Comebackdays -2,5,6,13,38,44,45

day 1
repl-read evaluate print loop(Shift+n in vs code terminal,ctrl+z or exit() to exit)
You cannot name your python file a keyword of python language due to clash of file and module search
You can run python file by python "04 python.py"
In replit after your code is xecuted program doesn't automatically stop and enters repl
import code
code.interact(local=dict(globals(), **locals()))
can be pasted after code to achieve the same result in vs code
os.system("python") also does same but starts a new interpreter in a subshell and pauses the original interpreter and any variables and functions in original script are not accessible in the new interpreter session.

day 2
come back to day 2 later-jarvis,fiverr,web scraping(automated?)
jarvis
flappy bird
snake game
face recognition
love calc

day 3
built in modules-(refrigerator)locally stored
external modules-pip(preferred installer program/package manager) gets(/installs to python interpreter) it from internet (shopping)
packages in replit cloud cant be used by device and that in device can't be used in replit cloud
replit automatically downloads some packages.some you have to download using shell

day 4
everything is treated as an object in python(int float are classes)
print-print function prints the object passed to it to the console(,
operator,str() FUNCTION , end="" termina char)

day 5
come back to day 5 later after file manipulation
print(object(s), sep=separator, end=end, file=file, flush=flush)
alt+down/up to move a line down/up
alt+shift+down/up to copy a line down/up
alt + right/left to move to next word(next space)
""" """ or ''' ''' are multilined strings but are ignored by compiiler
ctrl+/ to comment/uncomment selected content
" ' " and ' " ' are allowed
ctrl+click to open in new tab
tab to autocomplete

day 6
Come back to day 6 after data structures
variable=container in ram
All variables are pointers to objects
A variable can point to any type of object
variable type=data type=type of container
type(a)=<class type>(type can be
int,bool(True/False),str,noneType,float,tuple,dict,list,complex) is of type class
int() returns 0 str() returns null string list() returns emoty list and so on

knowing the type can help you avoid errors and operate on same datatypes with same datatypes
"hello"+str(5)
complex(a,b)
tuple=immutable list is a collection of objects
dict-key value pairs(value=f(key)).dict["name"]="akhil"
In python no type declaration is required(dynamically typed or type is assigned at runtime and can be changed)

```
for item in list([a,b,c,d])
  print(item)
```

A variable name cannot start with a number
A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
If you name a variable a keyword of python language(which is sometimes allowed e.g:int,str) and also use the keyword in question(str(),int()) python misbehaves

day 7

op	process	example
+	Addition	15+7
-	Subtraction	15-7
*	Multiplication	5*7
**	Exponential	5**3
/	Division	5/3(returns a float value unlike in c)
%	Modulus	15%7
//	Floor Division	15//7(floor(15/7))

you can also do A op=B equivalent to A= A op B
ternary operator x=a if a>b else b
loop x=a if bool else b if bool2 else c if bool 3 else d if bool4....

and,or,not in place of ||,&&,! for statements and boolean values

day 8
alt+left click to add a cursor there
ctrl+shift+l to select all instances of selected word

day 9
Typecasting
Explicit: Programmer does it
data type() function

not always possible(int("hellow")) but int("005") is converted to 5
does not use ascii value system(int("1")=1)
at runtime
Implicit:
automatic
lower order to higher order(int to float)
at run time by intrepreter

day 10
User Input
input(str) function prints str and returns a string from console
when input is prompted any text already present in the line other than the prompt passed is deleted in replit
but it works normally in vscode
if r precedes a string literal \ is treated as \ and not escape sequence
re.split(r"[\n,;:]+",str) makes list of str with mentioned seperators (+ to treate repeated seperators as single seperator)
import re has to be done
emotes are basically \unicode
input is considered to be taken in triple quote format where everything is taken as is
sys.stdin.read() takes input until ctrl+z is pressed

day 11
Strings(immutable in python)
""" - represents what you write as is even emotes(windows+.)
str[i] to access ith char(treated as a string with just ith character in it
-i== len(str)-i
when a string is printed all special characters(escape sequences,emotes) are replaced appropriately and cursor types everything character by character.when it encounters(for example) \n it goes to next line
for c in str
    print(c) #this is basically printing all characters in string with newline characters between them
\n \t \u00023432 etc are considered a single character

day 12
String slicing
str[0:len(str)}== str[:]
str[-x:-y]=str[len(str)-x:len(str)-y]
str[x:y]
from x to less than y(y not included)

day 13
come back to day 13 after data structures
Strings are immutable but you can use functions to return a new string
len(str)-returns length of the string
str.upper()-returns str in uppercase

.lower()-similar
.rstrip(c)-strips trailing c characters
.replace(substr1,substr2)-replaces all instances of substr1 with substr2
.split(c)-returns a list of strings with c used as seperation point
re.split(r"[\n,;:]+",str) makes list of str with mentioned seperators (+ to treate repeated seperators as single seperator)
(import re has to be done)
.capitalize()-first char capital rest small
.center(int) makes the string size int by adding padding before and after equally
.endswith(c,beginning,end)- checks whether the specified substring(full string by default if endswith(c)) ends with c
.find(c) returns index of first occurance of c(-1 if not found)
.index(c)-works the same as find but throws error if not found
.isalnum()-checks if only alphabets and numbers are present
.isalpha()-checks if only alphabets are present
.isprintable()-checks if entered string is printable.escape sequences that tell it to print a certain character(\\ 😂) instead of moving the cursor are printabe else unprintable(\n \t)
.isspace() checks if only space is present
.istitle() checks if first letter of every word is caps rest small
.startswith(c) checks if string starts with c
.swapcase()-swaps the case
.title()-makes it title
isupper/islower-check for lower or upper
string.join(iterable)-to join iterable to string.iterable can be loop
if you use str(list) it becomes "[1,2,3]"

"str"*2 ="strstr" lol

in keyword
  if "ha" in "harry"

day 14
if-else
Imagine curly brackets at change in level every level of indentation
all thing on same level of indentation are said to be in one block

```
if():
    line1
elif():
    line2
else:
    line3
```

day 15
time module-is a built in module
time.strftime("%H:%M:%S") gives time in 00:00:00 24 hours format

day 16
match case statement-was added in version 3.10 very recently
match x

```
 case 1:
   line1
 case 2:
   line2
 .
 .
 .
 case n:
   line n
 case _ if condition1(like x!=5 or x<5)
    line n+1
 .
 .
 .
 case _ if conditionN:
 case _:(this is basically no condition)
```

the statement is checked from top to bottom and once matched exits the indentation block of match

day 17
Loops
For loop:
for loops can iterate over a sequence of iterable objects in python

```
for var in str/list/range():
        print(var)
```

range() function returns a range object which contains iterable objects
range(initial value,exit value,increment) or range(initial value,exit value) or range(exit value)
when exit value is reached or exceeded loop exits initial value is where loop starts
increment is how much subsequent value is incremented
default increment is 1
default initial val is 0

Alternative explanation:
range(start ,stop,step)
start (optional) – The starting value of the sequence. If omitted, it defaults to 0.
stop – The end value of the sequence (exclusive). The sequence will go up to, but not include, this value.
step (optional) – The step or increment between values in the sequence. If omitted, it defaults to 1
step can be negative
this loop never goes into infinite loop and just doesnt run in such cases

day 18
While loops

```
while condition:
  code
```

```
else:
  else is run if while exits normalyy without encountering a break(finding
something youre looking for)

day 19
break and continue
do while loop syntax
while True:
  code
  if condition
     break

day 20
Functions-Reusable code
Function must be defined before it is called(not necessarily be defined above)
def function_name(parameters):
  pass
  # Code and Statements
if pass is not used function cannot be empty
Any indented block cannot be empty without pass or IndentationError is thrown
you can come back later and define function(in the same place) with pass
function call
 function(arguments)
mutable object operations inside function affect original object(list)
Immutable object operations do not as it involves creating a new
object(int,float,string,tuple)

day 21
Function arguments

Default arguments-def fun(c,d,a=5,b=6)
All Non default arguments must come before default arguments

Positional argument-Arguments are parsed by position and taken in the same order

keyword argument-Argument taken by keyword without regard to position

Positional arguments cannot follow keyword argument
fun(1,2,3,g=5,d=7)

Arguments as a tuple fun(a,*nums,b)-Multiple arguments b can only be accessed
through keyword

Arguments as a dict fun(a,**name)-No parameter can exist after **
fun(1,fname="akhil",lname="k")a=1 can also be written in place of 1 before or after
and it wont be part of dict and passsed into a only

Only one of * or ** parameter can exist in a function
** parameter can follow * parameter but otherway around is not allowed
```

day 22
Lists
List=[1,,"hello",3] is an object that is a collection of objects
Can be sliced and negative indexed similar to string
list[start:stop:jump] similar to increment index in range returns a sub object containing part of parent object and can be divided further
list[1:7][1:2][1]

in keyword
if (obj in list): here obj in list turns into boolean

list comprehension
List = [Expression(item) for item in iterable if Condition]
iterable - any object capable of returning its members one at a time
condition-any boolean

day 23
List methods
list(iterable) to create a list of iterable items in iterable(iterable can be loop)
e.g
list(for x in range(5)) returns [0,1,2,3,4]
len(l)-returns length of list(number of items in an iterable)
l[i]=obj to change ith index to obj
def l.sort(reverse=False)
l.sort()-sorts in ascending order
l.sort(reverse=True)-sorts l in descending order
l.reverse()-reverses l
l.index(item)-returns index offirst occurance of item in list
l.count(item)-returns number of occurances of item
l.copy() returns a copy of the list
l.append(obj) inserts obj at end of list
l.insert(index,obj) inserts obj at index otherwise preserving the order
l.extend(iterable)-inserts objs from iterable to end of list in order
l=l+list2-creates a new object and l now points to new object
l.pop(index) pops and returns the element aat index
l.pop() pops and returns last element

day 24
Tuples-(1,2,3,"hello",True)
t=(1) is an int
t=(1,) is a tuple
The objects in tuples cannot be replaced or removed or added but can be modified if possible(incase of lists inside)
Tuples hold references to the objects.As long as they refer to the same object it does not matter
Indexing and negative indexing and slicing(returns new tuple) similar to strings and lists possible (jump is also present)

day 25 Tuple methods

To change tuple

tuple=list(tuple)
modify list
tuple=tuple(tuple)
or
tuple=tuple+tuple2(returns new tuple and tuple now refers to it)
tuple.count(item) returns number of times item occurs
.index(item,start,stop) searches first occurance of item from start to stop
substring(throws an error if not found
start and stop or stop can be omitted
len(tuple) give length of tuple

day 26
looping ternary operator
x=a if bool else b if bool2 else c if bool 3 else d if bool4....

day 27
kbc
"1"!=1
"a"!=97
int("a")=valueerror
ord(a)==97
char(97)==a
ord("ab") is error
1==true,4!=True
Here True is converted to 1 and False to 0 and numbers are not converted to bool
bool(5)==true
when comparing two strings it is done lexographically from left to right

day 28
str="bla bla {0} {1} bla bla {2}....{4}"
str.format(var0,var1,var2,var3,var4) replaces the var in string at respective indexes and returns it
Here var3 is not used in str but still works aslong as all used {}s have an argument
or
str="{} bla bla {} bla {}"
str.format(var0,var1,var2) are placed from left to right(numbers are automatically assigned 0,1,2)
A mix of above formats is not allowed
or
str="bla bla{} {}{a} {b.2f}{}"
str.format(1,2,3,a=5,b=6.232)-positional arguments cannot appear after keyword ones
keyword arguments format can be mixed with either of the above two formats
.nf rounds off to nth decimal place

f strings

f"hello {var} bla bla {{var2}} bla bla {{{var3}}} {b.2f}"

here var 2 is not replace but appears as {var2}
var3 is replaced with pair of {}
Double {{}}-become single {}
Single {} becomes var
Odd-replace
Even-As is
After that remaining even number of {} become half

day 29
Docstrings
Multilined string right below a function heading that tells working of python
Should be above even pass statement
can be accessed using doc attribute
an attribute generally refers to a value associated with an object, which can be accessed using dot notation
function.__doc__ returns the doc string of function

PEP 8(Python Enhancement proposal) is a s et of conventions to be followed in python code
PEP 8 is a document that provides guidelines and best practices on how to write Python code.The primary focus of PEP 8 is to improve the readability and consistency of Python code.

The Zen of Python
import this in code have like a print statement that prints

The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than right now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

These are guiding principle of Python development

The 20th one doesnt print and the 20th aphorism is a mystery

day 30
recursion
define function in terms of itself
try to find relation between a term in function series and its previous terms
define base case to start recursion to avoid stack overflow
Backtrack

day 31
Sets-Unordered or order is not maintained
-cannot be accesed by index
Duplicate values are discarded
Sets cannot contain mutable objects like lists or dictionaries.
Hashability is a property that allows an object to be used as a dictionary key or stored in a set, and it requires that the object's value does not change over its lifetime.

Sets can be accessed using for loop
```
for x in set:
  print x
```
prints elements of set in random order

s={} is a dict
s=set() is an empty set

day 32
A.fun(b)

returns new set	modifes original set	
.union(B)	.update(B)	-AUB
.intersection(B)	.intersection_update(B)	-A^B
.symmetric_difference(B)	.symmetric_difference_update(B)	-AUB-A^B
.difference(B)	. _update(B)	-A-A^B

.isdisjoint(B)   for all x x does not belong to A and B
.issuperset(B)   B belongs to A
.issubset(B)   A belongs to B
.add(item) adds item to A
.remove(item)-raises error if not present
.discard(item)-does not raise error if not present
.pop() removes and returns the last element(which is random)
del keyword
del s deletes the set along with its reference
s.clear() clears the set(empties it)
in keyword can be used
```
if x in s:
  bla bla
```

day 33
dict-ordered now but used to be unordered

A mapping of key value pairs
dict={key:value,key2:value2}
dic[key] returns value and throws error if not present
dic.get(key) returns none if not present
dic.keys() returns a dict_keys iterable object containing all the keys in order
dic.values()- similar dict_values object
dic.items()-returns a iterable dict_items object that has key value pairs
for key,value in dic.items():
   bla bla
keys become key values become value

day 34
update()
dic.update(dic2) adds any new key value pairs present in dic2 and updates value of present keys to that in dic2 if there is a clash
.clear() empties dic
.pop(key) to delete and return a (key,value) tuple
.popitem() pops the last element
del dic - annihilates dic
del dic[key] deletes the key value pair

day 35
for/while loop with else
The else statement is executed after completion of loop and before exiting it
if loop breaks at some point else block is not executedS

day 36
Exception handling
In python code terminates at the point of exception if it is not handled after some halting by the interpreter
try:
  code
except ValueError as e:
  code
except IndexError as e:
  code
except Exception as e:
  code
Here when a exception is raised it is matched from top to bottom
But the type of exception raised depends on which exception occurs first as code stops running immediately after that

Exception handles all types of errors
ValueError to handle <class ValueError> exception(data type or value mismatch.input error)
IndexError to handle index error(Arrayoutofbounds or negative index)
just writing
except:
or
except Exception:

also handles all exceptions but exception is not caught as e

day 37
finally
```
try:
  code
except:
  code
finally:
  code
```

If the try block is entered the finally block is executed before it leaves regardless if it crashes,encounters a return or exit statement or whatever

 TypeError arises when an operation or function is applied to an object of inappropriate type.
 ValueError arises when a function receives an argument of the correct type but an inappropriate value.
 NameError arises when you use an undefined variable
 IndexError arises when index is outofbounds
 MemoryError occurs incase of infinite loop and ram runs out
 IndentaionError occurs when there is an error in indentation

Examples:

For TypeError, attempting to concatenate a string with an integer ("hello" + 5).ord("he") gives type error
For ValueError, attempting to convert a string that is not a valid representation of an integer (int("hello")).
For NameError,ord(a) instead of ord("a")
For IndexError,l=[1,2] print(l[100])

Code before the exception is raised is run safely and is stopped at the line which cause exception.The line in question does not run and then interpreter goes into the except blocks

day 38
Custom errors

you can raise an error like

```
 raise ValueError
```

the program halts at this line of code and throws an empty value error(unlike if python throws it there is a custom string value inside)

raise Error(str) to throw the error with str
Error:str
When captured as e and printed prints str

Come back to this day after classes to learn to create custom error class inheriting from a custom error

day 39
KBC
List[i][j][k] to access list of lists

day 40
Secret code language
random function?
Only manually created classes and object start with caps

day 41
Short hand if else
code if bool else code if bool else code
have to atleast include "" in place of code or indentation error is thrown
print("A") if a > b else print("=") if a == b else print("B")
result = value_if_true if condition else value_if_false

day 42
linter
-identifies and correct subtle programming mistakes or unconventional coding practices that can lead to errors.(shows red line or lint)

enumerate function
enumerate(iterable) returns an enumerate object which has an iterbale sequence of (index,value) tuple that can be unpacked
```
for index,value in enumerate(iterable):
   code
```
By default, the enumerate function starts the index at 0, but you can specify a different starting index by passing it as an argument to the enumerate function
 enumerate(iterable,start=num)
Now simply all indices become +num as enumerate counts from num to len(iterable)+num

day 43
Virtual environment-A Python virtual environment is an isolated environment that allows you to manage dependencies and packages for a specific project without affecting the global Python installation or other projects to avoid conflicts
Multiple interpreters using different packages working on seperate projects
This can be especially useful when working on projects that have conflicting package versions or packages that are not compatible with each other.
import package as p abbreviates package to p

Windows files and cmd are case insensitive
pip install package==1.2.3
pip uninstall package

```
# Create a virtual environment
python -m venv myenv
```

```
# Activate the virtual environment (Windows)
myenv\Scripts\activate.bat #.ps1 in pllace of .bat if in power shell

# Deactivate the virtual environment
deactivate
```

These commands are to be used in the folder containing myenv

nul > your_file.txt to create empty file
pip freeze lists all versions of packages installed if you run this inside (myenv)
pip freeze > file.txt creates a file with pip freeze content

pip -r file.txt installs all the packages inside pip freeze

This can be used to share projects

Replit automatically installs and maintains packages and resolves dependencies and you can share to someone through replit

day 44 (come back after classes and objects)
How import works
import math

Importing in Python is the process of loading code from a Python module into the current script as well as code from any other module that the imported module depends upon as an object of module class(<class module> named the name of module.
class module:
  all the attributes in the math class

You can change the name of the class using as keyword

import math as m

to access an attribute(function,variable) in math do m.attribute(followed by () for funtions)

from math import sqrt as s,pi:-
this just imports sqrt function and pi variable with name s and pi and can be used directly

from math import * -imports everything and is not recommended as it may cause name clashes when multiple packages are used

from math import * as m -throws an error

dir(module) returns a list(<class list>) of all the attribute names in module

a file in the same folder behaves as a module with file name as its name

day 45(come back after classes and objects)

if __name__ == "__main__" in Python

In python any code written outside a function in a module is executed automatically when module or any of its attributes are imported(from the module)
__name__ returns a str "__main__" if run normally and if it is run while being imported as a module object the name of the module file is returned
it basically returns the name of the current module that is running(being extracted) and is set to "__main__" by default

day 46 (Read the tutorial on replit to learn file handling after it is taught in class)
Os module
-Used to automate basic file handling tasks
import os
os.mkdir(str) creates a file of name str in the current working directory(not where the file is located but where the terminal is being run) or at the specfied path if the str is a path like "path/filename"
os.path.exists(str) checks if a file of path str exists in the working directory
os.mkdir(source,destination) turns source path into destination path(basically renames the file at the path to file at destination path)
os.listdir(path) returns a list of the contents(entries i.e files and folders) of file at the path

To get contents of folders within folders
```
folders=os.listdir(path)
for folder in folders:
  os.listdir(f"{path}/{folder}")
```

os.system(cmd) runs the cmd in the terminal
os.system("python") enters repl
os.getcwd()returns the full path of current working directory
os.chdir(path) to change the working directory to file of specified path

Use internet for more functions

day 47

Exercise

day 48
Variable scope

Scope of a var in python is similar to other languages.Any locally declared variables will override outside variables but only functions,modules and classes create a new block not loops and if else statements

global x makes it so that the global x is accessed now inside local block and if x is not not defined it is automatically defined when value is assigned as usual

day 49
File handling
f=open("filename","mode"). #mode is r if not specified
mode=r,w,a,rt(by default),rb(read as binary)
f is now a file type object
f.read() returns a string containing contents of file in read(r) mode
f.write(str) writes to the file in write(w) or append mode(a)
f.close() closes the file
whenever file is opened in write mode it erases all content of the file but in append mode it appends
with open(file,mode) as f:
   #f.read/write
This automaticaly closes file after with block is exited

day 50
f=f.open(file,"a") creates a file type object that can be used to interact with the file.Changed made in object reflect the file in real time.Stuff you write is written at the end of file without creating a new line.An internal pointer that keeps pointing to the end of the file is there in the object.A new file is created if it doesnt exist

f=f.open(file,"w") does the same but erases all content before writing and pointer is always at EOF which is also the beginning initially

f=f.open(file,"r") or f=f.open(file) creates a file type object that has an internal pointer initially pointing to the beginning of file
methods for r type file object:
f.read() reads the file from the current position of internal pointer to until pointer reaches EOF and returns it as a string
f.readline() reads the file from from current position to until pointer reaches EOF or \n and returns it as a string

f.write() writes from the location of internal pointer which is EOF
f.writelines(iterable) inserts all the iterations one by one from EOF (without any space or \n btw)

day 51
f.read(int) reads int lines from position of internal pointer
f.seek(int) moves pointer to int (0 to (int-1) chararacters are skipped)
f.tell() returns position of pointer as an integer
f.truncate(int)erases characters from int to EOf so size becomes int

day 52
lamda functions
one line anonymous weird functions
lamda x,y,z: x+y*z returns a function class object that can be stored in a variable or passed as an argument.

This function class object further can be called(callable object) takes arguments of the form (x,y,z) and returns the value of x+y*z
In Python, objects can be made callable by defining the __call__ method within their class
And ykw all functions are objects of function type.Fuck this shit.it's just that when you define functions normally they get binded to the definition name i.e. a variable of the name points to the object now

Example:
```
class Adder:
    def __call__(self, x, y):
        return x + y

# Create an instance of the Adder class
add = Adder()

# Call the object as if it were a function
result = add(3, 5)
```

day 53
Higher order functions
map(function, iterable) gives a iterable map of function(iteration)s which is convertible to list
filter(function,iterable) gives a filter object that conatains all iterations such that function(iteration) is considered true in boolean context
function must take 1 arguments here
from functools import reduce
reduce(function,iterable) repeatedly perfroms the function from left to right on iterations of iterable
function must take 2 arguments here

day 54
is vs ==

a is b iff a and b refer to the same memory location(same object)
a=mutable
b=mutable
a is not b

a=immutable
b=immutable
a is b
For immutable objects python reuses the same object and doesn't waste a new memory location
In functions,both immutable and mutable objects are passed by reference, meaning the function receives a reference to the original object, not a copy of it.

day 55
Exercise
Snake water gun game

Multiple assignment from bro code
a,b,c=x,y,z
a=b=c=d
a,b=b,a(swapping values)

day 56
OOPS
The basic idea of object-oriented programming (OOP) in Python is to use classes and objects to represent real-world concepts and entities.
A class is a blueprint or template for creating objects. It defines the properties and methods that an object of that class will have. Properties are the data or state of an object, and methods are the actions or behaviors that an object can perform.
An object is an instance of a class
encapsulation, means that the internal state of an object is hidden and can only be accessed or modified through the object's methods.This helps protect data safety

inheritance, which allows new classes to be created that inherit the properties and methods of an existing class.

Polymorphism, in the context of object-oriented programming (OOP), refers to the ability of different objects to be treated as instances of a common superclass or interface. It allows methods to be defined in a superclass and overridden by subclasses, providing a way for objects of different classes to be manipulated through a single, uniform interface.
Python supports overriding but not overloading of methods
If you attempt to define multiple methods with the same name in Python, only the last method definition will be used.
however a similar effect to overloading can be achieved using default arguments or variable-length argument lists (*args, **kwargs)

day 57
Classes and Objects
A class is a blueprint or a template for creating objects, providing initial values for state
User defined classes are defined with first letter caps as a convention to avoid naming collisons as all in built ones dont use caps

```
class classname:
  var1=5
  var2=bla bla
  def fun(self):
     bla bla  self.var
  def fun2(a,b)
  var3=bla bla
```

self parameter is a reference to the current instance of the class
to access attributes of class through an object use . notation
To create an object
object=classname()

day 58
Constructor
Constructor is a function that is invoked automatically when an object of a class is created.

```
class classname:
   def __init__(s,a,b):
     s.a=a
     s.b=b #There is no clash as local a and a within s are treated as seperate variable
     return None #This line is not necessary. fn always returns none if nothing is mentioned
classnane.cookie="cookie"
```

No declaration or assignment of values is necessary but is a good practice to do so if a constructor is not used

```
classnane.cookie="cookie"
```

first argument in any function of a class is always a reference to the object even if you dont write self.constructor must always return None

When the constructor accepts arguments along with self, it is known as parameterized constructor.Otherwise it is known as default constructor

day 59
Decorators

```
def dec(funct,*args,**kwargs):
  code
  funct(*args,**kwargs)
  code
@dec basically funct=dec(funct)
def funct():
  bla bla
funct()#this now calls dec(funct)
```

Whenever a function is defined a function type object is created and the name is a pointer to it and py runs line by line

Logging
In programming, logging refers to the process of recording or storing information or messages that help in understanding the behavior and execution of a program

```
import logging
```

can be used with decorators to track function calls and related info

day 60
Getters,Setters and deleters

```
class classname:
 @property
 def fun(self):
   bla bla value
   return something
```

now obj.fun returns the return value of the object and not a function object

```
@fun.setter
```

```
def fun2(self,arg):
  bla bla arg

obj.fun=arg #This runs the setter method with arg as arg

@fun.deleter
def fun3():
  bla bla
del obj.fun #This runs function deleter
```

This system can be used to help encapsulate data

if you use same name for both the getter method will keep calling itself and enter an infinite loop whenever you try to access the value
Usually the variable begins with _(_value) and getter name is same without _ (value)
Trying to access _value directly without getter is not recommended

day 61

```
class inheriter(inherited1,inherited2):
  bla bla
```

all attributes of the inherited are inherited by inheriter and can have extra attributes of its own and can also override attributes

Check day 61 tutorial 2nd page for types of inheritance definitions
1)Single
2)Multiple
3)Multilevel
4)Hierarchial
5)Hybrid

day 62
Access modifiers
By default all variables in class are public in python
By convention
_value is treated as protected(accessebile within the class and its sub classes)
__value is treated as private
Name Mangling:-
__value is changed to _classname__value automatically by compiler to avoid accidental misuse.
dir(obj) returns a list of all the attribute names of the class(functions and variables)

day 63
Exercise 5 solution
Snake water gun game
Watch the video later

day 64
Exercise
Create a library

day 65
Static methods-Just a method that happens to be inside a class to get shipped with the class
Static methods in Python are methods that belong to a class rather than an instance of the class

```
class Class:
  @staticmethod
  def fun():
    bla bla
```

this fun can be called as
Class.fun() or obj.fun()
This fn doesn't take self parameter as argument automatically
obj.fun(obj) must be done to pass the reference
Note:A non static function without passing self can be called by directly using classname but not recommended

day 66
Instance variables:-
va
defined using self.var inside class functions and obj.var outside class
These variabes can override class variables if same name otherwise class variables are accessed if you do obj.var
obj.fun(*args) translates to class.fun(obj,*args) unless the fun is static in which case it translates to class.fun(*args)

class variables:-

defined inside class but outside functions

```
class Class:
  var1=5
  var2=7
```

These variables are shared by all instances commonly and can be accessed inside class functions by Class.var

day 67
library solution

day 68
exercise 7
Write a program to clear the clutter inside a folder on your computer. You should use os module to rename all the png images from 1.png all the way till n.png where n is the number of png files in that folder. Do the same for other file formats.
For example:

sfdsf.png --> 1.png
vfsf.png --> 2.png
this.png --> 3.png
design.png --> 4.png
name.png --> 5.png

day 69
When no decorator
obj.fun() becomes class.fun(obj)
When @staticmethod
obj.fun() becomes class.fun()
When @classmethod
obj.fun() and class.fun() become class.fun(class)

Decorator	Call Syntax	Translation
None (Instance)	obj.fun()	Class.fun(obj)
None (Instance)	Class.fun()	Class.fun()
@staticmethod	obj.fun()	Class.fun()
@staticmethod	Class.fun()	Class.fun()
@classmethod	obj.fun()	Class.fun(Class)
@classmethod	Class.fun()	Class.fun(Class)

day 70
Classmethods as constructors

```
class cls:
@classmethod
def classmethod(cls,args):
  bla bla bla args
  return cls(bla bla args)
```

this modifies the arguments and passes them to the constructor and returns the object created

```
obj=cls.classmethod(args)
```

day 71
Object introspectors
dir(obj): The dir(obj) function returns a list of all the attributes and methods (including dunder methods) available for an object.
__bla bla __ methods are called dunder methods
help(obj): when this fn is called console enters an interactive mode a help documentation for the obj is presented and nothing is printed before or after the documentation until q is pressed
__dict__: The __dict__ attribute returns a dictionary representation of an object's attributes. It is a useful tool for introspection.
This does not exist for builtin classes
Builtin objects have fixed structure defined in C for performance and efficiency reasons and instance variables cannot be directly added

day 72
super keyword
super is a <class type> type callable object
super() returns a super type object which acts like a proxy object of the subsequent class in MRO order because this object dynaically binds to the class i.e whenever any variables or functions are accessed python fetches them from that class and automatically passes the current instance as first variable for instance methods,a reference to current class in class methods and nothing in static methods

All this internal stuff is coded in "Cpython" "C" or even assembly so it can be counterintuitive

Note:An object of child class inherits the parents attributes not the child class itself.they cant be used inside class without super keyword

MRO-Method resolution order determines the order in which classes are searched for an attribute when called
first there is the class itself then the parents in the order they are inherited and so on
 p1(x) p2(y) child(p1,p2) a tuple
now child.__mro__ gives (<class '__main__.child'>, <class '__main__.p1'>, <class '__main__.p2'>, <class 'object'>) tuple
child.mro() gives same but as list
object is always to top even if you explicitly inherit it in a different order
isinstance(obj,object) is always true

day 73
Dunder(double underscore methods) or magic methods:
  These methods allow you to define how objects of your class behave with various built-in operations, such as arithmetic operations, comparisons,function calling and type conversions.
Python automatically calls these methods when certain operations are called
__init__ is called when a new instance is created
__call__ method is called when you try to call the object
__add__ when addition is performed
__str__ when you run str(obj)
__repr__ when you run repr(obj)
When you try to print an object the obj is implicitly converted to str(obj) before printing if __str__ function exists otherwise it is converted to repr(obj)
If neither __str__() nor __repr__() is defined, Python uses the default implementation provided by the base object class, which usually looks like <__main__.ClassName object at 0x...>.
__main.__ implies that we are in the current module otherwise <class 'module.MyClass'>
__repr__ typically returns a string which if executed as code creates another identical object
Note:These dunder methods are typically instance methods but can be otherwise depending on the decorator

day 74
Method overriding
Self explanatory lol

day 75

exercise 7 solution

day 76
Exercise 8

Write a program to manipulate pdf files using pyPDF. Your programs should be able to merge multiple pdf files into a single pdf. You are welcome to add more functionalities

day 77
Operator overloading
-allows developers to redefine the behavior of mathematical and comparison operators for custom data types

__sub__(self1,self2)
  return Class(bla bla self1 self2)
now when an obj of Class is subtracted from another object of class
obj1-obj2 is replaced with Class(bla bla self1 self2) where self1 and self2 are references to the obj1 and obj2 respectively
Similarly __add__, __mul__,__truediv__,__floordiv can be used to override +,-,/,//
visit docs.python for official documentation of anything in python

day 78
Single inheritance
child(parent):
  pass

Now child is just a class identical to parent.Further features can be added or existing features ccan be overriden if required

day 79
Multiple inheritence
child(p1,p2)
if an attribute that is present in both p1 and p2 is called p1 attribute is accessed
MRO-Method resolution order determines the order in which classes are searched for an attribute when called
first there is the class itself then the parents in the order they are inherited and so on
 p1(x) p2(y) child(p1,p2) a tuple
now child.__mro__ gives (<class '__main__.child'>, <class '__main__.p1'>, <class '__main__.p2'>, <class 'object'>) tuple

child.mro() gives same but as list
object is always to top even if you explicitly inherit it in a different order
isinstance(obj,object) is always true

day 80
Multilevel inheritence
parent(grandfather)
child(parent)
through child you can access both the attribute of parent and grand father in method resolution order

day 81
Hybrid inheritance:A mix of more than one type of inheritance
hierarchial inheritance:A hierarchial structure where each class has multiple inheriters and the sub classes have inheriters

day 82
Exercise 8 solution

day 83
Exercise 9

Write a program to pronounce list of names using win32 API.

day 84