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 [1]:
```
print('hello')
```

```
hello
```

In [3]:
```
'hello'
```

Out[3]:
```
'hello'
```

In [4]:
```
print('/\')
      ('/ \')
       ('/  \')
```

```
  File "<ipython-input-4-f8b9263a5caf>", line 1
    print('/\')
                ^
SyntaxError: EOL while scanning string literal
```

insertion of cell and deletion of cells

Use b letter to insert cell (1-time)

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

In [7]:
```
print ('good')
```

```
good
```

In [8]:
```
print("hello")
```

```
hello
```

In [9]:
```
print('hi','world')
```

```
hi world
```

Hello world

Hello

hello

Hello

1. ananconda
2. welcome
3. hello

```
multi line comments
```

```
print('hello syntax')
```

In [7]:
```
print('hello world welcome anaconda ')
```

```
hello world welcome anaconda
```

In [8]:
```
a b = 10
```

```
  File "<ipython-input-8-039b1666f5cf>", line 1
    a b = 10
      ^
SyntaxError: invalid syntax
```

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

In [13]:
```
a=10
b=20
a+b
```

Out[13]:
```
30
```

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

Out[14]:
```
-10
```

In [15]:
```
b-a
```

Out[15]:
```
10
```

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

Out[16]:
```
200
```

In [17]:
```
a/2
```

Out[17]:
```
5.0
```

In [18]:
```
a*b/2
```

Out[18]:
```
100.0
```

In [19]:
```
a and b
```

Out[19]:
```
20
```

In [20]:
```
a or b
```

Out[20]:
```
10
```

In [21]:
```
a xor b
```

```
  File "<ipython-input-21-6637868c93bc>", line 1
    a xor b
        ^
SyntaxError: invalid syntax
```

In [22]:
```
a not b
```

```
  File "<ipython-input-22-fda1ac746f0e>", line 1
    a not b
          ^
SyntaxError: invalid syntax
```

In [23]:
```
not a
```

Out[23]:
```
False
```

In [24]:
```
a not
```

```
  File "<ipython-input-24-e6836cb86817>", line 1
    a not
          ^
SyntaxError: invalid syntax
```

In [25]:
```
help
```

Out[25]:
```
Type help() for interactive help, or help(object) for help about object.
```

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

In [27]:
```
keywords
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-27-1eb9771c7f0a> in <module>
----> 1 keywords

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

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

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

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

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
```

DOCUMENTATION FORMAT

HEADING1

Creation of Lists

- List1 -Sublist1
- List 2 -Sublist 2

To add Python syntax

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

commnets in Python

Two types :

1. sigle line comments

print(5) # displays informatio given

2. Multi line comments

``` **multi line comments**

iintialized in n- lines specified ```
```

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 signature.
 |
 |  ----------------------------------------------------------------------
 |  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 an 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 'big',
|        the most significant byte is at the beginning of the byte array.  If
|        byteorder is 'little', the most significant byte is at the end of the
|        byte array.  To request the native byte order of the host system, use
|        `sys.byteorder' as the byte order value.
|      signed
|        Determines whether two's complement is used to represent the integer.
|        If signed is False and a negative integer is given, an OverflowError
|        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 bytes.
|        Bytes and bytearray are examples of built-in objects that support the
|        buffer protocol.
|      byteorder
|        The byte order used to represent the integer.  If byteorder is 'big',
|        the most significant byte is at the beginning of the byte array.  If
|        byteorder is 'little', the most significant byte is at the end of the
|        byte array.  To request the native byte order of the host system, use
|        `sys.byteorder' as the byte order value.
|      signed
|        Indicates whether two's complement is used to represent the integer.
|
|  ----------------------------------------------------------------------
|  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
```

Data Conversions

In [ ]:
```
a = 5
```

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

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

In [2]:
```
c=3.5
```

In [3]:
```
c
```

Out[3]: 3.5

In [4]:
```
type(c)
```

Out[4]: float

In [5]:
```
str(c)
```

Out[5]: '3.5'

In [6]:
```
org ="JNTUACEA"
print(type(org))
```

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

In [7]:
```
avg = 12.7
```

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

Out[8]:
```
float
```

In [14]:
```
d=10
```

In [15]:
```
type(d)
```

Out[15]:
```
int
```

In [16]:
```
str(d)
```

Out[16]:
```
'10'
```

In [17]:
```
d=str(d)
```

In [18]:
```
d
```

Out[18]:
```
'10'
```

In [19]:
```
float(d)
```

Out[19]:
```
10.0
```

Multiple Variables

In [9]:
```
emp_name,emp_age,emp_id,emp_avg_salary ="JAIN",30,9999,40000
```

In [10]:
```
emp_name
```

Out[10]:
```
'JAIN'
```

In [11]:
```
emp_age
```

Out[11]:
```
30
```

In [12]:
```
emp-age,emp_id
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-12-9576b501d903> in <module>
----> 1 emp-age,emp_id

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

In [13]:
```
emp_id
```

Out[13]:
```
9999
```

Data Declerations

In [20]:
```
ab=56
```

In [21]:
```
a_=4
```

In [22]:
```
a
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-22-3f786850e387> in <module>
----> 1 a

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

In [23]:
```
a_
```

Out[23]: 4

In [24]:
```
ab
```

Out[24]: 56

In [25]:
```
_b=6
```

In [26]:
```
-b
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-26-236131f26970> in <module>
----> 1 -b

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

In [27]:
```
_b
```

Out[27]: 6

In [28]:
```
a1='abc'
```

In [30]:
```
a =5
```

In [31]:
```
a
```

Out[31]: 5

In [32]:
```
1a=345
```

```
  File "<ipython-input-32-9b0abace639b>", line 1
    1a=345
     ^
SyntaxError: invalid syntax
```

Special Symbols

In [33]:
```
*a=9
```

```
  File "<ipython-input-33-00417a518b29>", line 4
SyntaxError: starred assignment target must be in a list or tuple
```

In [34]:
```
@s =6
```

```
  File "<ipython-input-34-f0dc0f1e99f7>", line 1
    @s =6
       ^
SyntaxError: invalid syntax
```

OPERATORS DECELRATION

Arithematic operators are +,-,/,*,//,% (Addition , subtraction, division, multiplication, flow division , modulus

In [35]:
```
A=10
B=20
```

In [36]:
```
A+B
```

Out[36]: 30

In [37]:
```
A-B
```

Out[37]: -10

In [38]:
```
A*B
```

Out[38]: 200

In [39]:
```
A/2
```

Out[39]: 5.0

In [40]:
```
A*B/2
```

Out[40]: 100.0

In [47]:
```
A%B
```

Out[47]: 10

In [48]:
```
A // B
```

Out[48]: 0

Logical

In [42]:
```
A and B
```

Out[42]: 20

In [43]:
```
A or B
```

Out[43]: 10

In [44]:
```
not A
```

Out[44]: False

In [45]:
```
A not
```

```
  File "<ipython-input-45-e485ee7d67ba>", line 1
    A not
        ^
SyntaxError: invalid syntax
```

In [46]:
```
not B
```

Out[46]:
```
False
```

In [49]:
```
a=10
b=20

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

```
the sum is : 30
the sub is : -10
the div is : 0.5
the mul is : 200
the flowdiv is : 0
the mod is : 10
```

In [52]:
```
g=(input("Enter a value"))
```

```
Enter a value10
```

In [53]:
```
type(g)
```

Out[53]:
```
str
```

In [54]:
```
g=int(input('enter a Value'))
```

```
enter a Value10
```

In [55]:
```
type(g)
```

Out[55]:
```
int
```

In [51]:
```
a=int(input("Enter a value"))
b= int(input("Enter a Value"))
print("the sum is :", a+b)
print("the sub is :", a-b)
print("the div is :", a/b)
print("the mul is :", a*b)
print("the flowdiv is :", a//b)
print("the mod is :", a%b)
```

```
Enter a value20
Enter a Value30
the sum is : 50
the sub is : -10
the div is : 0.6666666666666666
the mul is : 600
the flowdiv is : 0
the mod is : 20
```

In [ ]: