# CLASS 5

# DATA STRUCTURES

# Data Structures in Python

aCubeIT
Institute of Artificial Intelligence

- Data structures are a special way of storing and accessing data.
- Every programming language has some built-in data structures
- Python data structures are
  - Tuples
  - Lists
  - Dictionaries
- These are compound data types

# TUPLES

# Tuples:

- An ordered sequence of elements
- Can mix element types
  - Ex: (2, “mit”, 3)
- Cannot change element values, **immutable**
- Represented with parentheses “ **()** “
- Indexing starts from 0

# Tuples: Examples

aCubeIT
Institute of Artificial Intelligence

```
# 1.
first_tuple = ("Monty Python", 30, "Baker Street", 5.8)
print(first_tuple[0])
```

```
Monty Python
```

```
# 2.
city_tuple = ("Mumbai", 18.9949521,   72.8141853)
print(city_tuple)
```

```
('Mumbai', 18.9949521, 72.8141853)
```

```
# 3.
new_tuple = city_tuple
print(new_tuple)
```

```
('Mumbai', 18.9949521, 72.8141853)
```

```
# Try
new_tuple[1] = 13.8877
```

# Tuples: Slicing Examples



- Let us take a Tuple of

  s = (‘A’,’B’,’C’,’D’,’AA’,’BB’,’CC’,’DD’)

  Output the slices

1. (‘B’,’C’) **s[1:3]**
2. (‘D’,’BB’,’DD’) **s[3::2]**

# Tuples: Swapping

aCubeIT
Institute of Artificial Intelligence

***Regular Swapping:***

- If we have **x,y,** how to swap and get

  x = y and y = x?

*Hint: Swapping using **Temp** variable*

```
# Swapping using "Temp" variable
temp = x
x = y
y = temp
print(x)
print(y)
```

```
6
5
```

- **How to swap x and y without using Temp?**

```
# Swapping without using "Temp" variable
x = 5
y = 6
x = x+y
y = x-y
x = x-y
print(x)
print(y)
```

# Tuples: Swapping using Tuples

aCubeIT
Institute of Artificial Intelligence

- Conveniently used to swap variable values

**(x, y) = (y, x)**

Used to return more than one value from a function

- Example: **x = 5, y = 6, Swap using Tuples**

```
x = 5
y = 6
(x, y) = (y, x)
print(x)
print(y)
```

```
6
5
```

# Nesting of Tuples /Tuple of Tuples



- Example:

```
tuple1 = (0, 1, 2, 3)
tuple2 = ('python', 'geek')
tuple3 = (tuple1, tuple2)
print(tuple3)
```

```
((0, 1, 2, 3), ('python', 'geek'))
```

# Tuples: Exercises

- Add the element ‘Python’ to a tuple

  **input_tuple = ('Monty Python', 'British', 1969).**

  **add_tuple = ('Python','named')**

```
input_tuple = ('Monty Python', 'British', '1969')
add_tuple = ('Python','named')
Output_tuple = (input_tuple + add_tuple)
print(Output_tuple)
```

```
('Monty Python', 'British', '1969', 'Python', 'named')
```

# Tuples: Exercises

aCubeIT
Institute of Artificial Intelligence

- Let us take Tuple_1 = (1,2,3,4) Tuple_2 = (‘A’,’B’,’C’,’D’)

  Multiply and verify:

  Tuple_1 * 2

  Tuple_2 * 3

```
Tuple_1 = (1,2,3,4)
Tuple_2 = ('A','B','C','D')
print(Tuple_1 * 2)
print(Tuple_2 * 3)
```

```
(1, 2, 3, 4, 1, 2, 3, 4)
('A', 'B', 'C', 'D', 'A', 'B', 'C', 'D', 'A', 'B', 'C', 'D')
```

# Tuples: Exercises

- Write a function to return quotient and remainder of x divided by y.

```
def quotient_and_remainder(x, y):
    q = x // y
    r = x % y
    return (q, r)
```

```
(quot, rem) = quotient_and_remainder(4, 5)
(quot, rem)
```

```
(0, 4)
```

# Tuples: Exercises HW

aCubeIT
Institute of Artificial Intelligence

1. Write a function to separate out a tuple of numbers & string tuples into two tuples. One with numbers and the other with words.

For example:

((1, “Sunday”), (2, “Monday”), …, (7, “Saturday”)

- numbers should be (1, 2, …., 7)
- words should be (“Sunday”, “Monday”, …, “Saturday”)

For the tuples:

((1, “Low”), (2, “Low”), (3, “Average”), (4, “High”), (5, “High”))

- words should be (“Low”, “Average”, “High”)

2. Write a Python program to unpack a tuple in several variables. Now add all the unpacked variables, repack it as a tuple including the sum value.

# LISTS

# Lists:

- Ordered sequence of information, accessible by index
  - A list is denoted by square brackets, [ ]
  - A list contains elements usually homogeneous (i.e., all integers)
  - Can contain mixed types (not common)

**Ex: [1,"mit",2]**

- List elements can be changed so a list is mutable

# Lists:

- Example:

List_1 = [1,2,3,4]

List_2 = [‘a’,’b’,’c’,’d’]

List_3 = [[1,2],[3,6]]

**List_4 = [1,’q’,[‘a’,’cc’],[13,14]]**

- A list can contain
  - Mixed types
  - List of lists

# Lists: Example

- Take L = [2, 'a', 4, [1,2]]. Print the following
  - len(L)
  - L[0]
  - L[2]+1
  - L[3]
  - L[4]

```
print(L[0])       Output: 2
print(L[2]+1)       Output: 5
print(L[3])       Output: [1,2]
print(L[4])       list index out of range
```

# Lists: Iterating over list



- Similar to strings, We can iterate over list elements directly

```
List_1 = [1, 3, 5, 7, 9]
total = 0
for i in range(len(List_1)):
   total = total +  List_1[i]
print(total)
```

Any other language, Syntax looks like this

```
List_1 = [1, 3, 5, 7, 9]
 total = 0
 # Using for loop
for i in list:
    total += L[i]
```

In Python, **we can iterate over list elements** . We observed this in strings also

- List elements are indexed 0 to len(list_1)-1
- range also goes from 0 to n-1

| |
|---|
| **append()** - Add an element to the end of the list |
| **extend()** - Add all elements of a list to the another list |
| **insert()** - Insert an item at the defined index |
| **remove()** - Removes an item from the list |
| **pop()** - Removes an element at the end of the list |
| **clear()** - Removes all items from the list |
| **index()** - Returns the index of the first matched item |
| **count()** - Returns the count of number of items passed as an argument |
| **sort()** - Sort items in a list in ascending order |
| **reverse()** - Reverse the order of items in the list |
| **copy()** - Returns a shallow copy of the list |

# Lists: Methods Examples

Take a list L = [1,2,3,4]

- Use method **append** to add element 5 to the list at the end
  - L.append(5)
- Lists are Python objects, **everything in Python is an object**
- **Objects have data**
- **Objects have methods and functions**
- Access this information by **object_name.do_something()**

# Lists: Methods Examples

- Use method remove() to delete item from the list
  - L.remove(4)
- Use method pop() to delete the last item of the list
  - L.pop()
- Use method reverse() to reverse the list
  - L.reverse()

# Lists: Methods Examples

aCubeIT
Institute of Artificial Intelligence

- **List: Add operation**
- To combine lists together use concatena>on, + operator, to give you a new list
- Mutate list with L.extend(some_list)
  - L1 = [2,1,3]
  - L2 = [4,5,6]
  - L3 = L1 + L2 then L3 is [2,1,3,4,5,6], L1, L2 unchanged
  - L1.extend([0,6]) à mutated L1 to [2,1,3,0,6]

# Lists – Convert Lists To Strings and back

aCubeIT
Institute of Artificial Intelligence

- **Convert string to list** with **list(s)**, returns a list with every character from s an element in L
- Can use **s.split()**, to split a string on a character parameter, splits on spaces if called without a parameter
- Use **''.join(L)** to turn a list of characters into a string, can give a character in quotes to add char between every element

```
# 1 split()
s = 'I<3 cs'
list(s)
print(s.split('<'),'\n')

# 2 join()
L = ['a','b','c']
print(''.join(L))
print('_'.join(L))
```

```
['I', '3 cs']

abc
a_b_c
```

# List: Examples

- Write a Python program to convert a list of characters into a string.
  - List: [‘P’,’Y’,’T’,’H’,’O’,’N’]

```
s = ['P','Y','T','H','O','N']
str1 = ''.join(s)
print(str1)
```

```
PYTHON
```

# List: Examples

aCubeIT
Institute of Artificial Intelligence

- Convert a string input_str = 'I love Data Science & Python' to a list by splitting it on '&'. The sample output for this string will be:

  **['I love Data Science ', ' Python']**

```
input_str = 'I love Data Science & Python'
output_list = input_str.split('&')
print(output_list)
```

```
['I love Data Science ', ' Python']
```

# List: Examples

aCubeIT
Institute of Artificial Intelligence

- Write a Python program to remove duplicates from a list. List is **a = [10,20,30,20,10,50,60,40,80,50,40]**

```
a = [10,20,30,20,10,50,60,40,80,50,40]
dup_items = []
uniq_items = []
for x in a:
    if x not in dup_items:
        uniq_items.append(x)
        dup_items.append(x)
print(uniq_items)
```

```
[10, 20, 30, 50, 60, 40, 80]
```

# List and Tuple: HW Exercises

aCubeIT
Institute of Artificial Intelligence

1. Work on the list examples in the previous slides replacing **Lists with Tuples**.
2. Write a Python program to find the index of an item of a tuple, list.
3. Write a Python program to count the number of even and odd numbers from a list/tuple of numbers.
4. Write a Python program that prints each item and its corresponding type from the following list.
   *Sample List* : datalist = [1452, 11.23, 1+2j, True, 'w3resource', (0, -1), [5, 12]]
5. Write a Python function to find the Max of numbers in a list.
6. Write a Python function to sum all the numbers in a list.
7. Write a Python function to multiply all the numbers in a list.
8. Write a Python program to print the even numbers from a given list.