1.TYPES OF INHERITANCE IN PYTHON

1.SINGLE INHERITANCE

In [2]:
```
class parent:
    def function_1(self):
        print("this funciton is in parent class.  ")
class children(parent):
    def function_2(self):
        print("this function is in child class.")
obj = children()
obj.function_1()
obj.function_2()
```

```
this funciton is in parent class.
this function is in child class.
```

2.MULTIPLE INHERITANCE

In [7]:
```


class Mother:
    mothername = ""

    def mother(self):
        print(self.mothername)


class Father:
    fathername = ""

    def father(self):
        print(self.fathername)


class Son(Mother, Father):
    def parents(self):
        print("Father :", self.fathername)
        print("Mother :", self.mothername)


# Driver's code
s1 = Son()
s1.fathername = "harry"
s1.mothername = "alice"
s1.parents()
```

```
Father : harry
Mother : alice
```

3.MULTILEVEL INHERITANCE

In [9]:
```
class Grandfather:

    def __init__(self, grandfathername):
        self.grandfathername = grandfathername


class Father(Grandfather):
    def __init__(self, fathername, grandfathername):
        self.fathername = fathername

        Grandfather.__init__(self, grandfathername)

class Son(Father):
    def __init__(self, sonname, fathername, grandfathername):
        self.sonname = sonname


        Father.__init__(self, fathername, grandfathername)

    def print_name(self):
        print('Grandfather name :', self.grandfathername)
        print("Father name :", self.fathername)
        print("Son name :", self.sonname)



s1 = Son('ROM', 'ALICE', 'HARRY')
print(s1.grandfathername)
s1.print_name()
```

```
HARRY
Grandfather name : HARRY
Father name : ALICE
Son name : ROM
```

4.HIERARCHICAL INHERITANCE

In [11]:
```
class Parent:
    def func1(self):
        print("This function is in parent class.")


class Child1(Parent):
    def func2(self):
        print("This function is in child 1.")


class Child2(Parent):
    def func3(self):
        print("This function is in child 2.")

object1 = Child1()
object2 = Child2()
object1.func1()
object1.func2()
object2.func1()
object2.func3()
```

```
This function is in parent class.
This function is in child 1.
This function is in parent class.
This function is in child 2.
```

5.HYBRID INHERITANCE

In [13]:
```
class School:
    def func1(self):
        print("This function is in school.")


class Student1(School):
    def func2(self):
        print("This function is in student 1. ")


class Student2(School):
    def func3(self):
        print("This function is in student 2.")


class Student3(Student1, School):
    def func4(self):
        print("This function is in student 3.")

object = Student3()
object.func1()
object.func2()
```

```
This function is in school.
This function is in student 1.
```

2.EXCEPTION HANDLING

1.ZeroDivisionError

marks = 10000 a = marks / 0 print(a)

In [19]:
```
marks = 10000
try:
    a = marks / 0
    print(a)
except:
    print ("An error occurred")
    print(marks)
```

```
An error occurred
10000
```

2.unexpected indent

In [21]:
```
a = [1, 2, 3]
try:
    print ("Second element = %d" %(a[1]))

    # Throws error since there are only 3 elements in array
    print ("Fourth element = %d" %(a[3]))

except:
    print ("An error occurred")
```

```
Second element = 2
An error occurred
```

In [ ]:
```
3.VALUE ERROR
```

In [59]:
```
while True:
...     try:
...         x = int(input("Please enter a number: "))
...         break
...     except ValueError:
...         print("Oops!  That was no valid number.  Try again...")
...
```

```
Please enter a number: 45
```

In [ ]:
```
4.FILE OPENING ERROR
```

In [55]:
```
try:
  f = open("demofile.txt")
  try:
    f.write("Lorum Ipsum")
  except:
    print("Something went wrong when writing to the file")
  finally:
    f.close()
except:
  print("Something went wrong when opening the file")
```

```
Something went wrong when opening the file
```

In [ ]:
```
5.IndentationError
```

In [49]:
```
a = 10
try:
     b = 19
     c = a+b
except:
    print("indentation error is occured")
print(c)
```

```
29
```

6.NAME ERROR

In [52]:
```
try:
  print(s)
except NameError:
  print("Variable s is not defined")
except:
  print("Something else went wrong")
```

```
Variable s is not defined
```

3. REGULAR EXPRESSION

1.FINDALL()

In [64]:
```
import re

word = "this is beautiful world"
x = re.findall("is", word)
print(x)
```

```
['is', 'is']
```

2.SEARCH()

In [73]:
```
import re

par = "This is beautiful world"
x = re.search("world", par)
print(x)
```

```
<re.Match object; span=(18, 23), match='world'>
```

In [74]:
```
import re

par = "This is beautiful world "
x = re.search("\s", par)

print("The located in position:", x.start())
```

```
The located in position: 4
```

3.SPLIT()

In [71]:
```
import re

sen = "This is beautiful world"
x = re.split("\s", sen)
print(x)
```

```
['This', 'is', 'beautiful', 'world']
```

4.sub()

In [77]:
```
import re
part = "this  is beautiful world"
x = re.sub("\s", "n", part)
print(x)
```

```
thisnnisnbeautifulnworld
```

4.aggregate function

In [78]:
```
import numpy as np
arr1 = np.array([10,20,30,40,50])
arr1
b = np.array([20,34,45,56,78])
print(b)
```

```
[20 34 45 56 78]
```

In [79]:
```
arr2 = np.array([[0,10,20],[20,40,50],[30,40,50]])
arr2
```

Out[79]:
```
array([[ 0, 10, 20],
       [20, 40, 50],
       [30, 40, 50]])
```

In [80]:
```
arr3 = np.array([[12,'str',-45,56.8],[34,45,45,89]])
arr3
```

Out[80]:
```
array([['12', 'str', '-45', '56.8'],
       ['34', '45', '45', '89']], dtype='<U32')
```

SUM FUNCTION

In [81]:
```
arr1.sum()
```

Out[81]:
```
150
```

In [83]:
```
arr2.sum(axis = 0)
#vertical
```

Out[83]:
```
array([ 50,  90, 120])
```

In [84]:
```
arr2.sum(axis = 1)
#hortizontal
```

Out[84]:
```
array([ 30, 110, 120])
```

PRODUCT FUNCTION

In [88]:
```
np.prod(arr1)
```

Out[88]:
```
12000000
```

In [89]:
```
#PRODUCT
np.product(arr2,axis = 0)
```

Out[89]:
```
array([    0, 16000, 50000])
```

In [90]:
```
np.prod(arr2,axis = 1)
```

Out[90]:
```
array([    0, 40000, 60000])
```

AVERAGE FUNCTION

In [85]:
```
np.average(arr1)
```

Out[85]:
```
30.0
```

In [86]:
```
#array average x and y axis
np.average(arr2,axis =0)
```

Out[86]:
```
array([16.66666667, 30.        , 40.        ])
```

In [87]:
```
np.average(arr2,axis = 1)
```

Out[87]:
```
array([10.        , 36.66666667, 40.        ])
```

MINIMUM FUNCTION

In [91]:
```
#minmun (min)
arr1.min()
```

Out[91]:
```
10
```

In [92]:
```
arr2.min(axis = 0)
```

Out[92]:
```
array([ 0, 10, 20])
```

In [93]:
```
arr2.min(axis = 1)
```

Out[93]:
```
array([ 0, 20, 30])
```

In [94]:
```
arr2.min(0)
```

Out[94]:
```
array([ 0, 10, 20])
```

In [103…
```
import numpy as np
x = np.random.randint(1, 10, size = (5, 5))
print(x)
print()

y = np.random.randint(1, 10, size = (5, 5))
print(y)

print('\nMinimum Array')
print(np.minimum(x, y))
```

```
[[9 6 4 1 7]
 [9 5 3 1 7]
 [5 5 3 1 3]
 [8 8 6 5 4]
 [7 7 3 3 3]]

[[2 2 9 7 5]
 [2 2 1 3 7]
 [6 1 5 8 6]
 [3 5 3 2 8]
 [1 6 5 6 5]]

-----Minimum Array----
[[2 2 4 1 5]
 [2 2 1 1 7]
 [5 1 3 1 3]
 [3 5 3 2 4]
 [1 6 3 3 3]]
```

MAXIMUM FUNCTION

In [102…
```
import numpy as np

a = np.array([1, 20, 5, 22, 8, 15])
print(a)

b = np.array([12, 9, 3, 42, 6, 33])
print(b)

print('\nMaximum Array')
print(np.maximum(a, b))
```

```
[ 1 20  5 22  8 15]
[12  9  3 42  6 33]

Maximum Array
[12 20  5 42  8 33]
```

MEAN FUNCTION

arr1.mean() arr2.mean()

MEDIAN FUNCTION

In [ ]:
```
np.median(arr1)
np.median(arr2)

np.median(arr2, axis = 0)
np.median(arr2, axis = 1)
```

VARIANCE FUNCTION

In [108…
```
arr1.var()
arr2.var()


x.var()
y.var()
```

Out[108]:
```
5.68
```

CUMULATIVE SUM

In [109…
```
arr1.cumsum()
arr2.cumsum()
```

Out[109]:
```
array([  0,  10,  30,  50,  90, 140, 170, 210, 260], dtype=int32)
```

CUMULATIVE PRODUCT

In [113…
```
arr1.cumprod()
arr2.cumprod()


# Let me find the cumulative product of array elements in x and Y axis
arr2.cumprod(axis = 0)
arr2.cumprod(axis = 1)
```

Out[113]:
```
array([[    0,     0,     0],
       [   20,   800, 40000],
       [   30,  1200, 60000]], dtype=int32)
```

In [ ]:
```
PERCENTILE FUNCTION
```

In [114…
```
np.percentile(arr1, 10)
np.percentile(arr1, 30)
np.percentile(arr1, 50)
np.percentile(arr1, 75)
np.percentile(arr1, 100)
```

Out[114]:
```
50.0
```

ARGMIN FUNCTION

In [115…
```
arr4 = np.random.randint(9, size = (9))
arr4

arr5 = np.random.randint(25, size = (5, 5))
arr5
```

Out[115]:
```
array([[ 9, 12, 20, 18,  4],
       [ 1,  4, 20, 24,  0],
       [ 5, 24,  3, 16, 19],
       [ 6,  9, 10, 14, 12],
       [ 2,  9, 17, 16,  0]])
```

In [ ]: