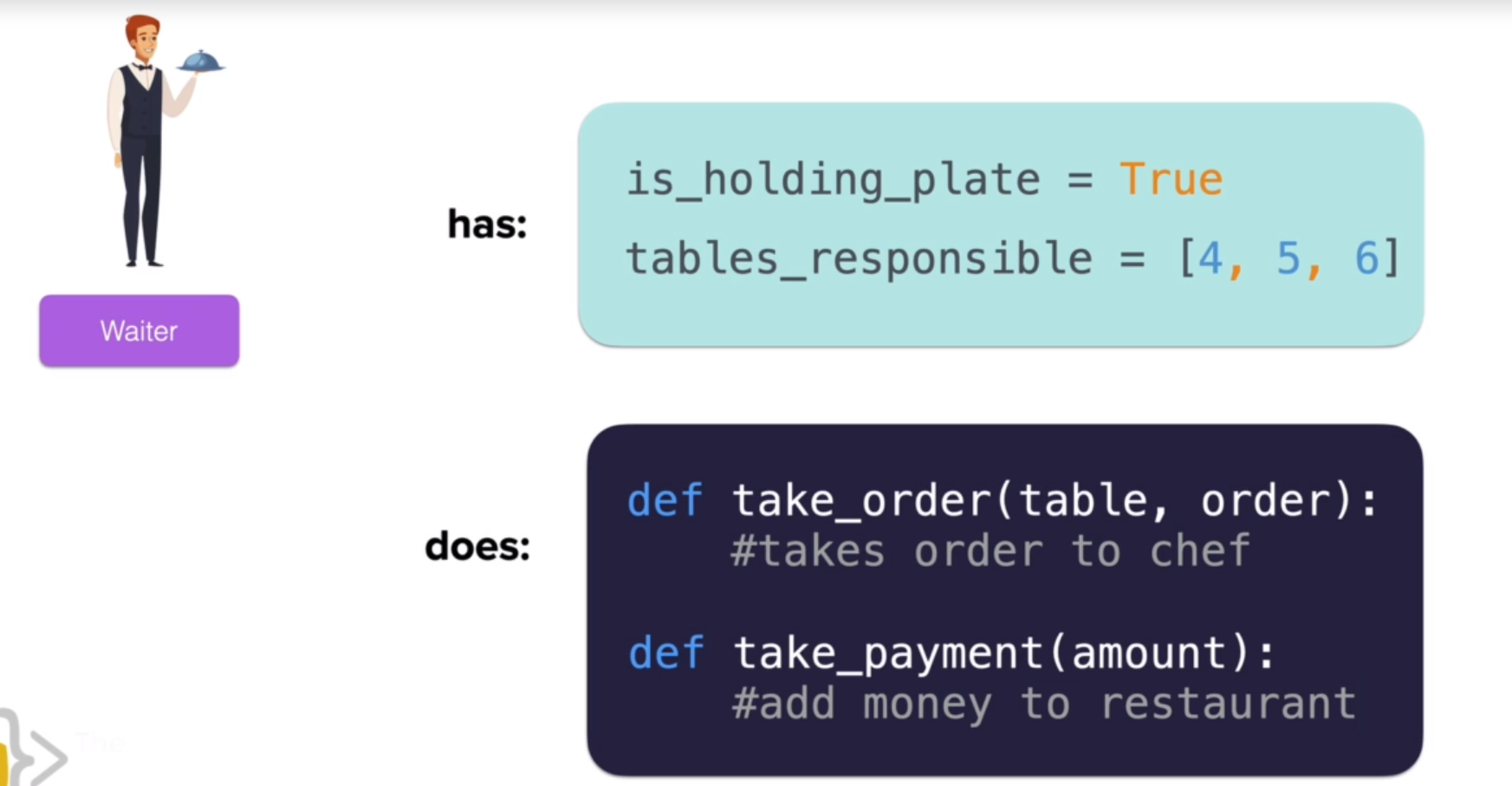
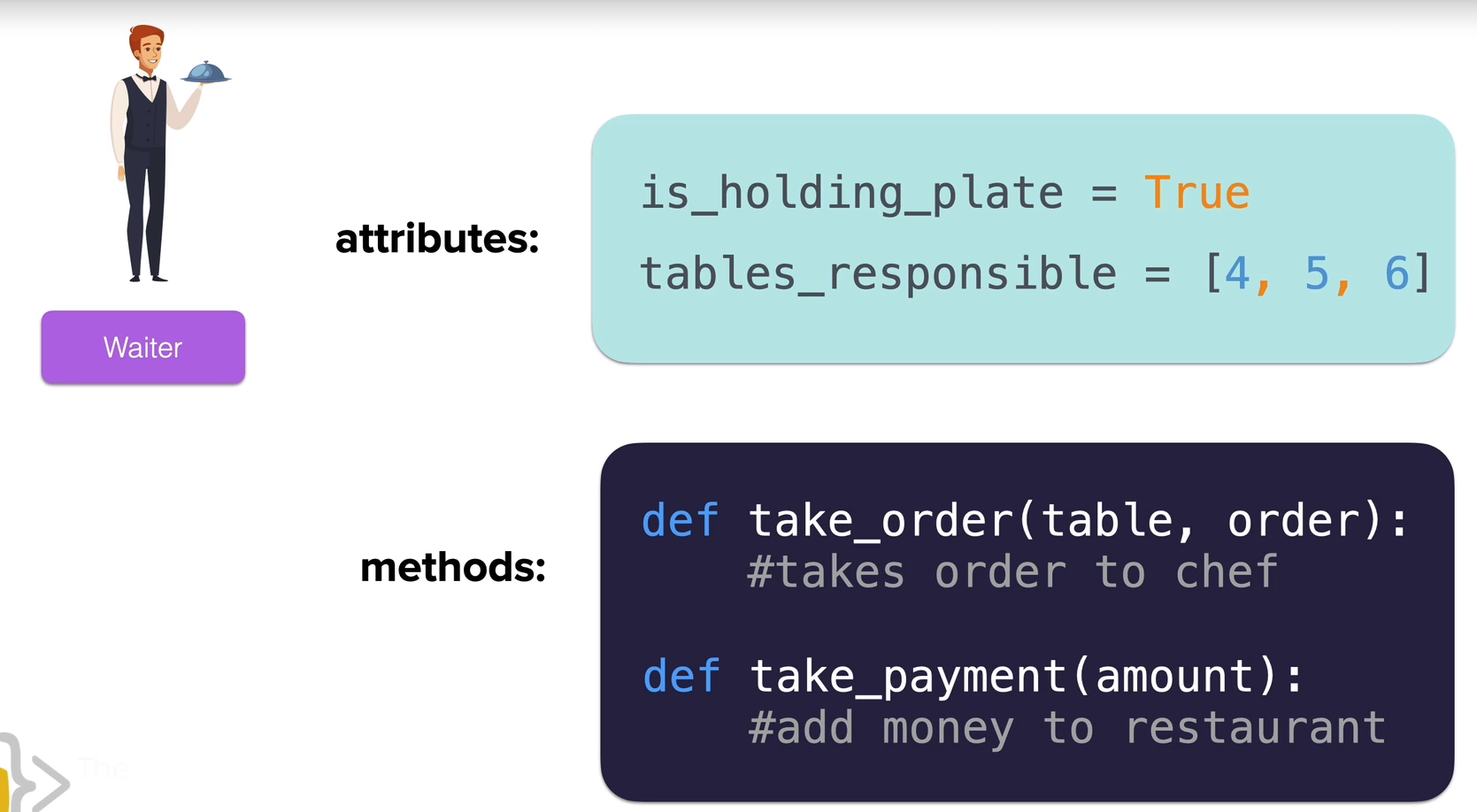
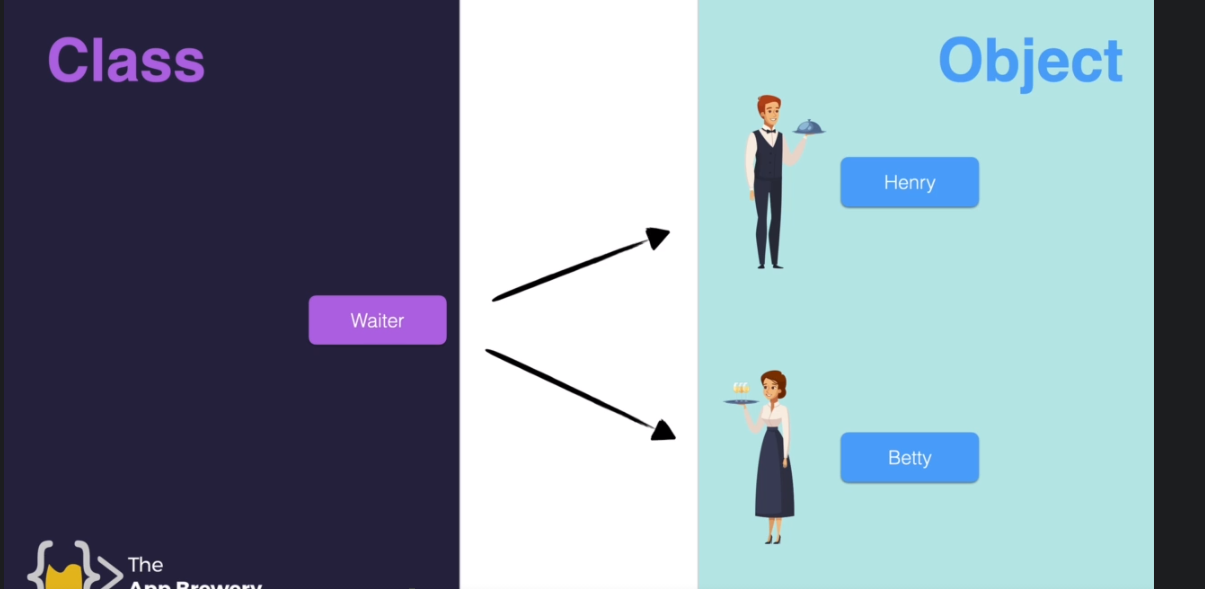
Python Notes

***Object Oriented Programming***

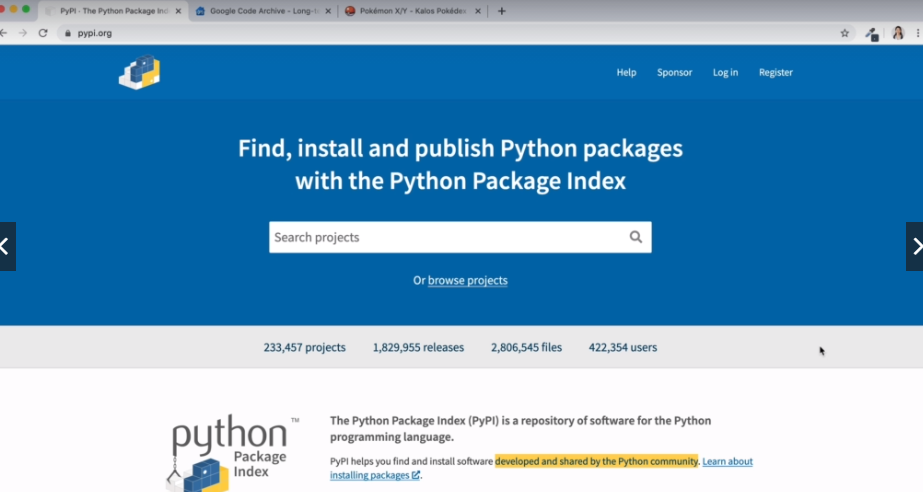






Syntax for importing other modules

From module import class1,class2…



Strings

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

Strings are ordered sequences it means you can use indexing and slicing to grab the strings

Character: h e l l o

Index: 0 1 2 3 4

Reverseindex 0 -4-3-2-1

Slicing grabs subsection of multiple characters

[start:stop:step]

Stop : Index where you will go upto but not include that number

step : Hops it takes

\n - new line

\t - tab space

len() - checks the length of string

Immutability

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

Strings are immutable. You cannot reassign them with whatever you want

Mutable means you can use index/position of element to modify the element. Immutable is the opposite

Concatenate two strings using the '+' sign

a= " Mahesh "

b = " Mammidibathula "

a= " Mahesh "

b= " Mamidibathula "

print(a+b)

Output : Mahesh Mamidibathula

# Objects in python have in built methods. This methods are nothing but functions inside objects

x = 'Hello World'

# Gives everything in upper case

print(x.upper())

Formatting : https://pyformat.info/

Lists

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

Ordered sequences that hold a variety of object types

my\_list = [1,2,3]

my\_list=['String',12,12.45]

len(my\_list)

Lists are mutable

new\_list.append() - to add new item to end of list

new\_list.pop()- by default pop the last element of list but you can specify any number

new\_list.sort()- sort order alphabetically

num\_list.reverse()- It reverses everything in your list

Dictionaries

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

Stores objects in key value pairs

{'key1':'value1','key2':'value2'}

Unorderd and cannot be sorted. You can retrieve an object without knowing index location

Ordered sequence can be indexed or sliced

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

my\_dictionary['key1']=value1

Example is prices in a store.

prices\_lookup={'Apples':456,'Oranges':532,'Milk':960}

d.keys(),d.values(),d.items()

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

Tuples

Tuples are similar to lists but they are not immutable. Once an element in tuple it

cannot be reassigned

tuple: (1,2,3)

t.count(),t.index()

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

Sets

They are unordered collection of unique elements.

It should have unique values.

It never accepts duplicates.

They don't have any particular unique order.

Casting a list to set to get unique values.

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

Booleans

Operators that allow you to convey True or False statements

They are very important with control flow and logic

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

Input/Output files

test.txt

%%writefile myfile.txt

Hello this is a text file

this is the second line

this is third line

myfile=open('myfile.txt')

myfile=open('whoops\_wrong.txt')

myfile.read() - To get the entire text as string.The cursor goes all to the end of line after its done

myfile.seek(0) - to bring the cursor back to first place

contents=myfiles.read()

myfile.seek(0)

myfile.readlines() - You get each lines as a list of objects

File Locations

to open a file in any location give the file path

myfile=open("C:\\Users\\Username\\Folder\\test.txt")

pwd- present working directory

for mac os myfile=open("/Users/Username...)

myfile.close() - to close the file. We have to manually close it.

myfile=open('myfile.txt')

with open('myfile.txt') as my\_new\_file:

contents=my\_newfile.read()

If we using with we dont need to worry about closing the file

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

contents= myfile.read()

r-read only, w-write only, a-append only, r+ - read and write, w+ - read and write

with open('my\_new\_file.txt',mode='r') as f:

print(f.read())

with open('my\_new\_file.txt', mode='a') as f:

f.write('\n Four on forth')

with open('ashdashdashd.txt',mode='w') as f:

f.write('I created this file')

with mode as 'w' it overrides an existing file or creates a new file if not present.

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

Loops

for loop in strings we can access each character in string

for \_ in 'HelloWorld:

print("Mahesh")

tuple unpacking

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

while loops:

while booleancondition:

do something

else:

do something else

break, continue and pass

break: Breaks out of the current closet enclosing loop

continue: Go the top of enclosing closest loop

pass : does nothing at all

x=[1,2,3]

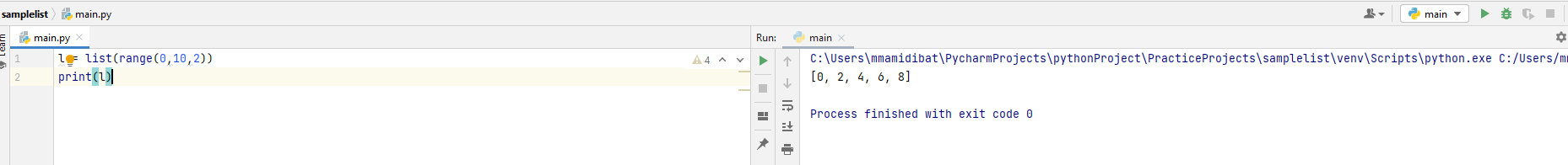
for item in x:

#comment

pass

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

l = list(range(0,10,2))  
print(l)



***Enumerate function***: It gives a tuple as an output.

a=**"Vijayawada"**for item in enumerate(a):  
 print(item)

***Output:***

(0, 'V')

(1, 'i')

(2, 'j')

(3, 'a')

(4, 'y')

(5, 'a')

(6, 'w')

(7, 'a')

(8, 'd')

(9, 'a')

***Zip***: It zips together two lists.

my\_list=[1,10,14]  
my\_list2=[12,45,31]  
for item in zip(my\_list,my\_list2):  
 print(item)

***Output***:

(1, 12)

(10, 45)

(14, 31)

from random import shuffle  
  
l = [1, 2, 3, 4]  
shuffle(l)  
print(l)

***Output***: [3, 1, 2, 4]

Gives some random integer

from random import randint  
  
print(randint(0,10))

***List Comprehensions***

If you find yourself in a scenario where you need to use a for loop with append function its good to use list comprehensions

Mystring = ’hello’

mystring=**'hello'**m=[]  
for l in mystring:  
 m.append(l)  
print(m)

***Equivalent list comprehension***

list\_compr = [l for l in mystring]  
print(list\_compr)

***Program to convert Celsius to Fahrenheit***

l=[0,10,20,30]  
x=[((9/5)\*m+32) for m in l ]  
print(x)

Output : [32.0, 50.0, 68.0, 86.0]

x=[10,20,30]  
y=[100,200,200]  
l=[m\*n for m in x for n in y ]  
print(l)

[1000, 2000, 2000, 2000, 4000, 4000, 3000, 6000, 6000]

***Program to check whether a given number is even in list***

x=[12,14,17,19]  
  
print([y for y in x if y%2 == 0 ])

Object Oriented Programming

def \_\_init\_\_(self, param1,param2):

self.param1=param1

self.param2=param2

def some\_method(self):

print(self.param1)

self is nothing the parameter to object itself

mylist=[1,2,3]

myset=set()

class Dog(): 🡪 Class

def \_\_init\_\_(self, breed) 🡪 This is called upon whenever we create an instance of class

self.breed=breed

mydog= Dog(breed=’lab’) 🡪 Instance of class

type(mydog)

mydog.breed=--- you can see this coming which means breed is attribute

init can be called as constructor of the class, it gets called automatically when we create the class

self represents the instance of object itself.

class Dog():  
 *# This is a method which is called upon when ever there is an instance of class* def \_\_init\_\_(self,name,breed,spots): *#-> This is like a construction which instantiate the attributes of class* self.name=name *#-> attribute* self.breed=breed *#->attribute* self.spots=spots  
  
mydog=Dog(**"Julia"**,**"Golden Retriever"**,True);  
print(mydog.name)  
print(mydog.spots)  
print(mydog.breed)

Class object attribute is same for any instance of class.

Methods are not but the functions inside a class.

***Inheritance and Polymorphism***

Able to create classes based on already created classes

Dog class inherited from animal class

class Animal():  
 def \_\_init\_\_(self):  
 print(**"I am an animal"**)  
 def eat(self):  
 print(**"I am eating"**)  
 def sleep(self):  
 print(**"I am sleepping"**)  
  
class Dog(Animal):  
 def \_\_init\_\_(self):  
 print(**"Dog created"**)  
 def eat(self):  
 print(**"I am a dog and I like eating"**)

Abstract class is one that never expects to be instantiated.

class Book():  
 def \_\_init\_\_(self, name, pages, author):  
 self.name = name  
 self.pages = pages  
 self.author = author  
  
 def \_\_str\_\_(self):  
 return **f"**{self.name} **by** {self.author}**"** def \_\_len\_\_(self):  
 return self.pages  
  
 def \_\_del\_\_(self):  
 print(**f"Object deleted"**)

Traceback (most recent call last):

File "C:\Users\mmamidibat\study\python\python programs\OOPS\main.py", line 20, in <module>

print(len (newbook))

NameError: name 'newbook' is not defined

Linger by Rahin

Object deleted

***Exception Handling in Python***

Try: This is the block of code that needs to be attempted.

Except: Block of code that will be executed if there is an error.

Finally: a block of code that will be executed regardless of error.

try:

result = a +b

except:

print(“Hey it looks like you are not adding correctly”)

finally:

def add(n1,n2):  
 n3=n1+int(n2)  
 print(**f"**{n3}\n**"**)  
  
n1=2  
n2=input(**"Enter a number to add"**)  
  
try:  
 add(n1,n2)  
except:  
 print(**"Something went wrong"**)  
else:  
 print(**"Addition done successfully"**)

def integercheck():  
 while True:  
 try:  
 i=int(input(**"Enter a number"**))  
 except:  
 print(**"Entered number is not a valid entry"**)  
 else:  
 print(**f"Valid number** {i}**"**)  
 break  
integercheck()

Practice Problems:

1. Write a python program to print the employee of month where employees are inserted in form of tuples?

employees = [(**"Abbie"**, 400), (**"Mike"**, 1500), (**"Jenny"**, 390),(**'uppal'**,900)]  
def employee\_best\_check(employees):  
 max\_hours = 0  
 for name, hours in employees:  
 if (max\_hours < hours):  
 new\_name=name  
 max\_hours = hours  
 return (new\_name, max\_hours)  
print(employee\_best\_check(employees))

1. How do you validate whether a user input is a digit or not?

It can be validate using the isdigit() method.

row1=[**''**,**''**,**''**]  
row2=[**''**,**''**,**''**]  
row3=[**''**,**''**,**''**]  
def displaygame(row1,row2,row3):  
  
 print(row1)  
 print(row2)  
 print(row3)  
  
displaygame(row1,row2,row3)  
  
def userinput():  
 input\_validator=**'Wrong'** inputrangevalidator= False  
 inputrange=range(0,10)  
 while input\_validator.isdigit() == False or inputrangevalidator == False:  
 input\_validator = input(**"Enter a valid number between (0-9)"**)  
 if input\_validator.isdigit()== False:  
 print(**"Please enter a valid number"**)  
 else:  
 if int(input\_validator) in inputrange:  
 inputrangevalidator=True  
 return input\_validator  
  
userinput()

1. Write a python program to define circle radius, its area and circumference?

class Circle():

pi: float = 3.14

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

self.radius=radius

self.area=Circle.pi\*self.radius\*\*2

self.circumference=2\*Circle.pi\*self.radius

def volume\_of\_circle(self):

volume=float(4/3\*Circle.pi\*(self.radius\*\*3))

return volume

newcircle=Circle(12)

print(newcircle.area)

print(newcircle.circumference)

print(newcircle.volume\_of\_circle())

1. Program to print the median of a number?

#Program to get the median of a list of numbers

n = int(input("Please enter the number of numbers you wish to enter"))

l = []

for \_ in range(n):

l.append(int(input("Please enter a number")))

l.sort()

median = [l[int(n - 1 / 2)] if n % 2 != 0 else (l[int(n/2)] + l[int(n/2) - 1]) / 2]

print(median.pop())

1. Program to get some 10 questions?

import random

with open('questions.txt', 'r') as myfile:

contents = random.sample(myfile.readlines(), 10)

x = [x.removesuffix('\n') for x in contents]

for \_ in range(10):

y=x.pop()

print(y[y.index(".")+1::])