IITM Programming in Python

Week 2

- shorthand operator:

```
count = count + 1 # instead of writing this
count += 1 # write this

count = count * 4
count *= 4
```

- "in" operator
  - checks if something exists "in" something

```
<IN> print("Akul" in "Hi, my name is Umang")
<OUT> False

<IN> print("laptop" in "I am taking notes on my laptop")
<OUT> True
```

- escape character: \

```
#this is wrong
print('It's a beautiful day')
```

```
print('It\'s a beautiful day') # \ ignores the next value

print("My name is:\t\tAkul") # "\t\t" will put a tab between "is:" and "Akul"

print("this is line1 and\nthis is line2") # "\n" will put a new line
```

- To store a multi-line string or a multi-line comment:

```
string1 = ''' line1 of string
line2 of string
'''
```

- String Methods

Method	Description	Code (x = 'pyThoN mETHods')	Output
lower()	entire string into lower	print(x.lower())	python methods
upper()	entire string into upper	print(x.upper())	PYTHON METHODS
capitalize()	first character into upper	print(x.capitalize())	Python methods
title()	first character of each word into upper	print(x.title())	Python Methods

Method	Description	Code (x = 'pyThoN mETHods')	Output
swapcase()	lower case becomes upper and vice versa	print(x.swapcase())	PYthOn MethODS
islower()	checks if the entire string is in lower case		
isupper()	checks if the string in upper case		
istitle()	checks if the string is in Title form		
isdigit()	checks if all the characters are digits		
isalpha()	checks if all the characters are alphabets		
isalnum()	checks if all the characters are alpha-numeric		
strip()	returns a trimmed version of string	x = "----pyth-on----" print(x.strip("-"))	pyth-on
lstrip()	left trim version	print(x.lstrip("-"))	pyth-on----
rstrip()	right trim version		
startswith()	if string starts with the specified value	x="Python" print(x.startswith("p"))	False
endswith()	if string ends with the specified value		
count()	number of times specified value appears		
index()	returns the position of the first occurence of the specified value	print(x.index("T"))	2
replace()	returns a string where specified value is replaced with another specified value	print(x.replace("T","Z"))	pyZHoN mEZHods

- Ceaser cipher on string:

```
alpha= "abcdefghijklmnopqrstuvwxyz"

# to shift all the letters in a word by 'k' in alphabetical order
word = "india"

new_word = ""
t = 0
k = 1 # shifts all the letters by 1
new_word += ( alpha[( ( ( alpha.index(word[t]) +k) %26))] )
new_word += ( alpha[( ( ( alpha.index(word[t+1]) +k) %26))] )
new_word += ( alpha[( ( ( alpha.index(word[t+2]) +k) %26))] )
new_word += ( alpha[( ( ( alpha.index(word[t+3]) +k) %26))] )
new_word += ( alpha[( ( ( alpha.index(word[t+4]) +k) %26))] )
print(new_word)

<OUT> joejb
```

Week 6

- sets take up more space than list.
- searching in sets is faster than list.
- sets are not subscriptable.
  - cannot iterate over the elements.
- if the values inside a tuple are also immutable, the the tuple is considered hashable.
  - if the values inside a tuple are mutable, the the tuple is considered non-hashable.

Week 8

- Sorting a list using recursive function:

```
def mini(mylist): # returns the minimum element
        mini = mylist[0]
        for x in mylist:
                if x < mini:
                        mini = x
        return mini

def sortlist(L):
        if L == [] or len(L) ==1:
                return L

        m = mini(L) # find the minimum element
        L.remove(m) # remove the minimum element
        return [m] + sortlist(L)
```

- Binary Search:

```
def binarysearch(L,k): # return 1 if k exist in L, return 0 otherwise
        begin = 0
        end = len(L) - 1


        while (end-begin) > 0:
                mid = (begin + end) // 2

                if (k == L[mid]) or (k == L[begin]) or (k == L[end]):
                        return 1
                if k > L[mid]:
                        begin = mid + 1
                if k < L[mid]:
                        end = mid - 1

        return 0
```

Week 9

- To open a file:

  ```
  f = open(filename, "r") # to open in Read-only format
  f = open(filename, "w") # to open in Write-only format
  ```

Week 10

- using *super()*. to execute a parent function:

```
class Person(): # Parent class
        def __init__(self, name, age):
                self.name = name
                self.age = age

        def display(self):
                print(self.name, self.age)
```

```python
class Student(Person):
        def __init__(self, name, age, marks):
                super().__init__(name, age)
                self.marks = marks

        def display(self):
                super().display()
                print(self.marks)
```

Week 11

- Exception Handling:

```python
# Dividing by Zero
a = int(input())
b = int(input())
try:
        f = open("abc.txt", "r")
        c = a/b
        print(c)
except ZeroDivisionError: # this will run if the c=a/b throws "ZeroDivisionError"
        print("Invalid input, divisor cannot be zero")
finally: # this block runs regardless of whether we get an exception or not
        f.close()
```

- How to create Exception conditions of your own:

```python
# Throw an error if age is less than 25
age = int(input())
if age < 25:
        raise Exception("You cannot consume alcohol")
```

- Custom example:

```python
# {"name":age}
friends = {"Preeti": 23,
           "Sanyam": 27,
           "Mayank": 34
           }

entry_in_club = []

for p in friends.keys():
    try:
        if friends[p] < 25:
            raise Exception("Underage")

        entry_in_club.append(p)

    except:
        print(f"{p} is not allowed to drink")
```

- Iterator

```python
mylist = ['apple', 'orange', 'pear', 'banana', 'watermelon']
basket = iter(mylist) # basket is now an iterator of the list 'mylist'
print(next(basket)) # will print the FIRST element
print(next(basket)) # will print the SECOND element
print(next(basket)) # will print the THIRD element
```

- Lambda function:

```
def add(x,y):
        return x + y
# add function can be written as a function without having to define the function
add = lambda x,y: x + y
division = lambda x,y: x/y
```

- Enumerate:

```
mylist = ['apple', 'orange', 'pear', 'banana', 'watermelon']

for i,fruit in enumerate(mylist):
        print(f"index of {fruit} in mylist is {i})
```

- Zip:

```
mylist = ['apple', 'orange', 'pear', 'banana', 'watermelon']
size = [5, 6, 4, 6, 10] # length of each element in "mylist"
print(list(zip(mylist, size))) # returns a list of tuples in format: (ith element mylist, ith
element size)
print(dict(zip(mylist, size))) # {"ith element of mylist": "ith element of size"}
```

Map:

```
a = [10, 20, 30, 40, 50, 60]
b = [5, 10, 15, 20, 25, 30]
# we want a list c, where ith element of c = (ith element of a) - (ith element of b)
subtract = lambda x,y: x - y

c = list(map(subtract, a, b)) # c is a mapping function
```