# 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

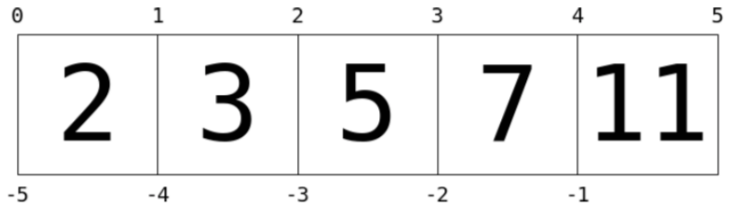
## Lists

A list defines a sequence of ordered objects(integers, floats, strings, etc…).
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.



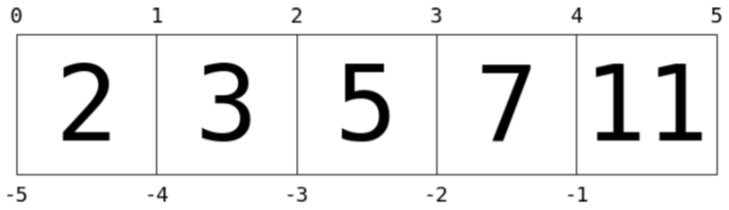
```
L = [2, 3, 5, 7, 11]
print(L[0])            # 2

print(L[1])            # 3

print(L[5])            # index out of bounds error.
```

## 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]
```

## String Indexing

Indexing also works with strings. We can retrieve a character of a string by specifying the *index* of that character, which is the integer that uniquely identifies that character's position in the string.

The built-in len() function returns the number of characters in a string.

```
message = "hello"
length = len(message)
print(length)                # 5
print(message[0])            # h
print(message[1])            # e
print(message[4])            # o
print(message[5])            # error! out of range!
```

Note: On the AP exam, The first index of the first character of a string is 1 not 0.

## String Indexing

Strings are **immutable**: once it is created, it cannot be changed!

```
message = "hello"
message[0] = "H"             # ERROR! A string is immutable!
```

Negative indices can be used to access characters of a string. The last character is at index -1, the second to last at index -2, etc…

```
print(message[-1])           # o
print(message[-2])           # l
```

## Traversing a list

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

1) for each loop(loop through values of list):

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

Looping through each value

2) loop using indices of list

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

Looping through each index
i takes on values: 0,1,2,3,4.

## Looping Through Each Character by Value

As we saw previously, we loop through each character of a string:

```
message = "hello"
for letter in message:
      print(letter)
```

Output:

```
h
e
l
l
o
```

## Looping Through Each Character by Index

Since each character of a string has index, we can also loop through the indices of the string and access each letter by its index.

```
message = "hello"
for i in range(len(message)):
      print(message[i])
```

Output:

```
h
e
l
l
o
```

## 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]
```

Thus, if we need to modify a list, we MUST loop through the indices!

## Creating a list

If you want to create a list containing the first five perfect squares, then you can complete these steps in three lines of code:

```
squares = []              # create empty list
for i in range(5):
     squares.append(i ** 2) # add each square to list
print(squares)
```

```
Output:
[0, 1, 4, 9, 16]
```

## Counting Letters(Version 1)

Write a function which accepts a string and returns the number of “A”, “a” in the string.

```
def countA(str):
    count = 0
    for i in range(len(str)):
        if str[i] == “a” or str[i] == “A”:
            count = count + 1
    return count
```

```
message = “abbA”
print(countA(message))   # 2
```

## Counting Letters(Version 2)

There is another way to loop through letters of a string. Here's the second way to do the previous problem. Note that this version is much easier since we are looping through each value of the string.

```
def countA(str):
    count = 0
    for letter in str:
        if letter == “a” or letter == “A”:
            count = count + 1
    return count
```

```
message = “abbA”
print(countA(message))   # 2
```

## 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_even(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]
a = maximum(lst)
print(a)            # 0 INCORRECT!
```

Incorrect!

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

## Find Maximum Function(Correct)

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

```
def maximum(nums):
    current_max = nums[0]     # the first value is maximum
    for x in nums:            # until a bigger value shows up
        if x > current_max:
            current_max = x
    return current_max

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

## Functions on Strings

Functions we discussed so far are isolated, independent entities. Sometimes functions are associated with some object and operates on the data of that object. In this context, functions are called **methods**.

Strings is an example of a type of objects which contains methods. These methods can be accessed through the **dot notation** applied to a string variable or literal.

| | |
|---|---|
| find(value) | returns the lowest index of a substring value in a string. If substring is not found, returns -1. |
| upper() and lower() | returns a copy of the string capitalizing(or lower casing) all characters in the string |

## String Methods

```
s = "Hi, Mike!"

index = s.find("Hi")
print(index)               # 0, first letter’s index is 0.
print(s.find(" "))         # 3
print(s.find("Mike"))      # 4

index2 = s.find("mike")    # -1, not found
b = "python"
print(b.upper())           # PYTHON
print("JAVA".lower())      # java
```

## String Methods

Note that upper(), lower() do not modify the original string but rather returns a new copy of the string.

```
s = "HI MIKE"
s.lower()                 # returned value "hi mike" is lost
print(s)                  # HI MIKE (s is unchanged)

s = s.lower()   # store the modified, returned string back in s
print(s)        # hi mike
```

## f-Strings

f-Strings is the new way to format strings in Python. (v 3.6)

Also called "formatted string literals," f-strings are string literals that have an f at the beginning and curly braces containing expressions that will be replaced with their values.

```
name = “Mike”
gpa = 3.2
f_str = f“I am {name} with a {gpa} gpa.”
print(f_str)
print("I am " + name + " with a " + str(gpa) " gpa.”)
Output:
I am Mike with a 3.2 gpa.
I am Mike with a 3.2 gpa.
```

## f-Strings

An f-string is special because it permits us to write Python code *within* a string; any expression within curly brackets, {}, will be executed as Python code, and the resulting value will be converted to a string and inserted into the f-string at that position.

```
grade1 = 1.5
grade2 = 2.5
ave = f“average is {(grade1+grade2)/2}”
print(ave)# average is 2.0
```

```
This is equivalent but it is preferable to use an f-string.
average = “average is " + str((grade1+grade2)/2)
```

## Slicing

Slicing is extracting a portion of a list or string. This is similar to the SUBSTRING function we see on the AP Exam questions. We can slice a list or string by specifying a start-index and stop-index, and the result is a subsequence of the items contained within the slice.

Slicing can be done using the syntax:

```
my_list[start:stop:step]
```

where

start: index of beginning of the slice(included), default is 0

stop:  index of the end of the slice(excluded), default is length of the list

step: increment size at each step, default is 1.

## List Slicing

```
L = [10, -2, 1, 6, 2]
print(L[1:3])          # [-2, 1]
print(L[:3])           # [10, -2, 1]
print(L[1:])           # [-2, 1, 6, 2]
print(L[0:4:2])        # [10, 1]
print(L[:])            # [10, -2, 1, 6, 2]
print(L[::-1])         # [2, 6, 1, -2, 10]
```

## String Slicing

```
language = "python"
print(language[0:4])    # pyth
                        # 0 up to but not including index 4
print(language[:4])     # pyth, default start index at 0
print(language[4:])     # on, default end index is length of string

print(language[:])      # python, 0 to end of string
print(language[0:5:2]) # pto, step size of 2
print(language[::-1])  # negative step size traverses backwards
                         # nohtyp
```

## AP Exam

There will be questions on the AP exam which requires students to manipulate strings. The exam will provide an API(application programming interface) for string manipulation. A sample API below was given in previous exams.

| Procedure Call | Explanation |
|---|---|
| concat(str1, str2) | Returns a single string consisting of str1 followed by str2. For example, concat("key", "board") returns "keyboard". |
| reverse(str) | Returns the reverse of the string str. For example, reverse("abcd") returns "dcba". |
| substring(str, start, length) | Returns a substring of consecutive characters from str, starting with the character at position start and containing length characters. The first character of str is located at position 1. For example, substring("delivery", 3, 4) returns "live". |
| substring(str, start, end) | Returns a substring of consecutive characters of str starting with the character at position start and ending with the character at position end. The first character of str is considered position 1. For example, substring("delivery", 3, 6) returns "live". |
| len(str) | Returns the number of characters in str. For example, len("pizza") returns 5. |

The substring function may take on different arguments on the exam. Read the API carefully as this may vary year to year.

# 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 to last element). |
| split() | splits a string into a list. A separator can be specified. The default separator is any whitespace. Note that this is a string method not a list method. |

## 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()                    # ok to not store popped value
print(L)                   # [3,"hello","hi",6]
```

## split()

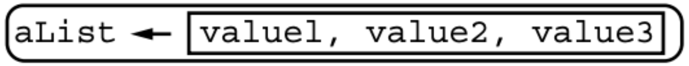
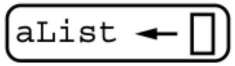
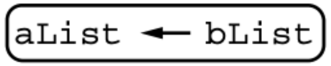
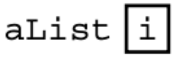
The split() method splits a string into a list. A separator can be specified.The default separator is any whitespace.

```
fruits = "apple mango banana grape"
fruits_lst = fruits.split()
print(list_fruits)   # [‘apple’, ‘mango’, ‘banana’, ‘grape’]

greeting = “hi,I am Mike,I just graduate.”
greet_lst = greeting.split(",")
print(greet_lst)    # [‘hi’, ‘I am Mike’, ‘I just graduate.’]

nums = “4 24 12”
nums_lst = nums.split()
print(nums_lst)    # [‘4’, ‘24’, ‘12’], these are still strings
```

# AP Exam: Lists API

| | |
|---|---|
| Text:<br>`aList ← [value1, value2, value3, ...]`<br>Block:<br><br>`aList ← value1, value2, value3` | Creates a new list that contains the values `value1`, `value2`, `value3`, and `...` at indices `1`, `2`, `3`, and `...` respectively and assigns it to `aList`. |
| Text:<br>`aList ← []`<br>Block:<br><br>`aList ←` | Creates an empty list and assigns it to `aList`. |
| Text:<br>`aList ← bList`<br>Block:<br><br>`aList ← bList` | Assigns a copy of the list `bList` to the list `aList`.<br><br>For example, if `bList` contains `[20, 40, 60]`, then `aList` will also contain `[20, 40, 60]` after the assignment. |
| Text:<br>`aList[i]`<br>Block:<br><br>`aList i` | Accesses the element of `aList` at index `i`. The first element of `aList` is at index `1` and is accessed using the notation `aList[1]`. |

# AP Exam: Lists API

| | |
|---|---|
| Text:<br>`x ← aList[i]`<br>Block:<br>$\boxed{\texttt{x} \longleftarrow \texttt{aList}\ \boxed{\texttt{i}}}$ | Assigns the value of `aList[i]` to the variable `x`. |
| Text:<br>`aList[i] ← x`<br>Block:<br>$\boxed{\texttt{aList}\ \boxed{\texttt{i}} \longleftarrow \texttt{x}}$ | Assigns the value of `x` to `aList[i]`. |
| Text:<br>`aList[i] ← aList[j]`<br>Block:<br>$\boxed{\texttt{aList}\ \boxed{\texttt{i}} \longleftarrow \texttt{aList}\ \boxed{\texttt{j}}}$ | Assigns the value of `aList[j]` to `aList[i]`. |

# AP Exam: Lists API

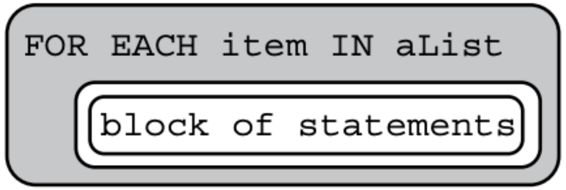
| | |
|---|---|
| Text:<br>**`INSERT(aList, i, value)`**<br>Block:<br>`INSERT` `aList, i, value` | Any values in **`aList`** at indices greater than or equal to **`i`** are shifted one position to the right. The length of the list is increased by 1, and **`value`** is placed at index **`i`** in **`aList`**. |
| Text:<br>**`APPEND(aList, value)`**<br>Block:<br>`APPEND` `aList, value` | The length of **`aList`** is increased by 1, and **`value`** is placed at the end of **`aList`**. |
| Text:<br>**`REMOVE(aList, i)`**<br>Block:<br>`REMOVE` `aList, i` | Removes the item at index **`i`** in **`aList`** and shifts to the left any values at indices greater than **`i`**. The length of **`aList`** is decreased by 1. |
| Text:<br>**`LENGTH(aList)`**<br>Block:<br>`LENGTH` `aList` | Evaluates to the number of elements in **`aList`**. |

# AP Exam: Lists API

Text:

```
FOR EACH item IN aList
{
 <block of statements>
}
```

Block:



```
FOR EACH item IN aList
    block of statements
```

The variable `item` is assigned the value of each element of `aList` sequentially, in order, from the first element to the last element. The code in `block of statements` is executed once for each assignment of `item`.

## 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).

## References

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