In [ ]:

```
DATA TYPES
```

In [1]:

```
a = 5
print("Type of a: ", type(a))
```

```
Type of a:  <class 'int'>
```

In [2]:

```
b = 5.0
print("\nType of b: ", type(b))
```

```

Type of b:  <class 'float'>
```

In [3]:

```
c = 2 + 4j
print("\nType of c: ", type(c))
```

```

Type of c:  <class 'complex'>
```

In [4]:

```
d ="saikumar"
print("\nType of d: ", type(d))
```

```

Type of d:  <class 'str'>
```

In [5]:

```
x = ["apple", "banana", "cherry"]
print("\nType of x: ", type(x))
```

```

Type of x:  <class 'list'>
```

In [6]:

```
x = {"name" : "John", "age" : 36}
print("\nType of x: ", type(x))
```

```

Type of x:  <class 'dict'>
```

In [7]:

```
x = {"apple", "banana", "cherry"}
print("\nType of x: ", type(x))
```

```

Type of x:  <class 'set'>
```

In [8]:

```
x = True
print(type(x))
```

```
<class 'bool'>
```

In [155]:

```
LISTS
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:\Users\SAIKUM~1\AppData\Local\Temp/ipykernel_21348/2929072608.py in <modul
e>
----> 1 LISTS

NameError: name 'LISTS' is not defined
```

In [9]:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

```
['apple', 'banana', 'cherry']
```

In [10]:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

```
banana
```

In [11]:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]
thislist[1:3] = ["blackcurrant", "watermelon"]
print(thislist)
```

```
['apple', 'blackcurrant', 'watermelon', 'orange', 'kiwi', 'mango']
```

In [12]:

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```

```
['apple', 'orange', 'banana', 'cherry']
```

In [13]:

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

```
['apple', 'cherry']
```

In [154]:

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

```
apple
banana
cherry
```

In [15]:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if "a" in x]

print(newlist)
```

```
['apple', 'banana', 'mango']
```

In [16]:

```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort()
print(thislist)
```

```
['banana', 'kiwi', 'mango', 'orange', 'pineapple']
```

In [ ]:

```
TUPLES:
```

In [17]:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

```
('apple', 'banana', 'cherry')
```

In [18]:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

```
('apple', 'banana', 'cherry')
```

In [19]:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
```

```
banana
```

In [20]:

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```

```
('apple', 'kiwi', 'cherry')
```

In [21]:

```
fruits = ("apple", "banana", "cherry")

(green, yellow, red) = fruits

print(green)
print(yellow)
print(red)
```

```
apple
banana
cherry
```

In [22]:

```
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")

(green, yellow, *red) = fruits

print(green)
print(yellow)
print(red)
```

```
apple
banana
['cherry', 'strawberry', 'raspberry']
```

In [23]:

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```

```
('a', 'b', 'c', 1, 2, 3)
```

In [ ]:

```
DICTIONARIES
```

In [26]:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

In [27]:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict["brand"])
```

```
Ford
```

In [28]:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "year": 2020
}
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

In [33]:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"year": 2020})
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

In [34]:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"color": "red"})
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

In [35]:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)
```

```
{'brand': 'Ford', 'year': 1964}
```

In [36]:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = thisdict.copy()
print(mydict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

In [ ]:

```
COMPREHENSIONS
```

In [37]:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []

for x in fruits:
  if "a" in x:
    newlist.append(x)

print(newlist)
```

```
['apple', 'banana', 'mango']
```

In [38]:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if "a" in x]

print(newlist)
```

```
['apple', 'banana', 'mango']
```

In [39]:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if x != "apple"]

print(newlist)
```

```
['banana', 'cherry', 'kiwi', 'mango']
```

In [40]:

```
newlist = [x for x in range(20)]

print(newlist)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

In [41]:

```
newlist = [x for x in range(22) if x < 15]

print(newlist)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

In [42]:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x.upper() for x in fruits]

print(newlist)
```

```
['APPLE', 'BANANA', 'CHERRY', 'KIWI', 'MANGO']
```

In [43]:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x if x != "banana" else "orange" for x in fruits]

print(newlist)
```

```
['apple', 'orange', 'cherry', 'kiwi', 'mango']
```

In [44]:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = ['SAI' for x in fruits]

print(newlist)
```

```
['SAI', 'SAI', 'SAI', 'SAI', 'SAI']
```

In [ ]:

```
GENERATORS
```

In [45]:

```
def simpleGeneratorFun():
    yield 1
    yield 2
    yield 3

for value in simpleGeneratorFun():
    print(value)
```

```
1
2
3
```

In [47]:

```
def my_gen():
    n = 1
    print('This is printed first')
    yield n

    n += 1
    print('This is printed second')
    yield n

    n += 1
    print('This is printed at last')
    yield n
```

In [48]:

```
def my_gen():
    n = 1
    print('This is printed first')

    yield n

    n += 1
    print('This is printed second')
    yield n

    n += 1
    print('This is printed at last')
    yield n
for item in my_gen():
    print(item)
```

```
This is printed first
1
This is printed second
2
This is printed at last
3
```

In [49]:

```
def rev_str(my_str):
    length = len(my_str)
    for i in range(length - 1, -1, -1):
        yield my_str[i]
for char in rev_str("hello"):
    print(char)
```

```
o
l
l
e
h
```

In [50]:

```
def simpleGeneratorFun():
    yield 1
    yield 2
    yield 3


x = simpleGeneratorFun()

print(x.next())
print(x.next())
print(x.next())
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
C:\Users\SAIKUM~1\AppData\Local\Temp/ipykernel_21348/2928282754.py in <modul
e>
      7 x = simpleGeneratorFun()
      8 
----> 9 print(x.next())
     10 print(x.next())
     11 print(x.next())

AttributeError: 'generator' object has no attribute 'next'
```

In [51]:

```
def simpleGeneratorFun():
    yield 1
    yield 2
    yield 3

# x is a generator object
x = simpleGeneratorFun()

# Iterating over the generator object using next
print(x.next()) # In Python 3, __next__()
print(x.next())
print(x.next())
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
C:\Users\SAIKUM~1\AppData\Local\Temp/ipykernel_21348/3373262647.py in <modul
e>
      8 
      9 # Iterating over the generator object using next
---> 10 print(x.next()) # In Python 3, __next__()
     11 print(x.next())
     12 print(x.next())

AttributeError: 'generator' object has no attribute 'next'
```

In [52]:

```
my_list = [1, 3, 6, 10]

a = (x**2 for x in my_list)
print(next(a))

print(next(a))

print(next(a))

print(next(a))

next(a)
```

```
1
9
36
100



---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
C:\Users\SAIKUM~1\AppData\Local\Temp/ipykernel_21348/1751448568.py in <modul
e>
     10 print(next(a))
     11
---> 12 next(a)

StopIteration:
```

In [ ]:

```
ITERATORS
```

In [53]:

```
mytuple = ("apple", "banana", "cherry")
myit = iter(mytuple)

print(next(myit))
print(next(myit))
print(next(myit))
```

```
apple
banana
cherry
```

In [54]:

```
mystr = "banana"
myit = iter(mystr)

print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
```

```
b
a
n
a
n
a
```

In [55]:

```
class MyNumbers:
  def __iter__(self):
    self.a = 1
    return self

  def __next__(self):
    x = self.a
    self.a += 1
    return x

myclass = MyNumbers()
myiter = iter(myclass)

print(next(myiter))
print(next(myiter))
print(next(myiter))
print(next(myiter))
print(next(myiter))
```

```
1
2
3
4
5
```

In [56]:

```
class MyNumbers:
  def __iter__(self):
    self.a = 1
    return self

  def __next__(self):
    if self.a <= 40:
      x = self.a
      self.a += 1
      return x
    else:
      raise StopIteration

myclass = MyNumbers()
myiter = iter(myclass)

for x in myiter:
  print(x)
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
```

```
39
40
```

In [57]:

```
mystr = "banana"

for x in mystr:
  print(x)
```

```
b
a
n
a
n
a
```

In [ ]:

```
LOOPS:
```

In [58]:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
apple
banana
cherry
```

In [60]:

```
fruits = ["apple", "mango","guava","banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

```
apple
mango
guava
banana
```

In [61]:

```
fruits = ["apple", "mango","guava","banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    continue
```

```
apple
mango
guava
banana
cherry
```

In [62]:

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

```
0
1
2
3
4
5
6
7
8
9
```

In [63]:

```
for x in range(2, 30, 3):
  print(x)
```

```
2
5
8
11
14
17
20
23
26
29
```

In [64]:

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

```
0
1
2
3
4
5
Finally finished!
```

In [65]:

```
for x in range(6):
  if x == 3: break
  print(x)
else:
  print("Finally finished!")
```

```
0
1
2
```

In [66]:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

In [67]:

```
i = 1
while i < 10:
  print(i)
  i += 1
```

```
1
2
3
4
5
6
7
8
9
```

In [69]:

```
i = 1
while i < 26:
  print(i)
  if i == 13:
    break
  i += 1
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

In [71]:

```
i = 0
while i < 26:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
1
2
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
```

In [72]:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

```
1
2
3
4
5
i is no longer less than 6
```

In [ ]:

```
EXCEPTIONS
```

In [74]:

```
try:
  print(Z)
except:
  print("An exception occurred")
```

```
An exception occurred
```

In [77]:

```
try:
  print(Z)
except NameError:
  print("Variable Z is not defined")
except:
  print("Something else went wrong")
```

```
Variable Z is not defined
```

In [78]:

```
try:
  print("Hello")
except:
  print("Something went wrong")
else:
  print("Nothing went wrong")
```

```
Hello
Nothing went wrong
```

In [80]:

```
try:
  print(Z)
except:
  print("Something went wrong")
finally:
  print("The 'try except' is finished")
```

```
Something went wrong
The 'try except' is finished
```

In [81]:

```
try:
  f = open("demofile.txt")
  try:
    f.write("SAIKUMAR")
  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 [82]:

```
x = "hello"

if not type(x) is int:
  raise TypeError("Only integers are allowed")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\Users\SAIKUM~1\AppData\Local\Temp/ipykernel_21348/1233933522.py in <modul
e>
      2
      3 if not type(x) is int:
----> 4   raise TypeError("Only integers are allowed")

TypeError: Only integers are allowed
```

In [ ]:

```
IF ELSE
```

In [83]:

```
a = 333
b = 2020
if b > a:
  print("b is greater than a")
```

```
b is greater than a
```

In [84]:

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

```
a and b are equal
```

In [86]:

```
a = 2020
b = 334
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a is greater than b
```

In [87]:

```
a = 2020
b = 333
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

```
b is not greater than a
```

In [88]:

```
if a > b: print("a is greater than b")
```

```
a is greater than b
```

In [89]:

```
a = 2
b = 330
print("A") if a > b else print("B")
```

```
B
```

In [90]:

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

```
Both conditions are True
```

In [91]:

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

```
At least one of the conditions is True
```

In [92]:

```
x = 50

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

```
Above ten,
and also above 20!
```

In [ ]:

```
FILES
```

In [93]:

```
f=open("demofile1.txt")
print(f.read())
```

```
Hello! Welcome to demofile.txt
This file is for practicing python.
```

In [94]:

```
f=open("demofile1.txt")
print(f.read(6))
```

```
Hello!
```

In [95]:

```
f=open("demofile1.txt")
print(f.read(50))
```

```
Hello! Welcome to demofile.txt
This file is for pr
```

In [96]:

```
f = open("demofile1.txt", "a")
f.write("Now the file has more content!")
f.close()

f = open("demofile1.txt", "r")
print(f.read())
```

```
Hello! Welcome to demofile.txt
This file is for practicing python.
Now the file has more content!
```

In [97]:

```
f = open("demofile1.txt", "r")
print(f.readline())
print(f.readline())
```

```
Hello! Welcome to demofile.txt

This file is for practicing python.
```

In [ ]:

```
STRINGS:
```

In [98]:

```
a = "Hello,Saikumar"
print(a)
```

```
Hello,Saikumar
```

In [99]:

```
a = "Hello,Saikumar"
print(a[1])
```

```
e
```

In [101]:

```
a = "Hello,Saikumar"
print(len(a))
```

```
14
```

In [102]:

```
txt = "The best things in life are free!"
print("free" in txt)
```

```
True
```

In [103]:

```
txt = "The best things in life are free!"
print("sai" not in txt)
```

```
True
```

In [104]:

```
for x in "saikumar":
  print(x)
```

```
s
a
i
k
u
m
a
r
```

In [ ]:

```
RANGE
```

In [105]:

```
x = range(15)
for n in x:
  print(n)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

In [106]:

```
a = "Hello, World!"
print(len(a))
```

```
13
```

In [107]:

```
x = range(30, 60)
for n in x:
  print(n)
```

```
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
```

In [110]:

```
x = range(3, 60, 5)
for n in x:
  print(n)
```

```
3
8
13
18
23
28
33
38
43
48
53
58
```

In [111]:

```
x = range(30, 860, 50)
for n in x:
  print(n)
```

```
30
80
130
180
230
280
330
380
430
480
530
580
630
680
730
780
830
```

In [ ]:

```
REGULAR EXPRESSIONS:
```

In [114]:

```
import re

txt = "The rain in INDIA"
x = re.search("^The.*INDIA$", txt)

if x:
  print("YES! We have a match!")
else:
  print("No match")
```

```
YES! We have a match!
```

In [115]:

```
import re

txt = "The rain in INDIA"
x = re.findall("in", txt)
print(x)
```

```
['in', 'in']
```

In [116]:

```
import re

txt = "The rain in INDIA"
x = re.findall("sai", txt)
print(x)
```

```
[]
```

In [117]:

```
import re

txt = "The rain in INDIA"
x = re.search("sai", txt)
print(x)
```

```
None
```

In [118]:

```
import re

txt = "The rain in INDIA"
x = re.split("\s", txt, 1)
print(x)
```

```
['The', 'rain in INDIA']
```

In [120]:

```
import re

txt = "The rain in INDIA"
x = re.split("\s", txt, 3)
print(x)
```

```
['The', 'rain', 'in', 'INDIA']
```

In [121]:

```
import re

txt = "The rain in Spain"
x = re.sub("\s", "9", txt, 2)
print(x)
```

```
The9rain9in Spain
```

In [122]:

```
import re

txt = "The rain in Spain"
x = re.search(r"\bS\w+", txt)
print(x.string)
```

```
The rain in Spain
```

In [123]:

```
import re

txt = "The rain in Spain"
x = re.search(r"\bS\w+", txt)
print(x.group())
```

```
Spain
```

In [ ]:

```
OOPS
```

In [125]:

```
class car:
    def __init__(self,modelname, year):
        self.modelname = modelname
        self.year = year
    def display(self):
        print(self.modelname,self.year)

c1 = car("RANGEROVER", 2016)
c1.display()
```

```
RANGEROVER 2016
```

In [127]:

```python
class Dog:
    attr1 = "mammal"

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

Rodger = Dog("Rodger")
Tommy = Dog("Tommy")


print("Rodger is a {}".format(Rodger.__class__.attr1))
print("Tommy is also a {}".format(Tommy.__class__.attr1))

print("My name is {}".format(Rodger.name))
print("My name is {}".format(Tommy.name))
```

```
Rodger is a mammal
Tommy is also a mammal
My name is Rodger
My name is Tommy
```

In [128]:

```python
class Dog:

    attr1 = "mammal"


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

    def speak(self):
        print("My name is {}".format(self.name))

Rodger = Dog("Rodger")
Tommy = Dog("Tommy")

Rodger.speak()
Tommy.speak()
```

```
My name is Rodger
My name is Tommy
```

In [129]:

```
class Bird:

    def intro(self):
        print("There are many types of birds.")

    def flight(self):
        print("Most of the birds can fly but some cannot.")

class sparrow(Bird):

    def flight(self):
        print("Sparrows can fly.")

class ostrich(Bird):

    def flight(self):
        print("Ostriches cannot fly.")

obj_bird = Bird()
obj_spr = sparrow()
obj_ost = ostrich()

obj_bird.intro()
obj_bird.flight()

obj_spr.intro()
obj_spr.flight()

obj_ost.intro()
obj_ost.flight()
```

```
There are many types of birds.
Most of the birds can fly but some cannot.
There are many types of birds.
Sparrows can fly.
There are many types of birds.
Ostriches cannot fly.
```

In [135]:

```
class Base:
    def __init__(self):
        self.a = "SAIKUMAR"
        self.__c = "GANJI"

class Derived(Base):
    def __init__(self):

        Base.__init__(self)
        print("Calling private member of base class: ")
        print(self.__c)


# Driver code
obj1 = Base()
print(obj1.a)
print(obj1.a)
```

```
SAIKUMAR
SAIKUMAR
```

In [ ]:

```
LAMBDA
```

In [136]:

```
six = lambda : 6

result = six()
print(result)
```

```
6
```

In [137]:

```
factorial = lambda a: a*factorial(a-1) if (a>1) else 1

result = factorial(5)
print(result)
```

```
120
```

In [138]:

```
factorial = lambda a: a*factorial(a-1) if (a>1) else 1

result = factorial(15)
print(result)
```

```
1307674368000
```

In [139]:

```
square = lambda a: a*a

result = square(10)
print(result)
```

```
100
```

In [140]:

```
mul = lambda a,b: a*b

result = mul(5,3)
print(result)
```

```
15
```

In [141]:

```
import math

def myfunc(n):
  return lambda a : math.pow(a, n)


square = myfunc(2)
cube = myfunc(3)
squareroot = myfunc(0.5)

print(square(3))
print(cube(3))
print(squareroot(3))
```

```
9.0
27.0
1.7320508075688772
```

In [142]:

```
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(map(lambda x: x * 2 , my_list))

print(new_list)
```

```
[2, 10, 8, 12, 16, 22, 6, 24]
```

In [143]:

```
my_list = [1, 5,343,5,777,8,44,33, 12]

new_list = list(map(lambda x: x * 2 , my_list))

print(new_list)
```

```
[2, 10, 686, 10, 1554, 16, 88, 66, 24]
```

In [ ]:

```
FUNCTIONS:
```

In [146]:

```
def my_function():
  print("Hello Saikumar")
my_function()
```

```
Hello Saikumar
```

In [147]:

```
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

```
Emil Refsnes
Tobias Refsnes
Linus Refsnes
```

In [149]:

```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("sai", "kumar")
```

```
sai kumar
```

In [151]:

```
def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("sai", "vamshi", "dimpu")
```

```
The youngest child is dimpu
```

In [152]:

```
def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

```
I am from Sweden
I am from India
I am from Norway
I am from Brazil
```

In [158]:

```
def my_function(country='australia'):
  print("I am from " + country)

my_function("USA")
my_function("INDIA")
my_function("CANADA")
my_function()
my_function("UK")
```

```
I am from USA
I am from INDIA
I am from CANADA
I am from australia
I am from UK
```

In [157]:

```
def my_function(x):
  return 15 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

```
45
75
135
```

In [ ]: