**PYTHON ASSIGNMENT 1**

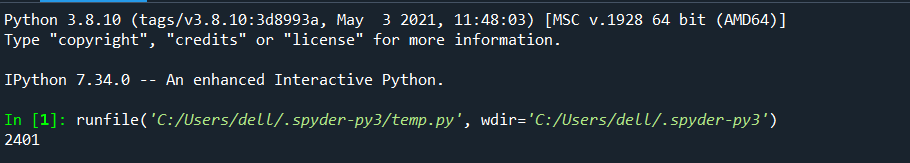
1. What is 7 to the power of 4 ?

CODE:

num = 7

p=7\*\*4

print(p)



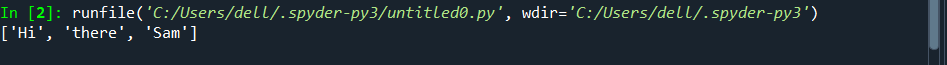
1. Split this string: s = “Hi there Sam!” into a list.

CODE:

s = 'Hi there Sam'

s.split()

print(s.split())



1. Given the variables:planet=”Earth” diameter=12742 Use .format() to print the following string: The diameter of Earth is 12742Km.

CODE:

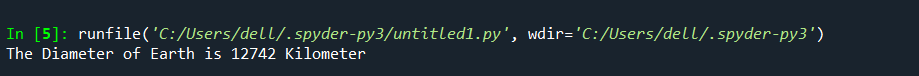
planet='Earth'

Diameter= '12742'

planet.format()

Diameter.format()

print("The Diameter of",planet.format(),"is",Diameter.format(),"Kilometer")



1. Given this nested list, use indexing to grab the world “hello” lst = [1,2,[3,4],[5,[100,200,[‘hello’]],23,11],1,7]

CODE:

lst = [1,2,[3,4],[5,[100,200,['hello']] ,23,11],1,7]

def find\_hello(lst):

for elem in lst:

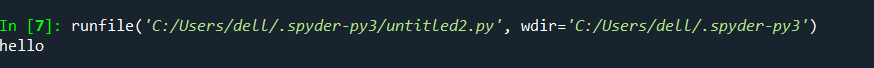
if isinstance(elem,list):

find\_hello(elem)

elif elem=='hello':

print("hello")

find\_hello(lst)



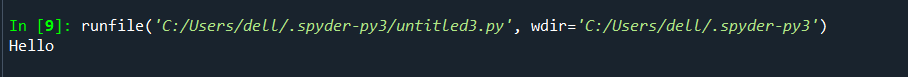
1. Given this nested dictionary grab the world “hello”.d = {‘k1’:[1,2,3,{‘tricky’:[‘oh,’man’,’inception’,’{‘target’:[1,2,3,’hello’]} ]} ]}

CODE:

d={'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'Hello']}]}]}

hello\_word=d['k1'][3]['tricky'][3]['target'][3]

print(hello\_word)



1. What is the main diiference between a tuple and list?



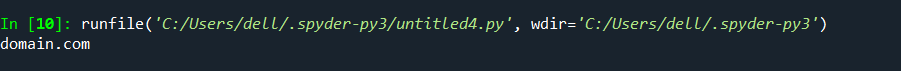
1. Create a function that grab the email website domain from a string in the form: [\*\*user@domain.com](mailto:**user@domain.com). So for example, passing [user@domain.com](mailto:user@domain.com) would return: domain.com

Code :

def domainGet(email):

return email.split('@')[-1]

print(domainGet('user@domain.com'))



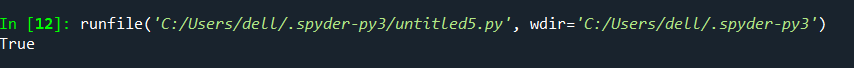
1. Create a basic function that returns True if the word ‘dog’ is contained in the input string. Don’t worry about edge cases like a punctuation being attached to the word dog, but do account for capitalization : findDog(‘Is there a dog here?’)

CODE:

def findDog(st):

return 'dog' in st.lower().split()

print(findDog('Is there a dog here?'))



1. Use lambda expressions and the filter() function to filter out words from a list that don't start with the letter 's'. For example:

seq = ['soup','dog','salad','cat','great']

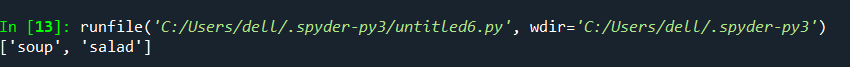
should be filtered down to:

['soup','salad']

CODE:

seq = ['soup','dog','salad','cat','great']

print(list(filter(lambda word: word[0]=='s',seq)))



1. You are driving a little too fast, and a police officer stops you. Write a function to return one of 3 possible results: "No ticket", "Small ticket", or "Big Ticket". If your speed is 60 or less, the result is "No Ticket". If speed is between 61 and 80 inclusive, the result is "Small Ticket". If speed is 81 or more, the result is "Big Ticket". Unless it is your birthday (encoded as a boolean value in the parameters of the function) -- on your birthday, your speed can be 5 higher in all cases.

CODE:

def caught\_speeding(speed, is\_birthday):

if is\_birthday:

speeding = speed - 5

else:

speeding = speed

if speeding > 80:

return 'Big Ticket'

elif speeding > 60:

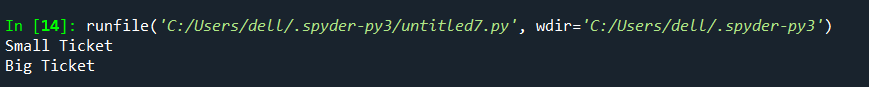
return 'Small Ticket'

else:

return 'No Ticket'

print(caught\_speeding(81,True))

print(caught\_speeding(81,False))



1. Write a program to input roll no, student name, marks of physics, chemistry and maths out of 100. (0-100). Calculate total, percentage, calculate STATUS (pass, fail) if students scores above 40 in all the 3 subjects the STATUS should be pass otherwise fail. Calculate GRADE if STATUS is pass.

Grade must be based on percentage value

if percentage is above 70, then grade must be DISTINCTION if percentage is above 60, then grade must be FIRST CLASS

if percentage is above 50, then grade must be SECOND CLASS if percentage is above 40, then grade must be PASS CLASS

CODE:

def calculate\_grade(percentage):

if percentage > 70:

return "DISTINCTION"

elif percentage > 60:

return "FIRST CLASS"

elif percentage > 50:

return "SECOND CLASS"

elif percentage > 40:

return "PASS CLASS"

else:

return "FAIL"

def main():

roll\_number = input("Enter Roll Number: ")

student\_name = input("Enter Student Name: ")

physics\_marks = int(input("Enter Marks in Physics (out of 100): "))

chemistry\_marks = int(input("Enter Marks in Chemistry (out of 100): "))

maths\_marks = int(input("Enter Marks in Maths (out of 100): "))

# Check if all marks are between 0 and 100

if not(0 <= physics\_marks <= 100) or not(0 <= chemistry\_marks <= 100) or not(0 <= maths\_marks <= 100):

print("Invalid marks. Marks should be between 0 and 100.")

return

total\_marks = physics\_marks + chemistry\_marks + maths\_marks

percentage = (total\_marks / 300) \* 100

# Check status (pass/fail)

status = "PASS" if physics\_marks >= 40 and chemistry\_marks >= 40 and maths\_marks >= 40 else "FAIL"

# Calculate grade if status is pass

grade = calculate\_grade(percentage) if status == "PASS" else None

# Display results

print("\nResults:")

print("Roll Number:", roll\_number)

print("Student Name:", student\_name)

print("Total Marks:", total\_marks)

print("Percentage:", round(percentage, 2))

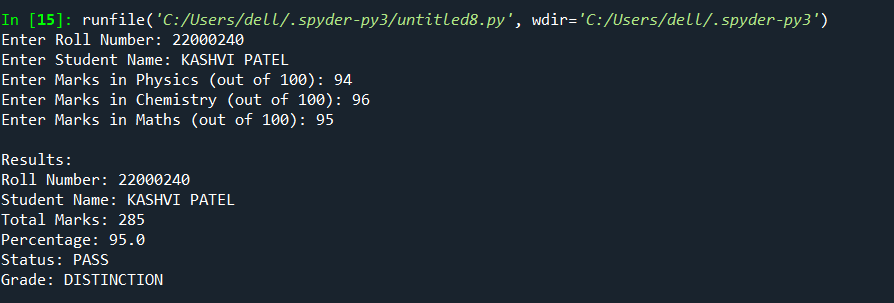
print("Status:", status)

if grade:

print("Grade:", grade)

if \_\_name\_\_ == "\_\_main\_\_":

main()



**ASSIGNMENT 2**

1. Write a OOP in python to input empid, name, basic salary, no. of experience in yrs.

Calculate hra(35% of basic), da (58% of basic) and pf (9.5% of basic).

Also calculate bonus based on experience in years.

If experience in years is >= 30, bonus must be 59% of basic,

If experience in years is >=23, bonus must be 51% of basic,

If experience in years is >=15, bonus must be 45% of basic,

If experience in years is >=7, bonus must be 33% of basic,

If experience in years is <7, bonus must be 16% of basic

Calculate netsalary as basic+da+hra-pf+bonus.

Create a class, constructor to create instance variables, getter-setter foreach variable,

calculative functions for operative variables. A class methods/function should not contain

display specific and input specific code. Such code should be added in driver part of

python program.

CODE:

class Employee:

def \_init\_(self, emp\_id, name, basic\_salary, experience\_years):

self.emp\_id = emp\_id

self.name = name

self.basic\_salary = basic\_salary

self.experience\_years = experience\_years

self.hra = 0

self.da = 0

self.pf = 0

self.bonus = 0

self.net\_salary = 0

def get\_emp\_id(self):

return self.emp\_id

def set\_emp\_id(self, emp\_id):

self.emp\_id = emp\_id

def get\_name(self):

return self.name

def set\_name(self, name):

self.name = name

def get\_basic\_salary(self):

return self.basic\_salary

def set\_basic\_salary(self, basic\_salary):

self.basic\_salary = basic\_salary

def get\_experience\_years(self):

return self.experience\_years

def set\_experience\_years(self, experience\_years):

self.experience\_years = experience\_years

def calculate\_hra(self):

self.hra = 0.35 \* self.basic\_salary

def calculate\_da(self):

self.da = 0.58 \* self.basic\_salary

def calculate\_pf(self):

self.pf = 0.095 \* self.basic\_salary

def calculate\_bonus(self):

if self.experience\_years >= 30:

self.bonus = 0.59 \* self.basic\_salary

elif self.experience\_years >= 23:

self.bonus = 0.51 \* self.basic\_salary

elif self.experience\_years >= 15:

self.bonus = 0.45 \* self.basic\_salary

elif self.experience\_years >= 7:

self.bonus = 0.33 \* self.basic\_salary

else:

self.bonus = 0.16 \* self.basic\_salary

def calculate\_net\_salary(self):

self.net\_salary = self.basic\_salary + self.da + self.hra - self.pf + self.bonus

# Driver code

if \_name\_ == "\_main\_":

emp\_id = input("Enter Employee ID: ")

name = input("Enter Employee Name: ")

basic\_salary = float(input("Enter Basic Salary: "))

experience\_years = int(input("Enter Years of Experience: "))

employee = Employee(emp\_id, name, basic\_salary, experience\_years)

# Calculate allowances and bonus

employee.calculate\_hra()

employee.calculate\_da()

employee.calculate\_pf()

employee.calculate\_bonus()

# Calculate net salary

employee.calculate\_net\_salary()

# Displaying the results

print("\nEmployee Details:")

print("Employee ID:", employee.get\_emp\_id())

print("Name:", employee.get\_name())

print("Basic Salary:", employee.get\_basic\_salary())

print("Experience (years):", employee.get\_experience\_years())

print("\nCalculated Components:")

print("HRA:", employee.hra)

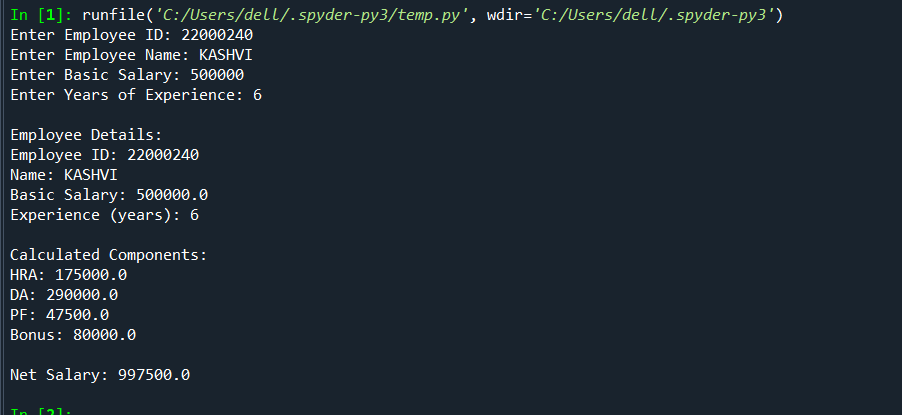
print("DA:", employee.da)

print("PF:", employee.pf)

print("Bonus:", employee.bonus)

print("\nNet Salary:", employee.net\_salary)

OUTPUT:



1. Write an OOP based Python program which inputs n numbers in a list from keyboard. If 8 numbers are inputted, calculate sum of 0th and 7th element and save it in another list say newlist[0], sum of 1st and 6th to newlist[1], sum of 2nd and 5th to newlist[2], sum of 3rd and 4th to newlist[3] and so on. Remove duplicates from newlist if any by converting to set. Later convert it to tuple and display.

CODE:

class NumberProcessor:

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

self.numbers = []

self.newlist = []

if n % 2 != 0:

print("Please input an even number of elements.")

return

self.n = n

def input\_numbers(self):

for i in range(self.n):

num = float(input(f"Enter number {i + 1}: "))

self.numbers.append(num)

def calculate\_and\_create\_newlist(self):

for i in range(self.n // 2):

sum\_pair = self.numbers[i] + self.numbers[self.n - 1 - i]

self.newlist.append(sum\_pair)

def remove\_duplicates\_and\_display(self):

unique\_newlist = list(set(self.newlist))

tuple\_newlist = tuple(unique\_newlist)

print("\nOriginal Numbers List:", self.numbers)

print("New List (Sum of Pairs):", self.newlist)

print("New List without Duplicates (as Tuple):", tuple\_newlist)

if \_\_name\_\_ == "\_\_main\_\_":

n = 8 # You can change this value as needed

processor = NumberProcessor(n)

if n % 2 == 0:

processor.input\_numbers()

processor.calculate\_and\_create\_newlist()

processor.remove\_duplicates\_and\_display()

OUTPUT:

A computer screen with numbers and letters

Description automatically generated

1. Write an OOP program to perform addition, base and power, concatenation, max, min of two numbers stored in two different objects created from same class. n1=MyNumber(2) n2=MyNumber() n2.setNum(5) n3=n1.add(n2) print("Addition is ",n3.getNum()) #7 n3=n1.raisedTo(n2) print(n1.getNum()," raised to ",n2.getNum()," is ",n3.getNum()) #32 n3=n1.concat(n2) print("Concat answer is ",n3.getNum()) #25 n3=n1.max(n2) print("Max is ",n3.getNum())

CODE:

class MyNumber:

def \_\_init\_\_(self, num=0):

self.num = num

def set\_num(self, num):

self.num = num

def add(self, other):

result = MyNumber(self.num + other.num)

return result

def raised\_to(self, other):

result = MyNumber(self.num \*\* other.num)

return result

def concat(self, other):

result = MyNumber(int(str(self.num) + str(other.num)))

return result

def max(self, other):

result = MyNumber(max(self.num, other.num))

return result

def min(self, other):

result = MyNumber(min(self.num, other.num))

return result

def get\_num(self):

return self.num

# Example usage

n1 = MyNumber(2)

n2 = MyNumber()

n2.set\_num(5)

n3 = n1.add(n2)

print("Addition is", n3.get\_num()) # 7

n3 = n1.raised\_to(n2)

print(n1.get\_num(), "raised to", n2.get\_num(), "is", n3.get\_num()) # 32

n3 = n1.concat(n2)

print("Concat answer is", n3.get\_num()) # 25

n3 = n1.max(n2)

print("Max is", n3.get\_num()) # 5

OUTPUT:

A computer screen with text

Description automatically generated