Guru Nanak Dev Engineering College

Artificial Intelligence

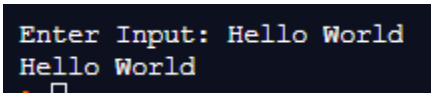
Practical

Submitted By: Taranjeet Singh (D3CSE) 1805996

Sr. No.	Program Name
1	Print Hello World
2	Addition of 2 Integer, Float, Complex Number
3	WAP to check whether a number is Odd or Even
4	WAP to find the Greatest of 4 numbers
5a	WAP to check whether a number is prime or not, using while loop
5b	WAP to print all the prime numbers in a range using for loop
6	WAP for a calculator using function without parameters
7	WAP for a calculator using a function with parameters
8	WAP to demonstrate Default parameters in a function.
9	WAP to illustrate function with *args as a parameter.
10	WAP to implement classes and objects.
11	WAP to implement queue using list.
12	WAP to implement Stack using list.
13.	WAP to implement water jug problem

Q1. WAP Print Hello World?

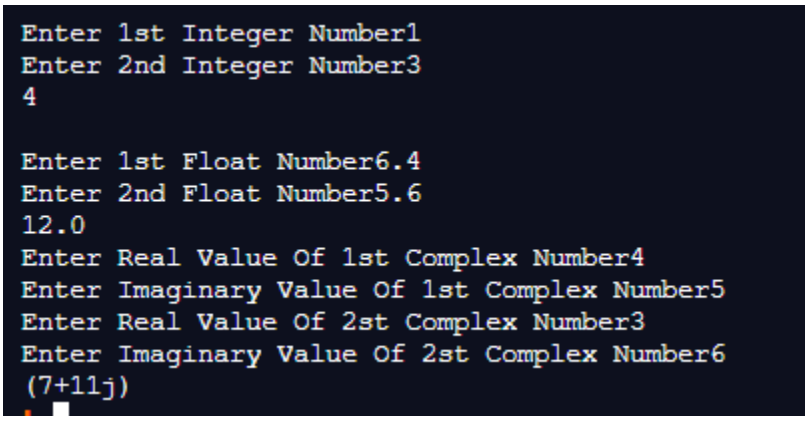
```
print (input ())
```



Q2. WAP Addition of 2 Integer, Float, Complex Number?

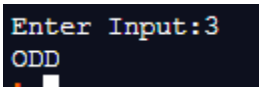
```
a, b = int(input("Enter 1st Integer Number")), int(
   input("Enter 2nd Integer Number"))
print(a + b)
```

```
c, d = float(input("Enter 1st Float Number")), float(
   input("Enter 2nd Float Number"))
print(c + d)


e = complex(
   int(input("Enter Real Value Of 1st Complex Number")),
   int(input("Enter Imaginary Value Of 1st Complex Number")))
f = complex(
   int(input("Enter Real Value Of 2st Complex Number")),
   int(input("Enter Imaginary Value Of 2st Complex Number")))
print(e + f)
```



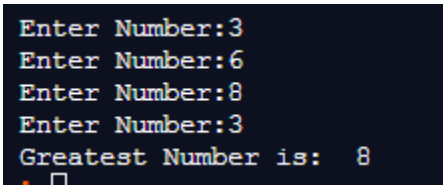
Q3.WAP to check whether a number is Odd or Even?

```
x = int(input("Enter Input:"))

if x%2 == 0:
  print("EVEN")
else:
  print("ODD")
```



Q4. WAP to find the Greatest of 4 numbers?

```
l = []

for x in range(4):
  x = int(input("Enter Number:"))
  l.append(x)

print(max(l))
```



Q5(a). WAP to check whether a number is prime or not using a while loop?

```
var = num = int(input("Enter Number You Wanna Test: "))
i = 2

while var>(num//2):

   if (num % i) == 0:
      print(num,"is not A Prime Number")
      break
   i+=1
   var-=1
else:
   print(num,"is a prime number")
```

```
Enter Number You Wanna Test: 17
17 is a prime number
```

Q5(b). WAP to print prime number using for loop?

```
for i in range(int(input("Enter Starting Point:")),int(input("Enter Ending Point:"))):
 if i>1:
  for j in range(2,i):
    if(i % j==0):
      break
  else:
    print(i)
```

```
Enter Starting Point:10
Enter Ending Point:20
11
13
17
19
```

Q6. WAP for a calculator using function without parameters?

```
num1,num2 = 3,5
print("Name: Taranjeet Singh \nURN: 1805996")
def add():
  print(num1+num2)
def sub():
```

```
    print(num1-num2)
def multiply():
    print(num1*num2)
def divide():
    print(num1/num2)
x = True
print("Please select operation\n 1. Add\n 2. Subtract\n 3. Multiply\n 4. Divide\n 5. Exit")
# Take input from the user
while x:
    select = int(input("Select operations form 1, 2, 3, 4, 5 :"))
    if select == 1:
        add()
    elif select == 2:
        sub()
    elif select == 3:
        multiply()
    elif select == 4:
        divide()
    elif select == 5:
        print("Thanks For Using.")
        x = False
    else:
        print("Invalid input")
```

```
Name: Taranjeet Singh
URN: 1805996
Please select operation
 1. Add
 2. Subtract
 3. Multiply
 4. Divide
 5. Exit
Select operations form 1, 2, 3, 4, 5 :5
Thanks For Using.
```

Q7. WAP for a calculator using function with parameters?

```
print("Name: Taranjeet Singh \nURN: 1805996")
def add():
   num1,num2 = int(input("Enter 1st Number: ")),int(input("Enter 2st Number: "))
   print(num1+num2)
def add():
   num1,num2 = int(input("Enter 1st Number: ")),int(input("Enter 2st Number: "))
   print(num1+num2)
def sub():
   num1,num2 = int(input("Enter 1st Number: ")),int(input("Enter 2st Number: "))
   print(num1-num2)
def multiply():
   num1,num2 = int(input("Enter 1st Number: ")),int(input("Enter 2st Number: "))
   print(num1*num2)
def divide():
   num1,num2 = int(input("Enter 1st Number: ")),int(input("Enter 2st Number: "))
   print(num1/num2
x = True
print("Please select operation\n 1. Add\n 2. Subtract\n 3. Multiply\n 4. Divide\n 5. Exit")
```

```
# Take input from the user
while x:
    select = int(input("Select operations form 1, 2, 3, 4, 5 :"))
    if select == 1:
        add()
    elif select == 2:
        sub()
    elif select == 3:
        multiply()
    elif select == 4:
        divide()
    elif select == 5:
        print("Thanks For Using.")
        x = False
    else:
        print("Invalid input")
```

```
Name: Taranjeet Singh
URN: 1805996
Please select operation
 1. Add
 2. Subtract
 3. Multiply
 4. Divide
 5. Exit
Select operations form 1, 2, 3, 4, 5 :3
Enter 1st Number: 5
Enter 2st Number: 7
35

Select operations form 1, 2, 3, 4, 5 :
```

Q8. WAP to demonstrate Default parameters in a function.

```
print("Name: Taranjeet Singh \nURN: 1805996")
def add(num = 10, num2 = 20):
   print(num+num2)
print("Without Parameters")
add()
print("With Parameters")
add(50,50)
```

```
Name: Taranjeet Singh
URN: 1805996
Without Parameters
30
With Parameters
100
```

Q9. WAP to illustrate function with *args as a parameter?

```
def add(*args):
   value = 0
   for x in args:
      value+=x
   print(value)
print("Name: Taranjeet Singh \nURN: 1805996")
add(1,3,5,7,9,2,4)
```

```
Name: Taranjeet Singh
URN: 1805996
31
```

Q10. WAP to implement classes and objects.

```
class student:
    def __init__(self, name, crn):
        self.name = name
        self.crn = crn
print("Enter name:")
s1.name = input()
print("Enter roll number:")
s1.crn = input()


print(s1.name)
print(s1.crn)
```

```
Enter name:
TARAN
```

Q11. WAP to implement queue using list.

```
class Queue:
    def __init__(self):
        self.queue = []


    def remove_element(self):
        self.queue.pop()


    def append_element(self, val):
        self.queue.append(val)
```

```
    def print_queue(self):
        print(self.queue)
q = Queue();

q.append_element(1)
q.append_element(2)
q.append_element(4)
q.append_element(8)

q.print_queue()

q.remove_element()
q.remove_element()
q.remove_element()

q.print_queue()
```

```
[1, 2, 4, 8]
[1]
```

Q12. WAP to implement Stack using list.

```
class Stack:
    def __init__(self):
        self.stack = []

    def remove_element(self):
```

```
        self.stack.pop(0)

    def append_element(self, val):
        self.stack = [val] + self.stack

    def print_stack(self):
        print(self.stack)

s = Stack()
s.append_element(1)
s.append_element(3)
s.append_element(6)
s.print_stack()
s.remove_element()
s.print_stack()
```

```
[6, 3, 1]
[3, 1]
```

Q13. WAP to implement water jug problem.

```
capacity = (12,8,5)
x = capacity[0]
y = capacity[1]
z = capacity[2]

memory = {}
```

```
ans = []

def get_all_states(state):

 a = state[0]
 b = state[1]
 c = state[2]

 if(a==6 and b==6):
    ans.append(state)
    return True

 if((a,b,c) in memory):
    return False

 memory[(a,b,c)] = 1
 memory

 if(a>0):

    if(a+b<=y):
       memory
       if( get_all_states((0,a+b,c)) ):
          ans.append(state)
          return True
    else:
       if( get_all_states((a-(y-b), y, c)) ):
```

```
            ans.append(state)
            return True

    if(a+c<=z):
        if( get_all_states((0,b,a+c)) ):
            ans.append(state)
            return True
    else:
        if( get_all_states((a-(z-c), b, z)) ):
            ans.append(state)
            return True

if(b>0):

    if(a+b<=x):
        if( get_all_states((a+b, 0, c)) ):
            ans.append(state)
            return True
    else:
        if( get_all_states((x, b-(x-a), c)) ):
            ans.append(state)
            return True

    if(b+c<=z):
        if( get_all_states((a, 0, b+c)) ):
            ans.append(state)
            return True
```

```
    else:
        if( get_all_states((a, b-(z-c), z)) ):
            ans.append(state)
            return True

if(c>0):

    if(a+c<=x):
        if( get_all_states((a+c, b, 0)) ):
            ans.append(state)
            return True
    else:
        if( get_all_states((x, b, c-(x-a))) ):
            ans.append(state)
            return True

    if(b+c<=y):
        if( get_all_states((a, b+c, 0)) ):
            ans.append(state)
            return True
    else:
        if( get_all_states((a, y, c-(y-b))) ):
            ans.append(state)
            return True

return False
```

```
initial_state = (12,0,0)

print("Starting work...\n")

get_all_states(initial_state)

ans.reverse()

for i in ans:

  print(i)
```

```
Starting work...

(12, 0, 0)
(4, 8, 0)
(0, 8, 4)
(8, 0, 4)
(8, 4, 0)
(3, 4, 5)
(3, 8, 1)
(11, 0, 1)
(11, 1, 0)
(6, 1, 5)
(6, 6, 0)
```