Code:

```
Part 1
================================================
print("Hello, ")
print("World")

#print in the same line using two prints
print("Hello", end=",")
print("World")

#print in the different lines using one print
print("Hello","World",sep="\n")


Part 2
=======================================================
# input in python
name = input("Enter your name please\n")
print(name)

'''
a = 10 #integer
a = "10" #integer or string? => String
a = 10.34 # float value
'''

# single line
'''
    Multi-line
'''

#type casting methods: int(), float(), complex(), etc.
a = int(input("Enter the value of a: ")) #input() will always returns a string
b= int(input("Enter the value of b: ")) #int will convert the input string into an
integer
# a = a+a # I can change values.
print(a+b)


Part 3
================================================
# Operators and precedence
#1. Arithmetic operations +, -, *, /, **, //, %

a = int(input("Enter the value of a: "))
b = int(input("Enter the value of b: "))

print("{x} + {y} = {z}".format(x=a, y=b,z=a+b)) # string formatting
print("{x} - {y} = {z}".format(x=a, y=b,z=a-b)) # string formatting
```

```
print("{x} * {y} = {z}".format(x=a, y=b,z=a*b)) # string formatting
print("{x} / {y} = {z}".format(x=a, y=b,z=a/b)) # string formatting

print("{x} ** {y} = {z}".format(x=a, y=b,z=a**b)) # power 10^2 = 100 like a^b
print("{x} // {y} = {z}".format(x=a, y=b,z=a//b)) # result is truncated
print("{x} % {y} = {z}".format(x=a, y=b,z=a%b)) # result is truncated

Part 4
==================================================
# operator precendence
# What is BODMAS
'''
    Brackets => Of => Division => Multiplication => Addition => Subtraction
'''

a = int(input("Enter the value of a: ")) #10
b = int(input("Enter the value of b: "))#2
print(a*b+a/b)
print(a**b*a)
print(a%b+a) # modulus is having higher precendence than *,/, + and -
```

TATA CONSULTANCY SERVICES

TATA

Python: Reading and Displaying Basic Types

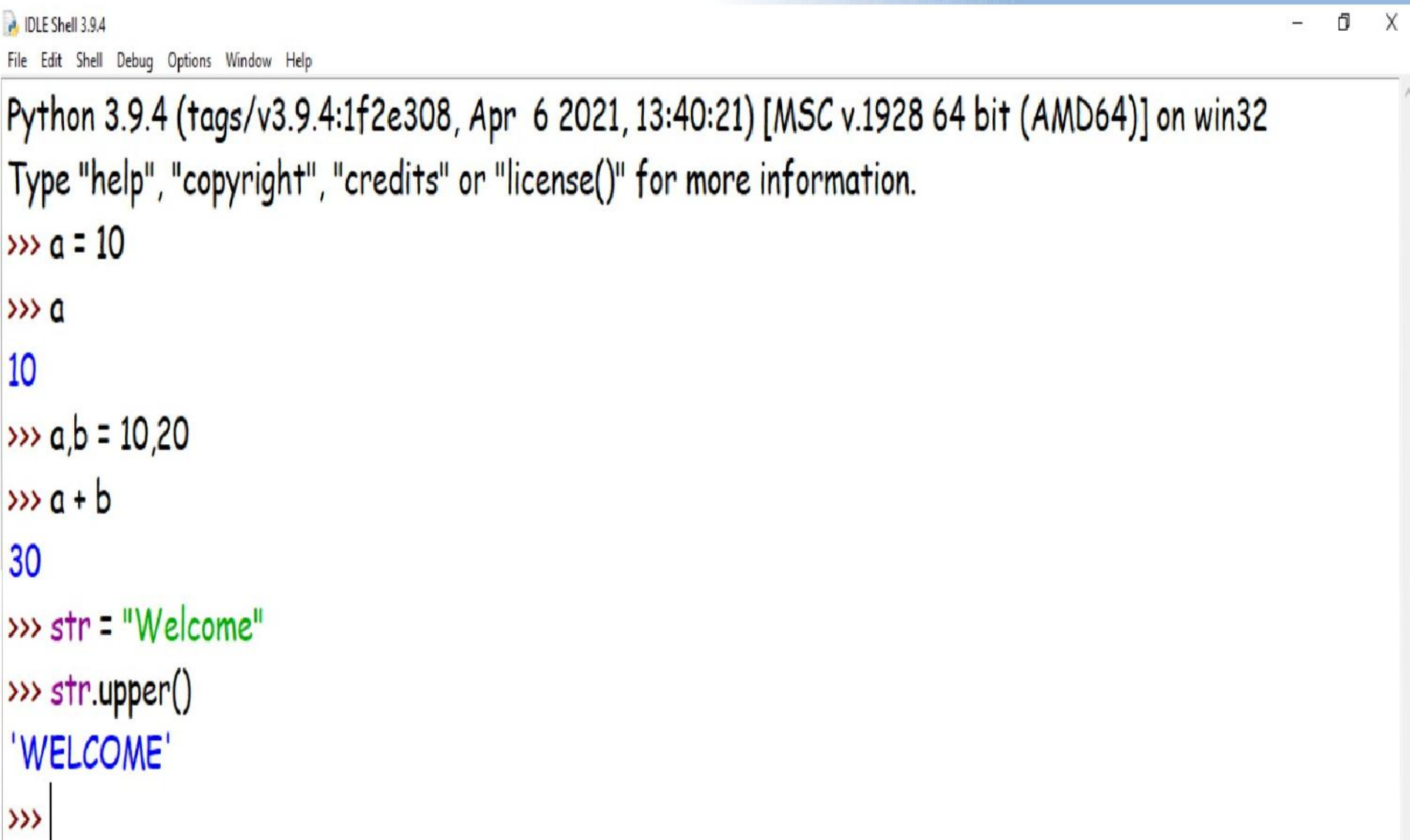
Agenda

- Introduction to Python programming.
- Software and Online compilers.
- Input and output in python.
- Basic data types.
- Type casting and formatting.
- Arithmetic operators and operations.
- Operator precedence.

Introduction to Python programming.

- Functional, scripted and object oriented.
- Easy to code.
- Open source.
- Interpreted and Platform independent.
- Dynamically typed.
- Wide external packages.

Python scripting



Software and Online compilers

- IDLE: It is used to write, edit and execute python programs. It comes with python interpreter, virtual machine, shell and the editor.

  IDLE download link: https://www.python.org/downloads/

  Supported OS: Windows, OSX, Linux.

- Online compilers: There are many options of online compilers available in www. Following are some commonly used online comilers.

  1. jDoodle:
  https://www.jdoodle.com/python3-programming-online/

  2. Open GDB:
  https://www.onlinegdb.com/online_python_compiler

  3. Programmiz:
  https://www.programiz.com/python-programming/online-compiler/

Input and Output

- We can read input from console using **input()** method.

  Syntax:

  variable=input(“Message”)

  Example:

  name = input(“Enter your name\n”)

  input() method is used to read a string however, we can read different data type values as well.

- We can print the output on console/ terminal using **print()** method.

  Syntax: print(expression)

  print() method automatically adds a new line at the end of the statement. We can change it using a special attribute called as “end”.

  Ex. print(“My name is khan”, end=' ')

  This will add space at the end of the line.

Basic Data Types

Following are data types in python:

- Text Type: str
- Numeric Types: int, float, complex
- Sequence Types: list, tuple, range
- Mapping Type: dict
- Set Types: set
- Boolean Type: bool

str, int, float, complex and bool are the examples of basic data types. List, tuples, range, dict, set are the advanced data types.

- You can check data type of any variable using type() method.
- Everything in python is an object, even functions are objects. You can check it using type() method.

Type casting and formatting

- Type casting is an art of converting data type of a variable to a new one.
- Data type of a variable is defined by the value it holds and decided at runtime.
- Ex.

  a=10 #here a is of type integer because it holds an integer value.

  a=20.13 #here a is automatically changed to a float value.
- This is called implicit type casting, performed by python internally.
- Explicit type casting is a process of converting the data type of a variable with the help of conversion methods.
- int(), float(), compelx(), str() are examples of conversion methods.

Arithmetic operators and precedence

There are 7 arithmetic operators in Python :

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Modulus
6. Exponentiation
7. Floor division

Operator precedence is an order of expression evaluation. You can consider operator precedence as BODMAS or VBODMAS rule.

Ex. “2+4*6%10+20”

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'td>
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators

```
PArt 1 shell
==================================

Python 3.9.4 (tags/v3.9.4:1f2e308, Apr  6 2021, 13:40:21) [MSC v.1928 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a  = int(input("Enter the value of a: \n"))
Enter the value of a:
20
>>> b  = int(input("Enter the value of b: \n"))
Enter the value of b:
30
>>> a > b
False
>>> s = "False"
>>> s
'False'
>>> a = 10
>>> b = 10
>>> a == b
True
>>> a is b
True
>>> a = 10
>>> b = 20
>>> id(a)
1532576557648
>>> id(b)
1532576557968
>>> a = b = 10
>>> id(a)
1532576557648
>>> id(b)
1532576557648
>>> 10 != 10
False
>>> not 10
False
>>> not True
False
>>> not False
True
>>> !=

Part 2
======================================
# combining conditions

n = int(input("Enter the number:\n"))
```

```
if n > 0 and n%2==0: #check if number is natural and if it is even
    print("AND: {number} is even natural".format(number=n))
else:
    print("AND: {number} is odd".format(number=n))

if n > 0 or n%2==1: #check if number is natural and if it is even
    print("OR: {number} is odd natural".format(number=n))
else:
    print("OR: {number} is odd".format(number=n))

'''
INput used in session
Enter the number:
101
AND: 101 is odd
OR: 101 is even natural
>>>
== RESTART: C:/Users/1783980/AppData/Local/Programs/Python/Python39/relational.py =
Enter the number:
-20
AND: -20 is odd
OR: -20 is odd'''

Part 3
====================================
# printing in python
a = 10
b = 20
c = a+b
print(a,"+",b,"=",c) #need to maintain comma and track the same
print("{x} + {y} = {z}".format(x = a, y = b, z=c)) #need not to maintain comma and
easy to read

#when to use formatting
# => When there are variables in the print statement

Part 4
============================================
marks = int(input())
year = int(input())
age = int(input())

if marks > 80 and year < 2020 and age > 18:
    print("{}:true".format(age))
else:
    print("{}:false".format(age))

'''
Input
97
```

```
2019
18
18:false
>>>
= RESTART: C:/Users/1783980/AppData/Local/Programs/Python/Python39/relational.py
100
2019
22
22:true'''
```

TATA CONSULTANCY SERVICES

TATA

Python: Relational operators and operations

Agenda

- Relational operators.
- Decision making with relational operators.
- Combining conditions.
- Problem statement – 1.
- Problem statement - 2.

Relational operators

Operator	Meaning	Expression	Value
>	is greater than	6 > 4	true
>=	is greater than or equal to	7>=6	true
<	is less than	9 < 8	false
<=	is less than or equal to	5 <= 5	true
==	is equal to	3 == -3	false
!=	is not equal to	8.0 != 8	fasle

Decision making with relational operators

Decision making is all about testing the condition.

Ex. if 10 > 2 then

print(10, “is greater than”, 2)

Relational operators returns either true or false.

Relational operators can also be used in loops and iterations.

Combining conditions

- Given a number n. Test if the number is natural even number or not.

  Steps:

  1. Check if the number is natural or not.

  2. If the number is natural then check if it is even or not.

- To check if the given number is even natural, we need two conditions.

- We can use following logical operators to combine multiple conditions.

  1. Logical AND (and).

  2. Logical OR (or).

Combining conditions

A	B	A AND B	A OR B	NOT A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

A and B are conditions. Above truth table demonstrates behaviour of each logical operator.

As a conclusion you can say,

1. and will return true iff both A and B are true, else false
2. or will return false iff both A and B are false, else true.
3. not will invert the condition result.

Problem statement - 1

John is a faculty in ABC college. Principal of the college wants to create group of all the students based on following criteria.

1. Student must have a 'A' grade in examination.
2. Student must have joined the college before 2020.
3. Student's age must be greater than 18.

Read marks (int) of the student as input followed by joining year (int) followed by age (int).

If marks are greater than 80, grade should be A.

Write a python program to check if the input student is eligible to be a part of the group or not.

if eligible: print “{student_age}:true”.

else: print “{student_age}:false”.

Problem statement - 2

Write a python program to read a string as an input. Check if the given string is palindrome or not.

Ex.

nitin => Palindrome

wasitacatoracatisaw=> Palindrome

welcome => Not a palindrome

If the input string is palindrome, print “{string} is a palindrome”

else, print “{string} is not a palindrome”

Instructions:

You will be given some questions based on the topics covered in the sessions. These questions will be coding based.

You are required to read the question carefully, analyze the sample input and output and write the code.

You can use online compilers or IDLE to make sure your code executes against the given sample input and output.

You can maintain a separate word file to store answers to all the questions. Once you write the code and get it tested using online compilers/ IDLE, you can copy paste your code below the question and follow the same for all the assignment question.

Ex.

Q.1 Write a python program to read and display a string.

Answer:

```
str = input("Enter your string")
print(str)
```

Q.2 Write a python program to read a number and print square of the number.

Answer:

```
number = int(input("Enter the number"))
print(number*number)
```

Maintaining a word file for assignment answers shall help you to revisit the topics.

Here is a list of some popular online compilers for python.

1. jDoodle: https://www.jdoodle.com/python3-programming-online/
2. Programmiz: https://www.programiz.com/python-programming/online-compiler/
3. OnlineGDB: https://www.onlinegdb.com/online_python_compiler

Solutions will be discussed in the session. However, it is recommended to try the questions before the discussion.

Questions based on "Reading and Displaying Basic Data Types".

Q. 1 Write a python program to read a string from the console and print each character of the string on a separate line.

```
Sample input 1
WELCOME

Sample output 1
W
E
L
C
O
M
E

Sample input 2
WELCOME HOME

Sample output 2
W
E
L
C
O
M
E

H
O
M
E
```

Q.2 Write a python program to read a 4-digit integer value as an input and perform sum of all the digits of input integer. Print the sum as shown in sample output.

```
Sample input 1
1234

Sample output 1
1+2+3+4 = 10

Sample input 2
1111

Sample output 2
1+1+1+1 = 4
```

Q.3 Write a python program to read a float value as an input and round it to 2 decimal digits.

```
Sample input 1
11.345

Sample output 1
11.35

Sample input 2
-123.248

Sample output 2
```

```
-123.25
```

Q.4 Write a python program to read a float value from console and print a number with 2 digits after the decimal point.

```
Sample input 1
2341.2341

Sample output 1
2341.23

Sample input 2
-11.111

Sample output 2
-11.11
```

Q.5 Write a python program to read a string expression from a console and print the length of the expression.

```
Sample input 1
2+3*7+12%2

Sample output 1
Length of the expression is 10

Sample input 2
2*3%10+12**2

Sample output 2
Length of the expression is 12
```

Questions based on "Basic conditional and Logical operations".

Q.1 Write a python program to read an integer value from console.

If the input number is a natural number, then please do the following.

1. If input number is even, print "{number} is an even natural number".
2. If input number is odd, print "{number} is an odd natural number".

If the input number is not a natural number, then please do the following.

1. If input number is even, print "{number} is an even number".
2. If input number is odd, print "{number} is an odd number".

A number is said to be natural if it is greater than 0.

```
Sample input 1
10

Sample output 1
10 is an even natural number

Sample input 2
0

Sample output 2
0 is an even number


Sample input 3
```

```
-11

Sample output 3
-11 is an odd number
```

Q.2 Write a python program to read a string from the console. Read another string which is a pattern. If the input starts or ends with the given input pattern, print "{input_string} contains {pattern}" else print "{input_string} does not contain {pattern}".

Note: String comparison is case insensitive.

```
Sample input 1
Mr. Rajesh Gopinathan is a CEO of TCS
Tcs

Sample output 1
Mr. Rajesh Gopinathan is a CEO of TCS contains Tcs

Sample input 2
Engineers builds the nation
country

Sample output 2
Engineers builds the nation does not contain country

Sample input 3
Engineers builds the nation
engiNeeR

Sample output 3
Engineers builds the nation contains engiNeeR.
```

Q.3 Write a python program to read an integer from console as "Marks". Assign and print grade for the input marks.

If marks >= 80, then print "{marks} will grant you a grade A".
If marks >= 60 and marks < 80, then print "{marks} will grant you a grade B".
If marks < 60, then print "{marks} will grant you a grade C".

```
Sample input 1
89

Sample output 1
89 will grant you a grade A

Sample input 2
55

Sample output 2
55 will grant you a grade C
```

Q.4 Write a python program to read a float number from console as "Salary". Print the salary with hike.

If salary is greater than or equal to 10000, hike should be 10% of the salary.
If salary is greater than or equal to 6000 and less than 10000, hike should be 8% of the salary.
If salary is less than 6000, hike should be 5% of the salary.

Round the updated salary to 2 digits.

```
Sample input 1
10000.34

Sample output 1
11000.37

Sample input 2
8560.45

Sample output 2
9245.29
```

Q.5 Write a menu driven python program. Read two integer values from console as "a" and "b". Create a following menu after reading a and b.

1. Addition
2. Subtraction
3. Exponent
4. Multiplication

Read an integer from console as "choice".

If choice is equal to 1, print "{a} + {b} = {a+b}".
If choice is equal to 2, print "{a} - {b} = {a-b}".
If choice is equal to 3, print "{a} ^ {b} = {a**b}".
If choice is equal to 4, print "{a} * {b} = {a*b}".

If choice is none of the above (1,2,3,4), print "Invalid choice! Exiting application"

```
Sample input 1
12
23
1

Sample output 1
12 + 23 = 35


Sample input 2
12
2
3

Sample output 2
12 ^ 2 = 144


Sample input 3
12
23
11

Sample output 3
Invalid choice! Exiting application
```

Mixed Questions.

Q.1 Write a python program to read an integer from console as "x". Print "true" if x is a palindrome number, otherwise print "false".

```
Sample input 1
121

Sample output 1
true
```

```
Explanation: 121 reads as 121 from left to right and from right to left.

Sample input 2
122

Sample output 2
false

Explanation: 122 reads as 122 from left to right and 221 from right to left.
```

Q.2 Write a python program to read two strings from console as "str1" and "str2". Write a logic to check if str1 and str2 are anagrams.

If strings are anagram, print "{str1} == {str2} is true".
If strings are not anagram, print "{str1} == {str2} is false".

An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Note: String comparison should be case sensitive.

```
Sample input 1
ate
eat

Sample output 1
ate == eat is true

Sample input 2
gate
etgs

Sample output 2
gate == etgs is false
```

TATA CONSULTANCY SERVICES

TATA

Python: Basic Conditional Operations

Agenda

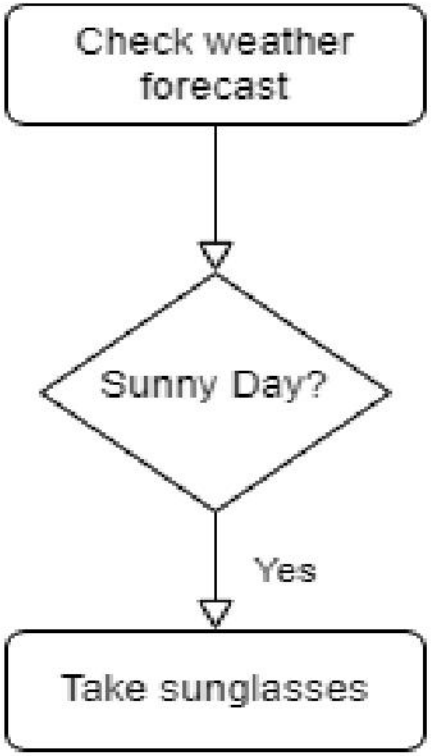
- Introduction to conditional logic
- The if statement
- Indentation while writing Python code
- The elif and else statements
- Merging two conditions with and/or

Some real life scenarios:

If sunny take sunglasses

If rainy take umbrella

Flowchart



How do we implement this logic in Python program?

The if statement

Basic Syntax

```
if <expr>:
    <statement>
```

If expression evaluates to True, the statement will be executed.

An example

Let us convert the earlier scenario to python code!

```
if weather == 'sunny':
    print('take sunglasses')

if weather == 'rainy':
    print('take an umbrella')
```

What is '==' ?

== is a comparison operator.

The == is used to check if one value is equal to another one.

Examples:

string_one = 'Python'
string_two = 'Java'
string_one==string_two will return False

Similarly 10==10 will also return True

Indenting the code

Python treats code with same indentation level as single block of code.

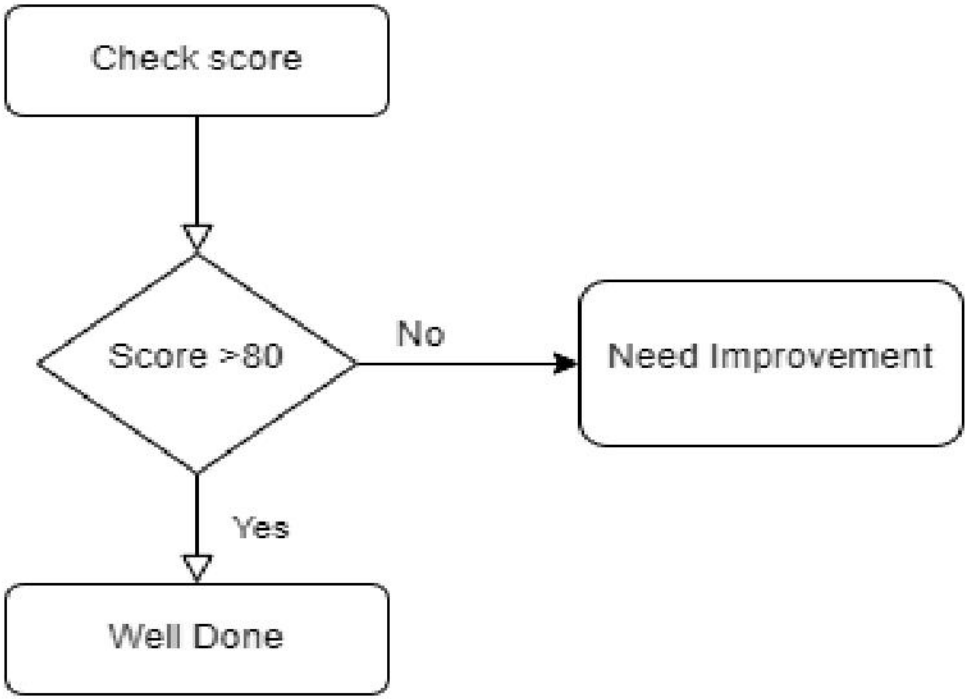
Example :

```
exam_score=89

if score>80:
    print('well done')
print('good luck')
```

Second print statement in the above example will be executed always.
Even if score is less than 80.

else clause

What if we have to do something if the expression is False?



else clause

Basic syntax

```
if <expr>:
    <statement(s)>
else:
    <statement(s)>
```

If the expression is False, statements under else clause will be executed.

elif clause

Basic Syntax

```
if <expr>:
    <statement(s)>
elif <expr>:
    <statement(s)>
elif <expr>:
    <statement(s)>
    ...
else:
    <statement(s)>
```

Question!

Take a score from user. Print the following grades based on score:

Score	Output grade
Greater than or equal to 90	A
Greater than or equal to 80 but less than 90	B
Greater than equal to 70 but less than 80	C
Less than 70	D

Two conditions with and

```
if english_score>=90 and science_score>=90:
    print('Good Job')
```

The print statement will be executed only if both the english score and science score are greater than or equal to 90.

Two conditions with or

```
if english_score<90 or science_score<90:
    print('Prepare well')
```

Print statement will be executed if score is less than 90 in any one of the subject.

Thank You

Code:

if statement
==========

```
# initializing a variable weather with a str value
weather = 'sunny'

if weather == 'sunny':
    print('take sunglasses')

if weather == 'rainy':
    print('take an umbrella')


if 10*10==100:
    print('You are correct')

# taking name as input from user
name = input('What is your name?')

if name == 'Raj':
    print('Hello Raj')
```

if else statement
==========

```
# initializing a variable score with a int value
score = 89

if score>80:
    print('well done')
else:
    print('Need Improvement')


# Question
# Take age as input from user. Print 'You are a kid' if the age is less than 18.
# Print 'You are an adult' if age is greater than or equal to 18. Print the statements without quotes.

# taking input from user
age = int(input('how old are you?'))

if age<18:
    print('You are a kid')
else:
    print('You are an adult')
```

indentation
==========

```
weather = input('How is the weather?')

if weather == 'sunny':
    print('take sunglasses')
```

```
# the below print statement will be executed even if above if condition is false
print('go outdoors')
```

if elif else statements
==========

```
# Question
# Take a score from user. Print the following grades based on score:

# Greater than or equal to 90                          A
# Greater than or equal to 80 but less than 90         B
# Greater than equal to 70 but less than 80            C
# Less than 70                                         D

# take the score as input from user
score = int(input('What is your score?'))

if score>=90:
    print('A')
elif score>=80:
    print('B')
elif score>=70:
    print('C')
else:
    print('D')
```

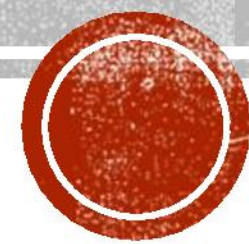
Conditions with and
==========

```
english_score = int(input('Enter your English score : '))
science_score = int(input('Enter your Science score : '))

if english_score>=90 and science_score>=90:
    print('Good Job')
```

Conditions with or
==========

```
english_score = int(input('Enter your English score : '))
science_score = int(input('Enter your Science score : '))

if english_score<90 or science_score<90:
    print('Prepare well')
```

NUMERIC LOGIC BUILDING-PYTHON



TCS Digital
Test Preparation Series

PROGRAM LOGIC

- Program logic is the implementation of the program's requirements and design.
- If the design of the application is bad, the program logic can nevertheless be professionally implemented.
- For example, if the user interface is poorly conceived, the program logic can execute that second-rate interface very efficiently.

ARMSTRONG NUMBER

WAP to find if a number is armstrong or not. (Take number as input from user)

Example:

```
153 = 1*1*1 + 5*5*5 + 3*3*3

153 is an armstrong number
```

Sample 1:

Input- Enter a number:153

Output- 153 is an armstrong number!

Sample 2:

Input- Enter a number:53

Output- 53 is not
an armstrong number!

LOGIC

1. Number=153 (input)
2. Separate 1 from 153 (Separating digits from a number)
3. Cube1= 1*1* 1
4. Separate 5 from 153
5. Cube2= 5*5*5
6. Separate 3 from 153
7. Cube3= 3*3*3
8. Sum= cube1+cube2+cube3
9. If Sum is equal to Number then it is Armstrong!

FLOW OF THE PROGRAM

number=153
temp=153
sum=0

while(temp>0)	digit=temp%10	sum+=digit**3	temp//=10
153 >0 True	153%10=3	0+(3*3*3)=27	153//10=15
15>0 True	15%10=5	27+(5*5*5)=27+125=152	15//10=1
1>0 True	1%10=1	152+(1*1*1)=152+1=153	1//10=0
0>0 False			

- **//** - floor division (300//10=30 whereas 300/10=30.0)
- %- modulus (gives remainder; ex- 300%10=0)
- **sum+=digit**3** also means sum=sum+(digit**3) OR sum=sum+(digit*digit*digit)
- **temp//=10** also means temp=temp//10

```
number = int(input("Enter a number: "))
sum = 0
temp = number
while temp > 0:
    digit = temp % 10
    sum += digit ** 3
    temp //= 10

if number == sum:
    print(number,"is an Armstrong number")
else:
    print(number,"is not an Armstrong number")
```

SUM OF ODD/EVEN DIGITS

WAP to print sum of even and odd digits of a number separately. (Take number as input from user)

Example:

Input-

Enter a number: 2345

Output-

Sum of even digits:6

Sum of odd digits: 8

LOGIC

1. num=2345 (input)
2. rem=num%10=2345%10=5 (separate digits)
3. if rem%2==0 (Even!) then sum+=rem
4. If rem%1==0 (Odd!) then sum+=rem
5. Repeat the process for all the digits.

FLOW OF THE PROGRAM

number=2345
sum_odd=0
sum_even=0

while num> 0	rem	if rem%2==0	sum_even+=rem	if rem%1==0	sum_odd+=rem	num//=10
2345>0 True	2345%10=5	False	-	True	0+5=5	2345//10=234
234>0 True	234%10=4	True	0+4=4	False	-	234//10=23
23>0 True	23%10=3	False	-	True	5+3=8	23//10=2
3>0 True	2%10=2	true	4+2=6	False	-	2//10=0
0>0 False						

```
number = int(input("Enter a number:"))

sum_even = 0
sum_odd = 0

while number > 0:
    rem = number % 10
    if rem % 2 == 0:
        sum_even = sum_even + rem
    else:
        sum_odd = sum_odd + rem
    number = number // 10

print("Sum of even digits:", sum_even)
print("Sum of odd digits:", sum_odd)
```

REVERSE A NUMBER

WAP to reverse a number. (Take number as input from user)

Example:

Input-

Enter a number:3456

Output-

Reversed Number: 6543

LOGIC

1. num=2593 (input)
2. reversed_num=0
3. digit=num%10=2593%10=3 (separate digits)
4. reversed_num=(reversed_num*10)+ digit = 0+3=3 (getting reveres number)
5. num=num//10=2593//10=259 (eliminate last digit)
6. Repeat the process.

while (num!=0)	digit	reversed_num	num
2593!=0 True	2593%10=3	0+3=3	2593//10=259
259!=0 True	259%10=9	(3*10)+9=30+9=39	259//10=25
25!=0 True	25%10=5	(39*10)+5=390+5=395	25//10=2
2!=0 True	2%10=2	395*10)+2=3950+2=3952	2//10=0
0!=0 False			

```
num = int(input("Enter a Number:"))
reversed_num = 0

while num != 0:
    digit = num % 10
    reversed_num = reversed_num * 10 + digit
    num //= 10

print("Reversed Number: " + str(reversed_num))
```

MIN, MAX AND SUM OF DIGITS OF A NUMBER

WAP to print the minimum(smallest) and maximum(largest) digit in a number. Also, print the sum of all the digits.

Sample Input-

Enter number: 8265

Sample Output-

Largest digit: 8

Smallest digit : 2

Sum of digits: 21

- **LOGIC**

1. num=8265 (input)
2. largest_num=0 and smallest_num=9
3. rem=num%10=8265%10=5 (separate digits)
4. count+=rem (will calculate sum of digits, 0+5=5 then 5+6=11 then 11+2=13 then 13+8=21)
5. if largest_num<rem (largest number) then largest_num=rem
6. if smallest_num>rem (smallest number) then smallest_num=rem
7. Repeat the process.

rem	largest_num<rem	largest_num = rem	smallest_num>rem	smallest_num = rem
5	0<5 True	5	9>5 True	5
6	5<6 True	6	5>6 False	-
2	6<2 False	-	5>2 True	**2**
8	6<8 True	**8**	2>8 False	-

```
num=int(input("Enter the Number :"))
largest_num=0
smallest_num=9
count=0

while (num > 0):
    reminder=num%10
    count=count+reminder
    if largest_num<reminder:
        largest_num = reminder
    elif smallest_num>reminder:
        smallest_num = reminder
    num =int(num / 10)
print("Largest Digit:", largest_num)
print("Smallest Digit:", smallest_num)
print("Sum of digits:",count)
```

MIN, MAX AND AVERAGE OF A NUMBERS WITHOUT USING ARRAY/COLLECTIONS

WAP to print the minimum(smallest) and maximum(largest) number of all the numbers (Do not use an array or collection to store numbers). Also, print the average of all the numbers entered.

Take 2 inputs from user:

1. The count; which means how many numbers would be entered.
2. The numbers

Sample Input:

Enter Count: 3

Enter Numbers:

2

3

4

Sample Output:

Average: 3.0

Minimum Number: 2

Maximum Number: 4

LOGIC

1.num=3(input count)
2.number =2 then 3 then 4 (3 times i.e., num)
3.min=max=num=3
4.if num < min (smallest number) then min=num
5.if num > max (largest number) then max=num

Flow of the Program

count = 0

total = 0.0

maximum = num

minimum = num

num	3			
while (count<num)	0<3 True	1<3 True	2<3 True	3<3 False
count+=1	0+1=1	1+1=2	2+1=3	
total+=num	0+3=3	3+3=6	6+3=9	
number(input)	2	3	4	
number < minimum	2<3 True	3<2 False	4<2 False	
minimum = number	**2**	-	-	
number > maximum	2>3 False	3>3 False	4>3 True	
maximum = number	-	-	**4**	

```
number=int(input("Enter a count:"))
total=0.0
count=0
minimum=0
maximum=0

print("Enter Numbers:")
for i in range(number):
    num=int(input())
    count=count+1
    total=total+ num

    if(i==0):
        minimum = num

    if num<minimum:
        minimum=num
    elif num>maximum:
        maximum=num

print("Average:", total/number)
print("Minimum number:", minimum)
print("Maximum number:", maximum)
```

Practice Code (Try it!)

```
num = 1
sum = 0.0

while(num <= 7):
    fact = 1
    for i in range(1,num+1):
        fact = fact * i
        i=i+1

    sum = sum + (num / fact)
    num=num+1

print("Sum of series is:", round(sum,5))
```

SUM OF SERIES

WAP to find sum of series:

1/1! + 2/2! + 3/3! + …. +n/n!

(n represents a number that we take as input from user; so, till n the sum would be calculated for the series)

Example:

1/1! + 2/2! + 3/3! + 4/4! + 5/5! + 6/6! +7/7!

Input- Enter a number: 7

Output- Sum of series is 2.718056

****Note**- Limit the decimal places to 5

ASSIGNMENT QUESTIONS- WEEK 2

Questions based on Iterations, Logic Building with String manipulation & Numeric Logic Building with conditions & iterations - Python

Q1. Write a Python program to read a word from the console. Print the total number of vowels in the word.

Sample input 1: Hello
Sample output 1: 2

Sample input 2: Consultancy
Sample output 2: 3

Q2. Write a Python program to read a number n. Print prime if the number is prime. Else print not prime.

Sample input 1: 9
Sample output 1: not prime

Sample input 2: 17
Sample output 2: prime

Q3. Read a word and a letter from console. Print the word after deleting all the occurrences of the letter.

Sample input 1: Moon o
Sample output 1: Mn

Sample input 2: Linux u
Sample output 2: Linx

Q4. Write a Python program to reverse a string.

Sample input 1: poker
Sample output 1: rekop

Sample input 2: Windows
Sample output 2: swodniW

Q5. Write a Python program to read a number from console and print the sum of their digits.

Sample input 1: 1112
Sample output 1: 5

Sample input 2: 567
Sample output 2: 18

Q6. Write a Python program to read numbers from a single line and print the missing numbers. Numbers will be in consecutive order.

Sample input 1: 1 2 5 6 7 9
Sample output 1: 3 4 8

Sample input 2: 102 110 115 116 117 120
Sample output 2: 102 110 115 116 117 120

Q7. Write a program to read a sentence and a word from console. Print the total number of occurrences of the word in the following format: The word 'x' appears 'y' times Replace x and y with the word and the total number of occurrence. Comparison must be case insensitive

Sample input 1: Linux is a open-source Unix-like OS based on the Linux kernel.
Linux
Sample output 1: The word 'Linux' appears 2 times

Sample input 2: Mars is the fourth planet from the Sun. Martians are from Mars.
Mars
Sample output 2: The word 'Mars' appears 2 times

Q8. Write a program to read a word from console. Print the length of longest repeating character streak.

Sample input 1: Loops
Sample output 1: 2

Sample input 2: Hellooooooo Wooorlddddddd
Sample output 2: 7

THANK YOU

Code

```
# Read a date as ddmmyyyy and displaying day, month and year

date = input()

print('day : '+ date[:2])
print('month : '+ date[2:4])
print('year : '+ date[4:])

# Read a string from console and print the count of vowels
word = input()

count = 0

for l in word:
    if l.lower() in ['a','e','i','o','u']:
        count += 1

print(count)

# Read a string from input and print the count of each vowels

sentance = input()

for char in ['a', 'e', 'i', 'o', 'u']:

    c = 0

    for letter in sentance:
        if letter.lower() == char:
            c += 1

    print('{} - {}'.format(char,c))


# Read a string from input and print word count in sentence

sentance = input()

words = sentance.split(' ')

print(len(words))


# read a string and print letters in odd position(E.g for input as "discover" output is - "dsoe")

word = input()

print(word[::2])

# read a string and print letters in even position(E.g for input as "discover" output is - "icvr")

word = input()

print(word[1::2])
```

TATA CONSULTANCY SERVICES

TATA

Python: String Manipulations

Agenda

- String reversal
- String formatting
- Questions

String reversal

Read a string and print its reverse:

Hai – iaH

Hello – olleH

```
word = input()
print(word[::-1])
```

String formatting

```
age = 10
print('my age is '+str(age))
print('my age is {}'.format(age))
print(f'my age is {age}')
```

Question1

Read a date as ddmmyyyy and displaying day, month and year

Question 2

Read a string from console and print the count of vowels

Question 3

Read a string from input and print the count of each vowels

Question 4

Read a string from input and print word count in sentence

Thank You

ITERATIONS IN PYTHON

TCS Digital Test Preparation Series

ABOUT ME

- Name – Aashina Arora
- Years of Experience- 4.5
- Areas of expertise- Python and Java
- Pursuing MCA

Programming Logic	**15 mins**
Coding Section	**45 mins**

PATTERN

- You must read all instructions carefully. If there is any negative marking, it will be clearly mentioned in the instructions.
- For more details, please visit this link - https://www.tcs.com/careers/tcs-off-campus-hiring .
- Number of Questions- 2

Section Name	Marks per Item	No. Of Items
Advanced Coding- Medium	60	1
Advanced Coding- Difficult	90	1

ITERATION

- This means repetition of a process.
- Iterations are performed through loops: WHILE, FOR
- Loops requires 3 conditions:
  - Initialization
  - Condition
  - Increment/Decrement

WHILE Loop

- With the while loop we can execute a set of statements as long as a condition is true.
- Syntax:

```
while expression:
    statement(s)
```

EXAMPLE

```
count = 0
while (count < 3):
    count = count + 1
    print("Hello World")
```

Sample Problem

WAP to find factorial of a number. (Take number for factorial as user input.)

Factorial:

$4! = 4*3*2*1 = 24$

Flow of Program

number	fact	i	while(i<=number)	fact	i
4	1	1	1<=4 True	fact=1*1=1	i=1+1=2
4	1	2	2<=4 True	fact=1*2=2	i=2+1=3
4	2	3	3<=4 True	fact=2*3=6	i=3+1=4
4	6	4	4<=4 True	fact=6*4=24	i=4+1=5
4	24	5	5<=4 False		

FOR Loop

- The for **loop in Python** is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like list, tuple, or dictionary.
- Syntax:

```
for iterating_variable in sequence:
    statement(s)
```

EXAMPLE

```
str = "Python"
for i in str:
    print(i)
```

Iterating a sequence

Iterating a List using FOR loop (Example):

```
list = [10,30,23,43,65,12]

sum = 0

for i in list:

    sum = sum+i

print("The sum is:",sum)
```

range()

- **Python range() function** returns the sequence of the given number between the given range.
- range() is a built-in function of Python. It is used when a user needs to perform an action a specific number of times.

Examples:

1. range(n)

```
for i in range(10):
    print("A")


for i in range(10):
    print(i)
```

range()

2. range(start,end)

```
for i in range(5,15):
    print(i)
```

3. range(start,end,stop)

```
for i in range(1,10,2):
    print(i)
```

Sample Problem

WAP to find fibonacci series. (Take as number as user input till where you want to print the series.)

Fibonacci Series:

0 1 1 2 3 5 8 13 21 34 55 89 144....

Online Editors:

https://www.jdoodle.com/python3-programming-online/

https://www.online-python.com/

https://www.onlinegdb.com/online_python_compiler

Program for Factorial

```
number=int(input("Enter number for factorial:"))
i=1
fact=1

while(i<=number):
    fact=fact*i
    i=i+1

print("Factorial of",number,"is:",fact)
```

Program for Fibonacci Series:

```
number=int(input("Enter a number:"))
a=0
b=1
print(a)
print(b)

for i in range(0,number):
    sum_of_numbers=a+b
    a=b
    b=sum_of_numbers

    print(sum_of_numbers)
```

Digital Test-Python-Assignment 1-Solutions

Q.1 Write a python program to read a string from the console and print each character of the string on a separate line.

Answer:

```
str = input()
for i in str:
    print(i)
```

Q.2 Write a python program to read a 4-digit integer value as an input and perform sum of all the digits of input integer. Print the sum as shown in sample output.

Answer:

```
number = int(input())
str = str(number)
res = int(str[0]) + int(str[1]) + int(str[2]) + int(str[3])
print(str[0] +"+"+str[1] + "+" +str[2]+ "+"+str[3]+" =",res)
```

Q.3 Write a python program to read a float value as an input and round it to 2 decimal digits.

Answer:

```
number = float(input())
print(round(number,2))
```

OR

```
number = float(input())
print("{:.2f}".format(number))
```

Q.4 Write a python program to read a float value from console and print a number with 2 digits after the decimal point.

Answer:

```
number = float(input())
integer = len(str(int(number))) '''should give us length of the
integer  part of a number'''
number = str(number)
print(number[:integer+3]) ''' Addition of 3 because of exclusive nature
of slicing'''
```

Q.5 Write a python program to read a string expression from a console and print the length of the expression.

Answer:

```
expression = input()
print("Length of the expression is",len(expression))
```

Q.1 Write a python program to read an integer value from console. If the input number is a natural number, then please do the following.

1. If input number is even, print “{number} is an even natural number”.
2. If input number is odd, print “{number} is an odd natural number”.

If the input number is not a natural number, then please do the following.

1. If input number is even, print “{number} is an even number”.
2. If input number is odd, print “{number} is an odd number”. A number is said to be natural if it is greater than 0.

Answer:

```
n = int(input())
if(n > 0): if(n%2==0):
   print("{} is an even natural number".format(n))
else:
   print("{} is an odd natural number".format(n))
else: if(n%2==0):
print("{} is an even number".format(n))
else:
   print("{} is an odd number".format(n))
```

Q.2 Write a python program to read a string from the console. Read another string which is a pattern. If the input starts or ends with the given input pattern, print “{input_string} contains {pattern}” else print “{input_string} does not contain {pattern}”.Note: String comparison is case insensitive.

Answer:

```
str = input()
pattern = input()
if(str.lower().startswith(pattern.lower()) or
str.lower().endswith(pattern.lower())):
   print("{} contains {}".format(str, pattern))
else:
   print("{} does not contain {}".format(str, pattern))
```

Q.3 Write a python program to read an integer from console as “Marks”. Assign and print grade for the input marks. If marks >= 80, then print “{marks} will grant you a grade A”. If marks >= 60 and marks < 80, then print “{marks} will grant you a grade B”. If marks < 60, then print “{marks} will grant you a grade C”.

Answer:

```
marks = int(input())
if(marks >= 80):
   print("{marks} will grant you a grade A".format(marks=marks))
elif(marks >= 60 and marks < 80):
   print("{marks} will grant you a grade B".format(marks=marks))
elif(marks < 60):
   print("{marks} will grant you a grade C".format(marks=marks))
```

Q.4 Write a python program to read a float number from console as "Salary". Print the salary with hike. If salary is greater than or equal to 10000, hike should be 10% of the salary. If salary is greater than or equal to 6000 and less than 10000, hike should be 8% of the salary. If salary is less than 6000, hike should be 5% of the salary. Round the updated salary to 2 digits.

Answer:

```
salary = float(input())

if(salary >= 10000):
   salary = salary + salary * 0.1
elif(salary >= 6000 and salary < 10000):
   salary = salary + salary * 0.08
elif(salary < 6000):
   salary = salary + salary * 0.05

salary = round(salary,2)
print(salary)
```

Q.5 Write a menu driven python program. Read two integer values from console as “a” and “b”. Create a following menu after reading a and b. 1. Addition2. Subtraction3. Exponent4. Multiplication Read an integer from console as “choice”. If choice is equal to 1, print “{a} + {b} = {a+b}”.If choice is equal to 2, print “{a} - {b} = {a-b}”.If choice is equal to 3, print “{a} ^ {b} = {a**b}”.If choice is equal to 4, print “{a} * {b} = {a*b}”.If choice is none of the above (1,2,3,4), print “Invalid choice! Exiting application”

Answer:

```
a = int(input())
b = int(input())
print("1. Addition","2. Subtraction", "3. Exponent", "4.Multiplication",sep="\n")
ch = int(input())
if(ch==1):
   print("{a} + {b} = {c}".format(a=a, b=b, c = a+b))
elif (ch==2):
   print("{a} - {b} = {c}".format(a=a, b=b, c = a-b))
elif (ch==3):
   print("{a} ^ {b} = {c}".format(a=a, b=b, c = a**b))
elif (ch==4):
   print("{a} * {b} = {c}".format(a=a, b=b, c = a*b))
else:
   print("Invalid choice! Exiting application")
```

Q.1 Write a python program to read an integer from console as “x”. Print “true” if x is a palindrome number, otherwise print “false”.

Answer:

```
x = int(input())
if(str(x)[::-1] == str(x)):
  print("true")
else:
  print("false")
```

Q.2 Write a python program to read two strings from console as "str1" and "str2". Write a logic to check if str1 and str2 are anagrams. If strings are anagram, print "{str1} == {str2} is true". If strings are not anagram, print "{str1} == {str2} is false". An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Answer:

```
str1 = input()
str2 = input()
if "".join(sorted(str1)) == "".join(sorted(str2)):
print("{str1} == {str2} is true".format(str1=str1, str2=str2))
else:
    print("{str1} == {str2} is false".format(str1=str1, str2=str2))
```

THANK YOU

Q1. Write a Python program to read a word from the console. Print the total number of vowels in the word.

```
word = input()

count = 0

for i in word:
   # check if each letter is in list of vowels
   if i.lower() in ['a', 'e', 'i', 'o', 'u']:
      count+=1

print(count)
```

Q2. Write a Python program to read a number n. Print prime if the number is prime. Else print not prime

```
n = int(input())

p = True

for i in range(2, int(n**.5)+1):

   if n%i==0:
      p = False

if p and n>1:
   print('prime')
else:
   print('not prime')
```

Q3. Read a word and a letter from console. Print the word after deleting all the occurances of the letter.

```
word = input()

letter  = input()

for i in word:
   if i!=letter:
      print(i, end = '')
```

Q4. Write a Python program to reverse a string.

```
word = input()

print(word[::-1]) # using string slicing
```

Q5. Write a Python program to read a number from console and print the sum of their digits

```
n = input()

sum = 0

for i in n:
   sum += int(i)

print(sum)
```

Q6. Write a Python program to read numbers from a single line and print the missing numbers. Numbers will be in consecutive order.

```
n = input()
input_numbers = n.split(' ')
input_numbers = [int(num) for num in input_numbers] # converting string input to integer

begin = input_numbers[0]
end = input_numbers[-1]

total_numbers = [i for i in range(begin, end+1)]

ans = list(set(total_numbers).difference(set(input_numbers))) #performing set difference like maths


for i in ans:
    print(i, end=" ")
```

Q7. Write a program to read a sentence and a word from console. Print the total number of occurances of the word in the following format:

The word 'x' appears 'y' times

Replace x and y with the word and the total number of occurance.
Comparison must be case insensitive

```
text = input().split(' ')

w = input()

count = 0

for word in text:
    if word.lower() == w.lower():
        count+=1

print('The word {} appears {} times'.format(w, count)) # string formatting
```

Q8. Write a program to read a word from console. Print the length of longest repeating charecter streak.

```
text = input()

max_count = 0
count = 1

for i in range(len(text)-1):

    # check if current letter is equal to next letter
    if text[i]==text[i+1]:
        print(text[i], text[i+1])
        count+=1
    else:
        max_count = max(max_count, count)
        count = 1
```

```
max_count = max(max_count, count)

if max_count>1:
    print(max_count)
```

```
Part 1:
#sample static list
ls = [1,2,3,"One",'Two',"""Three""",1.2,1.3,{"name":"Furkhan"}]
print(ls)

#String indexing
print(ls[0],ls[len(ls)-1],ls[-1], sep=" ")

#String slicing
print("\nSlicing output")
print(ls[0:], ls[::-1], ls[2:4],sep="\n")


print("\n For loop")
#Index based access
print("1. Index based")
for i in range(len(ls)):
    print(ls[i], end=" ") #here i is index not the value. ls[i] represents value

#Value based access
print("\n2. Value based")
for i in ls:
    print(i, end=" ") # here i is a value not index.

Part 2
================================================================================
ls = []
#append, extend, insert
ls.append(10) # add 10 to the end of ls
ls.extend([11,12,13]) #add 11, 12, 13 into the list as integer elements
ls.insert(1,22) #add 22 at index 1
print("Insertion results",ls,sep="\n")

#pop, pop(n), remove()
deleted_last = ls.pop() #returns the deleted element
print("ls.pop()",ls,sep=" => ")
deleted = ls.pop(1) #deletes an element from index 1
print("ls.pop(1)",ls,sep=" => ")
ls.remove(11) #deletes the specified element, do not return anything
print("ls.remove(11)",ls,sep=" => ")


Part 3
================================================================================
string = input("Enter the values to add\n")
ls = string.split(" ")

#map
ls = list(map(lambda x:int(x),ls))
print(sum(ls))
```

```
#write same code using list comprehension

Part 4
================================================================================
#list to string
ls = ["A","E","I","O","U"]
string = "".join(ls) # returns a string joined by ""
print(string)


================================================================================
Problem statement - 2
class Solution:
    def isValid(self, string):
        openStack = []
        openBrace = ["(","[","{"]
        closeBrace = [")","]","}"]
        flag = True

        for i in range(len(string)):
            if string[i] in closeBrace:
                if len(openStack) == 0:
                    flag = False
                    break
                else:
                    corr = openBrace[closeBrace.index(string[i])]
                    if corr == openStack[-1]:
                        openStack.pop()
                        flag = True
                        continue
                    else:
                        flag = False
                        break
            elif string[i] in openBrace:
                openStack.append(string[i])
        if len(openStack) > 0:
            flag = False
        return flag

Problem statement - 3:
string = input()
string_ls = string.split("#")
count = 0
for i in string:
    if i == '#':
        string_ls.append("#")
print("".join(string_ls))
```

TATA CONSULTANCY SERVICES

TATA

Python: List in python

Experience certainty.

IT Services
Business Solutions
Outsourcing

Agenda

- List
- Indexing and Slicing
- Iterating a list: for and foreach approach
- List methods
- Problem statement – 1
- Problem statement – 2
- Problem statement – 3

List in python

- Lists are used to store multiple items in a single variable.
- List items are ordered, changeable, and allow duplicate values.

  Ex.

  thislist = ["apple", "banana", "cherry"]

  print(thislist)
- The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.
- A list can contain different data types.
- It is also possible to use the list() constructor when creating a new list.

List indexing and slicing

- Lists can be indexed as well as sliced just like strings.
- List is an array but heterogeneous.
- First element of a list can be accessed using index 0 and last element using len(ls)-1.
- List supports reverse traversal as well. We can use -1 index to access last element of the list.
- We can access different parts of the list using list slicing.
- ls[start:end:step] #end is exclusive

Iterating a list

- You can iterate a list using any loop.
- Most recommended is a for loop. We can access a list using following two ways:
  1. Index based for loop.
  2. Element based for loop.
- Index based for loop will use range() function to iterate over a loop using index starting from 0 to len(ls) – 1.
- Element based for loop is a for-each loop, used to access the list element by element.

List methods

Method	Descitption
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
index()	Returns the index of the first element with the specified value
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Problem statement - 1

John is a faculty at ABC Pvt ltd. He have been given a list of names and have been asked to sort the list based on the length of the name.

Sample input 1:

ls = ['suresh','ram','Jayesh','Abhijeet']

Sample output 1:

ls = ['ram','suresh','Jayesh','Abhijeet']

Problem statement - 2

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

- Open brackets must be closed by the same type of brackets.
- Open brackets must be closed in the correct order.

Example 1:

Input: s = "()"

Output: true

Example 2:

Input: s = "(]"

Output: false

Problem statement - 3

Given a string with special character “#”. Write a python program to shift all the #s to the end of the string and print the output.

Input 1

#welc#om#e#

Output 1

welcome####

Input 2

##My##N#AM#E

Output 2

MyNAME######

Searching and Sorting with Collections

TCS Digital
Test Preparation Series

WAP to search input value in a list. Create a list by taking input from user. Also check if the search value is numeric or a string.

Sample Input 1:
Enter number of elements : 4
26
54
56
12
Enter element to search: 56
Sample Output 1:
56 is Numeric
Yes! 56 exists in list ['26', '54', '56', '12']

Sample Input 2:
Enter number of elements : 4
12
34
67
87
Enter element to search: 90
Sample Output 2:
90 is Numeric
No! 90 does not exists in list ['12', '34', '67', '87']

Sample Input 3:
Enter number of elements : 3
tom
32
a
Enter element to search: tom
Sample Output 3:
tom is a String
Yes! tom exists in list ['tom', '32', 'a']

Sample Input 4:
Enter number of elements : 3
tom
a
43
Enter element to search: dan
Sample Output 4:
dan is a String
No! dan does not exists in list ['tom', 'a', '43']

STEPS:

1. Create empty list; list=[]
2. Take input from user, how many elements in the list!
3. Take input of elements of list and store in a list.
4. Take input of the element(ele) to be searched.
5. Iterate the list to check whether 'ele' exist or not.
6. Also, check if 'ele' isnumeric or not.
7. Display the result in the same format shown in the samples.

SOLUTION

```
lst = []
flag=0
n = int(input("Enter number of elements : "))

for i in range(0, n):
    ele = input()
    lst.append(ele)

search_ele = input("Enter elements to search: ")
for i in lst:
    if(i == search_ele):
        flag+=1
if(str(search_ele).isdigit()):
    print(search_ele,"is Numeric")
else:
    print(search_ele,"is a String")

if(flag!=0):
    print("Yes!",search_ele,"exists in list",lst)
else:
    print("No!",search_ele,"does not exists in list",lst)
```

Search numeric values from array between given range of two input values (define a function to set logic to count in a range)

Sample Input 1:
Enter number of elements in array:6
12
3
5
7
9
10
Enter range start point1
Enter range end point:8
Sample Output 1:
Values between 1 and 8 are: 3

Sample Input 2:
Enter number of elements in array:5
23
34
56
90
76
Enter range start point1
Enter range end point:20
Sample Output 2:
No values found

SOLUTION

```
def countInRange(arr, n, x, y):

    count = 0;
    for i in range(n):
        if (arr[i] >= x and arr[i] <= y):
            count += 1
    return count

arr = []
n=int(input("Enter number of elements in array:"))
for i in range(0,n):
    ele=int(input())
    arr.append(ele)

arr_size = len(arr)
i = int(input("Enter range start point:"))
j = int(input("Enter range end point:"))

num=countInRange(arr, arr_size, i, j)
if(num!=0):
    print("Values between",i,"and",j,"are:",num)
else:
    print("No values found")
```

WAP to take input array from user and the find second largest value in numeric array.

Sample Input 1:
Enter number of elements in array:6
23
90
43
67
21
78
Sample Output 1:
In Array: [23, 90, 43, 67, 21, 78]
Second largest element in the list is: 78

Sample Input 2:
Enter number of elements in array:1
43
Sample Output 2:
Invalid Input!

Sample Input 3:
Enter number of elements in array:3
23
23
23
Sample Output 3:
In Array: [23, 23, 23]
Second largest element in the list is: 23

SOLUTION

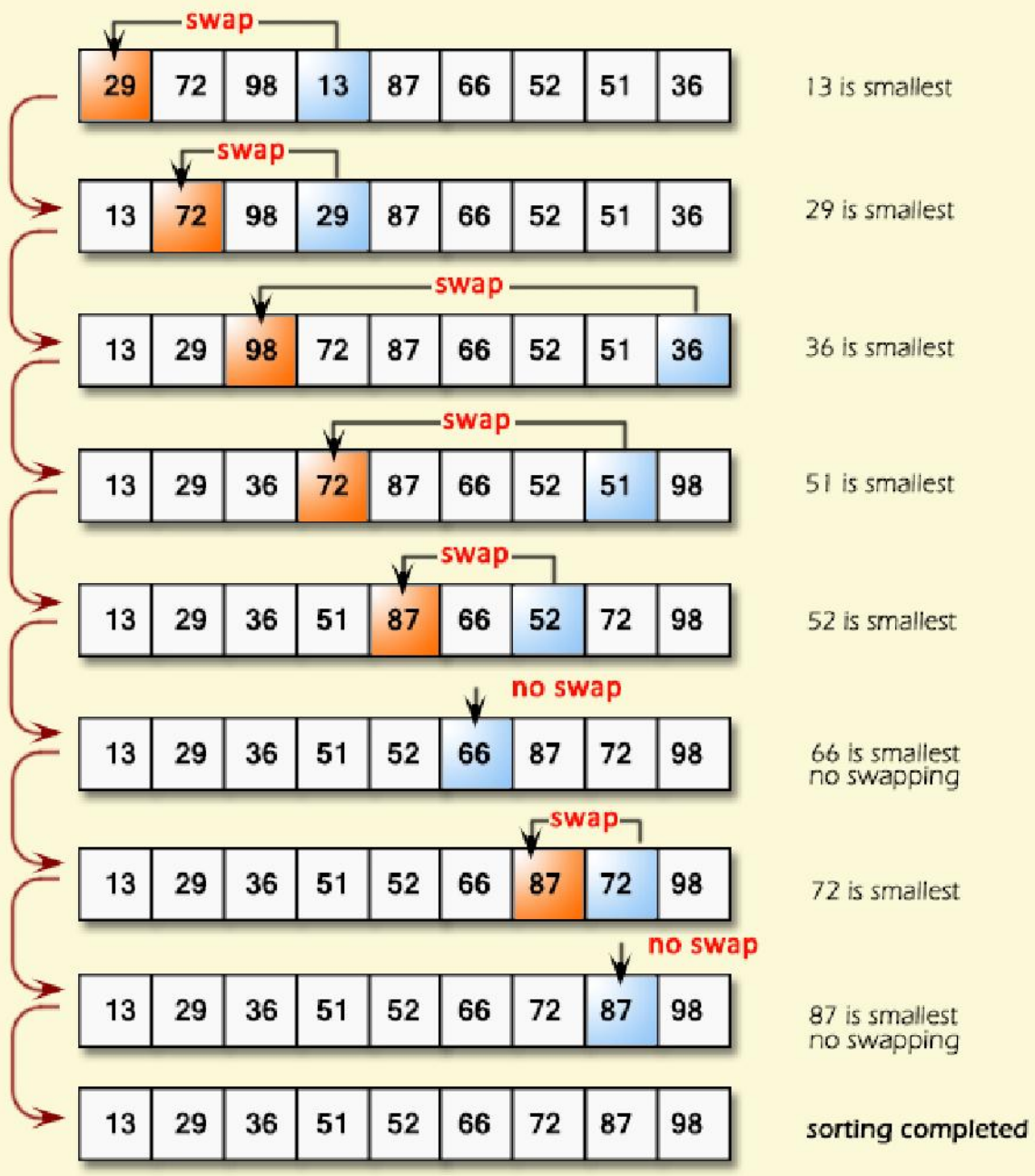
```
arr = []
n=int(input("Enter number of elements in array:"))
for i in range(0,n):
    ele=int(input())
    arr.append(ele)

if (len(arr) < 2):
        print("Invalid Input!")
else:
    arr.sort()
    print("In Array:",arr)
    print("Second largest element in the list is:", arr[-2])
```

WAP to create an array by taking input from user and then print the array and the sorted array (using selection sort algorithm).

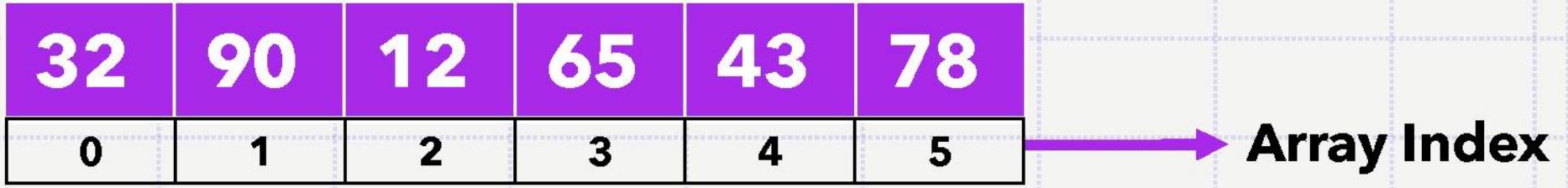
Sample Input 1:
Enter number of elements:6
Enter list elements:
32
90
12
65
43
78

Sample Output 1:
Original List is: [32, 90, 12, 65, 43, 78]
Sorted List is: [12, 32, 43, 65, 78, 90]

Selection Sort



Flow of the Program



Iteration 1 of loop1 :
x=0

Iteration 1 of loop2 :
y=1
if my_list[0]> my_list[1] 32>90 False

Iteration 2 of loop2 :
y=2
if my_list[0]> my_list[2] 32>12 True
temp=32
my_list[0]=12
my_list[2]=32

Iteration 3 of loop2 :
y=3
if my_list[0]> my_list[3] 32>65 False

Iteration 4 of loop2 :
y=4
if my_list[0]> my_list[4] 32>43 False

Iteration 5 of loop2 :
y=5
if my_list[0]> my_list[5] 32>78 False

array: 12 90 32 65 43 78

SOLUTION

```
my_list=[]
n=int(input("Enter number of elements:"))

print("Enter list elements:")
for i in range(n):
    ele=int(input())
    my_list.append(ele)

print("Original List is:",my_list)

for x in range(len(my_list)):
    for y in range(x+1,len(my_list)):
        if(my_list[x]>my_list[y]):
            temp=my_list[x]
            my_list[x]=my_list[y]
            my_list[y]=temp

print("Sorted List is:",my_list)
```

WAP to search how many numbers in given array are divisible by input value. Also, display those numbers.

Given Array: [10,20,34,57,33,21,30]
Sample Input 1:
Enter element for divisibility:2
Sample Output 1:
Array is : 10 20 34 57 33 21 30
4 elements in array are divisible by 2 which are: [10, 20, 34, 30]

Given Array: [10,20,34,57,33,21,30]
Sample Input 2:
Enter element for divisibility:3
Sample Output 2:
Array is : 10 20 34 57 33 21 30
4 elements in array are divisible by 3 which are: [57, 33, 21, 30]

SOLUTION

```
import array as arr

count=0
lst=[]
a = arr.array('i', [10,20,34,57,33,21,30])
ele=int(input("Enter element for divisibility:"))

for i in range (len(a)):
    if(a[i]%ele==0):
        count=count+1
        lst.append(a[i])

print ("Array is :",end=" ")
for i in range (0, len(a)):
    print (a[i],end=" ")

print("\n",count,"elements in array are divisible
by",ele,"which are:",lst)
```

Assignment_Questions_Week3

Questions based on Collections - Array & List basics, Logic Building with Collections and Searching/Sorting with Collections - Python

Q1. Write a Python program to get the number of occurrences of a specified element in an array. Create an array by taking inputs from user.

Sample Input:

Enter size of array:4

4

5

4

4

Enter element to check occurence:4

Sample Output:

Original Array:4 5 4 4

Number of occurrences of the number 4 in the said array: 3

Q2. Write a Python program to remove all duplicate elements from a given array and returns a new array

Sample 1:

Original array:1 3 5 1 3 7 9

After removing duplicate elements from the said array:1 3 5 7 9

Sample 2:

Original array:2 4 2 6 4 8

After removing duplicate elements from the said array:2 4 6 8

Q3. Write a Python program to count the number of strings where the string length is 2 or more and the first and last character are same from a given list of strings.

Sample List : ['abc', 'xyz', 'aba', '1221']
Expected Output: 2

Q4. Write a Python program to test if List contains elements in Range. Take list as input from user and range starting and end value. If elements in lists are present in mentioned range print True else False.

Sample Input 1:

Enter number of elements: 6
4
5
6
7
3
9
Enter range start value:3
Enter range end value:10

Sample Output 1:

The original list is: [4, 5, 6, 7, 3, 9]
Does list contain all elements in range: True

Sample Input 2:

Enter number of elements: 5
2
8
4
7
10
Enter range start value:2
Enter range end value:9

Sample Output 2:

The original list is: [2, 8, 4, 7, 10]
Does list contain all elements in range: False

Q5. Write a python program to sort tuples by total digits (in ascending order).

For example-

Input: [(3, 4, 6, 723), (1, 2), (134, 234, 34)]

Output: [(1, 2), (3, 4, 6, 723), (134, 234, 34)]

Explanation : Digits in first tuple=2, second tuple=6, then third tuple=8. So, 2 < 6 < 8, sorted by increasing total digits.

Sample-

The original list is: [(3, 4, 6, 723), (1, 2), (12345,), (134, 234, 34)]

Sorted tuples: [(1, 2), (12345,), (3, 4, 6, 723), (134, 234, 34)]

Q6. Write a python program to create a student record, which will give us final report in form of tuple, displaying all the records of all students.

We will take 5 inputs from user:
1. Students count
2. Student name
3. Student roll no.
4. Number of subjects for each student
5. Marks for each subject

And according to these inputs, every student data should be displayed in a single tuple and all data of all students should be a nested tuple.

Sample Input-

```
Student count:3
Enter name : Ash
Enter roll no : 1
Enter no. of subjects : 2
Enter marks : 20
Enter marks : 30
Enter name : Tom
Enter roll no : 2
Enter no. of subjects : 3
Enter marks : 10
Enter marks : 40
Enter marks : 50
Enter name : Zubi
Enter roll no : 3
Enter no. of subjects : 2
Enter marks : 50
Enter marks : 20
```

Sample Output-

```
Tuple :  (('Ash', '1', ['20', '30']), ('Tom', '2', ['10',
'40', '50']), ('Zubi', '3', ['50', '20']))
```

Q7. Write a Python program to combine two dictionaries by adding values for common keys.

For example-

d1 = {'a': 100, 'b': 200, 'c':300}

d2 = {'a': 300, 'b': 200, 'd':400}

({'a': 400, 'b': 400, 'd': 400, 'c': 300})

Q8. Create a dictionary by taking input from user. Take 3 inputs:

- Number of key:value pairs
- Keys
- Values

Write a logic to print this dictionary. Now take another input as a value that needs to be searched in the dictionary. Then print if that value is present in dictionary values or not and also convert it into Uppercase, in the format as shown in the samples below.

Sample Input 1-

Enter number of pairs:3
Enter key:1
Enter value: one
Enter key:2
Enter value: two
Enter key:3
Enter value: three
Enter value to search: two

Sample Output 1-

Dictionary is: {'1': 'one', '2': 'two', '3': 'three'}
Yes! TWO is present in this dictionary

Sample Input 2-

Enter number of pairs:2
Enter key:4
Enter value: four
Enter key:5
Enter value: five
Enter value to search: two

Sample Output 2-

Dictionary is: {'4': 'four', '5': 'five'}
No! TWO is not present in this dictionary

Q9. Write a python program to take input from user (irrespective of data type) and store it in a list and print the sorted list first in ascending order and then in descending order.

Sample Input 1-

90,12,45,32,87,45,65,13,74

Sample Output 1-

[12, 13, 32, 45, 45, 65, 74, 87, 90]
[90, 87, 74, 65, 45, 45, 32, 13, 12]

Sample Input 2-

42, -9, 0, 8.7

Sample Output 2-

[-9, 0, 8.7, 42]
[42, 8.7, 0, -9]

Sample Input 3-

orange, apple, mango, kiwi, strawberry

Sample Output 3-

['apple', 'kiwi', 'mango', 'orange', 'strawberry']
['strawberry', 'orange', 'mango', 'kiwi', 'apple']

Q10. Write a python program to sort the dictionary first on the basis of key and then values.

Sample Dictionary:

{'beth': 37, 'zane': 32,'john': 41, 'amy': 41, 'mike': 59, 'jane': 43}

Output:

[('amy', 41), ('beth', 37), ('jane', 43), ('john', 41), ('mike', 59), ('zane', 32)]
[('zane', 32), ('beth', 37), ('john', 41), ('amy', 41), ('jane', 43), ('mike', 59)]


THANK YOU

```
# Q1
# Read a list from input and print the minimum and maximum elements in the list

n = int(input())

l = []

for i in range(n):
    l.append(int(input()))

min_element = l[0]
max_element = l[0]

for element in l:

    if element>max_element:
        max_element = element

    if element<min_element:
        min_element = element


print(f'Max element is {max_element}')
print(f'Min element is {min_element}')

# Q2
# Read a list from input. Print average of all the elements in the list.

# sum(all elements)/total number of elements

n = int(input())

l = []

for i in range(n):
    l.append(int(input()))

sum = 0

for element in l:
    sum += element

# n is the total number of elements in the list l
print(f'Average of the list is {sum/n}')

# Q3
# Read a list from input. Print the subset array of all the odd numbers in the list.

# [1, 3, 4, 5, 6, 8]
# [1, 3, 5]

n = int(input())

l = []

for i in range(n):
    l.append(int(input()))
```

```
list_of_odds = []

for element in l:

    if element%2 !=0:
        list_of_odds.append(element)

print(list_of_odds)

# Q4
# Read a list from input. Swap the elements in the list in the middle.

# Sample case1
# Input: [1, 2, 3, 4, 5, 6] Output: [4, 5, 6, 1, 2, 3]

# Sample case2
# Input [1, 2, 3, 4, 5] Output: [4, 5, 3, 1, 2]


import math


n = int(input())

l = []

for i in range(n):
    l.append(int(input()))

# even number of elements
if n%2 == 0:

    elements_in_single_half = int(n/2)
    first_half = l[:elements_in_single_half]
    second_half = l[elements_in_single_half:]
    print(second_half + first_half)


# odd number of elements
else:
    middle = int(math.floor(n/2))
    first_half = l[:middle]
    second_half = l[middle+1:]
    print(second_half + [l[middle]] + first_half)
    # [4, 5] + [3] + [1, 2]


# Q5
# Read a list from input. Find count of prime numbers in the list

n = int(input())

l = []

for i in range(n):
    l.append(int(input()))

prime_count = 0
```

```
for element in l:
    if element == 1 or element ==0:
        continue
    else:
        prime = True

        for number in range(2,element):
            if element%number == 0:
                prime = False

        if prime == True:
            prime_count += 1

print(f'There are {prime_count} prime numbers in list')
```

TATA CONSULTANCY SERVICES

TATA

Python: Logic building with Collections

Experience certainty. IT Services
Business Solutions
Outsourcing

Agenda

- Find min and max in a list
- Find average of a list
- Create subset array of odd numbers in a list
- List swapping

Min and Max of a list

Read a list from input and print the minimum and maximum elements in the list

Average

Read a list from input. Print average of all the elements in the list.

Subset array

Read a list from input. Print the subset array of all the odd numbers in the list.

Question

Read a list from input. Swap the elements in the list in the middle.

Sample case1

Input: [1, 2, 3, 4, 5, 6] Output: [4, 5, 6, 1, 2, 3]

Question

Read a list from input. Find count of prime numbers in the list.

Thank You

```
Part - 1
'''
WAP to print sum of input numbers
'''
# traditional approch
numbers = input().split(" ")
nums = []
for i in numbers:
    nums.append(int(i))
numbers = nums.copy()
print(sum(numbers))

# list comprehension
numbers = input().split(" ")
numbers = [int(i) for i in numbers]
print(sum(numbers))

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

Part - 2
# dictionaries, tuples and set

#1. Tuple
numbers  = (1,2,3,4,5,6)
for i in numbers:
    print(i, end=" ")

#2. Dictionaries
data = {
    "Name":"Furkhan",
    "Age":22,
    "CTDT":"ABC@123"
}
print(data)

#3. set
s = {1,2,3,4,5}
print(s)

# given a list with duplicate data, print a list with unique values
ls = [1,1,1,2,3,4,4,5,5]
ls = list(set(ls))
print(ls)
========================================================================
```

Problem statement – 1:

In the game of scrabble, in order to avoid over-usage of the same letters in any word, Mario is trying to calculate if a letter appears more than three times in any word and wants to discard such words. In order to assist Mario, write a program to identify the number of times the most repeating letter would appear within any word. If the output number is more than three, Mario shall discard such words and choose another word for the game.

Sample input -1
trigger

Sample output -1
2

=>
using list comprehension:

```
data = input()
frequencies = [data.count(i) for i in data]
print(max(frequencies))
```

Without list comprehension:

```
data = input()
frequencies = []

for i in data:
   frequencies.append(data.count(i))
print(max(frequencies))
```

Problem statement – 2:

A faulty program stores values from the username and password fields as a continuous sequence of characters. When trying to fetch either the username or password, the entire string is displayed Given the string str, the task here is to find and extract only the letters and special characters from the whole string stored.

Note:

The letters and special characters should be displayed separately.

The output should consist of all the letters, followed by all the special characters present in the input string.

Sample input: admin@1234

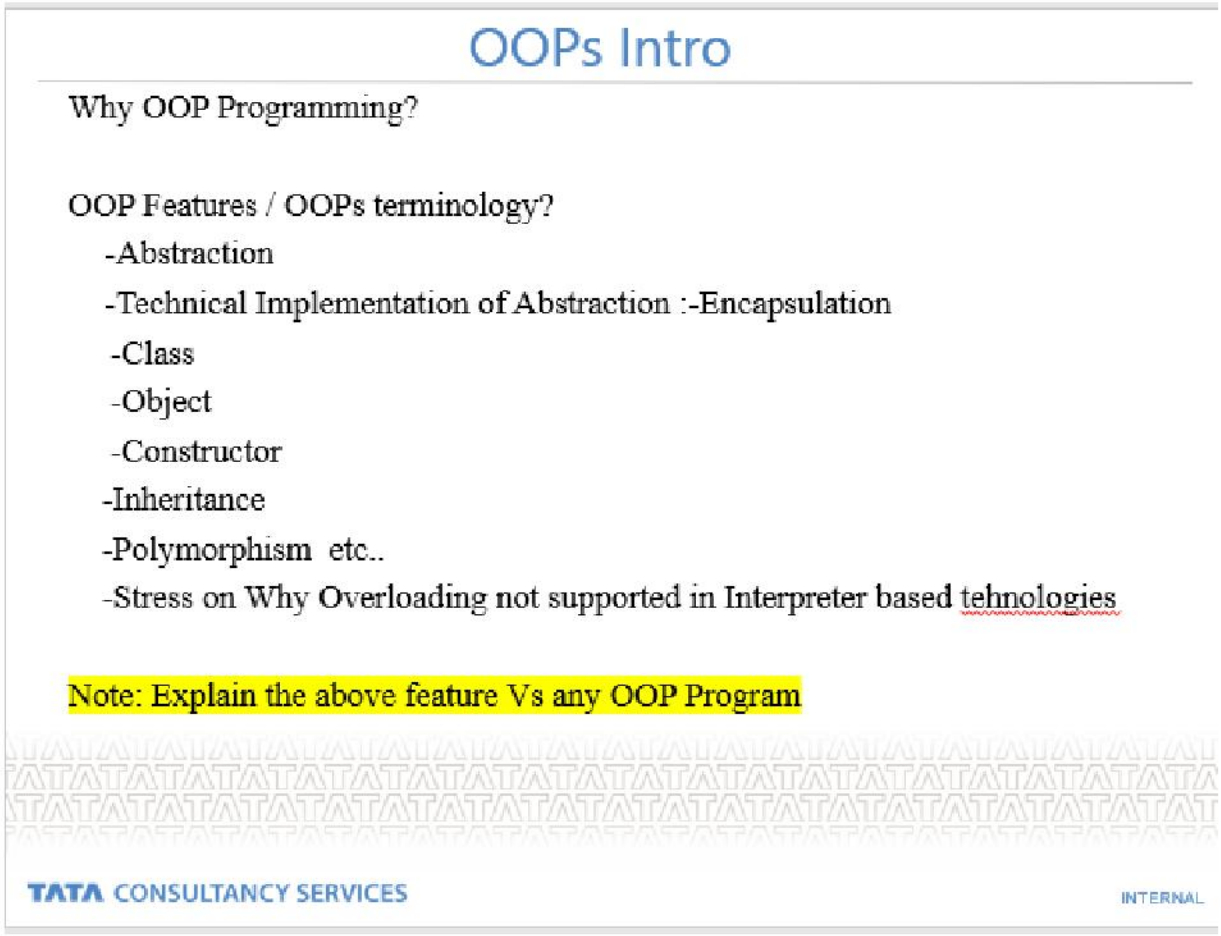
Sample output: admin@

Sample input: user@abc#12

Sample output: userabc@#

```
data = input()
str = [i for i in data if i.isdigit() == False]
special = [i for i in str if i.isalpha() == False]
str = [i for i in str if i.isalpha() == True]
print("".join(str)+"".join(special))
```

Presentation Slides:



Classes/Objects

Python has been an object-oriented language from day one. Because of this, creating and using classes and objects are downright easy.

Creating Classes:

The class statement creates a new class definition. The name of the class immediately follows the keyword class followed by a colon as follows:

```
class ClassName:
  'Optional class documentation string'
  class_suite
```

The class has a documentation string, which can be accessed via ClassName.__doc__.

The class_suite consists of all the component statements defining class members, data attributes and functions.

Classes/Objects

```
class Employee:
   'Common base class for all employees'
   empCount = 0

   def __init__(self, name, salary):
      self.name = name
      self.salary = salary
      Employee.empCount += 1

   def displayCount(self):
     print "Total Employee %d" % Employee.empCount

   def displayEmployee(self):
      print "Name : ", self.name,  ", Salary: ", self.salary
```

Classes/Objects

The variable empCount is a class variable whose value would be shared among all instances of a this class. This can be accessed as Employee.empCount from inside the class or outside the class.

The first method __init__() is a special method, which is called class constructor or initialization method that Python calls when you create a new instance of this class.

You declare other class methods like normal functions with the exception that the first argument to each method is self. Python adds the self argument to the list for you; you don't need to include it when you call the methods.

Classes/Objects

Creating instance objects:

To create instances of a class, you call the class using class name and pass in whatever arguments its __init__ method accepts.

```
"This would create first object of Employee class"
emp1 = Employee("Zara", 2000)
"This would create second object of Employee class"
emp2 = Employee("Manni", 5000)
```

Classes/Objects

Show Example?

Programs:

1. Simple OOP based Program with a function which reads the records from the text file .
   Converts the record into individual fields,
   creating the object with fields of data read from the file and
   Adding the object to the list of objects.
   Reading The list of objects and displaying the data.

```
#Doctor class
class Doctor:
    doctors=[]
    def __init__(self,doctor_name,doctor_specialisation,doctor_fee):
        self.doctor_name=doctor_name
        self.doctor_specialisation=doctor_specialisation
        self.doctor_fee=doctor_fee
    def __str__(self):
        return self.doctor_name+'\t'+self.doctor_specialisation+'\t'+str(self.doctor_fee)
    @classmethod
    def create_doctors(cls):
        with open('doctors_list.txt','r+') as doctor:
            d=doctor.readlines()
            for i in d:
                j=i.split('|')
                obj=Doctor(j[0],j[1],j[2])
                Doctor.doctors.append(obj)

if __name__=="__main__":
    Doctor.create_doctors()
    for i in Doctor.doctors:
        print (i)
```

```
#  textfile records below from file: doctors_list.txt - which is read by the above program #
Miky|cardio|500
Sai|heart|1000
Satish|eye|500
Vivek|dentist|5000
Murali|dermo|500
Rupak|rmp|100
Saketha|cardio|500
Pavan|kids|500
Nag|kids|500
```

Gopi|eye|500
Rajath|heart|500

Program2: On various types of access specifiers – and accessing them

```
#Public access specifiers
class employee:
   def __init__(self, name, sal):
      self.name=name
      self.salary=sal

e1=employee("Kiran",10000)
print(e1.salary)
e1.salary=20000
print(e1.salary)
#**********************************

#Protected sample - No Use

class employee1:
   def __init__(self, name, sal):
      self._name=name
      self._salary=sal

e2=employee1("Kiran",10000)
print(e2._salary)
e2._salary=20000
print(e2._salary)

#**********************************

#Private Sample - Still not much secured
class employee2:
   def __init__(self, name, sal):
      self.__name=name
      self.__salary=sal

#This way of private variable variable access would not work
#e3=employee2("Kiran",10000)
#print(e3.__salary)
#e3.__salary=20000
#print(e3.__salary)
```

 # Contradicting instruction – This was we can access Private variable still

```
e3=employee2("Kiran",10000)
e3._employee2__salary=22222
print(e3._employee2__salary)
```

Program3 and 4 below: Simple OOP based programs which reads,store the data into objects

Creating multiple entities as per the real time requirement – **Employee** and **Organization** and keeping the properties(variables/data members), constructor, behavior(methods/member functions) under the right entiry(class design) and defining the methods to access the data members using the object and Object creation

```
class Employee:
   def __init__(self,name,id,age,gender):
      self.eid=id
      self.ename=name
      self.age=age
      self.gender=gender

class Organisation:
      def __init__(self,name,elist):
         self.oname=name
         self.elist=elist

      def addEmployee(self,id,name,age,gender):
         print("In addEmployee function")
         e=Employee(id,name,age,gender)
         self.elist.append(e)
         print("End addEmployee function")

      def viewEmployees(self):
         print("In View Employee function")
         for e in self.elist:
            print(e.eid)
            print(e.ename)
            print(e.age)
            print(e.gender)
         print("End View Employee function")

      def getEmployeeCount(self):
         print("In countEmployee function")
         print("END countEmployee function")
         return len(self.elist)

      def findEmployeeAge(self,id):
```

```
        print("In findEmployee function")
        age=-1
        for e in self.elist:
            if e.eid == id:
                age=e.age
                break
        print("End findEmployee function")
        return age

    def countEmployees(self,age):
        count=0
        print("In countEmployee AGE WISE function")
        for e in self.elist:
            if e.age > age:
                count=count+1
        print("In countEmployee AGE WISE function")

        return count


if __name__ == '__main__':
    employees=[]
    o = Organisation('ABC',employees)
    n=int(input())
    for i in range(n):
        name=input()
        id=int(input())
        age=int(input())
        gender=input()
        o.addEmployee(name,id,age,gender)

    o.viewEmployees()

    print(o.getEmployeeCount())

    id=int(input())
    print(o.findEmployeeAge(id))

    age=int(input())
    print(o.countEmployees(age))
```

Program4:

```
class Student:
    def __init__(self,sid,sname,cname,score):
        self.studentId = sid
        self.studentName = sname
        self.courseEnrolled = cname
        self.studentScore = score


class Department:
    def __init__(self,dname,slist):
        self.departmentName = dname
        self.studentList = slist

    def findCourseWiseStudents(self):
        result={}
        cList = []
        for student in self.studentList:
            cList.append(student.courseEnrolled)
        courseList = list(set(cList))
        for course in courseList:
            result[course] = cList.count(course)
        return result

    def findStudentGrade(self,sid):
        grade = None
        for student in self.studentList:
            if student.studentId == sid:
                if student.studentScore >= 80:
                    grade = 'A'
                elif student.studentScore >= 65:
                    grade = 'B'
                elif student.studentScore >= 55:
                    grade = 'C'
                else:
                    grade = 'F'
        return grade
```

```
if __name__ == '__main__':
    count = int(input())
    sList = []
    for i in range(count):
        sid = int(input())
        sname = input()
        course = input()
        score = float(input())
        sList.append(Student(sid,sname,course,score))
    d = Department("Humanities",sList)
    inputId = int(input())
    outDict = d.findCourseWiseStudents()
    for course in sorted(outDict.keys()):
        print(course,outDict[course])
    outGrade = d.findStudentGrade(inputId)
    if outGrade is None:
        print('No Student Found')
    else:
        print(outGrade)
```

Program5:

Full-fledged program which demonstrates

-Object collaboration concept

-Constructors

-Members function which access the objects data , compare/process the two objects data

-Adding the Object into dictionary of objects , traversing and reading the data from dictionary of objects ->Creating the objects with the data read from the dictionary of objects and adding the object to lost of objects.

-Traversing the list / dictionary of objects to individual data members

```
class Book:
   def __init__(self,id,name,technology):
      self.id=id
      self.name=name
      self.technology=technology

class BookStore:
 #self.books={}
   def __init__(self,bookDict):
      self.bookdb=bookDict

   def searchByTechnology(self,tech):
      abc=[]
      restech=[]
      flag=0
      #Traversing the dictionary of object
      for ab in self.bookdb.keys():
            #print (self.bookdb[ab].id)
            #print(tech)
            #print(self.bookdb[ab].technology)
            if(tech==self.bookdb[ab].technology):
               print(self.bookdb[ab].technology)
               restech.append(self.bookdb[ab]) #Adding the object to the list of objects
               flag=1
      if flag==0:
            abc.append("NULL")
            abc.append("NULL")
            abc.append("NULL")
```

```
            abc.append("NULL")
            restech.append(abc)
        return restech
          #return (self.bookdb[ab])

    def compareBooks(self,a,b):
        if(a.name==b.name):
            print("Books are same")
        else:
            print("Books are not same")


if __name__ == '__main__':
    bookdb={}
    bookCount_master = int(input())
    for i in range(bookCount_master):
        id=input()
        name = input()
        technology= input()
        bookObj=Book(id,name,technology)
        bookdb.update({i: bookObj})
    bookStoreObj=BookStore(bookdb)

    technology_searchFor= input()
    restech=bookStoreObj.searchByTechnology(technology_searchFor)
    for k in restech:
        print(k.id)
        print(k.name)
        print(k.technology)

    id1=input()
    name1 = input()
    technology1= input()
    bookObj1=Book(id1,name1,technology1)

    id2=input()
    name2 = input()
    technology2= input()
    bookObj2=Book(id2,name2,technology2)

    bookStoreObj.compareBooks(bookObj1,bookObj2)
```