

Data Types

Python Data Types

- Number
 - Int
 - Float
 - Complex
 - Boolean
- String
- List
- Tuple
- Dictionary
- set

Number

- Which includes all number data type

Int

Float

Boolean or Logical

Creating a Number

Python 3.6.4 Shell

File Edit Shell Debug Options Window Help

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

```
>>> a=1452
```

```
>>> type(a)
```

```
<class 'int'>
```

```
>>> b=(-45)
```

```
>>> type(b)
```

```
<class 'int'>
```

```
>>> c=0
```

```
>>> type(c)
```

```
<class 'int'>
```

```
>>> d=0.003
```

```
>>> type(d)
```

```
<class 'float'>
```

```
>>> e=78E0987
```

```
>>> type(e)
```

```
<class 'float'>
```

```
>>> x=complex(1,2)
```

```
>>> type(x)
```

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

```
>>> print(x)
```

```
(1+2j)
```

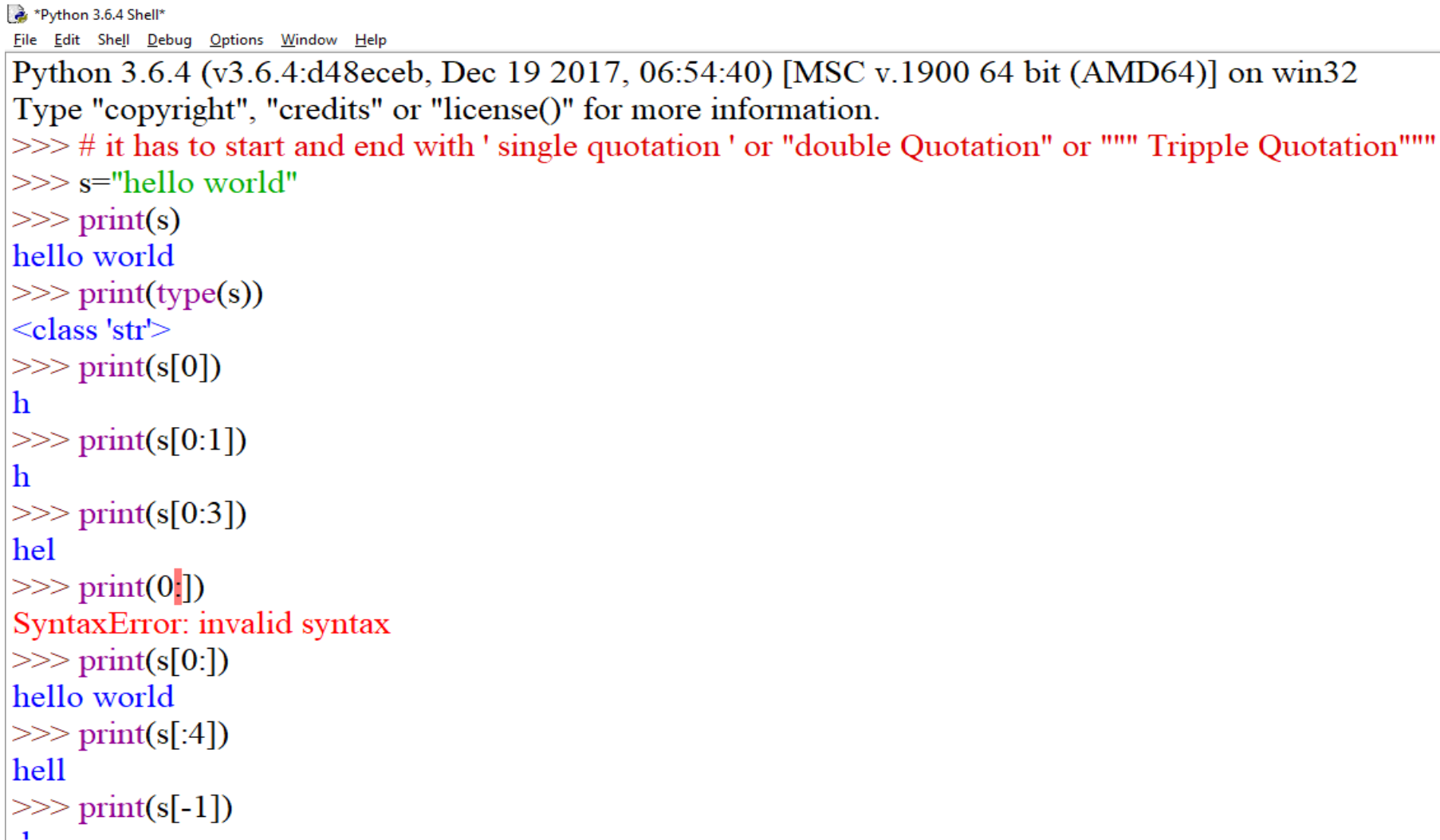
Complex and Boolean

```
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
>>> e=78E0987
>>> type(e)
<class 'float'>
>>> x=complex(1,2)
>>> type(x)
<class 'complex'>
>>> print(x)
(1+2j)
>>> y=4+7j
>>> type(y)
<class 'complex'>
>>> z=4+7J
>>> type(z)
<class 'complex'>
>>> f=True
>>> print(type(f))
<class 'bool'>
>>> g=False
>>> print(type(g))
<class 'bool'>
>>> |
```

String

- This is a group of characters
- it has to start and end with ' single quotation ' or "double Quotation" or "" Tripple Quotation""

Creating a string



```
*Python 3.6.4 Shell*
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> # it has to start and end with ' single quotation ' or "double Quotation" or "" Tripple Quotation""
>>> s="hello world"
>>> print(s)
hello world
>>> print(type(s))
<class 'str'>
>>> print(s[0])
h
>>> print(s[0:1])
h
>>> print(s[0:3])
hel
>>> print(0:1)
SyntaxError: invalid syntax
>>> print(s[0:])
hello world
>>> print(s[:4])
hell
>>> print(s[-1])
d
1
```

Indexing and Slicing

```
>>> # Index is []--> which we are defining which element you want to get
>>> # slicing is we are slice out the wanted elements from the sequence
>>> s="hello world"
>>> print(s[0]) # Python starts to count from 0
h
>>> print(s[0:1]) # This is Slicing and Indexing
h
>>> print(s[0:3]) # It will give upto the previous one here it will give upto 2
hel
>>> print(s[0:4]) # it will give upto 3
hell
>>> print(s[-7]) # when using '-' Symbol it will start to count from the reverse
o
>>>
```


List

- It may be a sequence of numbers, strings.
- This is dynamic , other words it is mutable
- Syntax: []

Creating a List

Slicing and Indexing

```
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
[2]
>>> print(type(list))
<class 'list'>
>>> # Creating a List
>>> list= [1,2,3,4,5]
>>> list1=["hai","hello","Python"]
>>> list2=["hai",1,"hello","Python"]
>>> print(list)
[1, 2, 3, 4, 5]
>>> print(list1)
['hai', 'hello', 'Python']
>>> print(list2)
['hai', 1, 'hello', 'Python']
>>> #Applying Slicing and Indexing in List
>>> print(list[0])
1
>>> print(list[1:2])
[2]
>>> print(type(list))
<class 'list'>
>>>
```

Activate Windows
Go to Settings to activate Windows.

Updating a value in List

```
>>> # creating a list sequenc of number and string
>>> list = ['physics', 'chemistry', 1997, 2000]
>>> print(list)
['physics', 'chemistry', 1997, 2000]
>>> print ("Value available at index 2 : ",list[2]) # get the index value of 2
Value available at index 2 : 1997
>>> list[2] = 2001 # To update the value of 2
>>> print ("New value available at index 2 : ",list[2])
New value available at index 2 : 2001
>>> print(list)
['physics', 'chemistry', 2001, 2000]
>>> |
```

Tuple

- We have two difference between list and tuple, that are:
 - Syntax of Tuple will be in this ()
 - Tuple is a static, other words is immutable

Creating a Tuple and Applying Slicing

```
>>> # Creating a tuple
>>> tup1 = ('physics', 'chemistry', 1997, 2000)
>>> tup2 = (1, 2, 3, 4, 5 )
>>> tup3 = ("a", "b", "c", "d")
>>> print(tup1)
('physics', 'chemistry', 1997, 2000)
>>> print(tup2)
(1, 2, 3, 4, 5)
>>> print(tup3)
('a', 'b', 'c', 'd')
>>> print(tup1[0])
physics
>>> print(tup1[2])
1997
>>> print(tup2[0:3])
(1, 2, 3)
>>>
```

Tuple immutable Example

```
>>> tup1 = (12, 34.56)
>>> tup2 = ('abc', 'xyz')
>>> # Following action is not valid for tuples
>>> tup1[0] = 100;
```

Traceback (most recent call last):

File "<pyshell#21>", line 1, in <module>

tup1[0] = 100;

TypeError: 'tuple' object does not support item assignment

```
>>> # So let's create a new tuple as follows
```

```
>>> tup3 = tup1 + tup2
```

```
>>> print(tup3)
```

```
(12, 34.56, 'abc', 'xyz')
```

```
>>> |
```

Dictionary

- Dictionary is act like a Hash Table
- Used to **store a values with the key element** which we can get it easily
- Use { } to construct a dict , [] is used to index a dict
- Dictionaries are mutable
- We can construct your dictionary by unique and by using, sting and number.

Creating a dictionary

```
>>> dict = {'Name': 'India QO', 'Age': 7, 'Category': 'Software'}
>>> # String value you have to given in ' / single or " double Quotation
>>> # Left side value is known as key and right side is value
>>> # to get difference between key and value element we are using :
```

```
>>> print(dict)
{'Name': 'India QO', 'Age': 7, 'Category': 'Software'}
```

```
>>> print(dict["Name"])
```

```
India QO
```

```
>>> dict['Age'] = 8 # To update Existing
```

```
>>> print(dict['Age'])
```

```
8
```

```
>>> print(dict)
```

```
{'Name': 'India QO', 'Age': 8, 'Category': 'Software'}
```

```
>>> |
```


Set

- Set will be in { }, it give the unique element in output
- Used to reduce the duplicate

Creating a set

```
>>> #creating a set
>>> set={1,1,1,2,2,2,3,3,3,4,4,4,5,5,5}
>>> print(set)
{1, 2, 3, 4, 5}
>>> my_set = {1.0, "Hello", (1, 2, 3)}
>>> print(my_set)
{1.0, 'Hello', (1, 2, 3)}
>>> x={1.0,3.4,0.004,"hai",1,1,3,4,6,2,3,5,"Python",45.09,0.8,0.877,"India QO"}
>>> print(x)
{0.8, 1.0, 2, 3.4, 3, 4, 6, 5, 0.877, 'India QO', 45.09, 'hai', 'Python', 0.004}
>>> |
```

Adding Element in set

```
>>> #Adding element in set
>>> my_set = {1, 2, 3}
>>> my_set.add(2)
>>> print(my_set)
{1, 2, 3}
>>> my_set.add(4)
>>> print(my_set)
{1, 2, 3, 4}
>>> my_set.update([2,3,4])
>>> print(my_set)
{1, 2, 3, 4}
>>> my_set.update([2,3,4,5])
>>> print(my_set)
{1, 2, 3, 4, 5}
>>> my_set.discard(4)
>>> print(my_set)
{1, 2, 3, 5}
>>> # discard to remove set
>>> my_set.remove(2) # remove
>>> print(my_set)
{1, 3, 5}
```

Great Job

Next Topic: Dir Functions