1st July Task/Practise Basics of Python

In [11]:
```
#ARITHMETIC OPERATORS
1 + 1
```

Out[11]: 2

In [2]:
```
2 * 2
```

Out[2]: 4

In [3]:
```
2-1
```

Out[3]: 1

In [4]:
```
4/6
```

Out[4]: 0.6666666666666666

In [5]:
```
12/6
```

Out[5]: 2.0

In [6]:
```
12//6
```

Out[6]: 2

In [7]:
```
12%6
```

Out[7]: 0

In [8]:
```
2 + 2 * 5
```

Out[8]: 12

In [10]:
```
(2+2) //4
```

Out[10]: 1

In [12]:
```
2e4
```

Out[12]: 20000.0

In [13]:
```
2**4
```

Out[13]: 16

In [14]:
```
16 %% 2
```

```
  Cell In[14], line 1
    16 %% 2
       ^
SyntaxError: invalid syntax
```

In [15]:
```
-10//3
```

Out[15]: -4

In [17]:
```
3 + 'nit' #cant add string with number but can mul
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[17], line 1
----> 1 3 + 'nit'

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [19]:
```
3 * ' nit'
```

Out[19]: ' nit nit nit'

In [26]:
```
a,b,c,d,e=10,10.7,10+2j,True,'nit' #declare mul var at same time
```

In [27]:
```
a
```

Out[27]: 10

In [28]:
```
b
```

Out[28]: 10.7

In [29]:
```
c
```

Out[29]: (10+2j)

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

Out[30]: True

In [31]:
```
e
```

Out[31]: 'nit'

In [38]:
```
type(e)
```

Out[38]: str

In [39]:
```
'Full Stack iT'
```

Out[39]: 'Full Stack iT'

In [40]:
```
print('Full Stack iT')
```

```
Full Stack iT
```

In [41]:
```
"naresh it tech"
```

Out[41]: 'naresh it tech'

In [42]:
```
s1='naresh it tech'
```

```
s1
```

Out[42]: 'naresh it tech'

In [43]:
```
a = 5
b = 'hi'
a + b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[43], line 3
      1 a = 5
      2 b = 'hi'
----> 3 a + b

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [45]:
```
print('max it's"Tech"')
```

```
  Cell In[45], line 1
    print('max it's"Tech"')
                        ^
SyntaxError: unterminated string literal (detected at line 1)
```

In [49]:
```
print('max it\'s "Tech"')
```

```
max it's "Tech"
```

In [50]:
```
print('max it','Tech')
```

```
max it Tech
```

In [51]:
```
print('I\'m "Affan"')
```

```
I'm "Affan"
```

In [52]:
```
'nit' + 'nit'
```

Out[52]: 'nitnit'

In [55]:
```
5 * ' nit '
```

Out[55]: ' nit nit nit nit nit '

In [56]:
```
print('c:\nit')
```

```
c:
it
```

In [58]:
```
print(r'c:\nit')
```

```
c:\nit
```

In [59]:
```
x = 5
y = 10
```

In [60]:
```
x + y
```

Out[60]: 15

In [68]:
```
name1='fine'
name1
```

Out[68]:
```
'fine'
```

In [69]:
```
name1[1]
```

Out[69]:
```
'i'
```

In [70]:
```
name[0]='d'
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[70], line 1
----> 1 name[0]='d'

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

In [75]:
```
name1[0:1]
```

Out[75]:
```
'f'
```

In [76]:
```
name1[0:1] = 'd'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[76], line 1
----> 1 name1[0:1] = 'd'

TypeError: 'str' object does not support item assignment
```

In [77]:
```
name1
```

Out[77]:
```
'fine'
```

In [81]:
```
'w'+name1[1:] #this is how u csn changes the string value through index slicing
```

Out[81]:
```
'wine'
```

In [87]:
```
help()
```

```
Welcome to Python 3.12's help utility! If this is your first time using
Python, you should definitely check out the tutorial at
https://docs.python.org/3.12/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To get a list of available
modules, keywords, symbols, or topics, enter "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",
enter "modules spam".

To quit this help utility and return to the interpreter,
enter "q" or "quit".
```

```
Help on class list in module builtins:

class list(object)
 |  list(iterable=(), /)
 |
 |  Built-in mutable sequence.
 |
 |  If no argument is given, the constructor creates a new empty list.
 |  The argument must be an iterable if specified.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return bool(key in self).
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, index, /)
 |      Return self[index].
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __iadd__(self, value, /)
 |      Implement self+=value.
 |
 |  __imul__(self, value, /)
 |      Implement self*=value.
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __mul__(self, value, /)
 |      Return self*value.
 |
```

```
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __reversed__(self, /)
 |      Return a reverse iterator over the list.
 |
 |  __rmul__(self, value, /)
 |      Return value*self.
 |
 |  __setitem__(self, key, value, /)
 |      Set self[key] to value.
 |
 |  __sizeof__(self, /)
 |      Return the size of the list in memory, in bytes.
 |
 |  append(self, object, /)
 |      Append object to the end of the list.
 |
 |  clear(self, /)
 |      Remove all items from list.
 |
 |  copy(self, /)
 |      Return a shallow copy of the list.
 |
 |  count(self, value, /)
 |      Return number of occurrences of value.
 |
 |  extend(self, iterable, /)
 |      Extend list by appending elements from the iterable.
 |
 |  index(self, value, start=0, stop=9223372036854775807, /)
 |      Return first index of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  insert(self, index, object, /)
 |      Insert object before index.
 |
 |  pop(self, index=-1, /)
 |      Remove and return item at index (default last).
 |
 |      Raises IndexError if list is empty or index is out of range.
 |
 |  remove(self, value, /)
 |      Remove first occurrence of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  reverse(self, /)
 |      Reverse *IN PLACE*.
 |
 |  sort(self, /, *, key=None, reverse=False)
 |      Sort the list in ascending order and return None.
 |
 |      The sort is in-place (i.e. the list itself is modified) and stable (i.e.
the
 |      order of two equal elements is maintained).
```

```
 |
 |      If a key function is given, apply it once to each list item and sort the
m,
 |      ascending or descending, according to their function values.
 |
 |      The reverse flag can be set to sort in descending order.
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  __class_getitem__(...)
 |      See PEP 585
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs)
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

```
Help on class tuple in module builtins:

class tuple(object)
 |  tuple(iterable=(), /)
 |
 |  Built-in immutable sequence.
 |
 |  If no argument is given, the constructor returns an empty tuple.
 |  If iterable is specified the tuple is initialized from iterable's items.
 |
 |  If the argument is a tuple, the return value is the same object.
 |
 |  Built-in subclasses:
 |      asyncgen_hooks
 |      MonthDayNano
 |      UnraisableHookArgs
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return bool(key in self).
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, key, /)
 |      Return self[key].
 |
 |  __getnewargs__(self, /)
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __hash__(self, /)
 |      Return hash(self).
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __mul__(self, value, /)
 |      Return self*value.
 |
```

```
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  count(self, value, /)
|      Return number of occurrences of value.
|
|  index(self, value, start=0, stop=9223372036854775807, /)
|      Return first index of value.
|
|      Raises ValueError if the value is not present.
|
|  ----------------------------------------------------------------------
|  Class methods defined here:
|
|  __class_getitem__(...)
|      See PEP 585
|
|  ----------------------------------------------------------------------
|  Static methods defined here:
|
|  __new__(*args, **kwargs)
|      Create and return a new object.  See help(type) for accurate signature.

You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)".  Executing "help('string')"
has the same effect as typing a particular string at the help> prompt.
```

In [88]:
```
PI = 3.14  #in math this is alway constant but python we can chang
PI
```

Out[88]: 3.14

In [91]:
```
PI = 3.69
PI #hence there are no constant values in python we can change them as when we w
```

Out[91]: 3.69

In [92]:
```
type(PI)
```

Out[92]: float

In [93]:
```
a = 5.0
b= int(a)
```

In [94]:
```
a
```

Out[94]: 5.0

In [95]:
```
b
```

Out[95]: 5

In [96]:
```
k=float(b)
```

In [97]:
```
k
```

Out[97]: 5.0

In [98]:
```
k1 = complex(b,k)
```

In [99]:
```
k1
```

Out[99]: (5+5j)

In [100…
```
type(k1)
```

Out[100… complex

In [101…
```
type(k)
```

Out[101… float

In [102…
```
condition = b>k
condition
```

Out[102… False

In [103…
```
type(condition)
```

Out[103… bool

In [104…
```
range(10)
```

Out[104… range(0, 10)

In [105…
```
r = range(0,10)
r
```

Out[105… range(0, 10)

In [107…
```
r1 = list(r)
r1
```

Out[107… [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [108…
```
print(type(r1))
```

```
<class 'list'>
```

In [109…
```
#ASSIGNMENT OPERATOR
```

In [110…
```
x = 2
```

In [111…
```
x = x + 2
x
```

Out[111… 4

In [112…
```
x += 2
```

In [113…
```
x
```

Out[113… 6

In [114…
```
x -=2
```

In [115…
```
x
```

Out[115… 4

In [116…
```
x *=2
x
```

Out[116… 8

In [117…
```
x /=4
x
```

Out[117… 2.0

In [118…
```
x%=8
x
```

Out[118… 2.0

`unary means 1 || binary means 2 `Here we are applying unary minus operator(-) on the operand n; the value of m becomes -7, `which indicates it as a negative value.

In [121…
```
n = -7
n
```

Out[121… -7

In [122…
```
-(n)
```

Out[122… 7

In [123…
```
m = -(n)
m
```

Out[123… 7

In [124…
```
-n
```

Out[124… 7

In [125…
```
# RELATIONAL OPERATOR
```

In [126…
```
a = 5
b = 6
```

In [127…
```
a < b
```

Out[127… True

In [128…
```
b > a
```
Out[128… True

In [129…
```
a != b
```
Out[129… True

In [130…
```
a<=b
```
Out[130… True

In [131…
```
a>=b
```
Out[131… False

In [132…
```
a ==b
```
Out[132… False

In [133…
```
#LOGICAL OPERATOR
```

In [136…
```
a>5 and b<5
```
Out[136… False

In [137…
```
b<a and a>b
```
Out[137… False

In [138…
```
a > 6 or b < 5
```
Out[138… False

In [140…
```
not a
```
Out[140… False

In [141…
```
not b
```
Out[141… False

Number System Conversion

In [142…
```
25
```
Out[142… 25

In [143…
```
bin(25)
```
Out[143… '0b11001'

In [144…
```
oct(25)
```

Out[144… '0o31'

In [145…
```
hex(25)
```

Out[145… '0x19'

In [152…
```
int(0b1110) #binary '0b'
```

Out[152… 14

In [153…
```
int(0o31) #octal '0o'
```

Out[153… 25

In [154…
```
int(0x19) #hexadecimal '0x'
```

Out[154… 25

In [155…
```
0o31
```

Out[155… 25

In [156…
```
type(0o31)
```

Out[156… int

In [158…
```
0xa
```

Out[158… 10

In [159…
```
0xb
```

Out[159… 11

In [160…
```
hex(1)
```

Out[160… '0x1'

In [161…
```
hex(25)
```

Out[161… '0x19'

Swap 2-var in python

In [174…
```
a = 10
b = 5
```

In [175…
```
a
```

Out[175… 10

In [176…
```
b
```

Out[176… 5

In [177…
```
temp = a
a = b
b = temp
print(a)
print(b)
```

```
5
10
```

In [178…
```
#another way to swap is:
```

In [180…
```
a,b = b,a
print(a)
print(b)
```

```
5
10
```

In [182…
```
a = 10
b = 5
a = b + a
b = a - b
a = a - b
```

In [183…
```
print(a)
print(b)
```

```
5
10
```

In [184…
```
0b101
```

Out[184… 5

In [185…
```
0b100
```

Out[185… 4

In [186…
```
type(0b100)
```

Out[186… int

In [187…
```
bin(11)
```

Out[187… '0b1011'

In [189…
```
#XOR
#0 0 - 0
#0 1 - 1
#1 0 - 1
#1 1 - 0
```

In [190…
```
a2 = 5
b2 = 9
```

In [191…
```
#there is other way to work using swap variable also which is XOR because it wil
a2 = a2 ^ b2
b2 = a2 ^ b2
a2 = a2 ^ b2
```

In [192…
```
print(a2)
print(b2)
```

```
9
5
```

In [193…
```
a ^ b
```

Out[193… 15

In [194…
```
a < 5 ^ b > 6
```

Out[194… True

In [195…
```
a2 = 10
b2 = 20
```

In [196…
```
a2,b2=b2,a2
print(a2)
print(b2)
```

```
20
10
```

#BITWISE OPERATORWE HAVE 6 OPERATORS COMPLEMENT (~) || AND (&) || OR (|) || XOR (^) || LEFT SHIFT (<<) || RIGHT SHIFT (>>)

In [198…
```
12 & 13 #in back it calculates with their binary form using 'and' table
```

Out[198… 12

In [199…
```
12 | 13
```

Out[199… 13

In [200…
```
1 & 0
```

Out[200… 0

In [201…
```
0 | 1
```

Out[201… 1

In [202…
```
1 ^ 1
```

Out[202… 0

In [203…
```
bin(13)
```

Out[203… '0b1101'

In [204…
```
35 & 40
```

Out[204… 32

In [205…
```
35 | 40
```

Out[205… 43

In [206…
```
25^30
```

Out[206… 7

In [209…
```
10<<1 #in backend it add zeros at the end
```

Out[209… 20

In [210…
```
10<<2
```

Out[210… 40

In [212…
```
10>>2 #it looses the digits by 2
```

Out[212… 2

In [213…
```
# BIT WISE LEFT SHIFT OPERATOR
# in left shift what we need to to we need shift in left hand side & need to shi
#bit wise left operator bydefault you will take 2 zeros ( )
#10 binary operator is 1010 | also i can say 1010
10<<2
```

Out[213… 40

In [214…
```
bin(10)
```

Out[214… '0b1010'

In [215…
```
10 >> 1
```

Out[215… 5

In [216…
```
10>>4
```

Out[216… 0

In [217…
```
10>>5
```

Out[217… 0

In [218…
```
10>>10
```

Out[218… 0

In [219…
```
10>>1
```

Out[219… 5

In [220…
```
10>>3
```

Out[220… 1

In [221…
```
bin(20)
```

Out[221… '0b10100'

basic python practise completed

In [ ]: