Principal amount is 65000 , rate of interest is 8 % . Tenure is 4 years. Find the maturity amount . Take inputs information from the user.

In [ ]:
```
P=float(input('enter the principal amount - '))
R=float(input('enter the rate of interest - '))
T=float(input('enter the time period in years - '))
Interest_Amount= (P*R*T)/100
Maturity_Amount= P + Interest_Amount
print(Maturity_Amount)
```

I am organizing a party for adults . Print 'true' or 'false' when user enters the age.

In [ ]:
```
age=int(input('enter the age - '))
print (age >= 18)
```

Write a program to check the dress code is true or false for a freshers competition. girl should wear _ and boys should wear__ .(we can form different questions with logical operators)

In [ ]:
```
girls=input('enter the dress color for girls - ')
boys=input('enter the dress color for boys - ')
girls=='black' and boys =='white'   # if both conditions are true the True
girls=='black' or boys =='orange'   # anyone of them is true then return True
girls=='black' and (not boys =='white') # not boys ==white meaning boys should
not wear white
girls=='black' or (not boys =='white') # delivers true if any one is satisfied
```

write a python program to calculate the hypotenous of right-angled triangle when the sides are given.

In [ ]:
```
import math                          #we have imported the math library over he
re

base = float(input('enter the base - '))
height=float(input('enter the height - '))
hypt = math.sqrt((base**2 + height**2))
print(hypt)
```

Euclidean division program using divmod function

write a program to show 8594 in h(hours) m(minutes) and s(seconds)

In [ ]:
```
raw_time=8594
minutes,seconds=divmod(raw_time,60)          #here we do 8594/60 which gives
 no of minutes in quotient & seconds in remainder
hours,minutes=divmod(minutes,60)             #we divide minutes/60 to get no
 of hrs and remaining minutes
print(f"{raw_time}s is {hours}h {minutes}m {seconds}s")
```

Nested string interpolation in python example : example 1

In [ ]:
```
name="Jose"
print(f"Hello,{name}")
```

example 2

In [1]:
```
name="vishal"
print("Hello,{}".format(name))
```

```
Hello,vishal
```

Write a program to take name and age as input and print a sentence.

In [ ]:
```
name=input('enter your name - ')
age=input('enter your age - ')
print("My name is",name,"and I am",age,"years old.")
```

if statement basic examples -1)WAP to print the typed number by the user if it is an odd number otherwise do nothing.

In [ ]:
```
num=int(input('enter the number -'))
if num%2!=0:
    print(num)
```

2)write a program to print the number if it is less than 10.

In [ ]:
```
num=int(input('enter the number - '))
if num<10:
    print(num)
```

if else basic examples - 1)WAP to check the given number is even or odd.

In [ ]:
```
num=int(input('enter a number - '))
if num%2==0:
    print(num,'is even number')
else:
    print(num,'is odd number')
```

WAP to check whether the given name of a person is in the list or not.

In [ ]:
```
group=('naveen','vishal','rahul','shekar','hemanth','srujan')
candidate=input('enter the name of the candidate - ')
if candidate in group:
    print(candidate,'belongs to the group')
else:
    print(candidate,'is not from the group')
```

WAP to obtain the marks of the students in maths , physics ,chemistry ....calculate the percentage and check whether the student has met the pass criteria.(pass criteria is more than 40% or equal to 40%)

In [ ]:
```
maths=float(input('enter the marks in mathematics - '))
physics=float(input('enter the marks in physics - '))
chemistry=float(input('enter the marks in chemistry - '))
percentage=(maths+physics+chemistry)/300*100
if percentage>=40:
    print('This candidate is eligible !!')
else   :
    print('This candidate has failed .')
```

WAP to toss the coin and guess its outcome . If the outcome is equal to guess player win or else player fails.

In [ ]:
```
guess=input('select heads or tails : ')
outcome='heads'
if guess==outcome:
    print('You won the toss')
else :
    print('you lost the toss')
```

One liner if else statement 1)write a program to check if the number is odd or even

In [ ]:
```
number=int(input('enter a number-'))
print('even number'if number%2==0 else 'odd number')
```

2)WAP to print the length of the input if the length exceeds more than 5 characters.

In [ ]:
```
word=input("please enter a word - ")
print(len(word) if len(word)>5 else word)
```

random module in python

In [ ]:
```
import random as r      #importing a random module(library)
r.random()
```

In [ ]:
```
r.randint(5,10)
```

In [ ]:
```
r.choice([5,4,7,8,88])
```

In [ ]:
```
r.choice(['ab','b','c666','5der','e0.5'])
```

In [ ]:
```
r.choice('abcde')
```

In [ ]:
```
r.choice('a,b@cd')
```

WAP to check whether you have won the coin toss or not. If you have won the toss print won the toss else print lost the toss.

In [ ]:
```
guess=input('guess heads or tails : ')
import random as r
outcome=r.choice(['heads','tails'])
print(outcome)
if guess==outcome:
    print('You won the toss!!')
else :
    print('You lost the toss.')
```

Modify the code to return a statement 'Invalid Input Given' is a person enters invalid guess other than heads and tails in a coin toss.

In [ ]:
```
guess=input('Guess heads or tails :')
options=['heads','tails']
if guess in options:
    import random as r
    outcome=r.choice(['heads','tails'])
    if guess==outcome:
        print('You won the toss!')
    else :
        print('You lost the toss!')
else:
    print('Invalid Input Given')
```

Write a program to check if the number is even . If it is an even number check if the number is divisible by 5.

In [ ]:
```
num=int(input('Enter a number : '))
if num%2==0:
    print(num)
    if num%5==0:
        print('number is divisible by 5' )
    else :
        print('number is not divisible by 5')
else :
     print('input is an odd number')
```

7 up and 7 down

Write a program to simulate below mentioned scenario:

1)Player enters game with initial amount as Rs. 1000/- 2)Generate a random variable 1 to 14 and store it in a variable (outcome). 3)If outcome=7,player hits a jackpot and wins Rs.10000000. 4)If outcome<7,player loses amount by (outcome*100) 5)If outcome>7,player earns amount by (outcome*100) 6)Print the final amount with the player.

In [ ]:
```
amount=1000
import random as r
outcome=r.randint(1,14)
print('Your score is',outcome)
if outcome<7:
    amount-=outcome*100
elif outcome>7:
    amount+=outcome*100
else:
    outcome==7
    amount+=10000000
print('final amount=',amount)
```

Write a program to take percentages as inputs and print the grades.

percentage>75 - A 60<percentage<75 - B 40<percentage<60 - C 40>percentage - D

In [ ]:
```
marks=int(input('enter the marks of the candidate : '))
if marks > 75 :
    print('A Grade')
elif 60 < marks < 75 :
    print('B Grade')
elif 40 < marks < 60 :
    print('C Grade')
else :
    print('D Grade')
```

Write a program to accept 3 numbers from the user and print the largest number .

In [ ]:
```
num1=int(input('Enter the first number : '))
num2=int(input('Enter the second number : '))
num3=int(input('Enter the third number : '))
if num2<num3>num1:
    print(num3,'is the largest number')
elif num1<num2>num3:
    print(num2,'is the largest number')
else:
    print(num1,'is the largest number')
```

Write a program to accept hours and rate per hour from the user and compute gross pay . Pay the hourly rate for the hours upto 40 and 1.5 times the hourly rate for all hours worked above 40hrs.

In [ ]:
```
hrs=float(input('enter the number of hours'))
rate=float(input('enter the rate per hour'))
if hrs<=40:
    gross_pay=hrs*rate
else:
    exceed_hrs=hrs-40
    exceed_rate=exceed_hrs*rate*1.5
    gross_pay=40*rate+exceed_rate
print(gross_pay)
```

WAP to accept single character from the user and check if it is vowel or not .

In [ ]:
```
ch=input('enter a character : ')
if ch in ('a','e','i','o','u'):
    print(ch,'is a vowel')
else :
    print(ch,'is a consonant')
```

WAP to check if the number is positive number or negative number or zero.

In [ ]:
```
num=int(input('enter the number : '))
if num<0 :
    print('negative number')
elif num>0 :
    print('positive number')
else :
    print('zero')
```

while loop and for loop examples :-

While loop examples

Print numbers 0-9 but that doesnt include 9.

In [9]:
```
count = -5    #defined a variable and initialized a value 0 to it

while count<9:  #this means the program should execute until the count<9, the
 moment it is more than 9 , it should come out of the loop.
    print("Number :",count) #print statement that will print the count value
    count=count+1

    #we have increased the count by 1

print("Good Bye")
```

```
Number : -5
Number : -4
Number : -3
Number : -2
Number : -1
Number : 0
Number : 1
Number : 2
Number : 3
Number : 4
Number : 5
Number : 6
Number : 7
Number : 8
Good Bye
```

There will be a random guessing game where we need to guess the numbers between 0 and 20 .

(print number is too small) (number is too large) (print Exit:Congratulations you have made it)

In [11]:
```
count = "vishal"  #defined a variable and initialized a value 0 to it
i=0
while count=="vishal":  #this means the program should execute until the count
<9, the moment it is more than 9 , it should come out of the loop.
    print("Number :",count)
    print(i)
    i=i+1#print statement that will print the count value
    count=count.upper()      #we have increased the count by 1
    print("Good Bye")
```

```
Number : vishal
0
Good Bye
```

In [ ]:
```
import random as r
n=20#because number should be in the range 0 to 20
to_be_guessed = int(n * r.random()) + 1 #right number+
guess = 0
while guess != to_be_guessed:
        guess = int(input("New number : "))
        if guess > 0 :
            if guess > to_be_guessed:
                print("number too large")
            elif guess < to_be_guessed:
                print("number is too small")
        else:
            print("Sorry that you are giving up!")
            break
else:
    print("Congratulations . You made it!")
```

In [16]:
```
for i in range(1,2,5):
    print(i)
```

```
1
```

For loop examples

In [ ]:
```
fruits = ['Mangoes' , 'Grapes' , 'Apples']

for fruit in fruits :
    print("current fruit:",fruit)

print("Good Bye")
```

In [23]:
```
tup=("asd","qwe","123")
dic={1:"as","a":"qwe",1.2:"qweqweqwe"}


for i in dic:
    print(dic[i])
```

```
as
qwe
qweqweqwe
```

Code to find the factorial of a number.

In [ ]:
```
num=int(input("Number:"))
factorial = 1

if num < 0:
    print("must be positive")
elif num == 0:
    print("factorial = 1")
else :
    for i in range(1,num + 1):# for i in range (1,6)
            factorial = factorial*i
print(factorial)
```

Write a program to accept the characters from a user till he enters "q". Print string of all the characters entered by the user .

In [26]:
```
ch = ""
string = ""
while ch != "q":
    ch=input("Enter a character")
    string += ch

print(string)
```

```
Enter a characterasdad
Enter a charactera
Enter a charactera
Enter a characters
Enter a characterq
134asfasdadaasq
```

Initialize n = 100 , iteratively divide n by 3 till n !=0 .

In [1]:
```
n=100
counter=0
while n!=0 :
    n = n//3
    counter+=1
    print(n)

print(counter)
```

```
33
11
3
1
0
5
```

Write a program to count the number of digits in a number entered by the user .

In [ ]:
```
num=int(input('enter a number : '))
count = 0
while num != 0 :
    num = num // 10
    count += 1
print(count)
```

Write a program to modify the 7up and 7down game , keep playing the game till user decides to quit or he is out of balance.

In [3]:
```
import random as r
amt = 1000
choice = "yes"

while choice == "yes" and amt > 0 :

    outcome = r.randint(1,14)
    print("your score - ", outcome)

    if outcome < 7 :
        amt -= (outcome * 100)
    elif outcome > 7 :
        amt += (outcome * 100)
    else:
        print("you have hit the jackpot !!!")
        amt += 1000000
    print('amount with you-',amt)

    choice = input("Do you wish to continue enter yes else no -")
```

```
your score -  4
amount with you- 600
Do you wish to continue enter yes else no -yes
your score -  9
amount with you- 1500
Do you wish to continue enter yes else no -yes
your score -  2
amount with you- 1300
Do you wish to continue enter yes else no -yes
your score -  10
amount with you- 2300
Do you wish to continue enter yes else no -yes
your score -  1
amount with you- 2200
Do you wish to continue enter yes else no -yes
your score -  5
amount with you- 1700
Do you wish to continue enter yes else no -yes
your score -  9
amount with you- 2600
Do you wish to continue enter yes else no -yes
your score -  5
amount with you- 2100
Do you wish to continue enter yes else no -yes
your score -  2
amount with you- 1900
Do you wish to continue enter yes else no -yes
your score -  11
amount with you- 3000
Do you wish to continue enter yes else no -yes
your score -  9
amount with you- 3900
Do you wish to continue enter yes else no -yes
your score -  1
amount with you- 3800
Do you wish to continue enter yes else no -no
```

Write a program to take a word as an input and print all the characters.

In [7]:
```
word = input("enter a word - ")

for ch in word :
    print(ch)

len()
```

```
enter a word - vishal
v
i
s
h
a
l

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-7-bd5241f67a5b> in <module>
      4     print(ch)
      5
----> 6 len()

TypeError: len() takes exactly one argument (0 given)
```

WAP to print the product of all the elements in a list = [4,5,7,8,1,2,6,2,2]

In [11]:
```
numbers = [4,5,7,8,1,2,6,2,2]
product = 1

for i in numbers :
    product = product * i

print(product)
```

```
26880
```

Write a program to print all the even numbers from the above list.

In [12]:
```
numbers = [4,5,7,8,1,2,6,2,2]

for i in numbers :
    if i % 2 == 0:
        print(i)
```

```
4
8
2
6
2
2
```

Print sum of even numbers and odd numbers from the above list.

In [13]:
```
numbers = [4,5,7,8,1,2,6,2,2]

sum_even = 0
sum_odd = 0

for i in numbers :
    if i%2==0 :
        sum_even += i
    else:
        sum_odd += i
print(sum_even , sum_odd)
```

```
24 13
```

WAP to reverse the number 1243.

In [14]:
```
num = 1243
rev = 0
while num != 0 :

    d = num % 10
    rev = rev * 10 + d
    print(rev)
    num //= 10
```

```
3
34
342
3421
```

Print the number of days as weeks and number of days.

In [15]:
```
days = 15

weeks = days // 7

d = days % 7

print(weeks , d)
```

```
2 1
```

Product and summation of first 10 natural numbers .

In [16]:
```
numbers = [1,2,3,4,5,6,7,8,9,10]
product = 1
total = 0

for i in numbers :
    product *= i
    total += i
print(total , product)
```

```
55 3628800
```

Product of first 10 natural numbers

In [18]:
```
product = 1

for i in range(1,11):
    product *= i

print(product)
```

```
3628800
```

WAP to accept an integer from the user and print a table for the given number .

In [19]:
```
num = int(input('enter a number : '))

for i in range(1,11):
    print(num,'x',i,'=',(num * i))
```

```
enter a number : 6
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
```

WAP to print prime numbers between 1 - 100

In [25]:
```
import math as m

for num in range(1,101) :
    for i in range(2,int(m.sqrt(num))+1):
        if num %i == 0:
            break


    else :
        print(num,end = " ")
```

```
1 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Write a program to find the first occurence of a vowel in a word

In [ ]:
```
word = 'barbeque'

for i in word :
    if i in "aeiou" :
        print(i)
        break
```

WAP to take 5 numbers from the user and to calculate the average.

In [26]:
```
print("Enter 5 numbers")

i = 0 # to iterate the loop

s = 0 # intializing the value to 0 to calculate the sum

while i < 5:

   n = int(input()) # taking input as integer

   s = s + n # modifying the variable to calculate the sum after every input

   i += 1 # incrementing the loop counter

avg = s / 5 # calculating the average of the values

print("The sum is = ",s)

print("The average is =",avg)
```

```
Enter 5 numbers
80
45
62
70
100
The sum is =  357
The average is = 71.4
```

Write a program to print the fibonacci series starting from 0,1 upto 10 terms.

In [27]:
```
# Program to display the Fibonacci sequence up to n-th term

nterms = int(input("How many terms? "))

# first two terms
n1, n2 = 0, 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
   print("Please enter a positive integer")
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
else:
   print("Fibonacci sequence:")
   while count < nterms:
       print(n1)
       nth = n1 + n2
       # update values
       n1 = n2
       n2 = nth
       count += 1
```

```
How many terms? 10
Fibonacci sequence:
0
1
1
2
3
5
8
13
21
34
```

In [24]:
```
n=0
m=1
sum=m
print(n)
print(m)
for i in range(n+m,10):
    sum=sum
    print(sum)
```

```
0
1
3
6
10
15
21
28
36
45
```

Write a program to check the entered string is a palindrome or not .

In [3]:
```
string = input('enter a string :-')
if string == string[::-1]:
    print('It is a palindrome')
else:
    print('not a palindrome')
```

```
enter a string :-noon
It is a palindrome
```

Programs on SEQUENCE (example programs for understanding)

Lists

In [ ]:
```
list_1 = list_2 = [1,2,3,4] # both names will point to the same memory locatio
n
list_1 = []
list_1 = list_2 # both names will point to the same list
list_1 = [1,2,3,4]
list_2 = [1,2,3,4] # independent lists
```

In [1]:
```
# nested list
my_list = [['Data Science','Machine Learning'],[135,232,321]]
my_list
```

Out[1]:
```
[['Data Science', 'Machine Learning'], [135, 232, 321]]
```

In [2]:
```
# define a list
course = ['data science', 'machine learning', 'python', 'html', 'big data' ]
# access the second item of a list at index 1
print(course[2])
```

```
python
```

In [3]:
```
# define a list
course = ['data science', 'machine learning', 'python', 'html', 'big data' ]
# change the third item
course[3] = 'statistics'
course
```

```
['data science', 'machine learning', 'python', 'statistics', 'big data']
```

list.append(item)

The method list.append(item) will add the element at the end of a list

In [13]:
```
# define a list
course1 = ['data science', 'machine learning', 'python', 'html', 'big data' ]
new_val = 'statistics'
# add element to the list
course1.append(new_val)
print(course1)
```

```
['data science', 'machine learning', 'python', 'html', 'big data', 'statistic
s']
```

list.insert(i, item)

This method will insert an element at the i index in a list

In [14]:
```
# define a list
course = ['data science', 'machine learning', 'python', 'html', 'big data' ]
# insert element at 2nd index in a list
course.insert(2, 'statistics')          #first argument is for the position of
the index
print(course)
```

```
['data science', 'machine learning', 'statistics', 'python', 'html', 'big dat
a']
```

list.extend(items)

The extend method concatenates lists

In [15]:
```
# define a first list
course1 = ['data science', 'machine learning', 'python', 'html', 'big data']
# define a second list
course2 = ["abcd", "India", "Mumbai"]
# concatenate the list
course1.extend(course2)
print(course1)
```

```
['data science', 'machine learning', 'python', 'html', 'big data', 'abcd', 'I
ndia', 'Mumbai']
```

list.sort(key, reverse)

The sort method sorts a list in-place without creating a new object

In [16]:
```
# declare a numeric list
income = [2500, 25000, 10000, 50000, 20000, 5000, 17500]
# sort() function is used to sort the numeric values in the list in ascending
 order
income.sort()
income
```

Out[16]:
```
[2500, 5000, 10000, 17500, 20000, 25000, 50000]
```

In [17]:
```
# set the reverse parameter to False to arrange the elements in the descending
order
income.sort(reverse = True)
income
```

Out[17]:
```
[50000, 25000, 20000, 17500, 10000, 5000, 2500]
```

In [19]:
```
# We can also sort a collection of strings by their length using the key param
eter
sentence = ['a', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'fro
g']
sentence.sort(key = max)
sentence
```

Out[19]:
```
['a', 'frog', 'the', 'quick', 'jumps', 'over', 'brown', 'fox', 'lazy']
```

Deleting List Elements

Use the del keyword to delete an item at specific index

In [46]:
```
# define a first list
course = ['data science', 'machine learning', 'python', 'html', 'big data' ]
# delete the third element from the list
del(course[2])
# print the output
course
```

Out[46]:
```
['data science', 'machine learning', 'html', 'big data']
```

In [6]:
```
# define a first list
course = ['data science', 'machine learning', 'python', 'html', 'big data' ]
# delete multiple items from the list
del course[0:2]
# print the output
course
```

Out[6]:
```
['python', 'html', 'big data']
```

list.clear()

The clear method resets the list to an empty state

In [22]:
```
sentence = ['a', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'fro
g']
sentence.clear()
sentence
```

Out[22]:
```
[]
```

list.remove(item)

It will search and remove only the first occurrence of an item

In [11]:
```
# define a first list
course = ['data science', 'machine learning', 'python', 'html', 'big data', 'h
tml' ]
course.remove('python')
course
```

Out[11]:
```
['data science', 'machine learning', 'html', 'big data', 'html']
```

list.pop()

Removes and returns the last item of a list

In [57]:
```
# define a first list
course = ['data science', 'machine learning', 'python', 'html', 'big data', 'h
tml' ]
course.pop(1)
course
```

Out[57]:
```
['data science', 'python', 'html', 'big data', 'html']
```

list.reverse()

It reverses the order of the items in a list

In [30]:
```
# define a first list
course = ['data science', 'machine learning', 'python', 'html', 'big data', 'h
tml' ]
course.reverse()
course
```

Out[30]:
```
['html', 'big data', 'html', 'python', 'machine learning', 'data science']
```

Lower , Upper and Title case

In [16]:
```
string1 = 'vishal'
print(str.upper(string1))
```

```
VISHAL
```

In [35]:
```
print(str.lower(string1))
```

```
vishal
```

In [36]:
```
print(str.title(string1))
```

```
Vishal
```

Reassigning means over writing the value

In [17]:
```
l2=[4,7,6,8,9]
l2[2]= 56
print(l2)
```

```
[4, 7, 56, 8, 9]
```

To find the maximum and minimum characters

In [45]:
```
l1 = [6,8,3,4,9,10,45]
print(max(l1))
print(min(l1))
```

```
45
3
```

In [58]:
```
import copy
l1 = [1,2,3,4, [5,6]]
l2 = copy.copy(l1) # Shallow copy
print(l1,l2)
l2[0] = 100
l2[4][1] = 10
print(l1,l2)
```

```
[1, 2, 3, 4, [5, 6]] [1, 2, 3, 4, [5, 6]]
[1, 2, 3, 4, [5, 10]] [100, 2, 3, 4, [5, 10]]
```

In [59]:
```
import copy
l1 = [1,2,3,4, [5,6]]
l2 = copy.deepcopy(l1) # deep copy
print(l1,l2)
l2[0] = 100
l2[4][1] = 10
print(l1,l2)
```

```
[1, 2, 3, 4, [5, 6]] [1, 2, 3, 4, [5, 6]]
[1, 2, 3, 4, [5, 6]] [100, 2, 3, 4, [5, 10]]
```

Tuples

objects inside the tuple are immutable(cannot be changed)

In [60]:
```
# define the tuple
my_tuple = ("mango", "yellow", "green", "blue", 353, 363.2, 'w')
# changing the element of tuple
# This throws an error since tuple elements are immutable
my_tuple[0] = 'orange'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-60-95f019197f11> in <module>
      3 # changing the element of tuple
      4 # This throws an error since tuple elements are immutable
----> 5 my_tuple[0] = 'orange'

TypeError: 'tuple' object does not support item assignment
```

In [18]:
```
my_tuple = (123, ['s', 'a', 'v'], "World")
print(my_tuple)
# changing the element of the list
# this is valid because list is mutable
my_tuple[1][0] = 9   # 1 is for the positional value inside the tuple and 0 is for the list inside the tuple
my_tuple
```

```
(123, ['s', 'a', 'v'], 'World')
```

Out[18]:
```
(123, [9, 'a', 'v'], 'World')
```

Python has built-in methods which can be used on tuples:

count() index()

count() It returns the number of times a specified element occurs in a tuple

index() It searches the tuple for the first occurance of a specified element and returns the index value for that particular position

In [68]:
```
my_tuple = ("u",'a','p','p','l','e','e','d','e', 'd','e','a','c', "u",'w')
my_tuple.count('e')
```

Out[68]:
```
4
```

Adding value to a tuple

In [2]:
```
mix_tuple = (['a', 1, True], 2, 'Science', -5)
mix_tuple=mix_tuple+(4,)
print(mix_tuple)
```

```
(['a', 1, True], 2, 'Science', -5, 4)
```

Dictionaries

Examples

In [69]:
```
# create dictionary
balance = {
 "Mia" : 83847,
 "John" : [83837, "abc"],
 "Jill" : 94766
}
print(balance)
```

```
{'Mia': 83847, 'John': [83837, 'abc'], 'Jill': 94766}
```

We can use get method to get the value of "Mia" key

In [70]:
```
x = balance.get("Mia")
x
```

Out[70]:
```
83847
```

Changing values in a dictionary

In [71]:
```
# create a dictionary
year_sales = {
 2015: 34500,
 2016: 34300,
 2017: 40000
}
year_sales
```

Out[71]:
```
{2015: 34500, 2016: 34300, 2017: 40000}
```

In [72]:
```
# Change the sales figure for 2015
year_sales[2015] = 45000
year_sales
```

Out[72]:
```
{2015: 45000, 2016: 34300, 2017: 40000}
```

Dictionary length

In [73]:
```
# print the length of dictionary
print(len(year_sales))
```

```
3
```

Adding Items

Add an item to the dictionary by using a new index key and assign a value to it

In [74]:
```
# create a dictionary
year_sales = {
 2015: 34500,
 2016: 34300,
 2017: 40000
}
year_sales[2018] = 55000
year_sales
```

Out[74]:
```
{2015: 34500, 2016: 34300, 2017: 40000, 2018: 55000}
```

Dictionary Method

Removing Items Following are the methods to remove items from a dictionary: keys() method values() method items() method pop() method popitem() method del keywords clear() method copy() method update() method

keys

calling the keys and values present in a dictionary

In [76]:
```
# declare a dictionary containing information of cars and their respective hor
sepowers
horsepower = {'BMW':949 , 'Mercedes':945 ,'Ferrari':954 ,'Volkswagen': 976,'Re
nault ':889}
horsepower.keys()        #calling only the keys
```

Out[76]:
```
dict_keys(['BMW', 'Mercedes', 'Ferrari', 'Volkswagen', 'Renault '])
```

In [77]:
```
# declare a dictionary containing information of cars and their respective hor
sepowers
horsepower = {'BMW':949 , 'Mercedes':945 ,'Ferrari':954 ,'Volkswagen': 976,'Re
nault ':889}
horsepower.values()
```

Out[77]:
```
dict_values([949, 945, 954, 976, 889])
```

items()

The function item() returns the key along with its value.

In [78]:
```
horsepower.items()
```

Out[78]:
```
dict_items([('BMW', 949), ('Mercedes', 945), ('Ferrari', 954), ('Volkswagen',
976), ('Renault ', 889)])
```

pop()

Remove the element with the specified key name

In [79]:
```
# create a dictionary
year_sales = {
 2015: 34500,
 2016: 34300,
 2017: 40000
}
# remove key from dictionary
year_sales.pop(2015)
year_sales
```

Out[79]:
```
{2016: 34300, 2017: 40000}
```

popitem()

It removes the last inserted item

In [80]:
```
# create a dictionary
year_sales = {
 2015: 34500,
 2016: 34300,
 2017: 40000
}
year_sales[2018] = 7890
# remove last item from the dictionary
year_sales.popitem()
year_sales
```

Out[80]:
```
{2015: 34500, 2016: 34300, 2017: 40000}
```

del keyword

delete the item with the specified key name

In [81]:
```
# create a dictionary
year_sales = {
 2015: 34500,
 2016: 34300,
 2017: 40000
}
# delete item from the dictionary
del year_sales[2015]
year_sales
```

Out[81]:
```
{2016: 34300, 2017: 40000}
```

clear()

The clear() keyword empties the entire dictionary

In [82]:
```
# create a dictionary
year_sales = {
 2015: 34500,
 2016: 34300,
 2017: 40000
}
# clear the dictionary
year_sales.clear()
year_sales
```

Out[82]:
```
{}
```

Copy()

You cannot copy a dictionary by using dict2 = dict1, because dict2 will only be a reference to dict1, and changes made in dict1 will also be made in dict2.

In [ ]:
```
Use copy() method to make copy of dictionary
```

In [21]:
```
# create a dictionary
year_sales1 = {
 2015: 34500,
 2016: 34300,
 2017: 40000
}
# make a copy of dictionary
copy_dict = year_sales1.copy()
```

update()

Add and modify dictionaries by using the dict.update() method

In [84]:
```
# create a dictionary
year_sales = {
 2015: 34500,
 2016: 34300,
 2017: 40000
}
new_dict = {2016:30000, 2019 : 6789098, 2020 :345678}
# update dictionary
year_sales.update(new_dict)
year_sales.popitem()
year_sales
```

Out[84]:
```
{2015: 34500, 2016: 30000, 2017: 40000, 2019: 6789098}
```

Using keys and values to create dictionaries

In [3]:
```
key = ['a','b','c','d']

value = [1, 2, 3, 4]
d=dict(zip(key,value))
d
```

Out[3]:
```
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
```

In [11]:
```
my_friends = {
    'Jose': {'last_seen': 6},
    'Rolf': {'surname': 'Smith'},
    'Anne': 6
}
#What would the value of this code be?

my_friends['Jose']['last_seen']
```

Out[11]:
```
6
```

Sets

Disjoint set: if the two sets have no common elements

In [87]:
```
print(set([11, 12, 22, 33, 4]).isdisjoint(set([4, 1])))
```

```
False
```

Checking for subset

Use issubset() method to check whether all elements of a set are contained in another set

In [88]:
```
# check whether one set 'a' is a subset of the 'b'
a = set([1, 3, 4, 5])
b = set([1, 3, 4, 5, 6])
print('Is a subset of b?', a.issubset(b))
```

```
Is a subset of b? True
```

Set Union

Use union() method to compute the union of two or more sets

In [89]:
```
a = set([1, 2, 3])
b = set([3, 4, 5, 6])
print (a.union(b))
```

```
{1, 2, 3, 4, 5, 6}
```

Set Intersection

Take intersection to find the common elements

In [90]:
```
a = set([12,34,56,78])
b = set([12,45,67,89,78])
c = set([12,34,56,78,56])
print (a & b)
print (a & b & c)
```

```
{12, 78}
{12, 78}
```

Set Difference

It returns a new set containing all items from the first set that are not in the other set

In [112]:
```
a = set([23,45,56,77,89])
b = set([23,45,54,33,23])
print(a-b)
```

```
{56, 89, 77}
```

In [113]:
```
a.difference(b)
```

Out[113]:
```
{56, 77, 89}
```

Symmetric Difference - Collection of all the uncommon elements

In [114]:
```
a = set([23,45,56,77,89])
b = set([23,45,54,33,23])
a.symmetric_difference(b)
```

Out[114]:
```
{33, 54, 56, 77, 89}
```

All elements from both the sets - Union

Common elements from both - Intersection

Uncommon elements from both - Symmetric difference

set difference - It returns a new set containing all items from the first set that are not in the other set

WAP to create a list of 5 integers and find their average

In [93]:
```
lst=[]
for i in range(5):
 x=int(input("enter a number-"))
 lst.append(x)
print(lst)
print(sum(lst)/len(lst))
```

```
enter a number-3
enter a number-7
enter a number-9
enter a number-10
enter a number-57
[3, 7, 9, 10, 57]
17.2
```

Fibonacii Series

0 1 1 2 3 5 8 ....

In [94]:
```
a = 0
b = 1
fibo = []
for i in range(10):
# print(a, end = " ")
 fibo.append(a)
 c = a + b
 a = b
 b = c

fibo
```

Out[94]: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

In [95]:
```
fibo = [0, 1]
for i in range(8) :
 fibo.append(fibo[-1] + fibo[-2])

fibo
```

Out[95]: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

WAP to print factorial of a number

In [97]:
```
n = 10
fact = 1
for i in range(1, n+1):
 fact *= i

fact
```

Out[97]: 3628800

WAP to reverse a string

In [98]:
```
sen=input("Enter the string: ")
rev=""
for i in sen:
 rev=i+rev
 print(rev)
```

```
Enter the string: vishal
v
iv
siv
hsiv
ahsiv
lahsiv
```

WAP to check if a number is multiple of 5

In [100]:
```
num = int(input("Enter a number - "))
print(num, "is multiple of 5" if num % 5 == 0 else "not multiple of 5")
```

```
Enter a number - 90
90 is multiple of 5
```

WAP to create a fibonacci series for first 10 values

In [101]:
```
# 0 1 1 2 3 5 8 13 ...
a, b = 0, 1
fibo = []
for i in range(10) :
# print(a, end = " ")
 fibo.append(a)
 c = a + b
 a = b
 b = c
fibo
```

Out[101]:
```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

In [102]:
```
fibo = [0, 1]
for i in range(8):
 fibo.append(fibo[-1] + fibo[-2])
 print(fibo)
fibo
```

```
[0, 1, 1]
[0, 1, 1, 2]
[0, 1, 1, 2, 3]
[0, 1, 1, 2, 3, 5]
[0, 1, 1, 2, 3, 5, 8]
[0, 1, 1, 2, 3, 5, 8, 13]
[0, 1, 1, 2, 3, 5, 8, 13, 21]
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

Out[102]:
```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

WAP to find factorial of a number

In [103]:
```
num = int(input("Enter a number - "))
fact = 1
for i in range(num, 0, -1) :
 fact *= i

print(fact)
```

```
Enter a number - 5
120
```

WAP to print following output -

A AA AAA AAAA AAAAA

In [104]:
```
for i in range(1, 6):
 print("A"*i)
```

```
A
AA
AAA
AAAA
AAAAA
```

WAP to print following output -

A AB ABC ABCD ABCDE

In [106]:
```
for i in range(1, 6):
 for j in range(65, 65+i) :
    print(chr(j), end = " ")
 print()
```

```
A
A B
A B C
A B C D
A B C D E
```

Read a number n and print an identity matrix of the same order

In [108]:
```
n = int(input("Enter a number - "))
for row in range(n) :
 for col in range(n) :
    print(1 if row == col else 0, end = " ")

 print()
```

```
Enter a number - 5
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

WAP to print following output -

1 22 333 4444 5555

In [110]:
```
n = 5
for i in range(1, n + 1) :
 for j in range(5, i-1, -1) :
    print(i , end = " ")
 print()
```

```
1 1 1 1 1
2 2 2 2
3 3 3
4 4
5
```

Lists to perform multiplication:

num_list_1 = [1, 2, 3, 4]

num_list_2 = [0, 5, 2, 1]

In [1]:
```
num_list_1 = [1,2,3,4]
num_list_2 = [0,5,2,1]
products = []
for num1,num2 in zip(num_list_1,num_list_2):
    products.append(num1 * num2)

print(products)
```

```
[0, 10, 6, 4]
```

fstring examples

In [1]:
```
name = "Rolf Smith"
street = "123 No Name Road"
postcode = "PY10 1CP"

address = f"""Name: {name}
Street: {street}
Postcode: {postcode}
Country: United Kingdom"""

print(address)
```

```
Name: Rolf Smith
Street: 123 No Name Road
Postcode: PY10 1CP
Country: United Kingdom
```

In [2]:
```
description = "{} is {} years old."
print(description.format("Bob", 30))
```

```
Bob is 30 years old.
```

In [3]:
```
description = "{} is {age} years old."
print(description.format("Bob", age=30))
```

```
Bob is 30 years old.
```

In [4]:
```
user_input = input('Please enter a number: ')
double = user_input * 2
print(f'Your number doubled is {double}.')
```

```
Please enter a number: 3
Your number doubled is 33.
```

In [10]:
```
nearby_people = {"Rolf", "Jen", "Anna"}
user_friends = set()  # This is an empty set, like {}

friend = input("Enter your friend name to see if he is nearby: ")

# Add the friend to the user_friends set
user_friends.add(friend)
print(user_friends)

# Print out the friends that are nearby... those which are in both sets!
```

```
Enter your friend name to see if he is nearby: vishal
{'vishal'}
```

Seperating lines we use \n.

In [1]:
```
a= "    *     "
b= "   ***    "
c= "  *****   "
d= " *******  "
e= "********* "
print(a,b,c,d,e,sep='\n')
```

```
    *
   ***
  *****
 *******
*********
```

Program for printing tables

In [2]:
```
num = int(input('enter a number : '))

for i in range(1,11):
    print(num,'x',i,'=',(num * i))
```

```
enter a number : 6
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
```

Iterating over dictionaries

In [1]:
```
friend_ages = {'Ram':56,'Shyam':65,'vodka':78}
for name in friend_ages :
    print(name)
```

```
Ram
Shyam
vodka
```

In [6]:
```
for age in friend_ages.values() :
    print(age)
```

```
56
65
78
```

In [7]:
```
for name,age in friend_ages.items():
    print(f'{name} is {age} years old .')
```

```
Ram is 56 years old .
Shyam is 65 years old .
vodka is 78 years old .
```

List Comprehensions

In [ ]:
```
numbers = [0, 1, 2, 3, 4]
doubled_numbers = []

for num in numbers:
    doubled_numbers.append(num * 2)

print(doubled_numbers)

# -- List comprehension --

numbers = [0, 1, 2, 3, 4]  # list(range(5)) is better
doubled_numbers = [num * 2 for num in numbers]
# [num * 2 for num in range(5)] would be even better.

print(doubled_numbers)

# -- You can add anything to the new list --

friend_ages = [22, 31, 35, 37]
age_strings = [f"My friend is {age} years old." for age in friend_ages]

print(age_strings)


# -- This includes things like --
names = ["Rolf", "Bob", "Jen"]
lower = [name.lower() for name in names]

# That is particularly useful for working with user input.
# By turning everything to lowercase, it's less likely we'll miss a match.

friend = input("Enter your friend name: ")
friends = ["Rolf", "Bob", "Jen", "Charlie", "Anne"]
friends_lower = [name.lower() for name in friends]

if friend.lower() in friends_lower:
    print(f"I know {friend}!")
```

Comprehensions with conditionals

In [ ]:
```
ages = [22, 35, 27, 21, 20]
odds = [n for n in ages if n % 2 == 1]

# -- with strings --

friends = ["Rolf", "ruth", "charlie", "Jen"]
guests = ["jose", "Bob", "Rolf", "Charlie", "michael"]

friends_lower = [f.lower() for f in friends]

present_friends = [
    name.capitalize() for name in guests if name.lower() in friends_lower
]

# -- nested list comprehensions --
# Don't do this, because it's almost completely unreadable.
# Splitting things out into variables is better.

friends = ["Rolf", "ruth", "charlie", "Jen"]
guests = ["jose", "Bob", "Rolf", "Charlie", "michael"]

present_friends = [
    name.capitalize() for name in guests if name.lower() in [f.lower() for f i
n friends]
]
```

Set and Dictionaries Comprehension

In [ ]:
```
friends = ["Rolf", "ruth", "charlie", "Jen"]
guests = ["jose", "Bob", "Rolf", "Charlie", "michael"]

friends_lower = {n.lower() for n in friends}
guests_lower = {n.lower() for n in guests}

present_friends = friends_lower.intersection(guests_lower)
present_friends = {name.capitalize() for name in friends_lower & guests_lower}

print(present_friends)

# Transforming data for easier consumption and processing is a very common tas
k.
# Working with homogeneous data is really nice, but often you can't (e.g. when
working with user input!).

# -- Dictionary comprehension --
# Works just like set comprehension, but you need to do key-value pairs.

friends = ["Rolf", "Bob", "Jen", "Anne"]
time_since_seen = [3, 7, 15, 11]

long_timers = {
    friends[i]: time_since_seen[i]
    for i in range(len(friends))
    if time_since_seen[i] > 5
}

print(long_timers)
```

zip function

In [ ]:
```
"""
friends = ["Rolf", "Bob", "Jen", "Anne"]
time_since_seen = [3, 7, 15, 11]

long_timers = {
    friends[i]: time_since_seen[i]
    for i in range(len(friends))
    if time_since_seen[i] > 5
}

print(long_timers)
"""

# While that is extremely useful when we have conditionals, sometimes we
# just want to create a dictionary out of two lists or tuples.
# That's when `zip` comes in handy!

friends = ["Rolf", "Bob", "Jen", "Anne"]
time_since_seen = [3, 7, 15, 11]

# Remember how we can turn a list of lists or tuples into a dictionary?
# `zip(friends, time_since_seen)` returns something like [("Rolf", 3), ("Bob",
7)...]
# We then use `dict()` on that to get a dictionary.

friends_last_seen = dict(zip(friends, time_since_seen))
print(friends_last_seen)
```

enumerate function

In [8]:
```
friends = ["Rolf", "John", "Anna"]

for counter, friend in enumerate(friends, start=1):
        print(counter, friend)
```

```
1 Rolf
2 John
3 Anna
```

In [9]:
```
friends = ["Rolf", "John", "Anna"]
friends_dict = dict(enumerate(friends))
```

functions

In [ ]:
```
# So far we've been using functions such as `print`, `len`, and `zip`.
# But we haven't learned how to create our own functions, or even how they rea
lly work.

# Let's create our own function. The building blocks are:
# def
# the name
# brackets
# colon
# any code you want, but it must be indented if you want it to run as part of
 the function.


def greet():
    name = input("Enter your name: ")
    print(f"Hello, {name}!")


# Running this does nothing, because although we have defined a function, we h
aven't executed it.
# We must execute the function in order for its contents to run.

greet()

# You can put as much or as little code as you want inside a function, but pre
fer shorter functions over longer ones.
# You'll usually be putting code that you want to reuse inside functions.

# Any variables declared inside the function are not accessible outside it.
print(name)  # ERROR!
```

arguments and parameters

In [10]:
```
# Imagine you've got some code that calculates the fuel efficiency of a car:

car = {"make": "Ford", "model": "Fiesta", "mileage": 23000, "fuel_consumed": 4
60}

mpg = car["mileage"] / car["fuel_consumed"]
name = f"{car['make']} {car['model']}"
print(f"{name} does {mpg} miles per gallon.")

# You could put this in a function:


def calculate_mpg():
    car = {"make": "Ford", "model": "Fiesta", "mileage": 23000, "fuel_consume
d": 460}

    mpg = car["mileage"] / car["fuel_consumed"]
    name = f"{car['make']} {car['model']}"
    print(f"{name} does {mpg} miles per gallon.")


calculate_mpg()

# But this is not a very reusable function since it only calculates the mpg of
a single car.
# What if we made it calculate the mpg of "any" arbitrary car?

car = {"make": "Ford", "model": "Fiesta", "mileage": 23000, "fuel_consumed": 4
60}


def calculate_mpg(car_to_calculate):  # This can be renamed to `car`
    mpg = car_to_calculate["mileage"] / car_to_calculate["fuel_consumed"]
    name = f"{car_to_calculate['make']} {car_to_calculate['model']}"
    print(f"{name} does {mpg} miles per gallon.")


calculate_mpg(car)

# This means that given a list of cars with the correct data format, we can ru
n the function for all of them!

cars = [
    {"make": "Ford", "model": "Fiesta", "mileage": 23000, "fuel_consumed": 460
},
    {"make": "Ford", "model": "Focus", "mileage": 17000, "fuel_consumed": 350
},
    {"make": "Mazda", "model": "MX-5", "mileage": 49000, "fuel_consumed": 900
},
    {"make": "Mini", "model": "Cooper", "mileage": 31000, "fuel_consumed": 235
},
]

for car in cars:
    calculate_mpg(car)
```

```
Ford Fiesta does 50.0 miles per gallon.
Ford Fiesta does 50.0 miles per gallon.
Ford Fiesta does 50.0 miles per gallon.
Ford Fiesta does 50.0 miles per gallon.
Ford Focus does 48.57142857142857 miles per gallon.
Mazda MX-5 does 54.44444444444444 miles per gallon.
Mini Cooper does 131.91489361702128 miles per gallon.
```

functions and return values in python

In [ ]:
```
def calculate_mpg(car):
    mpg = car["mileage"] / car["fuel_consumed"]
    return mpg  # Ends the function, gives back the value


def car_name(car):
    return f"{car['make']} {car['model']}"


def print_car_info(car):
    name = car_name(car)
    mpg = calculate_mpg(car)

    print(f"{name} does {mpg} miles per gallon.")
    # Returns None by default, as all functions do


cars = [
    {"make": "Ford", "model": "Fiesta", "mileage": 23000, "fuel_consumed": 460
},
    {"make": "Ford", "model": "Focus", "mileage": 17000, "fuel_consumed": 350
},
    {"make": "Mazda", "model": "MX-5", "mileage": 49000, "fuel_consumed": 900
},
    {"make": "Mini", "model": "Cooper", "mileage": 31000, "fuel_consumed": 235
},
]

for car in cars:
    print_car_info(car)
    # try print(print_car_info(car)), you'll see None


# -- Multiple returns --


def divide(x, y):
    if y == 0:
        return "You tried to divide by zero!"
    else:
        return x / y


print(divide(10, 2))  # 5
print(divide(6, 0))  # You tried to divide by zero!
```

default parameter values

In [ ]:
```
def add(x, y=3):  # x=2, y is not OK
    total = x + y
    print(total)


add(5)
add(2, 6)
add(x=3)
add(x=5, y=2)

# add(y=2)  # ERROR!
# add(x=2, 5)  # ERROR!


# -- More named arguments --

print(1, 2, 3, 4, 5, sep=" - ")  # default is " "

# You can use almost anything as a default parameter value.
# But using variables as default parameter values is discouraged, as that can
 introduce difficult to spot bugs

default_y = 3


def add(x, y=default_y):
    sum = x + y
    print(sum)


add(2)  # 5

default_y = 4
print(default_y)  # 4

add(2)  # 5

# Be careful when using lists or dictionaries as default parameter values. Unl
ike integers or strings, these will update if you modify the original list or
 dictionary.

# This is due to a language feature called mutability. It's not important to u
nderstand this now, but just know that they behave differently to integers and
strings behind the scenes when you change them.
```

lambda functions

In [ ]:
```
# Lambda functions are functions that are almost solely used to get inputs and
return outputs.
# That means we don't often use them to make actions.
# For example, the `print()` function is a function that performs an action. A
s such, it would not be suitable for lambda function.
# If we wanted a function that just divided two numbers, that might be suitabl
e for a lambda function.

# That's because that function takes inputs, processes them, and returns outpu
ts. And, it's a short, simple function. You'll see why that is relevant with t
his example.

divide = lambda x, y: x / y
# This spacing is common. After each comma in the parameters, after the colon
 but not before, and between operators (though that's optional, and sometimes
 will be seen without spaces).

# That is a lambda function, which takes two arguments and returns the result
 of dividing one by the other. It is almost identical to this function:


def divide(x, y):
    return x / y


# In both cases you would call it as a normal function:

print(divide(15, 3))

# While traditional functions _need_ the name (you can't define one without i
t), lambda functions don't have names unless you assign them to a variable.

result = (lambda x, y: x + y)(15, 3)
print(result)

# However you can see that lambda functions can be quite difficult to read, so
we won't be using them very often. The main reason to use lambda function is b
ecause they are short, so if we use them in conjunction with other functions t
hat can help make our programs a bit more flexible.

# Here's an example. Instead of this:


def average(sequence):
    return sum(sequence) / len(sequence)


students = [
    {"name": "Rolf", "grades": (67, 90, 95, 100)},
    {"name": "Bob", "grades": (56, 78, 80, 90)},
    {"name": "Jen", "grades": (98, 90, 95, 99)},
    {"name": "Anne", "grades": (100, 100, 95, 100)},
]

for student in students:
    print(average(student["grades"]))
```

```
# Since the average function just takes inputs and returns an output, we could
re-define it as a lambda function.

average = lambda sequence: sum(sequence) / len(sequence)

students = [
    {"name": "Rolf", "grades": (67, 90, 95, 100)},
    {"name": "Bob", "grades": (56, 78, 80, 90)},
    {"name": "Jen", "grades": (98, 90, 95, 99)},
    {"name": "Anne", "grades": (100, 100, 95, 100)},
]

for student in students:
    print(average(student["grades"]))
```

In [ ]: