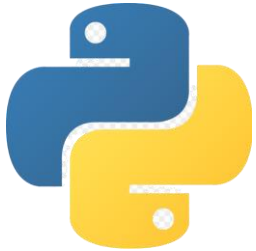


Lecture-5

Python Sequences

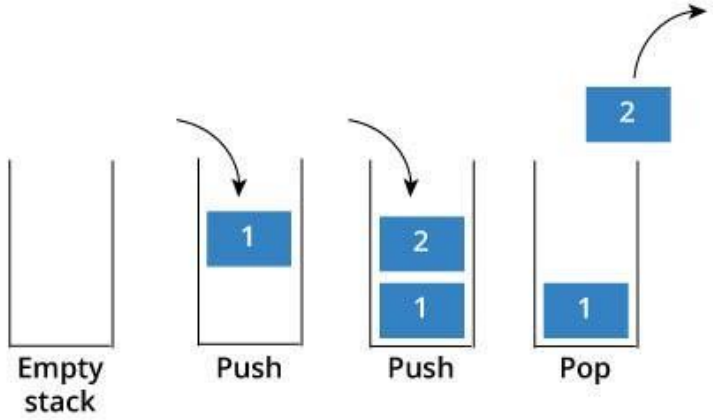
Stack, Expression and 2D list

Content

- Stack with lists
- List comprehension
- Expressions
- Sequence processing function
- Two-dimensional lists

Stack

Stack is a linear data structure. Order LIFO or FILO. The item that is added at first will came out last from the stack.



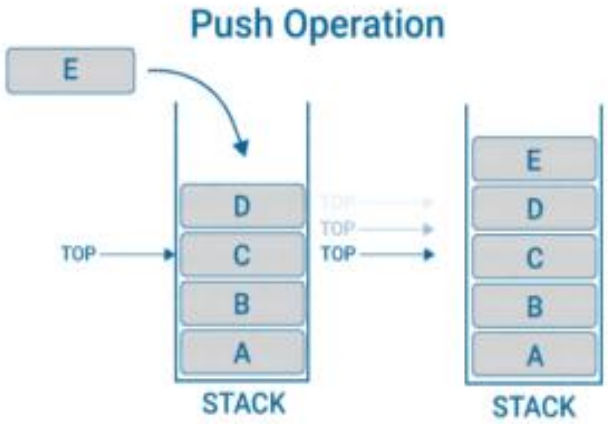
Create Stack Using List

Create a stack using [ ] or list()

```
[1]   1 # define stack
      2 stack = []
```

Push into Stack

- Add a value using **.append()** method
- Push from one by one




```
[2]   1 stack.append('A')
```

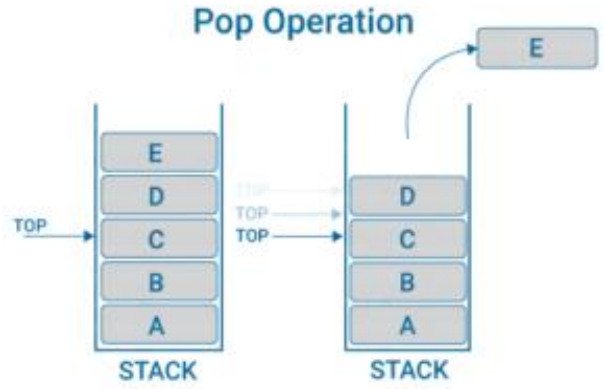
```
[3]   1 stack.append('B')
      2 stack.append('C')
      3 stack.append('D')
      4 stack.append('E')
```

```
[4]   1 stack
```

```
['A', 'B', 'C', 'D', 'E']
```

Pop from Stack

- Remove a value from stack using .pop() method
- Pop works from top element, means pop always remove the top element of the stack
- pop inside takes no parameter




```
[11]  1  stack
```

```
['A', 'B', 'C', 'D', 'E']
```

```
[12]  1  stack.pop()
```

```
'E'
```

List Comprehensions

- An elegant way to define and create lists based on existing lists
- Generally more compact and faster than normal functions and loops for creating list.

```
[6]   1 list2 = [item for item in range(1, 6)]
      2 list2
```

```
[1, 2, 3, 4, 5]
```

Here `item` iterate through over the range from 1 to 5

List Comprehensions(Cont)

```
[7]   1 list4 = [item * 3 for item in range(1, 6)]
      2 list4
```

```
[3, 6, 9, 12, 15]
```

Here `item` iterate through over the range from 1 to 5 and multiply with 3.

```
[8]   1 list5 = [item for item in range(1, 11) if item % 2 == 0]
      2 list5
```

```
[2, 4, 6, 8, 10]
```

Here item iterate through over the range from 1 to 10 and a condition if check whether %2 == 0 or not

Generator Expressions

- A generator is a function that produces a sequence of results instead of a single value
- Examples shows that it print out the numbers that are odd.

```
[9]  1 numbers = [10, 3, 7, 1, 9, 4, 2, 8, 5, 6]
     2 for value in (x ** 2 for x in numbers if x % 2 != 0):
     3     print(value, end=' ')
```

```
9 49 1 81 25
```

Sequence Processing Function

- Finding the Minimum and Maximum Values
- **ord( )** function used for checking the numerical values (ASCII)

```
[10]  1  'Red' < 'orange'
```

```
True
```

```
[11]  1 ord('R')
```

```
82
```

```
[12]  1 ord('o')
```

```
111
```

Reversed a Sequence

- **reversed( )** returns an iterator that enables you to iterate over a sequence's values backward.

```
[36]  1  num2 = [6, 10, 8, 7, 9, 4]
      2  num2
      3  re_num2 = [i for i in reversed(num2)]
      4  re_num2
```

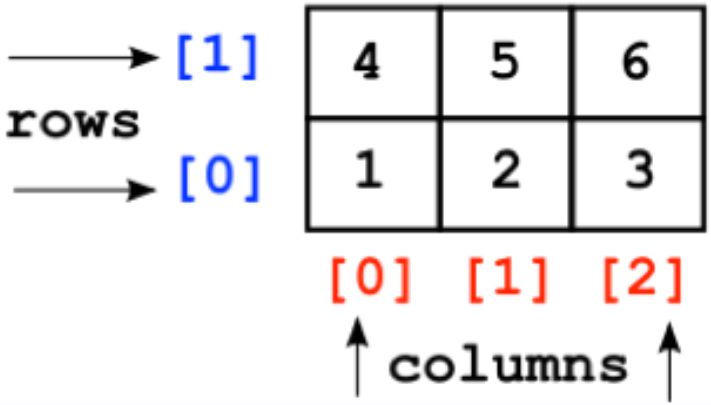
```
[4, 9, 7, 8, 10, 6]
```

```
[14]  1 numbers = [10, 3, 7, 1, 9, 4, 2, 8, 5, 6]
      2 reversed_numbers = [item * 2 for item in reversed(numbers)]
      3 reversed_numbers
```

```
[12, 10, 16, 4, 8, 18, 2, 14, 6, 20]
```

Two-dimensional lists

- 2D lists can contains another list as its' elements
- It's representation in like a table
- Contains two indices i.e- row and column where row specifies as first indices and second indices as column



Create 2D list

- Create a 2D list contains 3 students grades
- Inside a list we add another 3 lists(nested) of students grade
- Representations of this shown in fig

```
[15]  1 a = [[77, 68, 86, 73], # first student's grades
      2        [96, 87, 89, 81], # second student's grades
      3          [70, 90, 86, 81]] # third student's grades
      4 a

      [[77, 68, 86, 73], [96, 87, 89, 81], [70, 90, 86, 81]]
```

	Column 0	Column 1	Column 2	Column 3
Row 0	77	68	86	73
Row 1	96	87	89	81
Row 2	70	90	86	81

Iterate 2D list

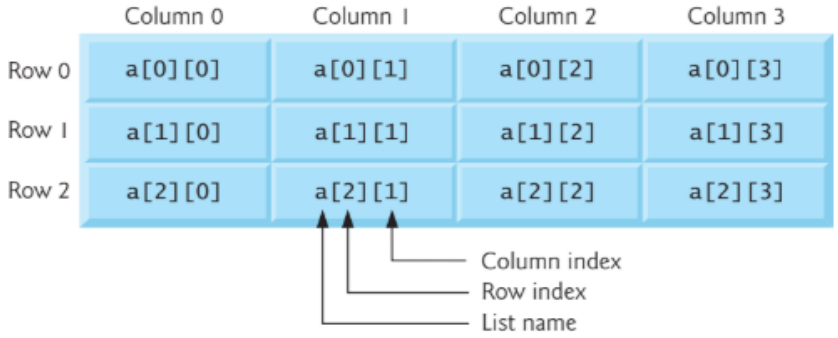
- 2d List contains rows, columns
- Accessing all elements needs 2 for loops as follows
- Nested for print out the rows elements

```
[17]  1 for row in a:
      2   for item in row:
      3     print(item, end=' ')
      4   print()
```

```
77 68 86 73
96 87 89 81
70 90 86 81
```

Identify an Item

- Element can be identified by a name of the form a[i][j]—a is the list's name, and i and j are the indices that uniquely identify each element's row and column, respectively.
- So, a[0][0] is 77




```
[30]  1 for i, row in enumerate(a):
      2   for j, item in enumerate(row):
      3     print(f'a[{i}][{j}]={item} ', end=' ')
      4   print()
```

```
a[0][0]=77  a[0][1]=68  a[0][2]=86  a[0][3]=73
a[1][0]=96  a[1][1]=87  a[1][2]=89  a[1][3]=81
a[2][0]=70  a[2][1]=90  a[2][2]=86  a[2][3]=81
```

Thank You