```
In [ ]:
```

Finding the missing number in an array is a popular coding interview question. According to LeetCode, this question is popular in the interviews of companies like Amazon, Adobe, Microsoft, LinkedIn, and many more. If you want to learn how to find the missing number in an array, this article is for you. This article will take you through how to find the missing number using Python.

```
In [ ]:
```

Fundamentals of Python

Python is a versatile programming language loved by so many data scientists, web developers, and even software engineers. To master Python for any profession, the most important step is to master the fundamentals of the Python programming language. So if you want to learn the fundamentals of the Python programming language, this article is for you. In this article, I'm going to walk you through the fundamentals of Python that you need to know before you jump into data science, web development, or any field that interests you.

Data Types in Python:

1. Function
2. List
3. Tuples
4. Dictionaries
5. If-Else Statment
6. Loops
7. Class

Function:

Python has a lot of built-in functions like print, len, input, etc. Besides the built-in functions of Python, you can also define your functions. A function is a block of code defined to perform a certain action. They are primarily used to replace repetitive statements in your code. You can read more about implementing functions in Python from here.

1. A Python function to print factorial of a number
2. A Python function to print the greatest number
3. 5 basic functions for calculations

```
In [ ]:
```

In [ ]:

In [1]:
```
def factorial(n) -> object:
    return 1 if (n==1 or n==0) else n * factorial(n-1)
n= int(input("enter number"))
print(factorial(n))


def greatest(x,y,z):
    if (x>y)and(x>z):
        print(x,"is greatest")
    elif (y>x)and (y>z):
        print(y,"is greatest")
    else:
        print(z,"is greatest")


def addtwo(x,y):
    return x+y

def subtracttwo(x,y):
    return x-y

def multiplytwo(x,y):
    return x*y

def dividetwo(x,y):
    return x/y

def square(x):
    return (x*x)
```

```
1
```

I hope you liked this concept on examples of functions in Python.

In [ ]:

List:

Lists are ordered collections of items and the most general data type provided by the Python programming language. They are mutable, which means that items stored in a list can be edited. A list is typically used to perform operations on a collection of items at a time. You can read more about implementing lists in Python from here.

In [2]:
```
## List Example:

L = [123, 'spam', 1.23]
len(L)
```

Out[2]:
```
3
```

we can also do indexing and slicing on it:

```
In [3]: L[0]
```

```
Out[3]: 123
```

```
In [4]: L[:-1]
```

```
Out[4]: [123, 'spam']
```

```
In [ ]:
```

Type Specific Operations

Lists in Python may be reminiscent of arrays in other languages, but they tend to be more powerful. On the one hand, they don't have a fixed type constraint, the list we just looked at, for example, contains three objects of completely different types. Also, the lists do not have a fixed size. In other words, they can grow and shrink on demand, in response to specific list operations:

```
In [6]: L.append('NI')
        L
```

```
Out[6]: [123, 'spam', 1.23, 'NI', 'NI']
```

```
In [8]: L.pop(2)
```

```
Out[8]: 1.23
```

```
In [10]: M = ['bb', 'aa', 'cc']
         M.sort()
         M
```

```
Out[10]: ['aa', 'bb', 'cc']
```

```
In [11]: M.reverse()
         M
```

```
Out[11]: ['cc', 'bb', 'aa']
```

```
In [ ]:
```

Nested Lists in Python

```
In [14]: M = [[1, 2, 3],
              [4, 5, 6],
              [7, 8 ,9]]
         M
```

```
Out[14]: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

I hope this concept understood List

Tuples:

A tuple is also a collection of items very similar to a list in Python. Like lists, they allow you to store an ordered collection of items to perform operations at one time. But unlike lists, a tuple cannot be modified once created. The only advantage of using tuples over lists is that tuples are slightly faster than lists. You can read more about implementing tuples in Python from here.

```
In [16]: T = (1, 2, 3, 4)
         len(T)

Out[16]: 4

In [18]: T + (5, 6)

Out[18]: (1, 2, 3, 4, 5, 6)

In [19]: T[0]

Out[19]: 1

In [20]: T.index(4)

Out[20]: 3

In [21]: T.count(4)

Out[21]: 1

In [22]: T[0]

Out[22]: 1

In [24]: T = (2,) + T[1:]
         T

Out[24]: (2, 2, 3, 4)

In [26]: T = 'spam', 3.0, [11, 22, 33]
         T[1]

Out[26]: 3.0

In [27]: T[2][1]

Out[27]: 22
```

I hope this concept understood Tuples

Dictionary

A dictionary in Python is completely different from lists and tuples, they are not sequences but mappings. Mappings are also collections of items but in the form of key and value pairs. Simply put, a dictionary contains indexes with keys mapped to certain values. You can read more about the implementation of dictionaries in Python from here.

In [31]:
```
D = {'food': 'Spam', 'quantity': 4, 'color': 'pink'}
D
```

Out[31]:
```
{'food': 'Spam', 'quantity': 4, 'color': 'pink'}
```

In [32]:
```
D['food']
```

Out[32]:
```
'Spam'
```

In [34]:
```
D['quantity'] += 1
D
```

Out[34]:
```
{'food': 'Spam', 'quantity': 5, 'color': 'pink'}
```

In [35]:
```
D = {}
D['name'] = 'Bob'
D['job'] = 'dev'
D['age'] = 40
D
```

Out[35]:
```
{'name': 'Bob', 'job': 'dev', 'age': 40}
```

In [ ]:

If Else Statement:

If-else statements are conditional statements found in all programming languages. These statements are used to write an event-driven program. Simply put, these statements are used when we want to execute a set of statements only if the given condition is met. You can read more about implementing if-else statements in Python from here.

1. Body Mass Index Calculator
2. Calculating Discount on Sale
3. Checking Eligibility for voting

In [ ]:

In [ ]:

1. Body Mass Index Calculator:

In [37]:
```
Height=float(input("Enter your height in centimetres: "))
Weight=float(input("Enter your Weight in Kg: "))
Height = Height/100
BMI=Weight/(Height*Height)
print("your Body Mass Index is: ",BMI)
if(BMI>0):
        if(BMI<=16):
                print("you are severely underweight")
        elif(BMI<=18.5):
                print("you are underweight")
        elif(BMI<=25):
                print("you are Healthy")
        elif(BMI<=30):
                print("you are overweight")
        else: print("you are severely overweight")
else:("enter valid details")
```

```
your Body Mass Index is:  19.531249999999996
you are Healthy
```

2. Calculating Discount on Sale

In [38]:
```
amount = int(input("Enter Sale Amount: "))
if(amount > 0):
        if amount <= 10000:
                discount = amount * 0.05
        elif amount <= 15000:
                discount = amount * 0.12
        elif amount <= 20000:
                discount = amount * 0.20
        else:
                discount = amount * 0.30
        print("Discount: ",discount)
        print("Net Payable Amount: ",amount-discount)
else:
        print("Invalid Amount")
```

```
Discount:  16500.0
Net Payable Amount:  38500.0
```

3. Checking Eligiblity for voting

In [39]:
```
age= int(input("Enter your Age : "))
if age >=18:
        status= "Eligible"
else:
        status="Not Eligible"
print("you are",status,"for vote.")
```

```
you are Eligible for vote.
```

Loops

Suppose we want to calculate the salary of several employees considering the number of hours each worked. Although the salary calculation is the same for each employee, it must be done for multiple employees. Thus, we can ask the computer to repeat the extraction and calculation of data for each employee until the data has been processed for all employees. Python has two types of loops called while and for a loop.

While Loop with Python to Count Grades:

In [40]:
```
total = 0
count = 0
value = int( input("Enter the first grade: ") )
while value > 0 :
    total += value
    count += 1
    value = int( input("Enter the next grade (or 0 to quit): ") )
avg = total / count
print( "The average grade is %4.2f" % avg )
```

```
The average grade is 2.40
```

In [ ]:

For Loop with Python to Print Table of any number:

In [41]:
```
n=int(input("Enter n :"))
for i in range(1,11):
        print(n,"x",i,"=",i*n)
```

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

I hope this understood concept Loops

In [ ]:

In [ ]:

Class:

Python, like all object-oriented programming languages, allows programmers to define their classes. A class definition is an instruction composed of a header and a body. The class header contains the class keyword followed by an identifier used to name the class. The body of the class definition contains one or more method definitions, all of which must be indented to the same level of indentation.

In [42]:
```
class worker:
    def __init__(self, name, payment):
        self.name = name
        self.payment = payment
    def lastName(self):
        return self.name.split()[-1]
    def giveRaise(self, percent):
        self.payment *= (1.0 + percent)
```

In [43]:
```
Aman = worker("Aman Kharwal", 1000000)
Akanksha = worker("Akanksha Kharwal", 500000)
print(Aman.lastName())
print(Akanksha.lastName())
Aman.giveRaise(.50)
print(Aman.payment)
```

```
Kharwal
Kharwal
1500000.0
```

I hope this concept preety clear fundamental of python topics

Find Missing Number using Python

Finding the missing number in an array means finding the numbers missing from the array according to the range of values inside the array. Most of the time, the question you get based on this problem is like:

Given an array containing a range of numbers from 0 to n with a missing number, find the missing number in the input array.

To find the missing number in an array, we need to iterate over the input array and store the numbers in another array that we didn’t find in the input array while iterating over it. Below is how you can find the missing number in an array or a list using the Python programming language:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [44]:
```
def findMissingNumbers(n):
    numbers = set(n)
    length = len(n)
    output = []
    for i in range(1, n[-1]):
        if i not in numbers:
            output.append(i)
    return output

listOfNumbers = [1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 13, 14, 16]
print(findMissingNumbers(listOfNumbers))
```

```
[4, 12, 15]
```

In [ ]:

Summary

Finding the missing number in an array is a popular coding interview question. To find the missing number in an array, we need to iterate over the input array and store the numbers in another array that we didn’t find in the input array while iterating over it. I hope you liked this article on how to find the missing number in an array or a list using Python. Feel free to ask valuable questions in the comments section below.

In [ ]: