Numbers

In [1]:
```
1+3
```

Out[1]:
```
4
```

In [2]:
```
print(1+3)
```

```
4
```

In [3]:
```
a=5
a
```

Out[3]:
```
5
```

In [4]:
```
a=5
print(a)
```

```
5
```

In [5]:
```
print("Hello World")
```

```
Hello World
```

In [7]:
```
# print is an inbuilt function that is used to print the thing that you want to
```

In [9]:
```
print("Hello World ! My name is Ajeet Kumar and i am currently working in Octopy
```

```
Hello World ! My name is Ajeet Kumar and i am currently working in Octopyder Se
rvices
```

Variable

In [10]:
```
# Variable is the name that is used to store the data and value within them.
# you can considered a variable as a container that is used to store the value w
```

In [11]:
```
name="Ajeet"
company= "Octopyde Services"
```

In [12]:
```
print(name)
print(company)
```

```
Ajeet
Octopyde Services
```

In [13]:
```
number =20
number
```

Out[13]:
```
20
```

In [14]:
```
type('')
```

Out[14]:
```
str
```

In [15]:
```
# type() function is used to check the type of the data type used
```

In [16]:
```
type(name)
type(company)
```

Out[16]:
```
str
```

In [17]:
```
type(12)
```

Out[17]:
```
int
```

In [18]:
```
float(2)
```

Out[18]:
```
2.0
```

In [19]:
```
type(23324.4)
```

Out[19]:
```
float
```

In [21]:
```
1+2j
```

Out[21]:
```
(1+2j)
```

In [22]:
```
print(type(1+2j))
```

```
<class 'complex'>
```

In [23]:
```
a= 23+ 34j
print(a)
```

```
(23+34j)
```

In [24]:
```
print(type(a))
```

```
<class 'complex'>
```

Rule for varible declaration

In [25]:
```
# a variable name can start with small as well as capital letter
# A variable name cannot start with number
# A varible name can have number in-between.
# A variable name contain alphabet, number, and underscore.
# A variable name can start with underscore
```

In [26]:
```
company="Octopyder services"
Company="Kivdanti"
print(company)
print(Company)
```

```
Octopyder services
Kivdanti
```

In [27]:
```
# variable name is case sensitive
```

In [30]:
```
#Reserved Keyword:
'''
int,float,len,complex,str,return, continue
'''
```

Out[30]:
```
'\nint,float,len,complex,str,return, continue\n'
```

In [32]:
```
# Boolean
True
type(True)
```

Out[32]: bool

In [33]:
```
print(0 and 1)
```

```
0
```

In [34]:
```
True and False
```

Out[34]: False

In [35]:
```
True or False
```

Out[35]: True

In [36]:
```
True and True
```

Out[36]: True

In [37]:
```
not True
```

Out[37]: False

In [38]:
```
not False
```

Out[38]: True

In [39]:
```
type(not False)
```

Out[39]: bool

In [40]:
```
bool(0)
```

Out[40]: False

In [41]:
```
bool(1)
```

Out[41]: True

In [42]:
```
bool(23)
```

Out[42]: True

In [43]:
```
bool(100)
```

Out[43]: True

In [46]:
```
a=1
if (bool(a))==True:
    print("Ajeet Kumar")
```

```
Ajeet Kumar
```

In [47]:
```
# Typecasting is a process of converting a data type into another data type valu
```

In [49]:
```
int("12")
```

Out[49]: 12

In [50]:
```
type(12)
```

Out[50]: int

Dynamic Typing

In [52]:
```
# Python is a dynamic type programming language it means that we did not need to
```

In [53]:
```
a=12
str1="Ajeet"
a
```

Out[53]: 12

In [54]:
```
str1
```

Out[54]: 'Ajeet'

In [57]:
```
int(234.4435)
```

Out[57]: 234

In [58]:
```
## Concatenation between different types
```

In [59]:
```
"1"+"1"
```

Out[59]: '11'

In [60]:
```
int("1"+"1")
```

Out[60]: 11

In [61]:
```
# concetenation is possible only for same data type
```

In [ ]:
```

```