**Ctrl alt l ---format code in pycharm**

**Python is very clean because it use indentation**

**section 1) Python objects and data structure basics**

pip install request

pip install colorama

python -m pip install --upgrade pip # to update python

Pip install pylint

Pip install requests======fetch content from website

Pip install lxml

Pip install bs4 =========beautiful soup

Pip install pillow (library)-image

pip install PyPDF2

Machine generated alternative text:
Complete Python 3 Bootcamp 
Name 
Integers 
Floating point 
Strings 
Lists 
Dictionaries 
Tuples 
Sets 
Booleans 
Type 
int 
float 
str 
list 
dict 
tup 
set 
bool 
Description 
Whole numbers, such as: 3 300 200 
Numberswith,a decimal point: 2.3 4.6 100.0 
Ordered sequence -Shprac rs: "hello" 'Sammy' "2000" 
Ord d vetiuenc of objects: 
Unordered Key:Value pairs: ("mykey" : "value" , "name" : "Frankie") 
Ordered immutable sequence of objects: (10,"hello",200.3) 
Unordered collection of unique objects: ("a","b") 
Logical value indicating True or False 

**1)Numbers**

print(2+1) # output 3

print(2-1) # output 1

print(2\*1) # output 2

print(3/2) # output 1.5

print(3%2) # output 1

print(3\*\*2) # output 9 **\*\* means power**

***2)variables assignment***

***Note : python is dynamic typing***

***not restriction in datatype***

a=5

print(a) # o:5

a=10

print(a) # o: 10

a=a+a

print(a) # o : 20

**# to know variable type**

print(type(a)) # int

a=10.1

print(type(a)) # float

**# more readable we can apply logic to variable name**

my\_income=100

tax\_rate=.1

my\_tax = my\_income\*tax\_rate

print(my\_tax) #o : 10.0

* 1. *String*

**# we can use single as well as double code for string**

print('hello')

print("hello")

print(" i'm going to school")

**# for new line use \n and for tab \t**

print("hello \n world") # output hello

# world

**# # to find length of string use len**

print(len("hello")) #output : 5

* 1. ***Indexing and slicing with string***

**# indexing - single character and slicing - substring of character**

**#indexing**

mystring ="hello world"

print(mystring[0]) # output h

**#reverse indexing**

print(mystring[-1]) # output d

mystring ="abcdefghijkl"

**# grab string from starting index to last**

print(mystring[2:]) # output : cdefghijkl

**# grab string upto**

print(mystring[:2]) # output : ab

**# grab middle of string def**

print(mystring[3:6])

**# grab string from beginning to end**

print(mystring[::])

**# jump string from beginning to end that is called step size**

print(mystring[::2]) # output : acegik

**# reverse a string using step size**

print(mystring[::-1]) # lkjihgfedcba

* 1. ***String Properties and Methods***

**# concatenation of string**

x="hello world"

x=x+" wow"

print(x) # output : hello world wow

**# multiplcation of letter**

x='a';

x=x\*10;

print(x) # output : aaaaaaaaaa

**# contatenation of string number**

print("4"+"2") # output : 42

# upper case

x="hello world"

x=x.upper()

print(x) # output : HELLO

**# split of string return a list**

x=x.split('L');

print(x) # output : ['HE', '', 'O WOR', 'D']

***NOTE:***

Strings are not mutable! (meaning you can't use indexing to change individual elements of a string)

* 1. **Print formatting with string**

my\_name="saurabh"

print("hello "+ my\_name)

**# formatting with .format method . it will grab a string and insert in curly braces.**

print("this is a string{}".format(" inserted")) # output : this is a string inserted

**# format insert in order but we can also insert as index position.**

print("tajmahal {} {} {}".format("is","white","colour")) # output : tajmahal is white colour

#requirement : tajmahal colour is white

print("tajmahal {2} {0} {1}".format("is","white","colour")) # output : tajmahal colour is white

**# we can use keywords also treat as variable assignment**

print("tajmahal {i} {w} {c}".format(i="is",w="white",c="colour")) # output : tajmahal is white colour

result = 100/777;

print("the result was {r} ".format(r=result)) # output : the result was 0.1287001287001287

**# float formatting follows " {value:width.precision f}"**

print("the result was {r:5.2f} ".format(r=result)) # output : the result was 0.13

**# f format type 1**

print(f"the result was {result}")

**# %s type 2**

print("the result was %s" %result)

name ="saurabh"

age="32"

print(f"{name} is {age} years old") # output : saurabh is 32 years old

print("{} is {} years old".format(name,age)) # output : saurabh is 32 years old

* 1. ***List in python***

**#list can be multiple object type**

**# list is ordered sequence of element so we can use indexing and slicing work just like string**

**#append ,pop,sort,reverse**

my\_list =[1,2,3]

print(my\_list)

my\_list =["saurabh",100,10.1]

print(my\_list)

print(len(my\_list))

**#indexing**

print(my\_list[0]) # output : saurabh

**#slicing**

print(my\_list[1:]) # output [100, 10.1]

**# can contatenate list also**

my\_list = ["1","2","3"]

another\_list =["4","5"]

my\_list = my\_list+another\_list

print(my\_list) # output : ['1', '2', '3', '4', '5']

**# we can change my\_list[0]**

my\_list[0]="10"

print(my\_list) # output ['10', '2', '3', '4', '5']

**# add element at last of list using append**

my\_list.append("19")

print(my\_list) # output ['10', '2', '3', '4', '5', '19']

**# pop will remove last element of list and also it will return the element**

popped\_item = my\_list.pop()

print(popped\_item) # output 19

print(my\_list) # ['10', '2', '3', '4', '5']

**# note by default pop index is -1 but we can also remove it by index**

print(my\_list.pop(2)) # output 3

print(my\_list) # output ['10', '2', '4', '5']

**# sort will sort list but note it will not return anything**

sort\_number =[1,5,4,3]

sort\_number.sort()

print(sort\_number) # output [1, 3, 4, 5]

**# or**

new\_list =['a','f','c']

new\_list.sort()

my\_sorted\_list=new\_list

print(my\_sorted\_list) # output ['a', 'c', 'f']

**# reverse it will also not return anything**

my\_sorted\_list.reverse()

print(my\_sorted\_list) # output ['f', 'c', 'a']

* 1. **Dictionaries: are unordered mappings for storing object**

Are unordered and cannot be sorted so when we don’t want exact index location we can use it

my\_dict ={'key1':'value1','key2':'value2'}

print(my\_dict) # output : {'key1': 'value1', 'key2': 'value2'}

**# want to get a value then just pass key**

print(my\_dict['key1']) # output : value1

**# note dictionary can hold int, list and even other dictionary**

dict ={'k1':1,'k2':[0,1,2],'k3':{'inside dict':4}}

print(dict['k3']) # output {'inside dict': 4}

print(dict['k3']['inside dict']) # output 4

**# do letter c in upper case**

dict\_new ={'k1':['a','b','c']}

letter=dict\_new['k1'][2].upper()

print(letter) # output C

**# add new value in dictionary**

d={'k1':1,'k2':2}

d['k3']=3

print(d) # output {'k1': 1, 'k2': 2, 'k3': 3}

**#overwrite a value**

d['k1'] ='new value'

print(d) # output {'k1': 'new value', 'k2': 2, 'k3': 3}

**#print keys and values and items**

print(d.keys()) # output dict\_keys(['k1', 'k2', 'k3'])

print(d.values())# output dict\_values(['new value', 2, 3])

print(d.items())# output dict\_items([('k1', 'new value'), ('k2', 2), ('k3', 3)])

NOTE:

**1. Do dictionaries keep an order? How do I print the values of the dictionary in order?**

Dictionaries are mappings and do not retain order! If you do want the capabilities of a dictionary but you would like ordering as well, check out the **ordereddict** object

* 1. **Tuples**

**List use square bracket while tuples use parenthesis**

* 1. Machine generated alternative text:
     Tuples are very similar to lists. However they 
     have one key differe c - immutability. 
     once an i 
     a tuple, it can not be 
     reassigned. 
     Tuples use parenthesis: (1,2,3) 

**# NOTE** :

tuple is important when we are passing object inside tuple and we don’t want to change it

tup =(1,2,3)

print(tup) # output (1, 2, 3)

print(type(tup)) # output <class 'tuple'>

print(len(tup)) # output 3

print(tup[0])# output 1

print(tup[1:])# output (2, 3)

**# two method present in tuple count and index**

**#count ----will count letter and index will return first index of letter**

t =('a','a','b')

print(t.count('a')) # output 2

print(t.index('a')) # output 0

NOTE:

Even though a list is one of the elements, the object constrained with () is a tuple!

Example :-(1,2,[3])

* 1. **Set**

**Declaration of set**

my\_set = set()

my\_set.add(1)

my\_set.add(2)

print(my\_set) # output {1, 2}

my\_list =[1,1,1,1,1,2,2,2,2,3,3,3,3]

my\_set =set(my\_list)

**# pass list inside set**

print(my\_set) # {1, 2, 3}

* 1. **Boolean**

**# note True and False T and F should be in capital letter**

print(type(True)) # output <class 'bool'>

print(type(False)) # output <class 'bool'>

print(1>2) # output False

print(1==1) # output True

* 1. **i/o With basics file in python**

Machine generated alternative text:
Reading, Writing, Appending Modes 
• mode:'r' is read only 
'E' is write only (will overwrite files or create new!) 
. mode: 
• mode:'a' is append only (will add on to files) 
• mode:'r+' is reading and writing 
• mode: 'w*' is writing and reading (Overwrites existing files or creates a 
new file!) 

**# reading a file**

myfile =open('myfile.txt')

print(myfile.read())

**# want to read file again then need to move cursor from 0**

myfile.seek(0)

print(myfile.readlines())

myfile.close()

**# no need to close file**

**# append mode**

with open('myfile.txt',mode='a') as file:

content=file.write(' India')

**# write # w will delete existing file and then write it**

with open('myfileNew.txt',mode='w') as myfileW:

content=myfileW.write(' jai hind')

**# read file**

with open('myfileNew.txt', mode='r') as file:

content = file.read()

print(content)

===============================

**Assesment**

String : ordered sequence of characters

List : ordered sequence of object(mutable)

Tuples: ordered sequence of object(immutable)

Dictionary: key value paring that is unordered

list3 = [1,2,[3,4,'hello']]

list3[2][2]='goodbye'

print(list3)

listsort=[2,1,5,4]

print(sorted(listsort))

**Section 2 ) python comparison operator**

**Comparison operator**

Machine generated alternative text:
Operator 
Description 
If the values of two operands are equal, then the condition 
becomes true. 
If values of two operands are not equal, then condition 
becomes true. 
If the value of left operand is greater than value of 
right operand, then condition becomes 
If the value of left operand is less than val 
Operand, then condition becomes true. 
If the value of left operand is greater than equal to 
value Of right operand, then condition becomes 
If the value Of left operand is less than or equal to the 
value of right operand, then condition becomes true. 
Example 
(a b) is not true. 
(a b) is true 
(a b) is not true. 
b) is true. 
(a b) is not true. 
(a b) is true. 

**Logical operator : and or not**

**# use of parenthesis is optional and check left and right both are true**

print('h'=='h' and 1==1 )

print(('h'=='h') and (1==1)) # output True

**# or one condition need to be true**

print('h'=='h' or 2==1 )

**# not like negation**

print(not (1==1)) # ouput False

**Section 3 ) Python Statements**

**1)Control flow :----If ,elif ,else**

**Note :**

**Indentation(whitespaces) crucial to python**

Machine generated alternative text:
Syntax of an if/else statement 
if some condition: 
# execute some code 
else 
# do something else 

Machine generated alternative text:
Syntax of an if/else statement 
if some condition: 
# execute some code 
elif some other condition: 
# do something different 
else 
# do something else 

**Example:**

if 1 == 1:

print('if')

elif 2 == 2:

print('elif')

else:

print('else')

**# another example**

hungry = True;

if hungry:

print('feed me')

else:

print('i am not hungry')

* 1. For loop

Machine generated alternative text:
Syntax of a for loop 
my_iterable 
for—in my_iterable: 
print(item name) 

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

for num in mylist:

# check for even

if num % 2 == 0:

print(f'even number {num}')

else:

print(f'odd number {num}')

**# find sum of every number in list**

list\_sum = 0

for sum in mylist:

list\_sum = list\_sum + sum # output : 15

# print is here outside of for loop

print(list\_sum)

**# iterate string**

my\_string = 'Hello World'

for letter in my\_string:

print(letter)

**# note if we don't use variable name then we can use \_ in place of variable name**

name = 'saurabh'

for \_ in name:

print('hello')

**# list of tuple**

mytuple = [(1, 2), (3, 4), (5, 6), (7, 8)]

for tupN in mytuple:

print(tupN)

**# tuple in packing means to print individual element of tuple parenthesis is optional**

for (a, b) in mytuple:

print(a) # output : 1 3 5 7

# iterate dictionary

dict = {'k1': 1, 'k2': 2}

**# important :----- by default when we iterate dictionary it print key**

for key in dict:

print(key) # output k1,k2

**# to iterate items**

for num in dict.items():

print(num) #output ('k1', 1)

#('k2', 2)

**# to get key and value seperately**

for key,value in dict.items():

print(key) #output k1,k2

print(value)# output 1,2

**# want only values from dictionary**

for num in dict.values():

print(num)

* 1. While loop

Machine generated alternative text:
Syntax of a while loop 
while some 6Téan condition: 
#do some h 

Machine generated alternative text:
You can combine with an else if you want 
while some boolean condition: 
#do something 
else: 
#do something different 

Machine generated alternative text:
break, continue, pass 
We can use break, continue, and pass statements in our loops to add additional functionality for 
various cases. The three statements are defined by: 
break: 
• st enclosing loop. 
continue: Goes to the top of the closest enclosing loop. 
pass: Does nothing at all. 

x=0

while x<3:

print(f'the current value of x is {x}')

x=x+1

#x+=1

else:

print('x is not less than 3')

#output :

# the current value of x is 0

#the current value of x is 1

#the current value of x is 2

#x is not less than 3

**# use of pass**

mylist =[1,2,3]

for num in mylist:

#comment note : if we dint write any statement it is expecting something in order to avoid that we use pass

pass

print('working')

**#use of continue dont want to print e**

myname='hello'

for letter in myname:

if letter=='e':

continue

print(letter) # output h l l o

**# use of break**

for letter in myname:

if letter=='e':

break

print(letter) # output h

4 ) useful operator

Range , enumerate ,zip, in , min,max

From random import shuffle

From random import randint

Note

in keyword operator is useful for checking

* 1. List comprehension

Don’t use list comprehension it hard to read

**# write string into list**

mystring ='hello'

mylist =[]

for name in mystring:

mylist.append(name)

print(mylist) # output ['h', 'e', 'l', 'l', 'o']

**# another way**

mylist = [ element for element in mystring]

print(mylist) # output ['h', 'e', 'l', 'l', 'o']

mylist=[num for num in range(0,11)]

print(mylist) # output [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

**# calculate fahrenheit**

celcius =[0,10,20,30.5]

fahrenheit = [ ((9/5)\*temp + 32)for temp in celcius]

print(fahrenheit) # output : [32.0, 50.0, 68.0, 86.9]

**# or caluclate fahrenheit using for loop**

fahrenheit =[]

for temp in celcius:

fahrenheit.append(((9/5)\*temp + 32))

print(fahrenheit) # output : [32.0, 50.0, 68.0, 86.9]

**# nested loop**

mylist=[]

for x in [1,2,3]:

for y in [10,20]:

mylist.append(x\*y)

print(mylist) # output [10, 20, 20, 40, 30, 60]

**# list comprehension way for nested loop**

mylist =[ x\*y for y in [10,20] for x in [1,2,3]]

print(mylist) # output [10, 20, 20, 40, 30, 60]

* 1. Assesment

**# Use for, .split(), and if to create a Statement that will print out words that start with 's'**

st = 'Print only the words that start with s in this sentence'

mylist = st.split()

for word in mylist:

if word.startswith('s'):

print(word) # output start s sentence

**# Use range() to print all the even numbers from 0 to 10.**

for x in range(0,11):

if x%2==0:

print(x) # output 2 ,4 ,6 , 8,10

**# or**

evenNumber=list(range(0,11,2))

print(evenNumber) # output [0, 2, 4, 6, 8, 10]

**#Use a List Comprehension to create a list of all numbers between 1 and 50 that are divisible by 3**.

mylist = [x for x in range(1,51) if x%3==0]

print(mylist) # output [3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48]

**# Go through the string below and if the length of a word is even print "even!"**

st = 'Print every word in this sentence that has an even number of letters'

mylist =st.split()

for word in mylist:

if len(word)%2==0:

print(word) # output word in this sentence that an even number of

**#**

**#Write a program that prints the integers from 1 to 100.**

**#But for multiples of three print "Fizz" instead of the number,**

**#and for the multiples of five print "Buzz".**

**#For numbers which are multiples of both three and five print "FizzBuzz".**

for x in range(1,101):

if x%3==0 and x%5==0:

print("FizzBuzz")

elif x%3==0:

print('Fizz')

elif x%5==0:

print('Buzz')

else:

print(x)

**#Use List Comprehension to create a list of the first letters of every word in the string below:**

st = 'Create a list of the first letters of every word in this string'

mylist =[ word[0] for word in st.split()]

print(mylist) # output ['C', 'a', 'l', 'o', 't', 'f', 'l', 'o', 'e', 'w', 'i', 't', 's']

**Section 4: Methods and function**

**Function**

* 1. **Function syntax**

Machine generated alternative text:
def name of _ function(): 
Keyword telling 
Python this is a 
function. 

Machine generated alternative text:
def name of _ function(): 
Snake casing is all lowercase 
with underscores between 
words 

Machine generated alternative text:
def name 
of 
function(): 
Parenthesis at the end. Later 
on we can pass in 
arguments/parameters into 
the function. 

Machine generated alternative text:
def name of _ function(): 
Docstring explains function. 
Optional: Multi-line string to 
describe function. 

Machine generated alternative text:
def name of _ function(): 
Docstring explains function. 
print("Hello") 
name of _ function() 
Resulting Outpl 

def my\_function(name):

'''

inside function

'''

print("hurrah function called" + name)

my\_function("India")

**# return in function add two number note return allow to save the result to a variable while print does not**

def add\_number(num1,num2):

return num1+num2

result =add\_number(1,2)

print(result) # output 3

**# if arguement is not provided we can give default to name**

def write\_name(name='defualt'):

print("name is "+name)

name=write\_name()

def even\_check(number):

return number%2==0

evencheck=even\_check(10)

print(evencheck)

**# check if any number is even in list using function return true**

**# once return execute then function will not execute**

def even\_check\_list(my\_list):

for num in my\_list:

if num%2==0:

return True

else:

pass # it will not do anything

return False # it will at for loop level

check\_even =even\_check\_list([1,2,3])

check\_list= even\_check\_list([1,5,3])

print(check\_even) # output True

print(check\_list) # output False

**# return even number list**

def even\_number(mynum):

even\_list=[]

for num in mynum:

if num % 2==0:

even\_list.append(num)

else:

pass

return even\_list

evencheck =even\_number([1,2,3,4,5,6])

print(evencheck) # output [2, 4, 6]

* 1. **Tuple unpacking with function**

stock\_prices =[('app',100),('goo',200),('msft',300)]

for item,value in stock\_prices:

print(item)

**# find best employee who work more hour**

def employee\_check(work\_hour):

max\_hour=0

employee\_name=''

for employee,hour in work\_hour:

if hour > max\_hour:

max\_hour =hour

employee\_name =employee

else:

pass

return (employee\_name, max\_hour)

work\_hour =employee\_check([('a',100),('b',200),('c',300)])

print(work\_hour) # output ('c', 300)

# or

name , hour = employee\_check([('a',100),('b',200),('c',300)])

print(name) # output c

print(hour) # output hour

* 1. Interaction between python function

# shuffle and player guess game

from random import shuffle

**# shuffle dont return anything but returning list from function**

def shuffle\_list(mylist):

shuffle(mylist)

return mylist

shufflelist=shuffle\_list(['','0',''])

print(shufflelist)

def player\_guess():

guess=''

while guess not in ['0','1','2']:

guess = input ('pack a number 0 ,1 0r 2')

return int(guess)

guess =player\_guess()

print(guess)

**# interatcion of two function**

def check\_guess(shufflelist,guess):

if shufflelist[guess]=='0':

print("correct")

else:

print('wrong')

print(shufflelist)

checkguess =check\_guess(shufflelist,guess)

print(checkguess)

* 1. **Practist**

Machine generated alternative text:
Functions #2: print Hello Name 
Define a function called myfunc that takes in a name, and prints 'Hello Name' 

def myfunc(name='Name'):

print("Hello "+name)

* 1. Args and kwargs in python

**# return 5 % of number a and b**

def myfunc(a,b):

return sum((a,b))\*.05

perc = myfunc(40,60)

print(perc)

**#Note: by default args set value as tuple and we can pass any number of arguement**

**# note args can be any other keyword**

def myvalue(\*args):

print(args) # output (100, 100, 100, 100, 100)

return sum(args)\*0.05

percentage =myvalue(100,100,100,100,100)

print(percentage) # output 25.0

**# note \*\*kwargs act as dictionary**

def checkfruit(\*\*kwargs):

if 'fruit' in kwargs:

print(kwargs) # \*\* default os dicctionary output : {'fruit': 'apple'}

print('fruit is {}'.format(kwargs['fruit']))

else:

print('fruit not found')

fruit =checkfruit(fruit='apple') # output : fruit is apple

**# pass both \*args and \*\*kwargs**

def myfunc(\*args,\*\*kwargs):

print(args) # output (1, 2, 3)

print(kwargs)# output {'fruit': 'apple', 'food': 'pizza'}

print(' i like {} {} '.format(args[0],kwargs['food'])) # output i like 1 pizza

like = myfunc(1,2,3,fruit='apple',food='pizza')

6) Practist of function

Machine generated alternative text:
Define a function called myfunc that takes in an arbitrary number of arguments, and returns a list containing only those arguments 
that are even. 
Remember, don't run the function, simply provide the definition. 

def myfunc(\*args):

listn =[]

for num in args:

if num%2==0:

listn.append(num)

return listn

Machine generated alternative text:
Define a function called myfunc that takes in a string, and returns a matching string where every even letter is uppercase, and every 
odd letter is lowercase. Assume that the incoming string only contains letters, and don't worry about numbers, spaces or punctuation. 
The output string can start with either an uppercase or lowercase letter, so long as letters alternate throughout the string. 

def myfunc(str):

result=""

for index,letter in enumerate(str):

if index%2==0:

result+=letter.upper()

else:

result+=letter.lower()

return result

**# warmup section**

**# 1========Write a function that returns the lesser of two given numbers if both numbers are even,**

**# but returns the greater if one or both numbers are odd**

**# lesser\_of\_two\_evens(2,4) --> 2**

**# lesser\_of\_two\_evens(2,5) --> 5**

def lesser\_of\_two\_evens(a, b):

if a % 2 == 0 and b % 2 == 0:

return min(a,b)

else:

return max(a,b)

checkEven = lesser\_of\_two\_evens(2, 4) # output 2

checkOdd = lesser\_of\_two\_evens(2, 5) # output 5

print(checkEven)

print(checkOdd)

**# 2 ====ANIMAL CRACKERS: Write a function takes a two-word string and**

**# returns True if both words begin with same letter**

**# animal\_crackers('Levelheaded Llama') --> True**

**# animal\_crackers('Crazy Kangaroo') --> False**

def animal\_crackers(letter):

new\_word = letter.split()

return new\_word[0][0] == new\_word[1][0]

check1= animal\_crackers('Levelheaded Llama')

check2 = animal\_crackers('Crazy Kangaroo')

print(check1)

print(check2)

**# 3 MAKES TWENTY: Given two integers, return True if the sum of the integers is 20**

**# or if one of the integers is 20. If not, return False¶**

**# makes\_twenty(20,10) --> True**

**# makes\_twenty(12,8) --> True**

**# makes\_twenty(2,3) --> False**

def makes\_twenty(a,b):

if a+b==20 or a==20 or b==20:

return True

else:

return False

check1 =makes\_twenty(20,10)

check2= makes\_twenty(12,8)

check3 = makes\_twenty(2,3)

print(check1)

print(check2)

print(check3)

**Level one**

**# level one**

**# OLD MACDONALD: Write a function that capitalizes the first and fourth letters of a name**

**# old\_macdonald('macdonald') --> MacDonald**

**# Note: 'macdonald'.capitalize() returns 'Macdonald'**

from re import split

def old\_macdonald(str):

first\_letter = str[0]

inbetween = str[1:3]

fourthletter = str[3]

rest = str[4:]

return first\_letter.upper() + inbetween + fourthletter.upper() + rest

result = old\_macdonald('macdonald')

print(result) # output MacDonald

**# In Python, the capitalize() method converts the first character of a string to capital (uppercase) letter**

**# another way**

def old\_macdonald\_new(str):

first\_half = str[0:3]

second\_half = str[3:]

return first\_half.capitalize() + second\_half.capitalize()

result\_new = old\_macdonald\_new('macdonald')

print(result\_new) # output MacDonald

#=

=====================================================

Note: The .join() method may be useful here. The .join() method allows you to join together strings in a list with some connector string. For example, some uses of the .join() method:

>>> "--".join(['a','b','c'])

>>> 'a--b--c'

This means if you had a list of words you wanted to turn back into a sentence, you could just join them with a single space string:

>>> " ".join(['Hello','world'])

>>> "Hello world"

===================================================================================

Reverse a string

**# joint will join space between every list**

def master\_yoda (str):

return ' '.join(str.split()[::-1])

rev = master\_yoda('I am Home')

print(rev) # output Home am I

**#Given an integer n, return True if n is within 10 of either 100 or 200¶**

#almost\_there(90) --> True

#almost\_there(104) --> True

#almost\_there(150) --> False

#almost\_there(209) --> True

#NOTE: abs(num) returns the absolute value of a number

def almost\_there(n):

return ((abs(100 - n) <= 10) or (abs(200 - n) <= 10))

resultabs = almost\_there(90)

print(resultabs)

**Level 2**

**# Given a list of ints, return True if the array contains a 3 next to a 3 somewhere.**

**# has\_33([1, 3, 3]) → True**

**# has\_33([1, 3, 1, 3]) → False**

**# has\_33([3, 1, 3]) → False**

def has\_33(num):

for i in range(0, len(num) - 1):

# if num[i]==num[i+1]:

# return True

if num[i:i + 2] == [3, 3]:

return True

return False

result = has\_33([1, 2, 3])

print(result)

**# 2)Given a string, return a string where for every character in the original there are three characters**

**# paper\_doll('Hello') --> 'HHHeeellllllooo'**

**# paper\_doll('Mississippi') --> 'MMMiiissssssiiippppppiii**'

def paper\_doll(str):

name = " "

mylist = list(str)

for num in mylist:

name = name + num \* 3

return name

result = paper\_doll('Hello')

print(result)

**# 3**

**# BLACKJACK: Given three integers between 1 and 11, if their sum is less than or equal to 21,**

**# return their sum. If their sum exceeds 21 and there's an eleven, reduce the total sum by 10.' \**

**# ' Finally, if the sum (even after adjustment) exceeds 21, return 'BUST'¶**

**# blackjack(5,6,7) --> 18**

**# blackjack(9,9,9) --> 'BUST'**

**# blackjack(9,9,11) --> 19**

def blackjack(a, b, c):

if sum((a, b, c)) <= 21:

return sum((a, b, c))

elif sum((a, b, c)) - 10 <= 21 and 11 in (a, b, c):

return sum((a, b, c)) - 10

else:

return 'BUST'

check1 = blackjack(5, 6, 7)

check2 = blackjack(9, 9, 9)

check3 = blackjack(9, 9, 11)

print(check1)

print(check2)

print(check3)

**# 4 SUMMER OF '69: Return the sum of the numbers in the array,**

**# except ignore sections of numbers starting with a 6 and extending to the next 9**

**# (every 6 will be followed by at least one 9). Return 0 for no numbers.¶**

**# summer\_69([1, 3, 5]) --> 9**

**# summer\_69([4, 5, 6, 7, 8, 9]) --> 9**

**# summer\_69([2, 1, 6, 9, 11]) --> 14**

def summer\_69(arg):

total = 0

add = True

for num in arg:

while add:

if num != 6:

total += num

break

else:

add = False

while not add:

if num != 9:

break

else:

add = True

return total

result1 = summer\_69([1, 3, 5])

result2 = summer\_69([4, 5, 6, 7, 8, 9])

result3 = summer\_69([2, 1, 6, 9, 11])

print(result1)

print(result2)

print(result3)

Challenge Problem

**# 1SPY GAME: Write a function that takes in a list of integers and returns True if it contains 007 in order**

**# spy\_game([1,2,4,0,0,7,5]) --> True**

**# spy\_game([1,0,2,4,0,5,7]) --> True**

**# spy\_game([1,7,2,0,4,5,0]) --> False**

def spy\_game(num):

code=[0,0,7,'x']

for i in num:

if i==code[0]:

code.pop(0)

return len(code)==1

result1 = spy\_game([1, 2, 4, 0, 0, 7, 5])

result2 = spy\_game([1, 0, 2, 4, 0, 5, 7])

result3 = spy\_game([1, 7, 2, 0, 4, 5, 0])

print(f" result are {result1}{result2}{result3}")

**# 2) COUNT PRIMES: Write a function that returns the number of prime numbers that exist up**

**# to and including a given number**

**# count\_primes(100) --> 25**

def count\_primes(num):

count = 0

for check in range(num):

if check <= 1:

continue

for test in range(2, check):

if check % test == 0:

break

else:

count += 1

return count

result = count\_primes(10)

print(result)

* 1. **Lamda expression , map and filter function**

**# lamda expression : annonymous function use only one time and never referenced again**

**# note dont add parenthesis when we pass function in map and map will apply to all element**

**# 1) map with number**

def square(num):

return num \* num

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

for item in map(square, mylist):

print(item) # output 1,4,9,16,25

**# want to list back**

print(list(map(square, mylist))) # output [1, 4, 9, 16, 25]

**# using lamda with map function**

print(list(map(lambda num:num\*num, mylist))) # output [1, 4, 9, 16, 25]

**# map with string**

def slicer(str):

if len(str) % 2 == 0:

return 'EVEN'

else:

return str[0]

mystring = ['saurabh', 'neha', 'ankit']

print(list(map(slicer, mystring))) # output ['s', 'EVEN', 'a']

**#================================================================================================**

**# 2)filter we apply in function it will return true or false based on condition it will apply to list**

def check\_even(num):

return num % 2 == 0

mylist = [1, 2, 3, 4, 5, 6]

**# we can use list**

print(list(filter(check\_even, mylist))) # output [2, 4, 6]

**# or we can loop it also**

for i in filter(check\_even, mylist):

print(i) # output 2,4,6

**# lamda with filter function**

print(list(filter(lambda num:num%2==0, mylist))) # output [2, 4, 6]

#============================================================

**# 3 lamda expression converting function into lamda expression**

square = lambda num : num\*\*2

print(square(5)) # output 25

**# find first character of name using lambda**

names=['saurabh','ankit']

print(list(map(lambda name:name[0],names))) #output ['s', 'a']

* 1. Nested statement and scope

Machine generated alternative text:
LEGB Rule: 
• L: Local — Names assigned in any way within a function (def or lambda), 
and not declared global'in thÅ€function. 
• E: Enclosing function' I 
Names in the local scope of any and all 
enclosing functions 
—mbda), from inner to outer. 
• G: Global (module) — 
ames—igned at the top-level of a module file, 
or declared global in a def within the file. 
• B: Built-in (Python) — Names preassigned in the built-in names module . 
open, range, SyntaxError,... 

**# global , enclose and local**

name = 'global'

def greet():

name = 'enclose'

def hello():

# name='local'

print(name)

hello()

result = greet()

**# want to change global variable inside function**

x=10

def change\_global():

global x

print(f'x is {x}')

# local reassignment on a global variable

x='new value'

print(f'locally changed global x to {x}')

print(x) # output x is 10

change\_global() # locally changed x to new value

print(x) # new value

**# in place of using global to change value we can do like this**

y=10

def change\_glo(y):

print(f'y is {y}') # output y is 10

# local reassignment on a global variable

y='new value'

print(f'locally changed y to {y}')

return y

print(y) # output 10

y=change\_glo(y) # locally changed y to new value

print(y) # new value

* 1. Method and function homework

**# 1) Write a function that computes the volume of a sphere given its radius.**

def volume(rad):

return (4 / 3 \* 22 / 7 \* rad \* rad \* rad)

volume\_calculate = volume(2)

print(f"volume of number is {volume\_calculate}")

**# 2) Write a function that checks whether a number is in a given range (inclusive of high and low)**

def ran\_check(num, low, high):

for num in range(low, high + 1):

return True

range\_test = ran\_check(5, 2, 7)

print(range\_test) # output True

**# 3)**

**# Write a Python function that accepts a string and calculates the number of upper case letters**

**# and lower case letters.**

**# HINT: Two string methods that might prove useful: .isupper() and .islower()**

def up\_low(s):

upperCase = 0

lowerCase = 0

for char in s:

if char.isupper():

upperCase += 1

elif char.islower():

lowerCase += 1

else:

pass

return (upperCase, lowerCase)

upper, lower = up\_low('Hello Mr. Rogers, how are you this fine Tuesday?')

print(f'No. of Upper case characters {upper}') # output No. of Upper case characters 4

print(f'No. of lower case characters {lower}') # output No. of lower case characters 33

**# 4)**

**# Write a Python function that takes a list and returns a new list with unique elements of the first list.**

def unique\_list(lst):

num = set()

return set(lst)

uniquelist = unique\_list([1, 1, 1, 1, 2, 2, 3, 3, 3, 3, 4, 5])

print(list(uniquelist)) # output [1, 2, 3, 4, 5]

**# or another way**

def unique\_list\_new(lst):

list = []

for a in lst:

if a not in list:

list.append(a)

return list

uniquelist = unique\_list\_new([1, 1, 1, 1, 2, 2, 3, 3, 3, 3, 4, 5])

print(uniquelist) # output [1, 2, 3, 4, 5]

**# 5) Write a Python function to multiply all the numbers in a list.**

def multiply(list):

fact = 1

for num in list:

fact = fact \* num

return fact

result = multiply([1, 2, 3, -4, 5])

print(result) # -120

**# 6) Write a Python function that checks whether a word or phrase is palindrome or not.**

def palindrome(s):

s=s.replace(" ","")

return s== s[::-1]

result = palindrome('ab a')

print(f"word is palindrome {result}") # output word is palindrome True

Section 5: Milestone project 1

* 1. **Display tic toc**

def display(row1,row2,row3):

print(row1)

print(row2)

print(row3)

row1 = [' ',' ',' ']

row2 = [' ',' ',' ']

row3 = [' ',' ',' ']

row2[1]='X'

display(row1,row2,row3)

* 1. Accepting user input

**# enter input // by default it is string to get int we need to typecast**

result =input("please enter a name")

print(f" name is {result}")

print(type(result)) # output <class 'str'>

num=input("enter number")

print(type(num))

int\_typecast= int(num) # output <class 'str'>

print(type(int\_typecast)) #output <class 'int'>

* 1. Validating user input

def choice():

choice = input("please enter a number between 0 and 10 ")

return int(choice)

num = choice()

print(num)

**# isdigit() use to check digit inside string**

def choice\_digit():

**# initial values**

choice='WRONG'

acceptable\_range = range(0,10)

within\_Range=False

**# two condition to check digit or within range**

while choice.isdigit()==False or within\_Range==False:

choice = input("please enter a number between 0 and 10 ")

**#Digit check**

if choice.isdigit()==False:

print("it is not a digit")

**# range check**

if choice.isdigit()==True:

if int(choice) in acceptable\_range: # we got it choice is digit since it already pass digit test

within\_Range=True

else:

within\_Range=False

return int(choice)

num = choice\_digit()

print(num)

**Section 6 object oriented programming**

Machine generated alternative text:
Object Oriented Programming (OOP) 
allows programm r. o create their own 

Machine generated alternative text:
class NameOfClass(): 
def 
init_(self,param1,param2): 
self.paraml paraml 
self.param2 param2 
def 

Machine generated alternative text:
Python Objects and Classes 
Python is an object oriented programming language. Unlike procedure oriented 
programming, where the main emphasis is on functions, object oriented programming 
stresses on objects. 
An object is simply a collection of data (variables) and methods (functions) that act on those 
data. Similarly, a class is a blueprint for that object. 
We can think of class as a sketch (prototype) of a house. It contains all the details about the 
floors, doors, windows etc. Based on these descriptions we build the house. House is the 
object. 
As many houses can be made from a house's blueprint, we can create many objects from a 
class. An object is also called an instance of a class and the process of creating this object is 
called instantiation. 

* 1. Python object and classes

"""

Note : very imp

everything in python is object

use class keyword to define user define object

class is blueprint of object and from classes we instantiate objects

The self is used to represent the instance of the class.

With self keyword, you can access the attributes and methods of the class in python.

It binds the attributes with the given argument

# init------called constructor of class .

it will called automatically when instance of class is created

self:--- it represent instance of object /class itself

# note init---constructor and self ---this in java

"""

class MyDog(): # create a class

def \_\_init\_\_(self,breed,name,spot): # it will called automatically when instance of class is created

self.breed=breed # self keyword basically connect to method to instance of class

self.name=name

# spot boolean

self.spot=spot

mydog =MyDog('Lab','pummy','true') # create instance of class

print(mydog.breed) # output Lab

print(mydog.name) # output pummy

print(mydog.spot) # output True

===================================

* 1. Python method

class MyDog(): # create a class

**# class object attribute**

**# same for any instance of class**

species = 'mammal'

def \_\_init\_\_(self, breed, name):

self.breed = breed

self.name = name

**# method are function inside a class that work on object in some way**

**# operation/action ---> methods**

def bark(self,number): **# not attaching number with self since user providing number**

return (f'WOOF dog name is {self.name} and the number is {number}')

mydog = MyDog('Lab', 'pummy') # create instance of class

print(mydog.breed) # output Lab

print(mydog.name) # output pummy

print(mydog.species) # output mammal

print(mydog.bark(10)) # output --- WOOF dog name is pummy and the number is 10

#============================================

**# another example calculate circumference of circle**

class Circle:

#class object attribute

pi=3.14

def \_\_init\_\_(self,radius=1):

self.radius=radius

self.area = self.pi\*radius\*radius

def get\_circumference(self):

return 2\***Circle.p**i\*self.radius

circum = Circle(30)

print(circum.get\_circumference()) # output 188.4

print(circum.area) # output 2826.0

===========================

* 1. Inheritance and polymorphism

Inheritance

**# base class**

class Animal:

def \_\_init\_\_(self):

print('Animal Created')

def who\_am\_i(self):

print('i am an animal')

def eat(self):

print('i am eating')

myanimal =Animal() # output Animal Created

myanimal.eat() # output i am eating

myanimal.who\_am\_i() # output i am an animal

**# derived class some of feature is same for dog so want to use same method**

class Dog(Animal):

def \_\_init\_\_(self):

Animal.\_\_init\_\_(self)

print("dog created")

def who\_am\_i(self): **# overwrite method**

print('animal is dog')

**# add our method**

def bark(self):

print('woof')

mydog= Dog() # output dog created constructor of dog called

mydog.eat() # output i am eating

mydog.who\_am\_i() # output animal is dog

mydog.bark() # output woof

**Poly example**

**class Dog:**

def \_\_init\_\_(self, name):

self.name = name

def speak(self):

return self.name + ' says Woof!'

**class Cat:**

def \_\_init\_\_(self, name):

self.name = name

def speak(self):

return self.name + ' says Meow!'

niko = Dog('Niko')

felix = Cat('Felix')

print(niko.speak())

print(felix.speak())

def pet\_speak(pet):

print(pet.speak())

pet\_speak(niko) # output Niko says Woof!

pet\_speak(felix) # output Felix says Meow!

====================

Abstract example

"""

**An abstract class is one that never expects to be instantiated.**

**For example, we will never have an Animal object**

**only Dog and Cat objects, although Dogs and Cats are derived from Animals:**

**real time example**

**open method defined in abstract class but we didnt implement it beacuse someone want to open pdf file someone word file**

**and someone word file**

"""

class Animal:

def \_\_init\_\_(self, name): # Constructor of the class

self.name = name

def speak(self): # Abstract method, defined by convention only

raise NotImplementedError("Subclass must implement abstract method")

class Dog(Animal):

def speak(self):

return self.name + ' says Woof!'

class Cat(Animal):

def speak(self):

return self.name + ' says Meow!'

fido = Dog('Fido')

isis = Cat('Isis')

print(fido.speak())

print(isis.speak())

* 1. **Special method**

class Book:

def \_\_init\_\_(self,title,author,pages):

self.title=title

self.author=author

self.pages=pages

**# string representation**

def \_\_str\_\_(self):

return (f"{self.title} by {self.author}")

**# want to use len variable**

def \_\_len\_\_(self):

return self.pages

**# to delete object**

def \_\_del\_\_(self):

print( " a book object has been deleted")

mybook =Book('Python rocks','jose',200)

print(mybook) # output Python rocks by jose

print(len(mybook)) # output 200

del mybook # output a book object has been deleted

**homework**

**"""**

**Problem 1**

**Fill in the Line class methods to accept coordinates as a pair of tuples and return**

**the slope and distance of the line.**

**"""**

class Line:

def \_\_init\_\_(self, coor1, coor2):

self.coor1=coor1

self.coor2=coor2

def distance(self):

x1,y1 =self.coor1

x2,y2=self.coor2

return ((x2-x1)\*\*2 + (y2-y1)\*\*2) \*\*.5

def slope(self):

x1, y1 = self.coor1

x2, y2 = self.coor2

return (y2 - y1) / (x2 - x1)

coordinate1 = (3,2)

coordinate2 = (8,10)

li = Line(coordinate1,coordinate2)

print(f"distance is{li.distance()}") # output 9.433981132056603

print(f"slope is{li.slope()}") # output 1.6

**#=============== second example**

class Cylinder:

def \_\_init\_\_(self, height=1, radius=1):

self.height=height

self.radius=radius

def volume(self):

#pir2h

return 3.14 \*self.radius\*\*2 \*self.height

#2πrh+2πr2

def surface\_area(self):

return 2\*3.14\*self.radius\*(self.radius+self.height)

c = Cylinder(2,3)

print((f"volume of cylinder is {c.volume()}")) # output volume of cylinder is 56.52

print((f"surface area of cylinder is {c.surface\_area()}")) # output surface area of cylinder is 94.2

94.2

**challenge**

class Account:

def \_\_init\_\_(self, owner, balance):

self.owner = owner

self.balance = balance

def deposit(self, deposit):

self.balance += deposit

print('deposit accepted')

def withdrawl(self, withdrawl):

if withdrawl < self.balance:

self.balance -= self.withdrawl

print('withdrawl accepted')

else:

print('insufficient fund')

def \_\_str\_\_(self):

return f" Account owner {self.owner} \n Account balance {self.balance}"

acct1 = Account('jose', 100)

print(acct1)

print(acct1.owner) # output jose

print(acct1.balance) # output 100

acct1.deposit(50) # output deposit accepted

print(acct1.balance) # output 150

acct1.withdrawl(500) # output insufficient fund

Section 7 ) Modules and Packages

**To install third party package**

**Note : to use package first we need to install it**

pip install requests # it will install request packages

pip install colorama

python -m pip install --upgrade pip # to update python

To create own modules and packages

Machine generated alternative text:
Now that we understand how to install 
external packages e 's explore how to 
s and packages. 
create our own 
Modules are just Sc apts that you call in 
another .py script. 
Packages are a collection of modules. 
Let's create some examples! 

Module interaction

**Mymodule.py**

def my\_func():

print("hey i am in mymodule.py")

**MyProgram.py**

from mymodule import my\_func

my\_func()

**Run MyProgram.py**

Output :# hey I am in mymodule.py

**Package**

**In order to tell python this is package we need to create \_\_init\_\_.py file inside that directory**

Defining package and creating module

Machine generated alternative text:
MyMainPackage 
SubPackage 
init_.py 
mysubscript.py 
init_.py 
some_main_scri t. 

**Some\_main\_script.py**

def report\_main():

print("hey i am dome\_main\_script in main package")

**Mysubscript.py**

def subreport():

print('hey i am function inside mysubscript')

**Calling module from package in other py file**

**Test.py**

**# import module from package then we can call function**

from MyMainPackage import some\_main\_script

from MyMainPackage.SubPackage import mysubscript

some\_main\_script.report\_main() # output hey i am dome\_main\_script in main package

mysubscript.subreport() # output hey i am function inside mysubscript

**Built\_in \_variable**

**\_ \_ name \_ \_**

**Note --- important point**

Machine generated alternative text:
#python one. py 
print( 'hello') 
if 
name 
myfunc() 
main 

**If we run one.py file directly**

**Then in backend python will execute:**

**If \_ \_ name \_ \_ == " \_ \_ main \_ \_":**

**One.py**

def func():

print("func() in one.py")

print("top level in one.py")

if \_\_name\_\_ == '\_\_main\_\_':

print('one.py is run directly')

else:

print('one.py is imported')

**Two.py**

import one

one.func()

print ("top level in two.py")

if \_\_name\_\_ == '\_\_main\_\_':

print('two.py is run directly')

else:

print('two.py is imported')

**Section 8 ) Errors and Exception handling**

Machine generated alternative text:
We use three keywords for this: 
o try: This is the block of code to be 
attempted (may lead to an error) 
o except: Block of code will execute in 
case there is an error in try block 
o finally: A final block of code to be 
executed, regardless of an error. 

# try and except demo

try:

**# want to attempt this code**

**# may have an error**

result =10+10

# we got an error so below line will never get executed

except:

print("hey you aren't adding correctly")

else:

print("add went well")

print(result)

**#----------another example try except and finally demo**

try:

f=open("testfile",'r')

f.write('write a test line')

except TypeError:

print('there was a type error')

except OSError:

print('there was an os error') #output there was an os error

except:

print('all other exception')

finally:

print('i always run') # i always run

**#================ another example**

def ask\_for\_int():

while True:

try:

number =int(input('please enter a number '))

except:

print('not a number')

continue

else:

print('thanku it is a number')

break

finally:

print('end of try/except/finally')

print('i will always run at the end')

ask\_for\_int()

**Output of all above 3 program**

add went well

20

there was an os error

i always run

please enter a number test

not a number

end of try/except/finally

i will always run at the end

please enter a number 10

thanku it is a number

end of try/except/finally

i will always run at the end

Homework

**"""**

**Problem 1**

**Handle the exception thrown by the code below by using try and except blocks.**

**"""**

try:

for i in ['a', 'b', 'c']:

print(i \*\* 2)

except TypeError:

print("power not occured")

else:

print('power occured')

**"""**

**Problem 2**

**Handle the exception thrown by the code below by using try and except blocks.**

**Then use a finally block to print 'All Done.'**

**"""**

try:

x = 5

y = 0

z = x / y

except ZeroDivisionError:

print('arithmetic exception')

finally:

print('all done')

**"""**

**Problem 3**

**Write a function that asks for an integer and prints the square of it.**

**Use a while loop with a try, except, else block to account for incorrect inputs.**

**"""**

def ask():

#waiting= True

while True:

#while waiting :

try:

n = int(input('Input an integer: '))

except:

print('An error occurred! Please try again!')

continue

else:

#waiting=False

break

print('Thank you, your number squared is: ', n \*\* 2)

ask()

# output

power not occured

arithmetic exception

all done

Input an integer: e

An error occurred! Please try again!

Input an integer: 2

Thank you, your number squared is: 4

**Pylint overview**

**Unit testing:**

Machine generated alternative text:
There are several testing tools, we will focus 
on two: 
o pylint: This is a library that looks at your 
code and reports back possible issues. 
o unittest: This built-in library will allow to 
test your own programs and check you 
are getting desired outputs. 

Step 1) install pylint library

Pylint is a source-code, bug and quality checker for the Python programming language. It is named following a common convention in Python of a "py" prefix, and a nod to the C programming lint program. It follows the style recommended by PEP 8, the Python style guide

**Pip install pylint**

a=1

b=2

print(a)

print(B)

Machine generated alternative text:
C: XPythonCodeXIB)Error unittest . py 
Module unittest 
unittest . 
unittest . 
unittest . 
unittest . 
unittest . 
•4. 
•6: 
ce3B4 : 
ce114: 
ceae3 : 
ceae3 : 
Eese2 : 
Final newline missing (missing-final-newline) 
Missing module docstring (missing-module-docstring) 
Constant name "a" doesn't conform to UPPER CASE naming style (invalid-name) 
Constant name "b" doesn 't conform to UPPER CASE naming style (invalid-name) 
Undefined variable 'B' (undefined-variable) 

**"""**

**a=1**

**b=2**

**print(a)**

**print(B)**

**"""**

**def myfunc():**

**"""**

**a simple function**

**"""**

**first =1**

**second=2**

**print(first)**

**print(second)**

**myfunc()**

**Unit test library**

**Cap.py**

def cap\_text(text):

'''

Input a string

output the capitalized string

'''

return text.title()

Test\_cap.py (test case)

import unittest # **built in function**

import cap

class TestCap(unittest.TestCase):

def test\_one\_word(self):

text='python'

result=cap.cap\_text(text)

self.assertEqual(result,'Python')

def test\_multiple\_words(self):

text='hello world'

result=cap.cap\_text(text)

self.assertEqual(result, 'Hello World')

if \_\_name\_\_=='\_\_main\_\_':

unittest.main()

Section 9 ) python decorator

Machine generated alternative text:
Decorators allow!q t? "decorate" a 
function, let's dis what that word 
means in this context: 

Machine generated alternative text:
Python has decorators that allow you to 
tack on extra functio ality to an already 
existing function 
They use the @ op agor and are then 
placed on top of the-Original function. 

Machine generated alternative text:
Now you can easily add on extra 
functionality with a decorator: 
@some decorator 
def simple_func(): 
# Do simple stuff 
rn something 

Assignment of function

def hello():

print('hello')

hello() # output hello

greet=hello

greet() # output hello

del hello

#hello() # output NameError: name 'hello' is not defined

greet() # output hello

Executing function inside function

# defining function inside other function

def hello():

print('hello function has been executed')

def greet():

return (' \t greet function executed inside hello function')

def welcome():

return('\t welcome function executed inside hello')

print(greet())

print(welcome())

print('this is the end of hello function')

hello()

**"""**

**output**

**hello function has been executed**

**greet function executed inside hello function**

**welcome function executed inside hello**

**this is the end of hello function**

**"""**

Returning a function

**#note we cann access hello function but we cant execute greet function outside hello function**

**#since it scope is defined inside hello function but if we want to execute greet function outside**

**#hello function it need to return function**

def hello\_new(name='jose'):

print('hello\_new function has been executed')

def greet\_new():

return (' \t greet\_new function executed inside hello\_new function')

def welcome\_new():

return('\t welcome\_new function executed inside hello\_new')

if name == 'jose':

return greet\_new

else:

return welcome\_new

my\_new\_func=hello\_new('sau')

print(my\_new\_func())

**"""**

**output**

**hello\_new function has been executed**

**welcome\_new function executed inside hello\_new**

**"""**

Another example

**# having a function and defining a function and returning a function**

def cool():

def super\_cool():

return ('i am super cool')

return super\_cool

result = cool()

print(result()) # output i am super cool

**# passing a function as an arguement**

def hello():

print("inside hello function")

def other(some\_def\_function):

print('other code run here')

some\_def\_function()

other(hello) **# passing a raw function and not executing it**

**"""**

**output**

**other code run here**

**inside hello function**

**"""**

################ decorator example

def new\_decorator(orig\_func):

def wrap\_func():

print('some extra code before original function')

orig\_func()

print('some extra code after original function')

return wrap\_func

**# do some decorator in orginal function . pass a function and then return as a wrap**

**# in future if we dont want that extra functionality we can remove new\_decorator**

@new\_decorator

def func\_needs\_decorator():

print('i want to be decorated')

func\_needs\_decorator()

**"""**

**@new\_decorator eqivalent to**

**#decorated\_func=new\_decorator(func\_needs\_decorator)**

**#decorated\_func() # returning wrap version of original function**

**output**

**some extra code before original function**

**i want to be decorated**

**some extra code after original function**

**"""**

**Section 10 ) Python Generator**

Machine generated alternative text:
This type of function is a generator in 
Python, allowing us-t generate a 
sequence of valu sov time. 
The main differe cc-lh yntax will be the 
use of a yield statement. 

Machine generated alternative text:
Generator functions will automatically 
suspend and resu 
state around the a 
generation. 
their execution and 
int of value 
The advantage ist a instead of having to 
compute an entire series of values up front, 
the generator computes one value waits 
is for. 

Machine generated alternative text:
If a user did need the list, they have to 
transform the generator to a list with 
list(range(O,10)) 
Let's explore how to create our own 
generators! 

def create\_cubes(n):

result =[]

for x in range (n):

result.append(x\*\*3)

return result

test=create\_cubes(5)

print(test) # we are storing in memory [0, 1, 8, 27, 64]

**# requirement we dont want to save in memory**

**# create own generator using yield**

**# it will not save in memory generate a value as we need them**

def create\_cubes(n):

for x in range (n):

yield x\*\*3

for x in create\_cubes(5):

print(x)

**# another example**

def simple\_gen():

for x in range(3):

yield x **# yield remembering the previous value and generating next value it not storing value**

for num in simple\_gen():

print(num) #output 0,1,2

g=simple\_gen()

print(next(g)) # output 0

print(next(g)) # output 1

print(next(g)) # output 2

#print(next(g))  **# StopIteration error**

**# iter function we cant call next direct on string function since next will call on generator function**

**# convert object into iterable**

str="string"

result =iter(str)

print(next(result)) # output s

Homework

**"""**

**Problem 1**

**Create a generator that generates the squares of numbers up to some number N.**

**"""**

def gensquares(N):

for x in range(N):

yield x\*\*2

for num in gensquares(4):

print(num)

"""

**Problem 2**

**Use the iter() function to convert the string below into an iterator:**

**"""**

s = 'hello'

s = iter(s)

print(next(s))

Section 11) Advanced Python Modules

Machine generated alternative text:
Modules Covered 
Collections 
Os module and Datetime 
Math and Random 
Python Debugger 
Timeit 
Regular Expressions 

* 1. **Python collection modules**

**# counter demo use for count it**

**# note : lot of different thing we can do with counter object**

**#example 1**

from collections import Counter

mylist =[1,1,1,1,2,2,2,2,3,3,3,3]

print(Counter(mylist)) # output Counter({1: 4, 2: 4, 3: 4})

**#example 2**

mylist=['a','a','a',1,1,1,1]

print(Counter(mylist)) # output Counter({1: 4, 'a': 3})

**# example 3**

print(Counter('aaaaaaaaabbbbbbbbbbcccccccccc')) # output Counter({'b': 10, 'c': 10, 'a': 9})

**# example 4**

sentence =" how many times does each word show in this sentence with a word"

print(Counter(sentence.split()))

# output Counter({'word': 2, 'how': 1, 'many': 1, 'times': 1, 'does': 1, 'each':

# 1, 'show': 1, 'in': 1, 'this': 1, 'sentence': 1, 'with': 1, 'a': 1})

**#Example 5**

c=Counter('aaaaaaaaabbbbbbbbbbcccccccccc')

print(c.most\_common()) # most\_common will return as a tuple # output [('b', 10), ('c', 10), ('a', 9)]

Default dictionary

**from collections import defaultdict**

#Example 1

d={'a':10}

print(d['a']) # 10

#print(d['wrong']) # output KeyError: 'wrong

**#NOTE defaultdict would add a some default key if there is no key occur in dictionary**

d=defaultdict(lambda:0)

d['correct']=100

print(d['correct']) # output 100

print(d['wrong']) # output 0

print(d) # output defaultdict(<function <lambda> at 0x000002131499D0D8>, {'correct': 100, 'wrong': 0})

Named tuple

Machine generated alternative text:
from collections import namedtuple 
Dog namedtuple 
o 
* , verbose-False, 
ield names 
Signature : 
e) 
Docstring: 
Returns a 
namedtuple(typename, 
new subclass of tuple 
rename-False, 
module—Non 
with named fields. 

**Named tuples assign meaning to each position in a tuple and allow for more readable**

**# named tuple**

**#note for very large tuple we dont remember value for what index so we can use nametuple**

from collections import namedtuple

Dog = namedtuple('Dog',['age','colour'])

sammy= Dog(age=5,colour='green')

print(sammy) # output Dog(age=5, colour='green')

print(sammy.age)

2)Opening and reading files and folders

Machine generated alternative text:
Deleting Files 
NOTE: The os module provides 3 methods for deleting files: 
• os.unlink(path) which deletes a file at the path your provide 
• os.rmdir(path) which deletes a folder (folder must be empty) at the path your provide 
this is the most dangerous, as it will remove all files and folders contained in the path. All Of these 
methods can not be reversed! Which means if you make a mistake you won't be able to recover the file. Instead 
we will use the send2trash module. A safer altemative that sends deleted files to the trash bin instead of 
permanent removal. 
Install the send2trash module with: 

f=open('practice.test','w+')

f.write("this is a text file")

f.close()

**# GET CURRENT WORKING DIRECTORY AND LIST ALL FINE IN DIRECTORY**

**# os has lot of functionalities**

import os

print(os.getcwd()) # it will give current working directory

print(os.listdir()) # it will list everything in current working directory

print(os.listdir('C:\\Users'))

**# movie file in different directory**

import shutil

shutil.move('practice.test','c:\\Users\\saurakes\\Desktop')

======================================

Os.walk================use to traverse folder,subfolder and files

**for** folder , sub\_folders , files **in** os.walk("Example\_Top\_Level"):  
   
 print("Currently looking at folder: "+ folder)  
 print('**\n**')  
 print("THE SUBFOLDERS ARE: ")  
 **for** sub\_fold **in** sub\_folders:  
 print("**\t** Subfolder: "+sub\_fold )  
   
 print('**\n**')  
   
 print("THE FILES ARE: ")  
 **for** f **in** files:  
 print("**\t** File: "+f)  
 print('**\n**')  
   
 *# Now look at subfolders*

====================================================================

* 1. Python datetime module

**# time info**

import datetime

mytime=datetime.time(2,20) # 2 am 20 minute

print(mytime.minute) # output 20

print(mytime) # output 02:20:00 note second will fill automatically

**# date info**

today =datetime.date.today()

print(today) # output 2020-09-28

print(today.year) # output 2020

print(today.month) # output 9

print(today.day) # output 28

print(today.ctime()) # output Mon Sep 28 00:00:00 2020

**# want both info combine together that is date and time**

from datetime import datetime

mydatetime= datetime(2012,10, 18, 12, 2 ,1)

print(mydatetime) # output 2012-10-18 12:02:01

# if we did mistake we can replace it

mydatetime=mydatetime.replace(year=2020)

print(mydatetime) #output 2020-10-18 12:02:01

**# want to see how much time user login into site**

from datetime import date

date1 =date(2020,11,3)

date2 =date(2019,11,3)

result =date1-date2

print(result) # output 366 days, 0:00:00

* 1. Python math and random modules

Math

import math

print(help(math)) # to see all method available in math

**#1 ) floor and ceil**

value=4.35

print(math.floor(4.35)) # output 4

print(math.ceil(4.35)) # output 5

**#)mathematical constant**

from math import pi

print(pi) # output 3.141592653589793

# or

print(math.pi) # output 3.141592653589793

# very important Numpy :----library mainly design for numeric caluclation

**# 3) mathemaitcal logarithmic**

print(math.e) # output 2.718281828459045

print(math.log(math.e)) # output 1

print(math.log(100,10)) # output 2

**# 4) trigonometric function**

print(math.sin(30)) # output in radians -0.9880316240928618

**Random**

import random

**# example 1**

num= random.randint(1,10) # it will return some random integer

print(num) # output 10

**# note if we set seed we will get same random integer**

**# example 2**

#random.seed(42)

result=random.randint(0,101)

print(result) # output 81

**# example 3**

mylist =list(range(0,20))

print(random.choice(mylist)) # output random integer

print(mylist) # output [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

**# sample with replacement**

print(random.choices(mylist,k=10)) # output [9, 14, 18, 16, 11, 6, 4, 17, 5, 6]

# sample without replacement

print(random.sample(mylist,10)) # output [8, 13, 19, 9, 12, 18, 6, 5, 11, 3]

**# shuffle a list**

random.shuffle(mylist)

print(mylist) # output [18, 16, 15, 9, 13, 2, 17, 12, 5, 7, 8, 19, 6, 3, 1, 4, 10, 14, 0, 11]

**# floating point allowed**

print(random.uniform(a=1,b=100)) # output 72.57615094294361

Python debugger :--to figure out error we can perform operation in mid

import pdb

x=[1,2,3]

y=2

z=3

result\_one=y+z

**pdb.set\_trace() # we can set it**

result\_two=x+y

**# to quit debugger we can use q**

* 1. **Regular expression**

**Part 1)**

**# search ,findall -will return list of matches and finditer**

text = 'the agent phone number is 408-555-1234. call soon'

**# we can search string like this if found it will return a boolean**

print('phone' in text) # output true

**# using regex**

import re

**# example 1**

**# pattern in text**

pattern = 'phone'

**# note this will search only first match**

result = re.search(pattern, text)

print(result) # it will return index also output <re.Match object; span=(10, 15), match='phone'>

**# to get index location**

print(result.span()) # output (10, 15)

**# start and end index**

print(result.start()) # output 10

print(result.end()) # output 15

**# pattern not in text**

pattern = 'not in text'

result = re.search(pattern, text)

print(result)

**# example 2 want to get multiple match use findall**

text ='my phone once my phone twice'

pattern='phone'

matches =re.findall(pattern,text)

print(matches) # output ['phone', 'phone']

for match in re.finditer(pattern,text):

print(match)

print(match.span()) # output (3,8) and(17,22)

print(match.group()) # output phone -----it will return actual match we are searching for

**# output : <re.Match object; span=(3, 8), match='phone'>**

**# <re.Match object; span=(17, 22), match='phone'>**

**Regex 2**

Machine generated alternative text:
Regular Expressions Patterns 
Character Identifiers 
Character 
NS 
Pattern Code Example Match 
Non-alphanumeric 
Non-whitespace 
file 
alsblsc 
'WWI'MWIW 
file 25 
abc 
ABC 
Yoyo 

Machine generated alternative text:
Quantifiers 
Character 
Description 
Occurs one or more times 
(3) 
Occurs exactly 3 times 
Occurs 2 to 4 times 
Occurs 3 or more 
• Occurs zero or more times 
Once or none 
Example Pattern Code 
Version NW-Xw+ 
'Id12,4) 
lw(3,) 
plurals? 
Example Match 
Version A-bl 1 
abc 
123 
anycharacters 
AAACC 
plural 

import re

**# example 1**

text ='my phone number is 408-555-1234'

**# but we dont know phone number**

**#pattern='408-555-1234'**

**#result=re.search(pattern,text)**

result=re.search(r'\d\d\d-\d\d\d-\d\d\d\d',text) # to tell python we are using backslash as regex will add r

print(result) # output <re.Match object; span=(20, 32), match='408-555-1234'>

**# to get actual match**

print(result.group()) # output 408-555-1234

**# notice we have written \d many times so what if 100 digit is there to resolve this we use quantifiers**

**# using quantifiers**

result=re.search(r'\d{3}-\d{3}-\d{4}',text)

print(result) # output <re.Match object; span=(20, 32), match='408-555-1234'>

**#Example 2 find phone number and also extract area code which is first 3 digit**

**# compile ----will do compile different regex pattern code**

**# it takes a mupltiple pattern code and each pattern code is seperated with parenthesis**

**#as a group and compiles them with single expression**

phone\_pattern =re.compile(r'(\d{3})-(\d{3})-(\d{4})') # parenthesis indicate a group of pattern

result = re.search(phone\_pattern,text)

print(result.group()) # output 408-555-1234

print(result.group(1)) # output 408

Part 3

import re

**#example 1**

result=re.search(r'cat','the cat is here')

print(result) # output <re.Match object; span=(4, 7), match='dog'>

**# if pattern not match**

result=re.search(r'cat','the dog is here')

print(result)

**# then will use pipe operator**

result=re.search(r'cat|dog','the dog is here')

print(result) # output <re.Match object; span=(4, 7), match='dog'>

**#example 2 use of wild card character**

result=re.findall(r'at','the cat in the hat sat there')

print(result) # output ['at', 'at', 'at']

**# requirement want to get all word use wild card character**

result=re.findall(r'.at','the cat in the hat sat there')

print(result) # output ['cat', 'hat', 'sat']

**# example 3 start with a number . it will search entire text start with a number**

result = re.findall(r'^\d','1 is a number ')

print(result) # output ['1']

**# example 4 end with a number**

result = re.findall(r'\d$',' number is a 2')

print(result) # output ['2']

**# example 5 want to exclude digit**

phrase =' there are 3 numbers 34 '

pattern =r'[^\d]'

result=re.findall(pattern,phrase)

print(result) # output [' ', 't', 'h', 'e', 'r', 'e', ' ', 'a', 'r', 'e', ' ', ' ', 'n', 'u', 'm', 'b', 'e', 'r', 's', ' ', ' ']

**# example 6)**

phrase =' there are 3 numbers 34 '

pattern =r'[^\d]+'

result=re.findall(pattern,phrase)

print(result) # output [' there are ', ' numbers ', ' ']

**# example 7) exclude ! . ? ======== this is grouping for exclusion**

test\_phrase = 'This is a string! But it has punctuation. How can we remove it?'

result =re.findall('[^!.? ]+',test\_phrase)

print(result) # output ['This', 'is', 'a', 'string', 'But', 'it', 'has', 'punctuation', 'How', 'can', 'we', 'remove', 'it']

**# example 8 :--- we can also use grouping of inclusion \w ==alphanumeric**

text = 'Only find the hypen-words in this sentence. But you do not know how long-ish they are'

result=re.findall(r'[\w]+-[\w]+',text)

print(result) # output ['hypen-words', 'long-ish']

**# example 9 Parenthesis for Multiple Options**

# Find words that start with cat and end with one of these options: 'fish','nap', or 'claw'

text = 'Hello, would you like some catfish?'

texttwo = "Hello, would you like to take a catnap?"

textthree = "Hello, have you seen this caterpillar?"

result=re.search(r'cat(fish|nap|claw)',text)

print(result) # output <re.Match object; span=(27, 34), match='catfish'>

* 1. Timing your python code

Machine generated alternative text:
We will focus on 3 ways of doing this: 
o Simply tracking time elapsed 
o Using the timeit module 
o Special %%timeit "magic" for Jupyter 
Notebooks 

Approach 1

**"""**

**Timing your code**

**Sometimes it's important to know how long your code is taking to run,**

**or at least know if a particular line of code is slowing down your entire project.**

**Python has a built-in timing module to do this.**

**Example Function or Script**

**Here we have two functions that do the same thing, but in different ways.**

**How can we tell which one is more efficient? Let's time it!**

**note this is not efficient way since input should be large**

**"""**

def func\_one(n):

**'''**

**Given a number n, returns a list of string integers**

**['0','1','2',...'n]**

**'''**

return [str(num) for num in range(n)]

result=func\_one(10)

print(result) # output ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

def func\_two(n):

**'''**

**Given a number n, returns a list of string integers**

**['0','1','2',...'n]**

**'''**

return list(map(str,range(n)))

result=func\_two(10)

print(result) # output ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

**"""**

**Timing Start and Stop**

**We can try using the time module to simply calculate the elapsed time for the code**

**"""**

import time

# STEP 1: Get start time

start\_time = time.time()

# Step 2: Run your code you want to time

result = func\_one(1000000)

# Step 3: Calculate total time elapsed

end\_time = time.time() - start\_time

print(end\_time) # **output 0.23803424835205078**

# STEP 1: Get start time

start\_time = time.time()

# Step 2: Run your code you want to time

result = func\_two(1000000)

# Step 3: Calculate total time elapsed

end\_time = time.time() - start\_time

print(end\_time) # **output 0.21996116638183594**

**# timeit function we can use it(just see in google)**

**7 unzipping and zipping files**

**Zipping a file**

f=open('fileone.txt','w+')

f.write('one file')

f.close()

f=open('filetwo.txt','w+')

f.write('two file')

f.close()

**# want to compress a file**

import zipfile

comp\_file= zipfile.ZipFile('comp\_file.zip','w')

comp\_file.write("fileone.txt",compress\_type=zipfile.ZIP\_DEFLATED)

comp\_file.write('filetwo.txt',compress\_type=zipfile.ZIP\_DEFLATED)

comp\_file.close()

**# extract file from zip**

zip\_obj = zipfile.ZipFile('comp\_file.zip','r')

zip\_obj.extractall("extracted\_content")

Section 12 ) web scrapping

Machine generated alternative text:
Web scraping is a general term for 
techniques involving automating the 
gathering of data from a website. 
In this section we will learn how to use 
Python to conduct web scraping tasks, 
such as downloading images or 
information off a website. 

Machine generated alternative text:
Limitations of Web Scraping 
o In general every website is unique, 
which means every web scraping 
script is unique. 
o A slight change or update to a 
website may completely break your 
web scraping script. 

Machine generated alternative text:
For effective basic web scraping we only 
need to have a basic understanding of 
HTML and CSS. 
Python can view these HTML and CSS 
elements programmatically, and then 
extract information from the website. 
Let's explore HTML and CSS in more detail. 

* 1. Setting of web scrapping libraries

Pip install requests======fetch content from website

Pip install lxml

Pip install bs4 =========beautiful soup

a)Grabbing a title

import requests

import bs4

**# Step 1: Use the requests library to grab the page**

**# Note, this may fail if you have a firewall blocking Python/Jupyter**

**# Note sometimes you need to run this twice if it fails the first time**

res = requests.get("http://www.example.com")

print(type(res)) # output <class 'requests.models.Response'>

print(res.text)

soup = bs4.BeautifulSoup(res.text,"lxml")

print(soup)

print(soup.select('title')) # output [<title>Example Domain</title>]

print(soup.select('p'))

**# requirement only want to grab title**

title\_tag = soup.select('title')

print(title\_tag[0].getText()) # output Example Domain

# paragraph

site\_paragrapgh = soup.select("p") # it will return a list

result=site\_paragrapgh[0].getText()

print(result)

* 1. Grabbing a class

Machine generated alternative text:
Syntax 
soup.select('div') 
soup.select('#some_id') 
soup.select('.some_class') 
soup.select('div span') 
soup.select('div span') 
Match Results 
All elements with 'div' tag 
Elements containing 
Elements containing class 'some_class' 
Any elements named span within a div 
element. 
Any elements named span directly within 
a div element, with nothing in between. 

import bs4

import requests

**# First get the request**

res = requests.get('https://en.wikipedia.org/wiki/Grace\_Hopper')

**# Create a soup from request**

soup = bs4.BeautifulSoup(res.text,"lxml")

print(soup.select(".toctext"))

for item in soup.select(".toctext"):

print(item.text)

"""

# output

Obituary notices

References

Further reading

External links

"""

Grabbing an image

import bs4

import requests

res = requests.get("https://en.wikipedia.org/wiki/Deep\_Blue\_(chess\_computer)")

soup = bs4.BeautifulSoup(res.text,'lxml')

#print(soup.select('.thumbimage'))

computer =soup.select('.thumbimage')[0]

print(computer)

print(computer['src'])

**"""**

**output //upload.wikimedia.org/wikipedia/commons/thumb/b/be/Deep\_Blue.jpg/220px-Deep**\_Blue.jpg

"""

Section 13 ) working with images with python

Pip install pillow (library)

Machine generated alternative text:


**# pip install pillow**

**Example 1)**

**# pip install pillow**

from PIL import Image

mac = Image.open('example.jpg')

print(type(mac)) # output <class 'PIL.JpegImagePlugin.JpegImageFile'>

**#mac.show() # it will open image**

**# image information**

print(mac.size) # output (1993, 1257) with width and height

# tell file name

print(mac.filename) # output example.jpg

**# croping a image**

mac.crop((0,0,100,100))

**Example 2)**

from PIL import Image

pencils = Image.open('pencils.jpg')

**##pencils.show()**

print(pencils.size)

x=0

y=0

w=1950/15

h=1300/1000

pencils.crop((x,y,w,h))

#pencils.show()

**# resize image**

pencils.resize((100,100))

pencils.show()

Section 14) work with pdf and csv files

Csv

Machine generated alternative text:
CSV stands for comma separated variables 
and is a very common output for 
spreadsheet programs. 
Example: 
o Name, Hours, Rate 
o David, 
20, 15 
o Claire, 
40, 20 

Machine generated alternative text:
Other libraries to consider: 
o Pandas 
Full data analysis library, can work 
with almost any tabular data type. 
• Runs visualizations and analysis. 
• One of my personal favorites, we 
teach it in various data science 
courses. 

Machine generated alternative text:
Other libraries to consider: 
o Openpyxl 
Designed specifically for Excel files. 
Retains a lot of Excel specific 
functionality. 
Supports Excel formulas. 
python-excel.org tracks various other 
Fyce! based Python libraries. • 

Machine generated alternative text:
Other libraries to consider: 
o Google Sheets Python API 
Direct Python interface for working 
with Google Spreadsheets. 
• Allows you to directly make changes 
to the spreadsheets hosted online. 
More complex syntax, but available in 

import csv

**#open the file**

data =open('example.csv',encoding='utf-8')

**#csv.reader**

csv\_data =csv.reader(data)

**# reformat it into a python object list of lists**

data\_lines=list(csv\_data)

print(csv\_data)

**# note to get rid of this error need to add encoding='utf-8'**

**# UnicodeDecodeError since it contain special character @**

**# output list of list first list will contain column names then second list 1 row data and so on**

print(data\_lines)

print(data\_lines[0]) # output column names ['id', 'first\_name', 'last\_name', 'email', 'gender', 'ip\_address', 'city']

**# to find total number of rows**

print(len(data\_lines)) # output 1001

for line in data\_lines[:3]:

print(line)

"""

**output**

**['id', 'first\_name', 'last\_name', 'email', 'gender', 'ip\_address', 'city']**

**['1', 'Joseph', 'Zaniolini', 'jzaniolini0@simplemachines.org', 'Male', '163.168.68.132', 'Pedro Leopoldo']**

**['2', 'Freida', 'Drillingcourt', 'fdrillingcourt1@umich.edu', 'Female', '97.212.102.79', 'Buri']**

**"""**

**# extract third value from row 10**

print(data\_lines[10][3]) # output hgasquoine9@google.ru

# grab all email

mylist=[]

for allemail in data\_lines[1:10]:

mylist.append(allemail[3])

print(mylist)

**# grab full name**

full\_names=[]

for line in data\_lines[1:5]:

full\_names.append(line[1]+' '+line[2])

print(full\_names) # output ['Joseph Zaniolini', 'Freida Drillingcourt', 'Nanni Herity', 'Orazio Frayling']

Write into csv file

**import csv**

**# newline controls how universal newlines works (it only applies to text**

**# mode). It can be None, '', '\n', '\r', and '\r\n'.**

file\_to\_output = open('to\_save\_file.csv','w',newline='') # it will overwrite a file

csv\_writer = csv.writer(file\_to\_output,delimiter=',')

csv\_writer.writerow(['a','b','c'])

csv\_writer.writerows([['1','2','3'],['4','5','6']])

file\_to\_output.close()

Machine generated alternative text:


**Pdf file in python**

Pdf -portable document format

Machine generated alternative text:
We've made sure that the PDF files 
included in this course material are 
readable by PyPDF2. 
Unfortunately we can't offer assistance for 
you own personal PDF files if they are not 
readable by PyPDF2. 

**pip install PyPDF2**

**Reading a pdf file**

**#pip install PyPDF2**

import PyPDF2

**"""**

**Similar to the csv library, we open a pdf, then create a reader object for it.**

**Notice how we use the binary method of reading , 'rb', instead of just 'r'.**

**"""**

**# Notice we read it as a binary with 'rb'**

f = open('Working\_Business\_Proposal.pdf','rb')

pdf\_reader = PyPDF2.PdfFileReader(f)

print(pdf\_reader.numPages) # output 5 pages

**# grab first page**

page\_one = pdf\_reader.getPage(0)

print(page\_one)

**# grab text on page**

page\_one\_text = page\_one.extractText()

print(page\_one\_text)

f.close()

# write one page from pdf into another pdf

**# add pages to pdf**

**# write first page of pdf into another pdf**

import PyPDF2

f = open('Working\_Business\_Proposal.pdf','rb')

pdf\_reader = PyPDF2.PdfFileReader(f)

first\_page = pdf\_reader.getPage(0)

pdf\_writer = PyPDF2.PdfFileWriter()

pdf\_writer.addPage(first\_page)

pdf\_output = open("Some\_New\_Doc.pdf","wb")

pdf\_writer.write(pdf\_output)

f.close()

Section 15) Emails with Python

Sending email

Machine generated alternative text:
To send emails with Python, we need to 
manually go through the steps of 
connecting to an email server, confirming 
connection, setting a protocol, logging on, 
and sending the message. 

Machine generated alternative text:
Each major email provider has their own 
SMTP (Simple Mail Transfer Protocol) 
Server. 
Provider 
SMTP server domain name 
Gmail (will need App Password) smtp.gmail.com 
Yahoo Mail 
Outlook.com/HotmaiI.com 
AT&T 
Verizon 
Comcast 
smtp.mail.yahoo.com 
smtp-mail.outlook.com 
smpt.mail.att.net (Use port 465) 
smtp.verizon.net (Use port 465) 
smtp.comcast.net 

section 16) Advanced python object and data structure

* 1. Advanced number

**# hexadecimal of number**

print(hex(12)) # output 0xc

**# binary representaion of number**

print(bin(4)) # output 0b100

**# power of number**

print(pow(2,4)) # output 16

**# three arguement will give mod x\*\*y%z**

print(pow(2,4,3)) # output 1

**# absolute value function**

print(abs(-2)) # output 2

**# round function**

print(round(3.91562,2)) # output 3.92

2)advanced string

s='hello world'

print(s.capitalize()) # output Hello world ( it will capitalize first letter of string)

print(s.upper()) # output HELLO WORLD

print(s.lower()) # output hello world

print(s.count('o')) # output 2

print(s.find('o')) # first occurence of o (is index) output 4

print('hello\thi') # output hello        hi

s='hello'

print(s.isalnum()) # output True

print(s.islower()) # return True if all character in s are in lower case

print(s.isspace()) # output =False return True if all character in s is space

s=' '

print(s.isspace()) # output True

s='hello'

print(s.endswith('o')) # True

#same as s[-1]=='o'

print(s.split('e')) # output ['h', 'llo']

s='aaaaassssssddddd'

print(s.partition('s')) # it will partition only for first instance and will return completely before and after

# output ('aaaaa', 's', 'sssssddddd')

* 1. Advanced set

**#1) add : add elements to a set. Remember, a set won't duplicate elements;**

**# it will only present them once (that's why it's called a set!)**

s=set()

s.add(1)

s.add(2)

s.add(2)

print(s) # output {1, 2}

**# 2) clear : removes all elements from the set**

s.clear()

print(s) # output empty set ---------set()

**# 3) copy : returns a copy of the set. Note it is a copy, so changes to the original don't effect the** copy.

s={1,2,3}

sc= s.copy()

print(sc) # output {1, 2, 3}

s.add(4)

print(s) # output {1, 2, 3, 4}

print(sc) # output {1, 2, 3}

**# 4) difference : difference returns the difference of two or more sets. The syntax is:**

# set1.difference(set2)

print(s.difference(sc)) # output {4}

**#5) difference\_update difference\_update syntax is:**

**#set1.difference\_update(set2)**

**#the method returns set1 after removing elements found in set2**

s1 = {1,2,3}

s2 = {1,4,5}

s1.difference\_update(s2)

print(s1) # output {2, 3}

**# 6) discard Removes an element from a set if it is a member. If the element is not a member, do nothing.**

s={1,2,3,4}

s.discard(2)

print(s) # output {1, 3, 4}

**#7) intersection**

**#Returns the intersection of two or more sets as a new set.(i.e. elements that are common to all of the sets.)**

s1 = {1,2,3}

s2 = {1,2,4}

print(s1.intersection(s2)) # output {1, 2}

**#8) isdisjoint This method will return True if two sets have a null intersection**

s1 = {1,2}

s2 = {1,2,4}

s3 = {5}

print(s1.isdisjoint(s2)) # output False

print(s1.isdisjoint(s3)) # output True

**#9) issubset This method reports whether another set contains this set.**

s1={1,2}

s2={1,2,4}

print(s1.issubset(s2)) # output True

**# 10) issuperset This method will report whether this set contains another set.**

print(s2.issuperset(s1)) # output True

print(s1.issuperset(s2)) # output False

**# 11) union Returns the union of two sets (i.e. all elements that are in either set.)**

print(s1.union(s2)) # output {1, 2, 4}

* 1. Advanced Dictionaries

**# 1) dictionary comprehension**

test = {x:x\*\*2 for x in range(10)}

print(test) # output {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}

**# example 2**

d = {'k1':1,'k2':2}

for k in d.keys():

print(k) # output k1,k2

for v in d.values():

print(v) # output v1,v2

* 1. Advanced List

list1 = [1,2,3]

**#1) append You will definitely have used this method by now, which merely appends an element to the end of a list:**

list1.append(4)

print(list1) # output [1, 2, 3, 4]

**#) 2 )count=== count() takes in an element and returns the number of times it occurs in your lis**

print(list1.count(10)) # output 0

print(list1.count(2)) # output 1

**# 3) extend**

**#append: appends whole object at end:**

x = [1, 2, 3]

x.append([4, 5])

print(x) # output [1, 2, 3, [4, 5]]

#extend: extends list by appending elements from the iterable:

x = [1, 2, 3]

x.extend([4, 5])

print(x) # output [1, 2, 3, 4, 5]

**# Note how extend() appends each element from the passed-in list. That is the key difference.**

**# 4) index: index() will return the index of whatever element is placed as an argument.**

**# Note: If the the element is not in the list an error is raised.**

print(list1.index(2)) # output 1

#print(list1.index(12)) it will give error since not in list

**# 5) insert insert() takes in two arguments: insert(index,object)**

**# This method places the object at the index supplied.**

list1 =[1,2,3,4]

# Place a letter at the index 2

list1.insert(2,'inserted')

print(list1) # output [1, 2, 'inserted', 3, 4]

**# 6) pop**

**#You most likely have already seen pop(), which allows us to "pop" off the last element of a list.** However,

# by passing an index position you can remove and return a specific element.

ele = list1.pop(1) # pop the second element

print(list1) # output [1, 'inserted', 3, 4]

print(ele) # output 2

**# 7) remove : The remove() method removes the first occurrence of a value**

print(list1) # output [1, 'inserted', 3, 4]

list1.remove('inserted')

print(list1) # output [1, 3, 4]

list2 = [1,2,3,4,3]

list2.remove(3)

print(list2) # output [1, 2, 4, 3]

**# 8) reverse reverse() reverses a list**

list2.reverse()

print(list2) # output [3, 4, 2, 1]

**# 9) sort The sort() method will sort your list in place:**

list2.sort()

print(list2) # output [1, 2, 3, 4]

**#**

**#The sort() method takes an optional argument for reverse sorting.**

**#Note this is different than simply reversing the order of items.**

list2.sort(reverse=True)

print(list2) # output [4, 3, 2, 1]