

Python

Python data types

- Integer
- Float
- String (single, double, triple quotes-for multi-line string)
- Complex ($t=10+20j$) (j is used for imaginary variable)
- Boolean (T should be capital in True, else error)

All fundamental data types are immutable

Dynamically type Programming Language (datatype is automatically interpreted)

Slicing

Data Structures of python: List, Tuples, Dictionary

List [] :

1. Heterogeneous objects are allowed
2. Duplicates are allowed
3. Order is preserved
4. Indexing and slicing are possible
5. Mutable
6. Enclosed in square brackets

Tuple () :

It is exactly same as list, only diff is tuple is immutable

SET:

1. Heterogeneous Object
2. Duplicates are not allowed
3. Order is not preserved
4. Slicing and indexing are not possible
5. Mutable

Dict { } :

1. Key Value
2. Heterogeneous
3. Duplicates are not allowed
4. Mutable

Flow Control: if-else, for loop



```
In [1]: print('Welcome')  
Welcome
```

```
In [2]: a=10  
type(a)
```

```
Out[2]: int
```

```
In [3]: a=10  
print(type(a))  
print(a)  
  
<class 'int'>  
10
```

```
In [4]: a=10.0  
type(a)
```

```
Out[4]: float
```

```
In [9]: c="""Hi  
welcome to shell"""  
type(c)
```

```
Out[9]: str
```

```
In [10]: a=True  
print(type(a))  
  
<class 'bool'>
```

```
In [11]: a=10  
b=20  
c=a>b  
print(c)  
  
False
```

```
In [12]: print(True+True)  
  
2
```

```
In [13]: print(True-False)  
  
1
```

```
In [15]: c=10+20j  
print(type(c))  
  
<class 'complex'>
```

```
In [16]: a=10  
print(id(a))  
  
8885288
```

```
In [17]: b=10  
print(id(b))  
  
8885288
```

```
In [18]: a=20  
print(id(a))  
  
8885608
```

```
In [20]: s="Data Engineering"
s[-2]
```

```
Out[20]: 'n'
```

```
In [21]: s[2]
```

```
Out[21]: 't'
```

```
In [22]: s[0:4]
```

```
Out[22]: 'Data'
```

```
In [24]: s="abcdefghijklmnopqrstuvwxy"
s[12:15]
```

```
Out[24]: 'mno'
```

```
In [25]: s[-3:]
```

```
Out[25]: 'xyz'
```

```
In [26]: s[:14]
```

```
Out[26]: 'abcdefghijklmnop'
```

```
In [27]: s[3:1000]
```

```
Out[27]: 'defghijklmnopqrstuvwxy'
```

```
In [28]: a=10
b=20
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a//b)
```

```
30
-10
200
0.5
0
```

```
In [29]: a=3
b=2
print(a/b)
print(a//b)
```

```
1.5
1
```

```
In [30]: a=3
b=2.0
print(a/b)
print(a//b)
```

```
1.5
1.0
```

```
In [33]: a=10.0
b=2.0
print(a/b)
print(a//b)
print(a%b)
print(a**b)
```

```
5.0
5.0
0.0
100.0
```

```
In [37]: is_geological_survey_complete = True
is_environmental_clearance_received = True
is_market_demand_high = False
```

```
In [39]: should_start_drilling = is_geological_survey_complete and is_environmental_clearance_received and is_market_demand_high
<----->
```

```
In [40]: True and True and False
```

```
Out[40]: False
```

```
In [41]: print(should_start_drilling)
```

```
False
```

```
In [42]: # Boolean Variables
HasExplorationPermit = True
HasDrillingRights = True
HasEnvironmentalApproval = False
HasOilDiscovery = True

# Logical Operations
IsExplorationAllowed = HasExplorationPermit and HasDrillingRights and not HasEnvironmentalApproval
IsDiscoveryProfitable = HasOilDiscovery and (HasExplorationPermit or HasDrillingRights)
```

```
In [43]: print("Is exploration allowed?", IsExplorationAllowed)
print("Is discovery profitable?", IsDiscoveryProfitable)
```

```
Is exploration allowed? True
Is discovery profitable? True
```

```
In [44]: 10+20
```

```
Out[44]: 30
```

```
In [45]: "ten"+"twenty"
```

```
Out[45]: 'tentwenty'
```

```
In [44]: 10+20
```

```
Out[44]: 30
```

```
In [45]: "ten"+"twenty"
```

```
Out[45]: 'tentwenty'
```

```
In [46]: 10*20
```

```
Out[46]: 200
```

```
In [49]: "ten"*"twenty"
```

```
-----
TypeError                                Traceback (most recent call last)
Cell In[49], line 1
----> 1 "ten"*"twenty"

TypeError: can't multiply sequence by non-int of type 'str'
```

```
In [50]: "ten"*2
```

```
Out[50]: 'tenten'
```

```
In [53]: a=[10,"python",10,19.5,True]
```

```
In [54]: print(a)
print(type(a))

[10, 'python', 10, 19.5, True]
<class 'list'>
```

```
In [55]: a[0]
```

```
Out[55]: 10
```

```
In [56]: a[-1]
```

```
Out[56]: True
```

```
In [57]: a[1:]
```

```
Out[57]: ['python', 10, 19.5, True]
```

```
In [58]: a.append(999)
```

```
In [59]: a
```

```
Out[59]: [10, 'python', 10, 19.5, True, 999]
```

```
In [65]: a=("Well A","2023-09-15","Pipeline",True,100,100)
type(a)
```

```
Out[65]: tuple
```

```
In [66]: a[2]
```

```
Out[66]: 'Pipeline'
```

```
In [67]: name="naval"
working="Data Engineer"
print("I am {} working as {}".format(name,working))

I am naval working as Data Engineer
```

```
In [69]: print(f"I am {name} working as {working}")

I am naval working as Data Engineer
```

```
In [68]: print("I am %s working as %s" %(name,working))

I am naval working as Data Engineer
```

```
In [70]: # Employee details
name = "John Doe"
job_title = "Senior Geologist"
department = "Geology"
email = "johndoe@email.com"
phone = "123-456-7890"
```

```
In [71]: employee_info=f"Employee Information: \nName: {name} \nJob Title: {job_title} \nDepartment: {department} \nEmail: {email} \nPhone: {phone}"
```

```
In [72]: print(employee_info)

Employee Information:
Name: John Doe
Job Title: Senior Geologist
Department: Geology
Email: johndoe@email.com
Phone: 123-456-7890
```

```
In [73]: c={10,10.5,"Shell",True,True,10,10,10,"Python"}
print(c)

{True, 'Shell', 10.5, 10, 'Python'}
```

```
In [74]: c.add(55)
print(c)

{True, 'Shell', 10.5, 55, 10, 'Python'}
```

```
In [75]: print(id(c))

140346222554784
```

```
In [76]: c.add("sql")
print(c)
print(id(c))

{True, 'Shell', 10.5, 'sql', 55, 10, 'Python'}
140346222554784
```

```
In [77]: d={"location":"WellA","Start_date":"2023-1-1","Duration":45,"End":True}
print(type(d))
print(d)

<class 'dict'>
{'location': 'WellA', 'Start_date': '2023-1-1', 'Duration': 45, 'End': True}
```

```
In [78]: d={"location":"WellA","Start_date":"2023-1-1","Duration":45, "Start_date":"2023-11-1"}
print(d)

{'location': 'WellA', 'Start_date': '2023-11-1', 'Duration': 45}
```

```
In [79]: d={"location":"WellA","Start_date":"2023-1-1","Duration":45, "End_date":"2023-1-1"}
print(d)

{'location': 'WellA', 'Start_date': '2023-1-1', 'Duration': 45, 'End_date': '2023-1-1'}
```

```
In [80]: d['location']='Well B'
print(d)

{'location': 'Well B', 'Start_date': '2023-1-1', 'Duration': 45, 'End_date': '2023-1-1'}
```

```
In [82]: # List of Equipment
refineryEquipment = ["Crude Distillation Unit", "Catalytic Cracking Unit", "Hydrotreating Unit", "FCC Unit"]

# Membership Operator
IsUnitInstalled = "Hydrotreating Unit" in refineryEquipment
IsUnitObsolete = "Thermal Cracking Unit" not in refineryEquipment

print("Is Hydrotreating Unit installed?", IsUnitInstalled)
print("Is Thermal Cracking Unit obsolete?", IsUnitObsolete)

Is Hydrotreating Unit installed? True
Is Thermal Cracking Unit obsolete? True
```

```
In [83]: employees = [
    "John Doe, Senior Geologist, Geology, johndoe@email.com, 123-456-7890",
    "Jane Smith, Drilling Engineer, Drilling, janesmith@email.com, 987-654-3210",
    "Bob Johnson, Reservoir Engineer, Reservoir Engineering, bobjohnson@email.com, 456-789-0123",
    "Alice Brown, Petrophysicist, Petrophysics, alicebrown@email.com, 789-012-3456"
]
```

```
In [84]: employees[0]
```

```
Out[84]: 'John Doe, Senior Geologist, Geology, johndoe@email.com, 123-456-7890'
```

```
In [90]: new_employee = "Eva Green, Drilling Technician, Drilling, evagreen@email.com, 111-222-3333"
employees.append(new_employee)
```

```
In [92]: age = 40
        if age>18:
            print("Eligible for voting")
        else :
            print("Not Eligible")
```

Eligible for voting

```
In [93]: current_fuel_level=input("enter current fuel level")
        print(current_fuel_level)
        print(type(current_fuel_level))
```

enter current fuel level1500
1500
<class 'str'>

```
In [ ] : low_fuel_thresold=1000
        critical_fuel_thresold=500
```

```
In [100]: if (int(current_fuel_level)<critical_fuel_thresold):
          print("Critical Fuel Level Reached. Take Immeditate Action")

          elif (int(current_fuel_level)<low_fuel_thresold):
              print("Send_fuel_low alter" )

          else:
              print("continue fueling")
```

continue fueling

```
In [103]: n=range(9)
```

```
In [104]: for i in n:
          print(i)
```

0
1
2
3
4
5
6
7
8

```
In [106]: for i in range(5,12):
          print(i)
```

5
6
7
8
9
10
11

```
In [107]: for i in range(5,51,5):
          print(i)
```

5
10
15
20
25
30
35
40
45
50

```
In [108]: l1=['a','b','c']
          l2=['Sales','IT','Finance']
          l=zip(l1,l2)
          print(l)

          <zip object at 0x7fa4d45c71c0>
```

```
In [109]: l=list(zip(l1,l2))
          print(l)

          [('a', 'Sales'), ('b', 'IT'), ('c', 'Finance')]
```

```
In [110]: oil_gas_data = [
          ("Field A", "Texas", 500000),
          ("Field B", "Alaska", 800000),
          ("Field C", "North Sea", 300000),
          ("Field D", "Gulf of Mexico", 600000),
          ]
```

```
In [111]: fields, locations, production = zip(*oil_gas_data)

print("Fields:", fields)
print("Locations:", locations)
print("Production:", production)

Fields: ('Field A', 'Field B', 'Field C', 'Field D')
Locations: ('Texas', 'Alaska', 'North Sea', 'Gulf of Mexico')
Production: (500000, 800000, 300000, 600000)
```

```
In [115]: e=list(enumerate(locations))
print(e)

[(0, 'Texas'), (1, 'Alaska'), (2, 'North Sea'), (3, 'Gulf of Mexico')]
```

```
In [116]: f=list(enumerate(fields))
print(f)

[(0, 'Field A'), (1, 'Field B'), (2, 'Field C'), (3, 'Field D')]
```

```
In [117]: for index, field_data in enumerate(oil_gas_data):
    fields, locations, production = field_data
    print(f"Field {index+1}: {fields} is located in {locations} with reserves of {production} barrels.")

Field 1: Field A is located in Texas with reserves of 500000 barrels.
```