# Introduction to Python

## Lists

# Topics

1) Lists
2) List indexing
3) Traversing and modifying a list
4) Summing a list
5) Maximum/Minimum of a list
6) List Methods

# Containers

Python includes several built-in sequences: *lists, tuples, strings*. We will discuss these in the next few lectures. Here's a broad overview:

*Lists* and *tuples* are *container sequences*, which can hold items of different type. They hold references to objects they contain(more on this later).

String is a flat sequence which holds item of one type(characters). Flat sequences physically store the value of each item within its own memory space.

Another way of grouping sequence types is by *mutability*. Lists are *mutable(can be modified)* sequences while strings and tuples are *immutable* sequences. We discuss lists in this lecture.

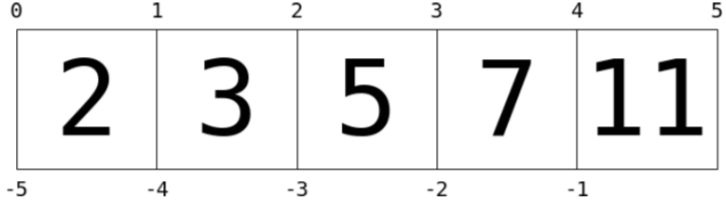
# Lists

Lists are the basic *ordered* and *mutable* data collection type in Python. They can be defined with comma-separated values between square brackets.

```
L = [2, 3, 5, 7]
print(len(L))        # 4, len() also works with strings
L.append(11)         # append to the end of the list
print(L)             # [2, 3, 5, 7, 11]
```

# Indexing

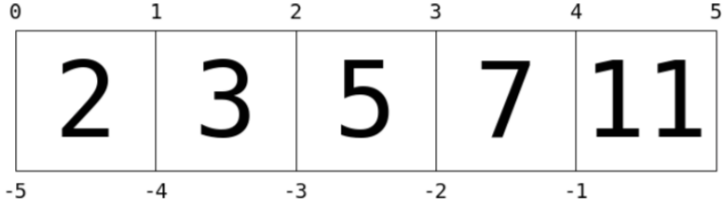
*Indexing* is a means the fetching of a single value from the list. This is a 0-based indexing scheme.




```
L = [2, 3, 5, 7, 11]
print(L[0])           # 2
print(L[1])           # 3
print(L[5])           # index out of bounds error.
```

# Indexing

Negative index wraps around the end.




```
L = [2, 3, 5, 7, 11]
print(L[-1])          # 11
print(L[-2])          # 7
```

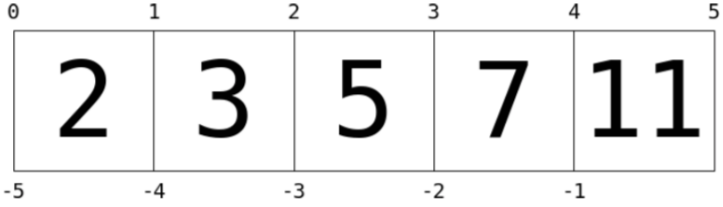
# Lists can contain different types of objects

List can contain different types and even other lists.

```
L = [1, 'two', 3.14, [-2, 3, 5]]
print(L[0])          # 1
print(L[1])          # two
print(L[3])          # [-2, 3, 5]
print(L[3][0])       # -2
print(L[3][1])       # 3
print(L[3][2])       # 5
```

# Modifying a List

Indexing can be used to set elements as well as access them.




```
L = [2, 3, 5, 7, 11]
L[0] = 100
print(L)        # [100, 3, 5, 7, 11]
L[2] = -4
print(L)        # [100, 3, -4, 7, 11]
```

# Traversing a list

We can traverse through a list using a for loop. There are two options:

1) for each loop:

```
nums = [2, -1, 3, 4, -3]
for x in nums:
    print(x, end=" ")
```

2) loop using indices

```
nums = [2, -1, 3, 4, -3]
for i in range(len(nums)):
    print(nums[i], end=" ")
```

# Modifying a list

Consider the following code that is intended to change all even numbers in a list to 0.

```
nums = [24, 3, 34, 6, -5, 4]
for x in nums:
    if x % 2 == 0:
        x = 0
print(nums)
```

```
Output:
[24, 3, 34, 6, -5, 4]
```

Note: The list is unchanged? Why? How can we fix it?

# Modifying a list

Here's the correct code to change all even numbers in a list to 0. Compare the following code to the previous slide.

```
nums = [24, 3, 34, 6, -5, 4]
for i in range(len(nums)):
     if nums[i] % 2 == 0:
          nums[i] = 0
print(nums)
```

```
Output:
[0, 3, 0, 0, -5, 0]
```

# Algorithms to know

The following algorithms are useful. Know how to implement these algorithms!

1) Find sum of a list of numbers.
2) Find the average of a list of numbers.
3) Find the maximum/minimum of a list of numbers.

# Sum of a list

Given a list, find the sum of its elements. We can do this by traversing through the list using a for loop.

```
nums = [2, -1, 3, 4, -3]
s = 0
for x in nums:
    s += x
print(s)
```

Whenever we have a piece of code that accomplish a useful task, we should put it in a function.

# Sum Function

Write a function that accepts a list of numbers as a parameter and returns its sum.

```
def sum(nums):
    s = 0
    for x in nums:
        s += x
    return s
lst = [2, -1, 3, 4, -3]
print(sum(lst))     # 5
lst2 = [1, 5, 4, 2]
a = sum(lst2)
print(a)       # 12
```

# Average Function

Write a function that accepts a list of numbers as a parameter and returns its average.

```
def average(nums):
    s = 0
    for x in nums:
        s += x
    return s/len(nums)


lst = [2, 5, 4, 3]
a = average(lst)
print(a)        # 3.5
```

# Conditional Summing

Write a function that accepts a list of numbers as a parameter and returns the sum of all even numbers in the list.

```
def sum(nums):
    s = 0
    for x in nums:
        if x % 2 == 0:
            s += x
    return s
```

# Find Maximum Function

Write a function that accepts a nonempty list of numbers as a parameter and returns its maximum value. Does the code below work?

```
def maximum(nums):
    current_max = 0
    for x in nums:
        if x > current_max:
            current_max = x
    return current_max

lst = [2, 5, 12, 3, 4, 11]
a = maximum(lst)
print(a)        # 12
```

No! What if the list contains only negative numbers? This function returns 0 which is not even in the list?

# Find Maximum Function

Here's the correct implementation of maximum. The minimum function is similar.

```
def maximum(nums):
    current_max = nums[0]
    for x in nums:
        if x > current_max:
            current_max = x
    return current_max

lst = [2, 5, 12, 3, 4, 11]
a = maximum(lst)
print(a)        # 12
```

# List Methods

The following is a short list of useful list methods.

| | |
|---|---|
| append(value) | appends value to the end of the list |
| insert(index, value) | inserts value at position given by index, shifts elements to the right. |
| pop(index) | removes object at index from list, shifts elements left and returns removed object. Returns last element if index is omitted. The index parameter is optional(default = -1) |

# List Methods

```
L = [3, "hi", -4, 6]
L.append(2)                 # L = [3,"hi",-4, 6, 2]
L.insert(1, "hello")        # L = [3,"hello","hi",-4, 6, 2]
a = L.pop(3)                # L = [3,"hello","hi",6, 2]
print(a)                    # -4
L.pop()
print(L)                    # [3,"hello","hi",6]
```

# Lab 1

Create a new repl on repl.it

1) Create this list and assign it to a variable [3,41,62,87,101, 88]. Use a for loop to compute the sum. Print out the sum.
2) Use a for loop to compute the sum of odd numbers from the list above.
3) Use a for loop to compute the sum of values located at even indices.(Use the len() function).

# Lab 2

Create a new repl on repl.it Use list comprehensions to create the following lists.

1) `[2,4,6,8,10,…,20]`
2) `[1,8,27,64,125]`
3) `[0.5,1.0,1.5,2.0,2.5,3.0]`
4) `['1','2','3','4','5']`
5) `[1,3,5,7,9,…,99]. Must use condition in the list comprehension.`
6) `The nested 2D list: [1,2,3,4,5] repeated 10 times.`
7) `The nested 2D list: [0,0,0,0,0] repeated 20 times.`

# References


1) Vanderplas, Jake, A Whirlwind Tour of Python, O’reilly Media.
2) Luciano, Ramalho, Fluent Python, O'reilly Media.