

Internship Program – AkashTechnolabs

Name: KamiyabHusen Bhatt

College: L.D. College of Engineering

Sem: 7th (BE-IT)

Date: 26th May 2021

Day-2-Report **(Work Summary)**

INTERNSHIP at AkashTechnoLabs

- ❖ Today it was the second day of our Internship.
- ❖ First, akash sir start with how was previous day experience?
- ❖ Then, sir solved doubts of previous day task.
- ❖ Afterwards, devanshi mam started our topic basic of python.

Day-2: What we learnt?

- ✓ How to define single line comments (with #) and multiline comments (with `""" """` or `''' '''`).
- ✓ How to define variables ex: name: 'string', number: 100.
- ✓ How to define various datatypes like numbers, tuple, list, dictionary, Boolean, set etc.
- ✓ How to do type casting:
 - `int()` – for interger type casting
 - `float()` – for float type casting
- ✓ `type()` function is used to know class of variable or value.
- ✓ `isinstance()` function used to check if an object belongs to a particular class.

Internship Program – AkashTechnolabs

Task – 1 (Basic): -

1) Comments:

```
#Single Line like this
'''
Multiline
Comments
like this
'''
"""
Multiline
Comments
also like this
"""
```

2) Variables:

```
#creating variables
print("-----creating variables-----")
x = 5
y = "Kamiyab"
print(x)
print(y)

#assign value to multiple variables
print("-----assign value to multiple variables-----")
x, y, z = "Kamiyab", "Husen", "Bhatt"
print(x)
print(y)
print(z)
```

Internship Program – AkashTechnolabs

Output:

```
-----creating variables-----  
5  
Kamiyab  
-----assign value to multiple variables-----  
Kamiyab  
Husen  
Bhatt
```

3) Various Datatypes:

```
# int datatype  
print("-----Int Datatype-----")  
num1 = 10 # decimal form  
num2 = 0B111 # binary form (B or b can use)  
num3 = 0O123 # octal form (O or o can use)  
num4 = 0Xabc # hexadecimal form (X or x can use)  
print(f'decimal: {num1}')print(f'binary: {num2}')print(f'octal: {num3}')print(f'hexadecimal: {num4}')  
# float datatype  
print("-----Float Datatype-----")  
num5 = 1.234 # decimal form  
num6 = 1.2e3 # exponential form (E or e can use)  
print(f'decimal: {num5}')print(f'exponential: {num6}')  
# complex datatype  
print("-----Complex Datatype-----")  
num7 = 3+5.5j  
print(f'number: {num7}')print(f'real part: {num7.real}')print(f'imaginary part: {num7.imag}')  
# boolean datatype  
print("-----Boolean Datatype-----")  
bool1 = True # True/False value
```

Internship Program – AkashTechnolabs

```
print(bool1)
print(True+False)    # True=1 & False=0

# string datatype
print("-----String Datatype-----")
str1 = "Python is Cool!!"
print(str1)
```

Output:

```
-----Int Datatype-----
decimal: 10
binary: 7
octal: 83
hexadecimal: 2748
-----Float Datatype-----
decimal: 1.234
exponential: 1200.0
-----Complex Datatype-----
number: (3+5.5j)
real part: 3.0
imaginary part: 5.5
-----Boolean Datatype-----
True
1
-----String Datatype-----
Python is Cool!!
```

4) List Datatype:

```
# collection of different type of elements.
# list is mutable(it can be add/modify/remove) and
duplicate elements are supported.
languages=['Python',
'Java', 'C', 'C++', 'HTML', 'CSS', 'C', 'JavaScript', '.NET', 10, 10
, 30]
number=[90, 25, 5, 80, 40]

print(f'languages:{languages}\n')

print("-----Accessing Elements-----")
print(f'{languages[1]}')
print(f'{languages[-3]}')
```

Internship Program – AkashTechnolabs

```
for item in languages:
    print(f'item:{item}')
print("\n")

print("-----Slicing Elements-----")
print(f'{languages[1:5]}')
print(f'{languages[1:]}')
print(f'{languages[:6]}')
print(f'{languages[:]}')
print(f'{languages[::-2]}')
```

Output:

```
languages:['Python', 'Java', 'C', 'C++', 'HTML', 'CSS', 'C', 'JavaScript', '.NET', 10, 10, 30]
```

```
-----Accessing Elements-----
```

```
Java
10
item:Python
item:Java
item:C
item:C++
item:HTML
item:CSS
item:C
item:JavaScript
item:.NET
item:10
item:10
item:30
```

```
-----Slicing Elements-----
```

```
['Java', 'C', 'C++', 'HTML']
['Java', 'C', 'C++', 'HTML', 'CSS', 'C', 'JavaScript', '.NET', 10, 10, 30]
['Python', 'Java', 'C', 'C++', 'HTML', 'CSS']
['Python', 'Java', 'C', 'C++', 'HTML', 'CSS', 'C', 'JavaScript', '.NET', 10, 10, 30]
[30, 10, 'JavaScript', 'CSS', 'C++', 'Java']
```

5) Tuple Datatype:

```
#collection of different elements.
#tuple is immutable(it cannot be add/modify/remove) and
duplicate elements are supported.
animals =
```

Internship Program – AkashTechnolabs

```
('tiger','lion','panther','leopard','tiger','elephant',10,20,30,10)

print(f'animals:{animals}\n')

print("-----Accessing Elements-----")
print(animals[4])
print(animals[-2])
for animal in animals:
    print(f'animal[{animal}]')
print("\n")

print("-----Slicing Elements-----")
print(animals[1:5])
print(animals[:6])
print(animals[2:])
print(animals[:])
print(animals[-3:])
print("\n")
```

Internship Program – AkashTechnolabs

Output:

```
animals:('tiger', 'lion', 'panther', 'leopard', 'tiger', 'elephant', 10, 20, 30, 10)
```

```
-----Accessing Elements-----
```

```
tiger
30
animal[tiger]
animal[lion]
animal[panther]
animal[leopard]
animal[tiger]
animal[elephant]
animal[10]
animal[20]
animal[30]
animal[10]
```

```
-----Slicing Elements-----
```

```
('lion', 'panther', 'leopard', 'tiger')
('tiger', 'lion', 'panther', 'leopard', 'tiger', 'elephant')
('panther', 'leopard', 'tiger', 'elephant', 10, 20, 30, 10)
('tiger', 'lion', 'panther', 'leopard', 'tiger', 'elephant', 10, 20, 30, 10)
(20, 30, 10)
```

6) Set Datatype:

```
#collection of different elements.
#set is mutable(it can be add/remove) and duplicate
elements are not supported.
numset = {10,20,45,60,10,50,'tiger'}
print(numset)
numset.remove(50)
print(numset)
numset.pop()
print(numset)
num1=numset.copy()
print(num1)
numset.add(65)
print(numset)
print(numset.difference(num1))
num2={'hii','byee'}
numset.update(num2)
print(numset)
```

Internship Program – AkashTechnolabs

Output:

```
{'tiger', 10, 45, 50, 20, 60}
{'tiger', 10, 45, 20, 60}
{10, 45, 20, 60}
{10, 20, 45, 60}
{65, 10, 45, 20, 60}
{65}
{65, 'hii', 10, 'bye', 45, 20, 60}
```

7) Dictionary Datatype:

```
#dictionaries are mutable(it can be add/modify/remove) and
duplicate values are not supported
import pandas as pd
colors = {
    1:"red",
    2:"blue",
    3:"black",
    4:"pink",
    5:"white"
}

print("-----Accessing Elements-----")
print(f'colors:{colors}')
print(f'item:{colors[2]}')
print(f'get:{colors.get(6)}')
print(f'keys:{colors.keys()}')
print(f'values:{colors.values()}')
```

Output:

```
-----Accessing Elements-----
colors:{1: 'red', 2: 'blue', 3: 'black', 4: 'pink', 5: 'white'}
item:blue
get:None
keys:dict_keys([1, 2, 3, 4, 5])
values:dict_values(['red', 'blue', 'black', 'pink', 'white'])
```


Internship Program – AkashTechnolabs

8) type():

```
num1 = 1000
num2 = 100.50
num3 = 3+5.5j
bool1 = True
str1 = 'kamiyab'
list1 = ["apple", "orange", 10, 20, 10]
tuple1 = (10, "apple", 20, 10)
set1 = {10, 20, 30, 'apple', 40, 20}
dict1 = {1: "apple", 2: "orange"}

print(type(num1))
print(type(num2))
print(type(num3))
print(type(bool1))
print(type(str1))
print(type(list1))
print(type(tuple1))
print(type(set1))
print(type(dict1))
```

Output:

```
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'bool'>
<class 'str'>
<class 'list'>
<class 'tuple'>
<class 'set'>
<class 'dict'>
```

Internship Program – AkashTechnolabs

Task – 2 (Advance): -

CRUD Operations using MySQL

GitHub Link:- [https://github.com/kamiyab786/Internship-AkashTechnolabs/blob/main/Task2/Python CRUD Operation.py](https://github.com/kamiyab786/Internship-AkashTechnolabs/blob/main/Task2/Python%20CRUD%20Operation.py)

1) Insert:

```
DB already exists
table already exists
choose option
1.insert data
2.read data
3.update data
4.delete data
enter your choice:1
-----We are inserting data-----
enter name:kamiyab
enter marks:98
insert into student(name,marks) values('kamiyab',98);
```

Internship Program – AkashTechnolabs

2) Read:

```
D:\PYTHON\kamiyab\venv\Scripts\python.exe D:/Python/kamiyab/Python_CRUD_Operation.py
DB already exists
table already exists
choose option
1.insert data
2.read data
3.update data
4.delete data
enter your choice:2
-----retrive data-----
id name marks
1 husen 100
2 kamiyab 98
```

3) Update:

```
D:\PYTHON\kamiyab\venv\Scripts\python.exe D:/Python/kamiyab/Python_CRUD_Operation.py
DB already exists
table already exists
choose option
1.insert data
2.read data
3.update data
4.delete data
enter your choice:3
-----Update Data-----
enter student id to update 1
select option for update
1.update only name
2.update only marks
3.update both
enter update option:2
----update only marks---
enter new marks :95
```

Internship Program – AkashTechnolabs

4) Delete:

```
D:\PYTHON\kamiyab\venv\Scripts\python.exe D:/Python/kamiyab/Python_CRUD_Operation.py
DB already exists
table already exists
choose option
1.insert data
2.read data
3.update data
4.delete data
enter your choice:4
Enter ID to be Deleted 2
```

```
-----retrive data-----
id name marks
1 husen 95
```

Internship Program – AkashTechnolabs

5) MySQL Database:

The screenshot displays the MySQL Workbench interface for a local instance of MySQL 8.0. The left sidebar shows the 'SCHEMAS' tree with a search filter. The 'demo' database is selected, and the 'student' table is highlighted under the 'Tables' folder. The main query editor shows a query: `SELECT * FROM demo.student;`. The 'Result Grid' tab is active, displaying the query results in a table format. The table has three columns: 'id', 'name', and 'marks'. The first row shows '1', 'husen', and '95'. The second row shows 'NULL', 'NULL', and 'NULL'. The bottom status bar indicates 'student 1' is selected. The 'Output' pane at the bottom shows the execution log with four entries: a successful query execution, an 'Apply changes to student' message, and two more successful query executions.

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Filter objects

SCHEMAS

- demo
 - Tables
 - student
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
 - sakila
 - sys
 - world

Query 1 student student - Table student student x

Limit to 1000 rows

1 • `SELECT * FROM demo.student;`

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid

	id	name	marks
▶ 1	1	husen	95
*	NULL	NULL	NULL

Form Editor

Field Types

Administration Schemas

Information

Table: student

Columns:

- id int AI PK
- name varchar(255)
- marks int

Object Info Session

student 1 x Apply Revert Context Help Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	10:12:49	SELECT * FROM demo.student LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
✓ 2	10:13:21	Apply changes to student	Changes applied	
✓ 3	10:13:38	SELECT * FROM demo.student LIMIT 0, 1000	1 row(s) returned	0.032 sec / 0.000 sec
✓ 4	12:18:10	SELECT * FROM demo.student LIMIT 0, 1000	1 row(s) returned	0.031 sec / 0.000 sec