NB 01 - Python Fundamentals (Basic Operations & Data Types)

August 16, 2021

1 Basic mathematical operations can be done

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

```
4
```

[2]:
```
print(1*3)
```

```
3
```

[3]:
```
print(3**2)
```

```
9
```

[4]:
```
print(pow(3,2))
```

```
9
```

[5]:
```
print(5/2)
```

```
2.5
```

[6]:
```
print(5//2)
```

```
2
```

[7]:
```
print(5%2)
```

```
1
```

2 Print function is not needed in Jupyter notebook for printing

[8]:
```
12+34-32
```

[8]: 14

3 Basic relational operations can be done

[9]:
```
45>20
```

[9]: True

[10]:
```
23==34
```

[10]: False

[11]:
```
32<=32
```

[11]: True

[12]:
```
24!=45
```

[12]: True

4 Basic logocal operations can be done

[13]:
```
3>2 and 3!=4
```

[13]: True

[14]:
```
3==4 and 5<7
```

[14]: False

[15]:
```
3==4 or 5<7
```

[15]: True

[16]:
```
not 3==4
```

[16]: True

5 Primitive data types

[17]:
```
type(2)
```

[17]: int

[18]:
```
type(3.123)
```

[18]: float

[19]:
```
type("Dog")
```

[19]: str

[20]:
```
type(True)
```

[20]: bool

[21]:
```
type(False)
```

[21]: bool

6 Checking whether a value is in a specific type or not

[22]:
```
isinstance(2,int)
```

[22]: True

[23]:
```
isinstance(3.455,float)
```

[23]: True

[24]:
```
isinstance(3.455,int)
```

[24]: False

7 Type casting

[25]:
```
int(3.212)
```

[25]: 3

[26]:
```
float(3)
```

[26]: 3.0

[27]:
```
int("45")
```

[27]: 45

8 Python variables

[28]:
```
x=20
y=30
print(x+y)
```

```
50
```

[29]:
```
z=x-y
print(z)
```

```
-10
```

[30]:
```
s=x>30
print(s)
```

```
False
```

[31]:
```
p="Today is a beatiful day!!"
print(p)
```

```
Today is a beatiful day!!
```

9 Special features of Python variables

[32]:
```
a=b=c=10
```

[33]:
```
a
```

[33]: 10

[34]:
```
b
```

[34]: 10

[35]:
```
c
```

[35]: 10

[36]:
```
p,q=10,20
```

[37]:
```
p
```

[37]: 10

[38]:
```
q
```

[38]: 20

10 User inputs

[39]:
```
name=input("Please enter your name: ")
print(name)
```

[40]:
```
age=input("Please enter your age: ")
print(age)
```

[41]:
```
type(age)
```

[41]:
```
str
```

[42]:
```
age_2years=int(age)+2
print(age_2years)
```

```
␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-42-28e26148eeb7> in <module>
   ----> 1 age_2years=int(age)+2
         2 print(age_2years)


       ValueError: invalid literal for int() with base 10: ''
```

[ ]:
```
num1=float(input("Enter the first value:"))
num2=float(input("Enter the second value:"))

num3=num1+num2
print(num3)
```

11 Python comments

[ ]:
```
#This is the first number
num1=20

#This is the second number
num2=30

print(num1+num2)
```

[ ]:
```
'''
This is the doc string
This can be used for multiple lines as well
'''
```

```
print(25+40)
```

12 Compound data types

13 Lists

[ ]:
```
L=[23,33,22,23,21,23]
print(L)
```

[ ]:
```
S=["Cat","Dog","Cow"]
print(S)
```

[ ]:
```
K=["Cat",23,33,43,True]
print(K)
```

[ ]:
```
len(L)
```

[ ]:
```
type(L)
```

14 Adding lists

[ ]:
```
L1=[23,33,21,23]
L2=[45,43,32]
L=L1+L2
print(L)
```

15 Nested lists

[ ]:
```
L=[12,22,23,21,[23,33,45,54],"Man"]
print(L)
```

[ ]:
```
len(L)
```

16 Subsetting lists

17 Indexing

[ ]:
```
L=[4,5,3,6,8,3,7,9,12,26,43]
```

[ ]:
```
L[0]
```

[ ]:
```
L[2]
```

[ ]:
```
L[-1]
```

[ ]:
```
L[-3]
```

18 Slicing

[ ]:
```
L[1:5]
```

[ ]:
```
L[2:]
```

[ ]:
```
L[:5]
```

[ ]:
```
L[::5]
```

[ ]:
```
L[:-1]
```

19 Lists are mutable

[ ]:
```
L=[23,34,45,56,66]
```

[ ]:
```
L[0]=100
```

[ ]:
```
L
```

20 Aliasing

[ ]:
```
a=[12,22,23,34,45,56,44,43,46]
b=a
```

[ ]:
```
b
```

[ ]:
```
b[0]=200
```

[ ]:
```
b
```

[ ]:
```
a
```

21 Cloaning

[ ]:
```
a=[12,22,23,34,45,56,44,43,46]
b=a[:]
```

[ ]:
```
b
```

[ ]:
```
b[0]=200
```

[ ]:
```
b
```

[ ]:
```
a
```

[ ]:
```
c=a.copy()
```

22 Appending new elements

[ ]:
```
L=[12,22,34,32]
L.append(100)
print(L)
```

[ ]:
```
L=[]
L.append(30)
L.append(40)
L.append(50)
print(L)
```

[ ]:
```
L=[12,22,34,32]
L.append([40,50,60])
print(L)
```

23 Extending lists

[ ]:
```
L=[12,22,34,32]
L.extend([40,50,60])
print(L)
```

24 Inserting a specific element to a specific place

[ ]:
```
L=[23,36,54,60,70]
L.insert(3,100)
print(L)
```

25 Removing a specific elemend from a list

[ ]:
```
L=[23,36,54,60,70]
L.remove(60)
print(L)
```

26 Removing an elemend in a specific index of a list

[ ]:
```
L=[23,36,54,60,70]
del(L[2])
print(L)
```

27 Pop out the last element in a list

[ ]:
```
L=[12,22,23,34,32,33]
L.pop()
```

[ ]:
```
L=[12,22,23,34,32,33]
L.pop(2)
```

28 Membership test

[ ]:
```
P=[34,33,32,34,32,33,56]
```

[ ]:
```
34 in P
```

[ ]:
```
60 in P
```

[ ]:
```
60 not in P
```

29 Sorting lists

[ ]:
```
L=[45,33,32,34,45,43,33,67,65]
```

[ ]:
```
L.sort()
```

[ ]:
```
L
```

[ ]:
```
L=[45,33,32,34,45,43,33,67,65]
```

[ ]:
```
P=sorted(L)
```

[ ]:
```
P
```

[ ]:
```
L
```

30 Counting elements

[ ]:
```
L=[33,32,33,34,45,44,33,45,67,33]
```

[ ]:
```
L.count(33)
```

31 Index of an element

[ ]:
```
L=[33,32,33,34,45,44,33,45,67,33]
```

[ ]:
```
L.index(45)
```

32 Tuple

33 Creating tuples

[ ]:
```
T=(23,33,34,45,56)
print(T)
```

```
(23, 33, 34, 45, 56)
```

[ ]:
```
F=23,33,34,45,56
print(F)
```

34 Indexing and slicing is similar to the lists

[ ]:
```
T=(23,33,34,45,56)
print(T[3])
```

[ ]:
```
T=(23,33,34,45,56)
print(T[1:3])
```

[ ]:
```
T=(55,45,56,67,32,[33,45,56,67])
```

[ ]:
```
T[5][2]
```

35 Tuples are not mutable

[ ]:
```
T=(55,45,56,67,32,[33,45,56,67])
```

[ ]:
```
T[0]=30 #This will give an error
```

[ ]:
```
T[5][2]=100
```

[ ]:
```
T
```

36 Unpacking tuples

[ ]:
```
T=(33,32,34,45,56)
a,b,c,d,e=T
```

[ ]:
```
a
```

[ ]:
```
b
```

[ ]:
```
c
```

[ ]:
```
d
```

[ ]:
```
e
```

37 Membership test in tuples

[ ]:
```
T=(23,33,32,34,45)
```

[ ]:
```
23 in T
```

[ ]:
```
60 in T
```

[ ]:
```
60 not in T
```

38 Index of an element

[ ]:
```
T=(23,33,34,56,78)
```

[ ]:
```
T.index(34)
```

39 Count elements

[ ]:
```
T=(33,32,34,34,45,66,34,78)
```

[ ]:
```
T.count(34)
```

40 Sets

41 Creating sets

[ ]:
```
S={23,33,32,22,24,56,43,33,33,33,56}
```

[ ]:
```
S
```

42 Adding an element to a set

[ ]:
```
S={23,33,45}
S.add(50)
print(S)
```

43 Adding multiple elements to a list

[ ]:
```
S={23,33,45}
S.update([50,50,60,70])
print(S)
```

44 Removing elements

[ ]:
```
S={23,33,45}
S.remove(45)
print(S)
```

[ ]:
```
S={23,33,45}
S.discard(45)
print(S)
```

45 Union between two sets

[ ]:
```
A={2,3,4,5,6}
B={4,5,6,7,8}
```

[ ]:
```
A.union(B)
```

[ ]:
```
B.union(A)
```

46 Intersection between two sets

[ ]:
```
A.intersection(B)
```

[ ]:
```
B.intersection(A)
```

47 Casting compund data

[ ]:
```
L=[2,3,4,5,6,6,6,7,8,9]
```

[ ]:
```
T=tuple(L)
print(T)
```

```
[ ]: S=set(L)
     print(S)
```

```
[ ]: T=(4,5,6,7)
```

```
[ ]: L=list(T)
     print(L)
```

48 Dictionary

49 Creating dictionaries

```
[ ]: d={"Name":["Sam","Kane","Jane"],"Age":[23,33,45]}
```

```
[ ]: d
```

```
[ ]: g=dict([("A",20),("B",[40,30,20])])
```

```
[ ]: g
```

50 Keys & values

```
[ ]: d={"Name":["Sam","Kane","Jane"],"Age":[23,33,45]}
```

```
[ ]: d.keys()
```

```
[ ]: d.values()
```

51 Membership test

```
[ ]: d={"Name":["Sam","Kane","Jane"],"Age":[23,33,45]}
```

```
[ ]: "Name" in d
```

```
[ ]: True
```

52 Accessing elements

```
[ ]: d={"Name":["Sam","Kane","Jane"],"Age":[23,33,45]}
```

```
[ ]: d["Name"]
```

```
[ ]: d.get("Age")
```

53 Changing elements

[ ]:
```
k={"A":30,"B":70}
```

[ ]:
```
k["A"]=100
```

[ ]:
```
k
```

54 Adding a new element

[ ]:
```
k={"A":30,"B":70}
```

[ ]:
```
k["C"]=200
```

[ ]:
```
k
```

55 Removing an element

[ ]:
```
k={"A":30,"B":70}
```

[ ]:
```
del k["A"]
```

[ ]:
```
k
```