red”]
>>> list2 == list1
False
>>> list2 >= list1
False
>>> list2 < list1
True
>>> list2 <= list1
True
```

# List comprehensions

- List comprehensions provide a concise way to create a sequential list of elements.
- A list comprehension consists of brackets containing an expression followed by a for clause, then zero or more for or if clauses.

```
>>> list1 = [x for x in range(5)]
>>> list1
[0, 1, 2, 3, 4]
>>> list2 = [0.5*x for x in list1]
>>> list2
[0.0, 0.5, 1.0, 1.5, 2.0]
>>> list3 = [x for in list2 if x < 1.5]
>>> list3
[0.0, 0.5, 1.0]
```

# List methods

| *Function* | *Description* |
|---|---|
| `append(x: object): None` | Adds an element x to the end of the list. |
| `count(x: object): int` | Returns the number of items element x appears in the list. |
| `extend(l :list): None` | Appends all the elements in l to the list. |
| `index(x: object): int` | Returns the index of the first occurrence of element x in the list. |
| `insert(index: int, x: object): None` | Inserts an element x at a given index. |
| `pop(i): object` | Removes the element at the given position and returns it. |
| `remove(x: object): None` | Removes the first occurrence of element x from the list. |
| `reverse(): None` | Reverses the elements in the list. |
| `sort(): None` | Sorts the elements in the list in ascending order. |

# Splitting a string into a list

- The str class conatins the `split` method, which is useful for splitting items in a string into a list.

```
>>> items = ”Jane John Peter Susan”.split()
['Jane', 'John', 'Peter', 'Susan']
>>> items = ”04-12-2001”.split(”-”)
['04', '12', '2001']
```

# Inputing lists

- Reading data from the console into a list:

  - enter one data item per line and append it to a list in a loop

```
lst = []
print(”Enter 10 numbers: ”)
for i in range(10):
     lst.append(eval(input()))
```

  - enter the data in one line separated by spaces

```
s = input(”Enter 10 numbers separated by spaces: ”)
items = s.split(”-”)
lst = [eval(x) for x in items] #convert items to
                                numbers
```

# Problem 1

**List processing –** an application that performs the following functions:

- Creates a list, reads numbers from console and adds them to the list until entered data is not numeric
- Finds a maximum value from the list and returns the value and its position in a list
- Finds a minimum value from a list and returns the value and its position in the list
- Sum numbers between the minimum and the maximum numbers in a lits
- Inserts a number into a list. To do this enter a number and its position from console. Check whether entering data is numeric.

# Problem 1 - solution

- Source code

```
def createlist():
    numlist=[]
    print("Enter a number or the word 'end' to finish:")
    i=0
    while True:
        x=input()
        if not x.isnumeric():
            break
        numlist.append(int(x))
    return numlist

def maximum(nl):
    return max(nl),nl.index(max(nl))

def minimum(nl):
    return min(nl),nl.index(min(nl))
```

# Problem 1 - solution

- Source code

```
def minmaxsum(nl,min,max):
    s = 0
    minv,minp = min
    maxv,maxp = max
    if minp < maxp:
        for i in range(minp+1,maxp,1):
            print(nl[i])
            s += nl[i]
    else:
        for i in nl[maxp+1:minp]:
            print(i)
            s += i
    return s

def insertToList(list1):
    print("Enter a number:")
    x=input()
    print("Enter a position:")
    pos=input()
    if x.isnumeric():
        list1.insert(int(pos),int(x))
    return list1
```

# Problem 1 - solution

- Source code

```
x=createlist()
print("Max=",maximum(x))
print("Min=",minimum(x))
print("Sum between min and max =",minmaxsum(x,minimum(x),maximum(x)))

print(insertToList(x))
```

# Problem 2

**Counting the occurences of each digit –** a program that counts the occurrence of each number among 10 numbers from 0 to 9 in a list of 100 random numbers.

- Guidelines:
  - Generate 100 numbers randomly and assign them to a list of numbers.
  - Count the occurrences of each number in the list. To do so, create a list named counts that has 10 int values, each of which counts the occurrences of a number.
  - Display the numbers in the list with 20 on each line.
  - Display the counts list.
  - Display the results in the form of histogram.

# Problem 2

- Here is a sample run of the program:

```
Random numbers:
2 7 1 4 6 5 7 9 0 1 4 2 9 0 6 3 3 5 5 1
5 9 9 2 5 1 7 8 5 3 6 3 7 4 7 9 5 4 7 6
7 9 0 9 5 2 9 1 5 1 7 9 6 2 4 0 6 6 6 1
0 2 7 0 1 1 0 9 5 0 8 5 0 0 8 1 3 9 0 0
9 3 1 8 2 4 2 5 7 8 0 2 0 2 9 1 8 5 8 9
Occurrences of numbers from 0 to 9:
[14, 12, 10, 6, 6, 13, 8, 10, 7, 14]
Histogram
0**************
1************
2**********
3******
4******
5*************
6********
7**********
8*******
9**************
```

# Problem 2 - solution

- Source code

```
from random import randint
def randomList1():
    rl=list()
    for i in range(0,100):
        rl.append(randint(0,9))
    return rl

def randomList2():
    return [randint(0,9) for a in range(0,100)]

def countNumbers(numbers):
    counts = 10 * [0]
    for i in range(len(numbers)):
        counts[numbers[i]] += 1
    return counts
```

# Problem 2 - solution

- Source code

```
def displayList(numbers):
    for i in range(len(numbers)):
        if(i+1)%20==0:
            print(numbers[i])
        else:
            print(numbers[i],end=' ')

def histogram(counts):
    for i in range(len(counts)):
        print(i,end='')
        for j in range(counts[i]):
            print('*',end='')
        print()
```

# Problem 2 - solution

- Source code

```
def main():
    numlist = randomList1()
    print('Random numbers:')
    displayList(numlist)
    print('Occurrences of numbers from 0 to 9:')
    print(countNumbers(numlist))
    print('Histogram')
    histogram(countNumbers(numlist))

main()
```

# Problem 3

**Eliminate duplicates –** a program with a function that returns a new list by eliminating the duplicate values in the list.

- Guidelines:
  - Use the following function header:

    ```
    def eliminateDuplicates(lst):
    ```

  - Write the program that use 10 random numbers in the range of 0-9, invokes the function, and displays the results.
  - The sample run of the program:

```
Generated list:
[9, 3, 1, 4, 7, 5, 5, 5, 2, 1]
Sorted list:
[1, 1, 2, 3, 4, 5, 5, 5, 7, 9]
The distinct numbers are:
[1, 2, 3, 4, 5, 7, 9]
```

# Problem 3 - solution

- Source code

```
from random import randint,random
def randomList1():
    rl=list()
    for i in range(0,10):
        rl.append(randint(0,10))
    return rl

def randomList2():
    return [randint(0,10) for a in range(0,10)]

def eliminateDuplicates(lst):
    y=[]
    for i in lst:
        if i not in y:
            y.append(i)
    return y
```

# Problem 3 - solution

- Source code

```
def main():
    a = randomList1()
    print("Generated list:")
    print(a)
    a.sort()
    print("Sorted list:")
    print(a)
    print("The distinct numbers are:")
    print(eliminateDuplicates(a))

main()
```

# Mulitdimensional lists

- A value in a two-dimensional list can be accessed through a row and column index.
- A two-dimensional list is a list that consists of rows. Each row is a list that contains the values.
- The rows can be accessed using the index, conveniently called a row index.
- The values in each row can be accessed through another index, called a column index.
- Each value in matrix can be accessed using matrix[i][j], whre i and j are the row and column indexes.

# Problem 4

**Processing Two-Dimensional Lists -** an application that performs the following functions:

- Initialize the matrix2D with following values:

```
matrix2D = [[1,2,1,3,4],
            [3,2,5,4,3],
            [5,4,3,2,1],
            [2,1,1,1,1],
            [7,8,9,8,7]
            ]
```

- Sum numbers on the main diagonal
- Sum numbers on the second diagonal
- Print a two-dimensional list

# Problem 4 - solution

- Source code

```
matrix2D = [[1,2,1,3,4],
            [3,2,5,4,3],
            [5,4,3,2,1],
            [2,1,1,1,1],
            [7,8,9,8,7]
            ]
def sumDiagonal1(m):
    s = 0
    for i in range(5):
        s += m[i][i]
    return s

def sumDiagonal2(m):
    s = 0
    for i in range(5):
            s += m[i][4-i]
    return s
```

# Problem 4 - solution

- Source code

```
def show(m):
    for i in range(5):
        for j in range(5):
            print(format(m[i][j],'2d'),end='')
        print()

def main():
    sd1 = sumDiagonal1(matrix2D)
    print("Sum of numbers on the main diagonal =",sd1)
    sd2 = sumDiagonal2(matrix2D)
    print("Sum of numbers on the second diagonal =",sd2)
    show(matrix2D)

main()
```