Python

```
Python is a simple and easy to understand language which feels like reading simple English this pseudo code nature of python next it easy to learn and understand by beginners.

Features of python

•   easy to understand

•   free and open source

•   high level language

•   portable.
```

What is programming?

```
Programming is a way to instruct the computer the perform various task.
```

Modules Comments Pip

Modules

```
modules is a file containing code written by somebody(usually)  which can be imported and used in our program.

Types of modules

Are two types of models in Python

built In models : pre install in Python

Example : OS , abc etc

```

```
External models : need to install using pip

Example: numpy etc
```

Pip

```
pip is a packet manager for Python.

we can use pip  to install a module on our system.

exa : pip install flask

```

Comments :

```
Comments are used to write something which the program does not want to execute.

Types of comments
Single line comments: using '#'

Multiline comments:    using ''' ''' or """ """.
```

In [ ]:
```
a =5 # this is comment
```

In [ ]:
```

```

Variables and data types

In [ ]:
```
Variables:
Variable is a container to store a value.
A variables is the name given to a memory location in program.
Reserved word in Python cannot be a variable.

For example:
Rules for defining a variable name:
A variables name can only start with an alphabet or underscore.
A variable name can contain alphabets digit and underscore.
A variable name can't start with a digit.
No white space is allowed to be used inside a variable name.

```

In [ ]:
```
A = 11
B = 'Sourabh'
C = 55.55
```

In [ ]:
```
# Example :
a    =  2
_va  = [2,]
val_ = ('ganesh',)
var1 = {'raja',}
```

Data Types

int

In [ ]:
```
Numerical Data types
```

In [ ]:
```
represents whole number (+ve or -ve)

example
a1 = 15
a2 = -15
```

float

In [ ]:
```
containing decimal number
a1 = 15.48
a2 = -15.88
```

complex

In [ ]:
```
two parts real and imaginary parts
realy part can be float or int (+ve or -ve)

a1 = 2 + 3j
a2 = 2 - 3j
```

Type casting or type conversion or type coersion

In [ ]:
```
we can convert one datatype value to another datatpe,this conversion is called as Type casting or type co

1) Implicit : (Python internally do type conversion depending on condition)
  Example :
    a = 15
    print(type(a))


2) Explicit : ( We do type casting or converiosn as per our requirement)
str1 = "15"
print(type(str1))# str
a =  int(str1)
print(type(a)) # int
```

In [68]:
```
a = 15
print(type(a))
```

```
<class 'int'>
```

In [69]:
```
a = 15.0
print(type(a))
```

```
<class 'float'>
```

str to int

In [70]:
```
a1 = "15"
print(f"before{type(a)}")
a2  = int(a1) #  type casting
print(f"after {type(a2)}")
```

```
before<class 'float'>
after <class 'int'>
```

int to float

In [72]:
```
a = 11
print(f" a = {a} and data type is {type(a)}")
b = float(a) #  type casting
print(f" b = {b} and data type is {type(b)}")
```

```
a = 11 and data type is <class 'int'>
b = 11.0 and data type is <class 'float'>
```

float to int

In [74]:
```
a = 11.52
print(f" a = {a} and data type is {type(a)}")
b = int(a) #  type casting
print(f" b = {b} and data type is {type(b)}")
```

```
a = 11.52 and data type is <class 'float'>
b = 11 and data type is <class 'int'>
```

int to complex

In [76]:
```
a = 11
print(f" a = {a} and data type is {type(a)}")
b = complex(a) #  type casting
print(f" b = {b} and data type is {type(b)}")
```

```
a = 11 and data type is <class 'int'>
b = (11+0j) and data type is <class 'complex'>
```

float to complex

In [77]:
```
a = 11.76
print(f" a = {a} and data type is {type(a)}")
b = complex(a) #  type casting
print(f" b = {b} and data type is {type(b)}")
```

```
a = 11.76 and data type is <class 'float'>
b = (11.76+0j) and data type is <class 'complex'>
```

complex to int

In [ ]:
```
not possible
```

complex to float

In [ ]:
```
not possible
```

int to str

In [78]:
```
a = 11
print(f" a = {a} and data type is {type(a)}")
b = str(a) #  type casting
print(f" b = {b} and data type is {type(b)}")
```

```
a = 11 and data type is <class 'int'>
b = 11 and data type is <class 'str'>
```

float to str

In [79]:
```
a = 11.76
print(f" a = {a} and data type is {type(a)}")
b = str(a) #  type casting
print(f" b = {b} and data type is {type(b)}")
```

```
 a = 11.76 and data type is <class 'float'>
 b = 11.76 and data type is <class 'str'>
```

str to int

In [81]:
```
a= "11"
b= int(a)
b,type(b)
```

Out[81]:
```
(11, int)
```

str to float

In [82]:
```
a= "11.76"
b= float(a)
b,type(b)
```

Out[82]:
```
(11.76, float)
```

str to complex

In [83]:
```
a= "11.76"
b= complex(a)
b,type(b)
```

Out[83]:
```
((11.76+0j), complex)
```

eval :

In [ ]:
```
internally recongnized data type of input
```

In [87]:
```
a = eval(input(" enter salary value "))
print(f" a = {a} and data type is {type(a)}") # checking datatype
```

```
enter salary value 11.76
a = 11.76 and data type is <class 'float'>
```

ceil and floor

In [88]:
```
import math
```

In [89]:
```
math.ceil(11.23)
```

Out[89]: 12

In [90]:
```
math.floor(11.23)
```

Out[90]: 11

STRING

```
String is a data type in Python.

string is a sequence of character and unclosed in quote.

we can primarly write a string in this way:

single quoted string

double quoted string

triple quoted string

```

```
13 string in Python can be sliced for getting a part of the string.
```

In [ ]:

```
1 str1 = "Data Science"
2  D   a    t    a       S    c   i  e    n   c   e
3  0   1    2    3   4   5    6   7  8    9   10  11  :> +ve indexing (read left to right) default is 0
4 -12 -11  -10  -9  -8   -7  -6  -5  -4  -3   -2  -1  :> -ve indexing (read right to left)  default is -1
5
```

POSITIVE INDEXING

```
1 The index in a string start from 0 to (length -1)(left to right) in Python in order to slice a string
  we use the following syntax.
```

In [5]:

```
1 name = "sourabh"
2 for i in name:
3     print(f'{i} >> {name.find(i)}',end =",")
```

```
s >> 0,o >> 1,u >> 2,r >> 3,a >> 4,b >> 5,h >> 6,
```

NEGATIVE INDEXING

```
1 The index in a string start from -1 to (-len)(right to left) in Python in order to slice a string we
  use the following syntax.
```

In [7]:

```
1 name = "sourabh"
2 n=-1
3 for i in name[::-1]:
4     print(f'{i} >> {n}',end =",")
5     n-=1
```

```
h >> -1,b >> -2,a >> -3,r >> -4,u >> -5,o >> -6,s >> -7,
```

In [ ]:
```
return index and char and default is 0
# Syntax
for index, value in enumerate(iterable):
    print(index,value)
```

In [10]:
```
name = "sourabh"
for i,val in enumerate(name):
    print(f' {i} >>.. {val}',end = " ")
```

```
 0 >>.. s  1 >>.. o  2 >>.. u  3 >>.. r  4 >>.. a  5 >>.. b  6 >>.. h
```

range

In [14]:
```
name = "sourabh"
for i in range(len(name)):
    print(f'{i}.>>>.{name[i]}',end =" ,")
```

```
0.>>>.s ,1.>>>.o ,2.>>>.u ,3.>>>.r ,4.>>>.a ,5.>>>.b ,6.>>>.h ,
```

STRING SLICING

```
A string in Python Can be sliced for getting a part of the string.
```

In [ ]:
```
str1[start_index (include) : end_index(exclude): step_size(default=1)]
start_index = default 0 (include)
end_index = len(str1)-1
step_size = 1 default (char to skip)

# note
step size = 1  >> 0 char skip
step size = 2  >> 1 char skip
step size = 3  >> 2 char skip
step size = 4  >> 3 char skip
```

In [8]:
```
name = "sourabh"
name[2:6]
```

Out[8]: 'urab'

Slicing with skip value

```
We can provide a skip value as a part of our slice like this.
```

In [9]:
```
name  = "sourabh"
name[2:6:3]
```

Out[9]: 'ub'

String functions

```
 Some of the mostly used Functions to perform operation on or Manipulate strings are as follows:
```

Upper()

```
In this case, All the alphabets are converted into upper case(A-Z).
```

In [10]:
```
name = 'sourabh'
name.upper()
```

Out[10]: 'SOURABH'

isupper()

```
True- If all characters in the string are uppercase.

```

```
3 False- If the string contains 1 or more non-uppercase characters.
```

In [16]:
```
1 a ="MNDFJK5215"
2 B ="knkdshkjHJG57"
3 a.isupper(),B.isupper()
```

Out[16]: (True, False)

Lower()

```
1 In this case, All the alphabets are converted into lower case(a-z).
```

In [12]:
```
1 name = 'sourabh'
2 name.lower()
```

Out[12]: 'sourabh'

islower()

```
1 True- If all characters in the string are lowercase.
2
3 False- If the string contains 1 or more non-lowercase characters.
```

In [19]:
```
1 a="hjbzchjb58"
2 B ="knkdshkjHJG57"
3 a.islower(),B.islower()
```

Out[19]: (True, False)

Capitalize

```
In this case,Zeroth index of the string is converted into upper case(if their is any alphabet).
```

In [13]:
```
a = 'india is my country'
a.capitalize()
```

Out[13]: 'India is my country'

In [14]:
```
a = ' india is my country'
a.capitalize()
```

Out[14]: ' india is my country'

Swap case

```
Swapcase() method converts all uppercase characters to lowercase and vice versa of the given string and returns it.
```

In [21]:
```
name = 'SouraBH'
name.swapcase()
```

Out[21]: 'sOURAbh'

Title

```
It converts the first letter of every word to uppercase and other letters to lowercase and then returns this new string.
```

In [22]:
```
aa ="india is my country"
aa.title()
```

Out[22]: 'India Is My Country'

istitle()

In [28]:
```
aa = 'India Is My Country'
bb = 'india Is mY COUNTRY'
aa.istitle(),bb.istitle()
```

Out[28]: (True, False)

Case fold()

```
casefold() method is used to convert string to lowercase.

It is similar to the Python lower() string method, but the case removes all the case distinctions present in a string.
```

In [24]:
```
aa = "SOURABH ß"
print("Using lower():", aa.lower())
print("Using casefold():", aa.casefold())
```

```
Using lower(): sourabh ß
Using casefold(): sourabh ss
```

center()

```
Returns a centered string sorrounded by white spaces.
```

In [31]:
```
name = 'sourabh'
name.center(12)
```

Out[31]: ' sourabh   '

count()

```
1 Returns the number of times a specified value occurs in a string.
```

In [34]:
```
1 aa = "sourabh bh"
2 aa.count("b")
```

Out[34]: 2

endswith()

```
1 Returns true if the string ends with the specified value
```

In [35]:
```
1 name = "sourabh"
2 name.endswith("bh")
```

Out[35]: True

find()

```
1 Searches the string for a specified value and returns the position of where it was found
```

In [2]:
```
1 name = "sourabh"
2 name.find("b")
```

Out[2]: 5

index()

```
1 Searches the string for a specified value and returns the position of where it was found
```

In [4]:
```
name = "sourabh"
name.index("o")
```

Out[4]: 1

isalnum()

```
Returns True if all characters in the string are alphanumeric
```

In [6]:
```
aa = "hjbhug584"
aa.isalnum()
```

Out[6]: True

isalpha()

```
Returns True if all characters in the string are in the alphabet
```

In [7]:
```
aa = "hjbhug584"
aa.isalpha()
```

Out[7]: False

In [8]:
```
aa = "hjbhug"
aa.isalpha()
```

Out[8]: True

isdecimal()

```
Returns True if all characters in the string are decimals

True : If string containing only numbers (0-9)

```

```
True : if the string contains only digit or unicode for digits (0-9)

for
   0 "\u0030"
   .
   .
   9 "\u0039"
```

In [10]:
```
aa = "51564"
aa.isdecimal()
```

Out[10]: True

In [11]:
```
aa = "5156fgn4"
aa.isdecimal()
```

Out[11]: False

isdigit()

```
Returns True if all characters in the string are digits

True :  for digits (0-9) or unicode of 0-9 or subscript or superscript (chemical compounds)

superscript

x^2 + y^2

subscript

 H2O
```

In [12]:
```
aa = "51564"
aa.isdigit()
```

Out[12]: True

In [13]:
```
aa = "51564hgfhudsy"
aa.isdigit()
```

Out[13]: False

isnumeric()

```
Returns True if all characters in the string are numeric

True :  if all charcater are digits, substript, superscript, vulger fraction (1/2 or 3/4)
```

In [16]:
```
aa = "51564hgfhudsy"
aa.isnumeric()
```

Out[16]: False

isspace()

```
Returns True if all characters in the string are whitespaces
```

In [1]:
```
aa = "       "
aa.isspace()
```

Out[1]: True

In [2]:
```
aa = "      vhjfh "
aa[:5].isspace()
```

Out[2]: True

concatenation

```

```

```
concatenation takes place between strings only.
```

In [18]:
```
aa = "raja"
bb = "rani"
aa+" "+ bb
```

Out[18]: 'raja rani'

Multiplication

```
Multiplication takes place between string and positive intger only.
```

In [20]:
```
aa = "++ "
bb = 12
aa * bb
```

Out[20]: '++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ '

lstrip()

```
(remove all leading white spaces (starting)
```

In [26]:
```
name = "   sourabh"
name.lstrip()
```

Out[26]: 'sourabh'

rstrip()

```
remove all trailing white spaces (ending).
```

In [27]:
```
name = "   sourabh      "
name.rstrip()
```

Out[27]:
```
'   sourabh'
```

strip()

```
remove all leading and Trailing white spaces (starting and ending)
```

In [28]:
```
name = "   sourabh      "
name.strip()
```

Out[28]:
```
'sourabh'
```

replace()

```
str1.replace(old_str,new_str,count(optional))
```

In [31]:
```
str1 = "hello,hii!, good morning," #
str1.replace("good morning","good evening")
```

Out[31]:
```
'hello,hii!, good evening,'
```

Split()

```
return a list of the words in the string seperated by white spaces
```

In [32]:
```
name = "Gautam       Gambhir"
lst = name.split()
print(lst)
pan_card_name1 = " ".join(lst)
print(pan_card_name1)
```

```
['Gautam', 'Gambhir']
Gautam Gambhir
```

Index()

In [ ]:
```
if substring present it will give the index no of that substring,if not raise an error
```

In [37]:
```
aa = "sourabh"
aa.index("b")
```

Out[37]: 5

In [38]:
```
aa = "sourabh"
aa.index("p")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[38], line 2
      1 aa = "sourabh"
----> 2 aa.index("p")

ValueError: substring not found
```

rindex()

In [ ]:
```
Searches the string for a specified value and returns the last position of where it was found
```

In [50]:
```
aa = "sourabh bh"
print("from left to right 'b' ",aa.index("b"))
print("from right to left 'b' ",aa.rindex("b"))
```

```
from left to right 'b'  5
from right to left 'b'  8
```

find()

```
string.find(substring, [start_index],[end_index])

it return always 1st index of char / substring/word occurnace only

if char / substring/word not found in given string the it will return -1
```

In [34]:
```
aa = "sourabh"
aa.find("o")
```

Out[34]: 1

rfind()

In [ ]:
```
Searches the string for a specified value and returns the last position of where it was found
```

In [49]:
```
aa = "sourabh bh"
print("from left to right 'b' ",aa.find("b"))
print("from right to left 'b' ",aa.rfind("b"))
```

```
from left to right 'b'  5
from right to left 'b'  8
```

join()

In [36]:
```
aa = "sourabh"
"-".join(aa)
```

Out[36]: 's-o-u-r-a-b-h'

zfill()

```
it adds zeros at the beginning of string

str1.zfill()
```

In [39]:
```
aa = "sourabh"
aa.zfill(15)
```

Out[39]: '00000000sourabh'

partition()

In [ ]:
```
Returns a tuple where the string is parted into three parts
```

In [41]:
```
string = "Hello, world! How are you?"
partitioned = string.partition(", ")
print(" ".join(partitioned))
print(partitioned)
print("Before:", partitioned[0])
print("Separator:", partitioned[1])
print("After:", partitioned[2])
```

```
Hello ,  world! How are you?
('Hello', ', ', 'world! How are you?')
Before: Hello
Separator: ,
After: world! How are you?
```

Everification

In [23]:
```
pan_name = "Gautam Gambhir".lower()
aadhar_name = "Gautam GAMBHIR".lower()
if pan_name == aadhar_name:
    print("Both names are same")
else:
    print("Both names are not same")
```

```
Both names are same
```

In [24]:
```
pan_name = "Gautam GamBHIr".title()
aadhar_name = "Gautam GAMBHIR".title()
if pan_name == aadhar_name:
    print("Both names are same")
else:
    print("Both names are not same")
```

```
Both names are same
```

In [64]:
```
val = "Data Science"
for i in val:
    print(f'{i} >> {val.index(i)}')
```

```
D >> 0
a >> 1
t >> 2
a >> 1
  >> 4
S >> 5
c >> 6
i >> 7
e >> 8
n >> 9
c >> 6
e >> 8
```

In [65]:

```
val = "Data Science"
for i in val:
    print(f'{i} >> {val.find(i)}')
```

```
D >> 0
a >> 1
t >> 2
a >> 1
  >> 4
S >> 5
c >> 6
i >> 7
e >> 8
n >> 9
c >> 6
e >> 8
```

In [67]:

```
val = "Data Science"
for i in val[::-1]:
    print(f'{i} >> {val.find(i)}')
```

```
e >> 8
c >> 6
n >> 9
e >> 8
i >> 7
c >> 6
S >> 5
  >> 4
a >> 1
t >> 2
a >> 1
D >> 0
```