Data Types and Operations

- **Sequence** types
- **Mapping** types
- Other types

Sequence Types

- List

  [1, 2, 3]

- Tuple

  (1, 2, 3)

- Range

trainer.cpp@gmail.com

List

- **Mutable** type with elements separated by a comma.
- Ex:

```
# List creation
a = []          # creates empty list
a = list()      # creates empty list
a = [1,2,3]     # list with values
# mutability
a.append(5)  # appending to the end of list
a[1] = 4        # changing value of existing item
a.insert(2,33)
a.sort()        # sorts the list in place, all items must be of same type
```


Tuple

- **Immutable** sequences. Represented by a **()**
- Supports slicing and indexing operations, but changing values not allowed
- Ex:

```
x = ()          # empty tuple
x = tuple()     # empty tuple
x = (1,2,3)
x = 1,2,3
x = 1,          # singleton tuple
x = tuple([1,2,3])
# mutation not allowed
x[1] = 3  # error immutable type
```

Range

- Represents **immutable sequence** of numbers.
- **range()** method returns a **range object** in python 3
  range(start [,end [, step size]])
- Employed in range based for loops
- Ex:

```
range(10)          # returns object with values 0 till 9
range(5,10)        # 5 till 9
range(20,100, 5)   # 20 till 95 with step size of 5
```

trainer.cpp@gmail.com

Mapping : dict

- Mutable mapping type. Represented using {}

 # Creation

```
d = {}                    # empty dictionary
d = dict()                # empty dictionary
d = dict(one=1, two=2, three=3)
d = {'one': 1, 'two': 2, 'three': 3}
d = dict([('two', 2), ('one', 1), ('three', 3)])  # list of tuples
```

 # Operations

d[<Key>] to access a value. Exception if key not found.
d[<Key>] = <Value> creates or overwrites **Value** for a **Key**

trainer.cpp@gmail.com

Dict : Operations

```
del d[key]              # delete the entry for Key
pop(key [, default] ) # deletes and returns value, exception if key not
                        found and Default not provided
key in <d>              # checks for membership of key in dictionary d
key not in <d>
```

```
# Accessing elements
    items()     # returns list of tuples of form (key, value)
    keys()      # returns list of keys
    values()    # returns list of values
```

trainer.cpp@gmail.com

Methods on sequence types

- **len()** : returns the number of elements
- All sequence types support **slicing**.
- Membership check
  in , **not in** # returns Boolean **True** or **False**
- Finding minimum and maximum values:
  min, **max**
- Concatenation and Replication
  +, *****

trainer.cpp@gmail.com

Copying Lists

- l1 = [1,2,3]
  l2 = l1 # both now reference to the same list
  l2.append(4)
  print(l1, l2)
- l3 = list(l1) # create a new list with same values in l3 as l1
  l4 = l1[:] # same as above

Iterating Sequence types

- Use **for** loop:

  for <*variable*> in <*sequence type*>:

  # operations using **<variable>**

- Printing a List:

  l = [1,2,3,4,5]

  for *item* in *l*:

  print(*item*)

Exercises

- **Lists**:
  Print a list in reverse order
  From a list of integers, print only Even elements
  Print Elements at Odd indexes from a list
- **Tuples**:
  Convert a List to a Tuple
  Search for an element in a Tuple
  Search and print the index of element in tuple

trainer.cpp@gmail.com

Questions

- Dictionary
  _ Create a mapping of number to word from 0-9. **(0:'zero'……)**
  _ Ask user for a single digit number and print the corresponding word format
  _ Print all keys of a dictionary
  _ Print all Values of a dictionary
  _ Print all Key and Values of a dictionary

trainer.cpp@gmail.com

Questions

- Print the output of (mention if the syntax is incorrect):

```
x = 1,2,3; print(type(x))
x = (1) ; print(type(x))
x = 1 ; print(type(x))
x = 1, ; print(type(x))
```

trainer.cpp@gmail.com

Questions

- WAP to input a string from user and count occurrence of each alphabet in the string (Hint: use dictionaries). Upper and lower case alphabets are the same

ex: sunny DaY

s:1 u:1 n:2 y:2 d:1 a:1

trainer.cpp@gmail.com