PYTHON PROGRAMMING BASICS

Select the cell operation with Markdown to write comments and documents

To Run cell use keys : Shift and Enter to Run

Concept 1: Strings

In [7]:
```
print('BIA CLASS')
```

```
BIA CLASS
```

In [8]:
```
print("CLASS Started")
```

```
CLASS Started
```

In [3]:
```
print("hello","world")
```

```
hello world
```

Heading1

Heading1

Heading 2

Heading3

In [4]:
```
print('hello')
```

```
hello
```

In [5]:
```
print("welcome to class BIA ")
```

```
welcome to class BIA
```

In [6]:
```
print('welcome', 'Rajesh to class')
```

```
welcome Rajesh to class
```

insertion or deletion of cells

Use b letter to insert cell (1-time)

use d letter Two times to delet cell (2- times)

In [1]:
```
print('/\')
print ('/ __ \')
```

```
  File "<ipython-input-1-6df7a14735b5>", line 1
    print('/\')
                ^
SyntaxError: EOL while scanning string literal
```

Heading1 : To write a documentation

Heading 2

Heading 3

Headhing 4

To create list

- List1

  -sublist1
  - List2

    -sublist2

To add Python syntax within markdown

```
print('This is a markdown syntax')
```

Comments in Python

- 1. Single line comments

print(5) # displays content given

- 2.Multi line comments

```
 multi line commnets
print('hello world')
print('start python')
```

Keywords in python

In [ ]:
```
help()
```

```
Welcome to Python 3.7's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.7/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> keywords

Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not

help> False
Help on bool object:

class bool(int)
 |  bool(x) -> bool
 |
 |  Returns True when the argument x is true, False otherwise.
 |  The builtins True and False are the only two instances of the class bool.
 |  The class bool is a subclass of the class int, and cannot be subclassed.
 |
 |  Method resolution order:
 |      bool
 |      int
 |      object
 |
 |  Methods defined here:
 |
 |  __and__(self, value, /)
 |      Return self&value.
 |
 |  __or__(self, value, /)
 |      Return self|value.
 |
 |  __rand__(self, value, /)
 |      Return value&self.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
```

```
 |  __ror__(self, value, /)
 |      Return value|self.
 |
 |  __rxor__(self, value, /)
 |      Return value^self.
 |
 |  __str__(self, /)
 |      Return str(self).
 |
 |  __xor__(self, value, /)
 |      Return self^value.
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signatur
e.
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from int:
 |
 |  __abs__(self, /)
 |      abs(self)
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __bool__(self, /)
 |      self != 0
 |
 |  __ceil__(...)
 |      Ceiling of an Integral returns itself.
 |
 |  __divmod__(self, value, /)
 |      Return divmod(self, value).
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __float__(self, /)
 |      float(self)
 |
 |  __floor__(...)
 |      Flooring an Integral returns itself.
 |
 |  __floordiv__(self, value, /)
 |      Return self//value.
 |
 |  __format__(self, format_spec, /)
 |      Default object formatter.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
```

```
 |
 |  __getnewargs__(self, /)
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __hash__(self, /)
 |      Return hash(self).
 |
 |  __index__(self, /)
 |      Return self converted to an integer, if self is suitable for use as a
n index into a list.
 |
 |  __int__(self, /)
 |      int(self)
 |
 |  __invert__(self, /)
 |      ~self
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __lshift__(self, value, /)
 |      Return self<<value.
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __mod__(self, value, /)
 |      Return self%value.
 |
 |  __mul__(self, value, /)
 |      Return self*value.
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __neg__(self, /)
 |      -self
 |
 |  __pos__(self, /)
 |      +self
 |
 |  __pow__(self, value, mod=None, /)
 |      Return pow(self, value, mod).
 |
 |  __radd__(self, value, /)
 |      Return value+self.
 |
 |  __rdivmod__(self, value, /)
 |      Return divmod(value, self).
 |
 |  __rfloordiv__(self, value, /)
 |      Return value//self.
 |
 |  __rlshift__(self, value, /)
 |      Return value<<self.
```

```
|
|  __rmod__(self, value, /)
|      Return value%self.
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  __round__(...)
|      Rounding an Integral returns itself.
|      Rounding with an ndigits argument also returns an integer.
|
|  __rpow__(self, value, mod=None, /)
|      Return pow(value, self, mod).
|
|  __rrshift__(self, value, /)
|      Return value>>self.
|
|  __rshift__(self, value, /)
|      Return self>>value.
|
|  __rsub__(self, value, /)
|      Return value-self.
|
|  __rtruediv__(self, value, /)
|      Return value/self.
|
|  __sizeof__(self, /)
|      Returns size in memory, in bytes.
|
|  __sub__(self, value, /)
|      Return self-value.
|
|  __truediv__(self, value, /)
|      Return self/value.
|
|  __trunc__(...)
|      Truncating an Integral returns itself.
|
|  bit_length(self, /)
|      Number of bits necessary to represent self in binary.
|
|      >>> bin(37)
|      '0b100101'
|      >>> (37).bit_length()
|      6
|
|  conjugate(...)
|      Returns self, the complex conjugate of any int.
|
|  to_bytes(self, /, length, byteorder, *, signed=False)
|      Return an array of bytes representing an integer.
|
|      length
|        Length of bytes object to use.  An OverflowError is raised if the
|        integer is not representable with the given number of bytes.
|      byteorder
|        The byte order used to represent the integer.  If byteorder is 'bi
```

```
g',
 |       the most significant byte is at the beginning of the byte array.  I
f
 |       byteorder is 'little', the most significant byte is at the end of t
he
 |       byte array.  To request the native byte order of the host system, u
se
 |       `sys.byteorder' as the byte order value.
 |     signed
 |       Determines whether two's complement is used to represent the intege
r.
 |       If signed is False and a negative integer is given, an OverflowErro
r
 |       is raised.
 |
 |  ----------------------------------------------------------------------
 |  Class methods inherited from int:
 |
 |  from_bytes(bytes, byteorder, *, signed=False) from builtins.type
 |      Return the integer represented by the given array of bytes.
 |
 |      bytes
 |        Holds the array of bytes to convert.  The argument must either
 |        support the buffer protocol or be an iterable object producing byte
s.
 |        Bytes and bytearray are examples of built-in objects that support t
he
 |        buffer protocol.
 |      byteorder
 |        The byte order used to represent the integer.  If byteorder is 'bi
g',
 |        the most significant byte is at the beginning of the byte array.  I
f
 |        byteorder is 'little', the most significant byte is at the end of t
he
 |        byte array.  To request the native byte order of the host system, u
se
 |        `sys.byteorder' as the byte order value.
 |      signed
 |        Indicates whether two's complement is used to represent the intege
r.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from int:
 |
 |  denominator
 |      the denominator of a rational number in lowest terms
 |
 |  imag
 |      the imaginary part of a complex number
 |
 |  numerator
 |      the numerator of a rational number in lowest terms
 |
 |  real
 |      the real part of a complex number
```

```
help> symbols

Here is a list of the punctuation symbols which Python assigns special meanin
g
to. Enter any symbol to get more help.

!=                  +                   <=                  __
"                   +=                  <>                  `
"""                 ,                   ==                  b"
%                   -                   >                   b'
%=                  -=                  >=                  f"
&                   .                   >>                  f'
&=                  ...                 >>=                 j
'                   /                   @                   r"
'''                 //                  J                   r'
(                   //=                 [                   u"
)                   /=                  \                   u'
*                   :                   ]                   |
**                  <                   ^                   |=
**=                 <<                  ^=                  ~
*=                  <<=                 _

help> modules

Please wait a moment while I gather a list of all available modules...


C:\Users\RAJESH RAI\AppData\Roaming\Python\Python37\site-packages\IPython\ker
nel\__init__.py:13: ShimWarning: The `IPython.kernel` package has been deprec
ated since IPython 4.0.You should import from ipykernel or jupyter_client ins
tead.
  "You should import from ipykernel or jupyter_client instead.", ShimWarning)
```

Variables decelration and Data types

In [5]:
```
ab=50
```

In [6]:
```
ab
```

Out[6]: 50

In [7]:
```
a=10
b=20
a+b
```

Out[7]: 30

In [8]:
```
type(a)
```

Out[8]: int

In [9]:
```
org = 'JNTUACEA'
print(type(org))
```

```
<class 'str'>
```

In [10]:
```
avg = 10.57
type(avg)
```

Out[10]:
```
float
```

Multiple Variables

In [16]:
```
emp_name, emp_age, emp_id, emp_avg_salary ='Rajesh',30,9999,40000
```

In [17]:
```
emp_name
```

Out[17]:
```
'Rajesh'
```

In [18]:
```
emp_age
```

Out[18]:
```
30
```

In [21]:
```
emp_age, emp_avg_salary
```

Out[21]:
```
(30, 40000)
```

data conversions

In [24]:
```
c=3.5
```

In [25]:
```
c
```

Out[25]:
```
3.5
```

In [26]:
```
type(c)
```

Out[26]:
```
float
```

In [27]:
```
c=str(c)
```

In [31]:
```
c
```

Out[31]:
```
'3.5'
```

In [28]:
```
d=20
type(d)
```

Out[28]:
```
int
```

In [29]:
```
d=str(d)
```

In [30]:
```
d
```

Out[30]:
```
'20'
```

In [32]:
```
float(d)
```

Out[32]:
```
20.0
```

In [33]:
```
d='c'
```

In [34]:
```
type(d)
```

Out[34]:
```
str
```

In [35]:
```
ab=54
```

In [36]:
```
ab
```

Out[36]:
```
54
```

In [37]:
```
a_=4
```

In [38]:
```
a_
```

Out[38]:
```
4
```

In [40]:
```
_b=10
```

In [41]:
```
_b
```

Out[41]:
```
10
```

In [42]:
```
a1='abc'
```

In [43]:
```
a1
```

Out[43]:
```
'abc'
```

In [44]:
```
ab=10
```

In [45]:
```
a
```

Out[45]:
```
10
```

In [46]:
```
1a=345
```

```
  File "<ipython-input-46-9b0abace639b>", line 1
    1a=345
     ^
SyntaxError: invalid syntax
```

In [47]:
```
*a=10
```

```
  File "<ipython-input-47-a6d0f5818dd4>", line 4
SyntaxError: starred assignment target must be in a list or tuple
```

In [48]:
```
@f=20
```

```
  File "<ipython-input-48-50ca9a04d007>", line 1
    @f=20
      ^
SyntaxError: invalid syntax
```

Operators

1. Arithematic Operators

2. Assignment Operators

3. Comparison Operators

4. Logical Operations

5. Bitwise Operators

6. Identify Operators

7. membership operators

1. Arithematic Operators

In [ ]:
```
# +,-,*,/,//,%

a=20
b=30

a=int(input('enter the value'))
b=int (input('enter the value'))

print("the sum is:", a+b)
print("the sub is:", a-b)
print("the mul is:", a*b)
print("the div is:", a/b)
```

In [ ]:
```
a= int(input('enter the value'))
b=int(input('enter the value'))

print("the sum is:", a+b)
print("the sub is:", a-b)
print("the mul is:", a*b)
print("the div is:", a/b)
```

2. Assignement Operator

In [2]:
```
# =,*=,+=,-=,/=,//=

a=10
b=30
print(a,b)
```

```
10 30
```

In [5]:
```
a=10
b=20
```

In [6]:
```
print(a)
```

```
10
```

In [7]:
```
a-=5
print(a)
```

```
5
```

In [8]:
```
a/=5
print(a)
```

```
1.0
```

In [9]:
```
type(a)
```

Out[9]:
```
float
```

In [10]:
```
b+=20
```

In [11]:
```
print(b)
```

```
40
```

3. Comparision operations

In [12]:
```
#, <,>.<=,>=,==,!=

print(a,b)
```

```
1.0 40
```

In [13]:
```
a>b
```

Out[13]: False

In [14]:
```
a<b
```

Out[14]: True

In [15]:
```
7<=2
```

Out[15]: False

In [16]:
```
a==b
```

Out[16]: False

In [17]:
```
7==7
```

Out[17]: True

In [18]:
```
a!=b
```

Out[18]: True

In [19]:
```
4!=3
```

Out[19]: True

4. Logical Operators

In [20]:
```
# amd, or, not

#and

a=5
b=7

a and b
```

Out[20]: 7

In [21]:
```
a= 7>7 or 2>1
print(a)
```

```
True
```

In [22]:
```
7 and 0 or 5
```

Out[22]: 5

In [3]:
```
a=10
not(a)
```

Out[3]: False

In [4]:
```
a not
```

```
  File "<ipython-input-4-e6836cb86817>", line 1
    a not
        ^
SyntaxError: invalid syntax
```

In [5]:
```
not a
```

Out[5]: False

5. Bitwise Operations

In [6]:
```
# &,|,^,~,<<,>> <<- leftshift and >>- right shift

a=2
b=5
a&b
```

Out[6]: 0

In [7]:
```
a|b
```

Out[7]: 7

In [8]:
```
3>>2
```

Out[8]: 0

In [9]:
```
3>>1
```

Out[9]: 1

In [10]:
```
2<<2
```

Out[10]: 8

In [11]:
```
x=10
x is 20
```

Out[11]: False

In [12]:
```
x is 10
```

Out[12]: True

In [13]:
```
x= "JNTUACEA"
x is "JNTU"
```

Out[13]: False

In [14]:
```
x is not "jntua"
```

Out[14]: True

In [15]:
```
name = ['rajesh','jain','sarfaraz',14,15,16.255,'tarun']

name
```

Out[15]: ['rajesh', 'jain', 'sarfaraz', 14, 15, 16.255, 'tarun']

condition execution

In [16]:
```
10 == 9
```

Out[16]: False

In [17]:
```
type(True)
```

Out[17]: bool

In [20]:
```
name = input("enter the name") # JNTUACEA
password = input("enter the password") #IOT

if name =="JNTUACEA" and password =="IOT":
    print("User data")
    print("welcome", name)
elif name == "JNTUACEP" and password =="IOT":
           print("welcome", name)
else:
                 print("invalid")
```

```
enter the nameJNTUACEP
enter the passwordIOT
welcome JNTUACEP
```

In [22]:
```
a=10
if a%2 == 2:
    print("even")
else:
    print("odd")
```

```
odd
```

In [23]:
```
# gratest of 3 numbers

a=10
b=15
c=20

if a>b and a>c:
    print(a,"is grater")
elif b>c:
    print(b,"is grater")
else:
    print(c,"is grater")
```

```
20 is grater
```

while loop

syntax

while expression:

statemnts1

statements 1

In [1]:
```
count = 0
while (count<9):
    print(count)
    count = count+1
```

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

In [6]:
```
count = 0
while (count<9):
    #print(count)
    count = count+1
    print(count,end=',') # horizontal line output
```

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

In [7]:
```
count = 0
while (count<9):
    #print(count)
    count = count+1
    print(count,end='') # horizontal line output
```

```
123456789
```

In [8]:
```
count = 0
while (count<9):
    #print(count)
    count = count+1
    print(count,end='@') # horizontal line output
```

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

For loop

syntax:

for iterator name in interator:

```
### print(iteratorname) # interator variable name
```

In [9]:
```
for i in range(1,10+1): # range (lb,ub-1)
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

In [10]:
```
for i in range(1,10): # range (lb,ub-1 #lb- lower bound and ub - upper bound
    print(i)
```

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

In [11]:
```
for i in range(1,10): # range (lb,ub-1 #lb- lower bound and ub - upper bound
    print(i,end=' ')
```

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

Example : 1

write a python program to get all even numbers in given range

In [7]:
```
a= int(input("enter the value:"))
b= int(input("enter the value:"))
for i in range(a,b+1):
        if(i%2==0):
            print(i)
```

```
enter the value:1
enter the value:100
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
66
68
70
72
74
76
78
80
82
84
86
88
90
92
94
96
98
100
```

In [9]:
```
a= int(input("enter the value:"))
b= int(input("enter the value:"))
for i in range(a,b+1):
        if(i%2==0):
            print(i,end=",")
```

```
enter the value:1
enter the value:100
2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,
56,58,60,62,64,66,68,70,72,74,76,78,80,82,84,86,88,90,92,94,96,98,100,
```

example 2

write a program for multiplication table

In [10]:
```
a= int(input("enter the table number:"))
for i in range(1,11):
    print(a,"*",i,"=",a*i)
```

```
enter the table number:5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

Auto incerement for every 2 values

syntax:

for in range (start,endvalue,increment of range value)

In [11]:
```
for i in range(1,100,2): # odd values
    print(i,end=",")
```

```
1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53,5
5,57,59,61,63,65,67,69,71,73,75,77,79,81,83,85,87,89,91,93,95,97,99,
```

In [13]:
```
for i in range(2,100,2): # even values
    print(i,end=",")
```

```
2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,
56,58,60,62,64,66,68,70,72,74,76,78,80,82,84,86,88,90,92,94,96,98,
```

Example

python program to check given numbers is prime number or not ?

In [17]:
```
a= int(input("enter the value:"))
if a>1:
    for i in range(2,a):
        if(a%i==0):
            print(a,'is not prime number')
        break
    else:
        print(a,'is prime number')
```

```
enter the value:12
12 is not prime number
```

In [18]:
```
a= int(input("enter the value:"))
b=0
for i in range(1,a):
        if(a%i==0):
            b=b+1
if(b==2):
    print(a,'is not prime number')

else:
        print(a,'is prime number')
```

```
enter the value:5
5 is prime number
```

In [19]:
```
a= int(input("enter the value:"))
b=0
for i in range(1,a+1):
        if(a%i==0):
            b=b+1
if(b==2):
    print(a,'is not prime number')

else:
        print(a,'is prime number')
```

```
enter the value:5
5 is not prime number
```

In [39]:
```
# program to print prime number given range 1 to n

a= int(input("enter the value:"))
b= int(input("enter the value:"))
for i in range(1,b+1):
    if(i>1):
        for j in range (2,i):
                if (i%j==0):
                   break
        else:
            print(i,end=',')
```

```
enter the value:1
enter the value:100
2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97,
```

In [ ]: