1. Create a Integer Variable and convert it to Float, Boolean, String.

```
a = 11
print(type(a))
b = float(a)
c = bool(a)
d = str(a)
print(type(b))
print(type(c))
print(type(d))
```

```
    <class 'int'>
    <class 'float'>
    <class 'bool'>
    <class 'str'>
```

2 Create a Float Variable and convert it to Integer, Boolean and String.

```
a = 1.2
print(type(a))
b = int(a)
c = bool(a)
d = str(a)
print(type(b))
print(type(c))
print(type(d))
```

```
    <class 'float'>
    <class 'int'>
    <class 'bool'>
    <class 'str'>
```

3 Create a Boolean Variable and convert it to Integer, Float and String.

```
a = True
print(type(a))
b = int(a)
c = float(a)
d = str(a)
print(type(b))
print(type(c))
print(type(d))
```

```
    <class 'bool'>
    <class 'int'>
    <class 'float'>
    <class 'str'>
```

4 Create a String Variable and convert it to Integer, Float and Boolean

```
a = '20'
print(type(a))
b = int(a)
c = float(a)
d = bool(a)
print(type(b))
print(type(c))
print(type(d))
```

```
    <class 'str'>
    <class 'int'>
    <class 'float'>
    <class 'bool'>
```

5 Find out values in String, Integer and Float when converting to Boolean it gives False

```
s = ''
i = 0
f = 0.0
print(type(s))
print(type(i))
print(type(f))
print(bool(s))
```

```
print(bool(i))
print(bool(f))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
False
False
False
```

6 Perform operations with all the Arithmetic Operators

```
#int operations
#Addition
a = 33 + 58
print("Addition : ",a)
#Subtraction
s = 34 - 23
print("Subtraction : ",s)
#Multiplication
m = 7 * 6
print("Multiplication : ",m)
#Division
d = 60/4
print("Division : ",d)
#Modulus
mod = 43%5
print("Modulus : ",mod)
#Exponentiation
e = 4**3
print("Exponentiation : ",e)
#Floor division
f = 44//5
print("Floor division : ",f)
```

```
Addition :  91
Subtraction :  11
Multiplication :  42
Division :  15.0
Modulus :  3
Exponentiation :  64
Floor division :  8
```

```
#float operations
#Addition
a = 23.67 + 57.9
print("Addition : ",a)
#Subtraction
s = 34.98 - 23.45
print("Subtraction : ",s)
#Multiplication
m = 7.9 * 6.2
print("Multiplication : ",m)
#Division
d = 60.67/4.2
print("Division : ",d)
#Modulus
mod = 43.9%5
print("Modulus : ",mod)
#Exponentiation
e = 4.78**3
print("Exponentiation : ",e)
#Floor division
f = 44.69//5
print("Floor division : ",f)
```

```
Addition :  81.57
Subtraction :  11.529999999999998
Multiplication :  48.980000000000004
Division :  14.445238095238095
Modulus :  3.8999999999999986
Exponentiation :  109.21535200000001
Floor division :  8.0
```

```
#bool operations
#Addition
a1 = True + True
a2 = True + False
a3 = False + False
print("Addition1 : ",a1)
print("Addition2 : ",a2)
print("Addition3 : ",a3)
#Subtraction
```

```
s = 6 - True
print("Subtraction : ",s)
#Multiplication
m = True * 3
print("Multiplication : ",m)
#Division
d = True/2
print("Division : ",d)
#Modulus
mod = True%5
print("Modulus : ",mod)
#Exponentiation
e = True**3
print("Exponentiation : ",e)
#Floor division
f = True//5
print("Floor division : ",f)
```

```
Addition1 :  2
Addition2 :  1
Addition3 :  0
Subtraction :  5
Multiplication :  3
Division :  0.5
Modulus :  1
Exponentiation :  1
Floor division :  0
```

7 Perform operations with all the Bitwise Operators

```
#& AND Returns 1 if both the bits are 1 else 0.
#| OR Returns 1 if either of the bit is 1 else 0.
#~ NOT Returns one’s complement of the number.
#^ XOR Returns 1 if one of the bits is 1 and the other is 0 else returns false.
#>> Right shift  Shifts the bits of the number to the right and fills 0 on voids left( fills 1 in the case of a negative numb
#<< Left shift  Shifts the bits of the number to the left and fills 0 on voids right as a result. Similar effect as of multip
#Bitwise operators works only on integers (convert integer into binary then return it in decimal)
a = 10
b = 1
print("Bitwise AND : ",a & b)
print("Bitwise OR : ",a|b)
print("Bitwise NOT : ",~b)
print("Bitwise XOR : ",a^b)
print("Bitwise Right Shift : ",a>>1)
print("Bitwise Left Shift : ",a<<1)
```

```
Bitwise AND :  0
Bitwise OR :  11
Bitwise NOT :  -2
Bitwise XOR :  11
Bitwise Right Shift :  5
Bitwise Left Shift :  20
```

8 Perform operations with all the Relational Operators

```
a = 10
b = 5
print(a==b)
print(a!=b)
print(a>=b)
print(a<=b)
print(a>b)
print(a<b)
```

```
False
True
True
False
True
False
```

9 Perform operations with all the Logical Operators

```
# and or not
a = 24
b = 78
if(a<20 and b>60):
  print("and")
if(a>20 or b>90):
```

```
  print("or")
if not(a<20 and b>60):
  print("not")
```

```
   or
   not
```

10 Create a python script/program that will take input from the user for 3 numbers and result will print the biggest number and the smallest number using 'input' and 'print'.

```
no1 = int(input("Enter no1 : "))
no2 = int(input("Enter no2 : "))
no3 = int(input("Enter no3 : "))
l = [no1,no2,no3]
print("Biggest number : ",max(l))
print("Smallest number : ",min(l))
```

```
    Enter no1 : 20
    Enter no2 : 10
    Enter no3 : 5
    Biggest number :  20
    Smallest number :  5
```

```
no1 = int(input("Enter no1 : "))
no2 = int(input("Enter no2 : "))
no3 = int(input("Enter no3 : "))
b = 0
s = 0
if(no1>no2):
  if(no1>no3):
    b = no1
    s = no3
    if(no3>no2):
      s = no2
  else:
    b = no3
    s = no2
elif no2>no1:
  if(no2>no3):
    b = no2
    s = no3
    if no3>no1:
      s = no1
  else:
    b = no3
    s = no1
print("Biggest number : ",b)
print("Smallest number : ",s)
```

```
    Enter no1 : 12
    Enter no2 : 30
    Enter no3 : 4
    Biggest number :  30
    Smallest number :  4
```

11 Create another script/program using 'input' and pass all the three parameters as a single input and execute the same program as mentioned above.

```
no1,no2,no3 = [int(x) for x in input("Enter three values: ").split()]

b = 0
s = 0
if(no1>no2):
  if(no1>no3):
    b = no1
    s = no3
    if(no3>no2):
      s = no2
  else:
    b = no3
    s = no2
elif no2>no1:
  if(no2>no3):
    b = no2
    s = no3
    if no3>no1:
      s = no1
  else:
    b = no3
```

```
    s = no1
print("Biggest number : ",b)
print("Smallest number : ",s)
```

```
Enter three values: 29 67 23
Biggest number :  67
Smallest number :  23
```

12 Print odd numbers between 1 to 10 in reverse order using while.

```
i = 10
while(i!=0):
  if i % 2 != 0:
    print(i)
  i-=1
```

```
9
7
5
3
1
```

13 Perform the same operation with for loop.

```
for i in range(10,0,-1):
  if i % 2 != 0:
    print(i)
```

```
9
7
5
3
1
```

14 Print odd numbers between 1 to 10 using continue in both for and while loop.

```
i = 0
while(i<10):
  i+=1
  if i % 2 == 0:
    continue
  print(i)
```

```
1
3
5
7
9
```

```
for i in range(1,11,1):
  if i%2==0:
    continue
  print(i)
```

```
1
3
5
7
9
```

15 Take 10 numbers in a list(array) and print only first 3 numbers using loop.

```
l = [1,2,3,4,5,6,7,8,9,10]
for i in range(0,3):
  print(l[i])
```

```
1
2
3
```

16 Write a function with recursion to give the factorial of a number.

```
def factorial(no):
  if no == 1:
    return 1
  else:
```

```
  else:
    return no * factorial(no-1)
print(factorial(4))
```

```
24
```

17 Create a two funcitons. Call one function from another function

```
def first():
  return "first"

def sec():
  return first() + "Second"
print(sec())
```

```
firstSecond
```

18 Create a function that will take 5 arguments 2 will be mandatory and 3 will be keyword parameters. If 2 parameters are passed perform multiplication of 2 parameters. If 3 parameters are passed print all the 3 parameters. If 4 parameters are passed addition of 4 parameters. If 5 parameters are passed multiply 2 mandatory parameters and then separately multiply 3 keyword parameters and add both of them.

```
def fun(no1,no2,**kwargs):
  if(len(kwargs) == 0):
    print(no1 * no2)
  if(len(kwargs) == 1):
      for key,value in kwargs.items():
        print(no1,no2,value)
  if(len(kwargs) == 2):
    v = 0
    for key,value in kwargs.items():
      v += value
    v = v + no1 + no2
    print(v)
  if(len(kwargs) == 3):
    m = no1*no2
    v = 1
    for key,value in kwargs.items():
      v *= value
    v = v + m
    print(v)

fun(9,8,a=7,b=9,c=10)
```

```
702
```

19

```
def options(argument):
  no1,no2,no3=0,0,0
  if argument == 1 or argument == 2 or argument == 3:
    no1 = int(input("Enter no1 : "))
    no2 = int(input("Enter no2 : "))
    no3 = int(input("Enter no3 : "))
  else:
    no1 = int(input("Enter no1 : "))
    no2 = int(input("Enter no2 : "))
  switcher = {
        1: add(no1,no2,no3),
        2: sub(no1,no2,no3),
        3: mul(no1,no2,no3),
    4: div(no1,no2),
    5: exp(no1,no2),
    6: floor(no1,no2)
    }
  return switcher.get(argument, "Invalid Input...Enter 1 to 6")

def add(no1,no2,no3):
  return no1+no2+no3

def sub(no1,no2,no3):
  return no1-no2-no3

def mul(no1,no2,no3):
  return no1*no2*no3

def div(no1,no2):
  return no1/no2
```

```
def exp(no1,no2):
  return no1**no2

def floor(no1,no2):
  return no1//no2

if __name__ == "__main__":
  print("1.Addition\n2.Substraction\n3.Multiplication\n4.Division\n5.Exponential\n6.Floor Division")
  argument=int(input("Enter your choice : "))
  print("Answer is ",options(argument))
```

```
1.Addition
2.Substraction
3.Multiplication
4.Division
5.Exponential
6.Floor Division
Enter your choice : 5
Enter no1 : 22
Enter no2 : 11
Answer is  584318301411328
```

20

```
class Example:
  no1 = 0
  no2 = 0
  def __init__(self,no1,no2):
    self.no1 = no1
    self.no2 = no2

  def add(self):
    self.no1 = self.no1 + self.no2

  def show(self):
    print(self.no1)


e = Example(23,45)
e.add()
e.show()
```

```
68
```

21

```
class Parent:
  def a(self):
    print('Parent a method')
  def b(self):
    print('Parent b method')
class Child(Parent):
  def c(self):
    print('Child c')


p = Parent()
p.a()
p.b()
p.c()
```

```
Parent a method
Parent b method
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call
last)
<ipython-input-156-d0ba4d2ba892> in <module>
      2 p.a()
      3 p.b()
----> 4 p.c()

AttributeError: 'Parent' object has no attribute 'c'
```

```
c = Child()
c.a()
c.b()
c.c()
```

```
Parent a method
Parent b method
Child c
```

22

```
class Parent:
  def __init__(self):
    print('Constructer created')

  def __del__(self):
    print('Destructer called')

  def a(self):
    print('Parent a method')
  def b(self):
    print('Parent b method')
class Child(Parent):
  def c(self):
    print('Child c')


p = Parent()
p.a()
p.b()
```

```
Constructer created
Parent a method
Parent b method
```

```
del p
```

```
Destructer called
```

```
c = Child()
c.a()
c.b()
c.c()
```

```
Constructer called
Parent a method
Parent b method
Child c
```

23

```
class Parent():
    def __init__(self):
        self.value = "Inside Parent"

    def show(self):
        print(self.value)

class Child(Parent):
    def __init__(self):
        self.value = "Inside Child"

    def show(self):
        print(self.value)

obj1 = Parent()
obj2 = Child()
obj1.show()
obj2.show()
```

```
Inside Parent
Inside Child
```

```
# multiple inheritance
class Parent1():
    def show(self):
        print("Inside Parent1")

class Parent2():
    def display(self):
        print("Inside Parent2")
```

```
class Child(Parent1, Parent2):
    def show(self):
        print("Inside Child")

obj = Child()
obj.show()
obj.display()
```

```
Inside Child
Inside Parent2
```

```
#multilevel inheritance
class Parent():
    def display(self):
        print("Inside Parent")

class Child(Parent):
    def show(self):
        print("Inside Child")

class GrandChild(Child):
    def show(self):
        print("Inside GrandChild")

g = GrandChild()
g.show()
g.display()
```

```
Inside GrandChild
Inside Parent
```

Colab paid products - Cancel contracts here
✓ 0s completed at 12:56