Ex.No.1	USE OF CONTROL STATEMENTS	Reg.No: URK24CS1189
16.12.24		

1a. Write a python program to find the factorial of a number using a while loop.

Aim: To implement decision making and looping using while, for and if-else loops for the given programs.

Description / Algorithm:
Step1: Get the number (n) whose factorial is to be found.
Step2: Give variable 'f' value 1.
Step3: Use a while loop where the condition is i < n+1.
Step4: While i < n+1, do f = f * i.
Step5: Print 'f'.

Program:

```
#URK24CS1189
num=int(input("enter the number:"))
f=1
i=1
while i<num+1:
    f=f*i
    i=i+1
print("factorial of ",num," = ",f)
```

Output:

```
enter the number:5
factorial of  5  =  120
```

Result: Thus, a python program to find the factorial of a number using a while loop has been done successfully.

1b. Write a program to check whetehr a number is prime or not.

Aim: To implement decision making and looping using while, for and if-else loops for the given programs.

Description / Algorithm:
Step1: Get the number (n) as an integer.
Step2: If n <= 1, then print "not prime".
Step3: For i = 2 to n/2, if (n%i == 0), then print "not prime".
Step4: Else print "prime number".

Program:

```
#URK24CS1189
num=int(input("enter a number:"))
if(num>1):
    for i in range(2,num):
        if num%i==0:
            print(f"{num} is a prime number")
            break
    else:
        print(f"{num} is a prime number")
else:
    print(f"{num} is negative")
```

Output:

```
enter a number:7
7 is a prime number

=== Code Execution Successful ===
```

Result: Thus, a program to check whetehr a number is prime or not has been done successfully.

1c. Write a program to check whether the given number is palindrome or not.

Aim: To implement decision making and looping using while, for and if-else loops for the given programs.

Description / Algorithm:
Step1: Get the number as string.
Step2: Check whether the number is equal to n [::-1].
Step3: If it is equal then print 'number is a palindrome'.
Step4: If it is not equal then print 'number is not a palindrome'.

Program:

```
#URK24CS1189
num=input("enter a  number:")
if num==num[::-1]:
    print("palindrome")
else:
    print("not palindrome")
```

Output:

Result: Thus, a program to check whether the given number is palindrome or not has been done successfully.

1d. Write a program to check whether a number is armstrong or not.

Aim: To implement decision making and looping using while, for and if-else loops for the given programs.

Description / Algorithm:
Step1: Take an integer input n.
Step2: Initialize a variable num_digits = 0.
Step3: Set temp = n.
Step4: While temp > 0, increment num_digits by 1 and update temp to temp//10.
Step5: Initialise sum = 0.
Step6: Set temp = n again.
Step7: While temp > 0, get last digit using digit = temp % 10.
Step8: Add digits^num_digits to sum.
Step9: Update temp to temp // 10.
Step10: If sum = n, print 'Armstrong number' else, print 'Not an armstrong number'.

Program:

```
n=int(input("enter a number:"))
numdig=0
temp=n
while temp>0:
    numdig+=1
    temp=temp//10
sum=0
temp=n
while temp>0:
    dig=temp%10
    sum=sum+dig**numdig
    temp=temp//10
if sum==n:
    print("armstrong")
else:
    print("not armstrong")
```

Output:

Result: Thus, a program to check whether a number is armstrong or not has been done successfully.

Ex.No.2	LIST, TUPLE AND SET IN PYTHON	Reg.No: URK24CS1189
9.1.25		

2a. Write a python program to remove duplicate elements in a list.

Aim: To implement list, tuple and set methods for the given programs.

Description / Algorithm:
Step1: Initialize an empty list output_list.
Step2: Iterate through each item in input_list.
Step3: If item is not in output_list, append it to output_list.
Step4: Return output_list after the iteration.

Program:

```
#URK24CS1189
inp_lst=[1,2,3,4,5,1,3]
out_lst=[]
for i in inp_lst:
    if i not in out_lst:
        out_lst.append(i)
print("original list: ",inp_lst)
print("list after removing duplicates :",out_lst)
```

Output:

```
original list:  [1, 2, 3, 4, 5, 1, 3]
list after removing duplicates : [1, 2, 3, 4, 5]

=== Code Execution Successful ===
```

Result: Thus, a python program to remove duplicate elements in a list has been done successfully.

2b.Write a python program to print each and every element in reverse order.

Aim: To implement list, tuple and set methods for the given programs.

Description / Algorithm:
Step1: Iterate through the input_list in reverse order.
Step2: Print each item as you iterate.

Program:

```
inp_lst=[1,2,3,4,5]
new_lst=[]
print("original list: ",inp_lst)
for i in inp_lst[::-1]:
    new_lst.append(i)
print("reversed list :",new_lst)
```

Output:

```
original list:  [1, 2, 3, 4, 5]
reversed list : [5, 4, 3, 2, 1]

=== Code Execution Successful ===
```

Result: Thus, a python program to print each and every element in reverse order has been done successfully.

2c. Write a pyhton program demonstrating all the methods in set.

Aim: To implement list, tuple and set methods for the given programs.

Description / Algorithm:
Step1: Create a set: Initialize a set with some elements.
Step2: Demonstrate the following set methods: add( ) , update( ) , remove( ) , discard( ) , pop( ) , clear( ) , copy( ) , union( ) , intersection( ) , difference( ) , symmetricdifference( ) , issubset( ) , issuperset( ) , isdisjoint( ) , len( ) , max( ) , & min( ) .

Program:

```
#URK24CS1189
set={1,2,3,4,5}
print("Original set : ",set)
print("\nAfter adding an element : ",set.add(6))
print("\nAfter updating : ",set.update([7,8,9]))
print("\nAfter removing 9 : ",set.remove(9))
print("\nAfter removing an element that does not exist : ",set.discard(10))
popped_element = set.pop()
print("\nPopped Element: ", popped_element)
print("Set after pop: ", set)
print("\nAfter clearing the set : ",set.clear())
set={1,2,3,4,5}
my_set_copy = set.copy()
print("\nOriginal Set:", set)
print("Copied Set:", my_set_copy)
set_b = {4, 5, 6, 7}
union_set = set.union(set_b)
print("\nUnion of my_set and set_b:", union_set)
intersection_set = set.intersection(set_b)
print("\nIntersection of my set and set_b:", intersection_set)
difference_set = set.difference(set_b)
print("\nDifference between my set and set_b:", difference_set)
symmetric_difference_set = set.symmetric_difference(set_b)
print("\nSymmetric Difference between set and set_b:", symmetric_difference_set)
is_subset = set.issubset(set_b)
print("\nIs my_set a subset of set_b?", is_subset)
is_superset = set.issuperset(set_b)
print("\nIs my_set a superset of set_b?", is_superset)
is_disjoint = set.isdisjoint(set_b)
print("\nIs my_set disjoint with set_b?", is_disjoint)
set_length = len(set)
print("\nLength of my_set:", set_length)
max_element = max(set)
min_element = min(set)
print("\nMax element in my_set:", max_element)
print("Min element in my_set:", min_element)

```

Output:

Result: Thus, a python program demonstrating all the methods in set has been done successfully.