Data_Type

August 15, 2024

[3]:
```
import sys
import keyword
import operator
from datetime import datetime
import os
```

[ ]:
```

```

[ ]:
```

```

KEYWORDS

[6]:
```
print(keyword.kwlist) #List all Python Keywords
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break',
'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or',
'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

[8]:
```
len(keyword.kwlist) #Python contains 35 keywords
```

[8]: 35

Identifiers

[10]:
```
1var = 10 #Identifier can't start with a digit
```

```
  Cell In[10], line 1
    1var = 10
    ^
SyntaxError: invalid decimal literal
```

[12]:
```
va12@ = 35 #Indentifire can't use special symbols
```

```
  Cell In[12], line 1
    va12@ = 35 #Indentifire can't use special symbols
         ^
```

```
SyntaxError: invalid syntax
```

[14]:
```
import = 125 #Keywords can't be used as identifiers
```

```
  Cell In[14], line 1
    import = 125 #Keywords can't be used as identifiers
           ^
SyntaxError: invalid syntax
```

[23]:
```
"""
kjfjdn fnnjrn nfjdnjf enjmwj fnjdfdm fndm
fmdn d fndfdslsj jnejfndnnuehunss fnj fmnfm
hune nbjennnejhufenjefjnefjnejnf
"""
va12 = 10
```

[25]:
```
val_ = 99
```

Comments in Python

[29]:
```
#single line comment\
val1 = 10
```

[31]:
```
#Multiple
#line
#comment
val1 =10
```

[35]:
```
'''
MUMBAI IS THE
CAPITAL OF
India
'''
val1 = 10
```

[41]:
```
"""
Multiple
line
comment
"""
val1 = 10
```

Satements

[47]:
```
p = 20
q = 20
r = q
p, type(p), hex(id(p)) #Output: The same address in hexadecimal format
                       #(id(p)) # Output: A decimal number representing the␣
 ↪memory address
```

[47]: (20, int, '0x7ff99b3f3c18')

[49]:
```
r, type(r), hex(id(p))
```

[49]: (20, int, '0x7ff99b3f3c18')

[51]:
```
(id(r))
```

[51]: 140710028196888

[56]:
```
p = 20
p = p+10 #Variable Overwriting
p
```

[56]: 30

Variable Assigment

[60]:
```
intvar = 10 #Integer variable
floatvar = 2.57 #Float Variable
strvar = "Python Variable" #string variable
print(intvar)
print(floatvar)
print(strvar)
```

```
10
2.57
Python Variable
```

Multiple Assignments

[64]:
```
intvar, floatvar, strvar = 10, 2.57, "Python Variable" #Using commad to separat
print(intvar)
print(floatvar)
print(strvar)
```

```
10
2.57
Python Variable
```

[66]:
```
p1 = p2 = p3 = p4 = 44 #all variable pointing the same value
print(p1,p2,p3,p4)
```

```
44 44 44 44
```

DATA TYPES Numeric

[80]:
```
val1 = 10 # Integer data type
print(val1)
print(type(val1)) #Type of Object
print(sys.getsizeof(val1)) #Sie of integer object in bytes
print(val1, "is Integer?", isinstance(val1, int)) #val1 is an instance of int
```

```
10
<class 'int'>
28
10 is Integer? True
```

[82]:
```
val2 = 92.78 # Float data type
print(val2)
print(type(val2)) #Type of Object
print(sys.getsizeof(val2)) #Sie of float object in bytes
print(val2, "is float?", isinstance(val2, int)) #val2 is an instance of int
```

```
92.78
<class 'float'>
24
92.78 is Integer? False
```

[84]:
```
val3 = 25 + 10j # Complex data type
print(val3)
print(type(val3)) #Type of Object
print(sys.getsizeof(val3)) #Sie of integer object in bytes
print(val3, "is Complex?", isinstance(val3, int)) #val3 is an instance of int
```

```
(25+10j)
<class 'complex'>
32
(25+10j) is Complex? False
```

[86]:
```
sys.getsizeof(int()) #size of integer object in bytes
```

[86]: 28

[88]:
```
sys.getsizeof(float()) #size of float object in bytes
```

[88]: 24

[90]:
```
sys.getsizeof(complex()) #size of complex in bytes
```

[90]: 32

BOOlean

[96]:
```
bool1 = True
```

[98]:
```
bool2 = False
```

[100]:
```
print(type(bool1))
```

```
<class 'bool'>
```

[102]:
```
print(type(bool2))
```

```
<class 'bool'>
```

[104]:
```
isinstance(bool1, bool)
```

[104]: True

[106]:
```
bool(0)
```

[106]: False

[108]:
```
bool(1)
```

[108]: True

[110]:
```
bool(None)
```

[110]: False

[112]:
```
bool(False)
```

[112]: False

String

creation

[120]:
```
str1 = "Hello Python"
print(str1)
```

```
Hello Python
```

[122]:
```
mystr = 'Hello World' #Define string using single quotes
print(mystr)
```

```
Hello World
```

[ ]:
```
mystr = "Hello World" #Define string using double quotes
print(mystr)
```

[124]:
```
mystr = '''Hello
            World''' #Define string using triple quotes
print(mystr)
```

```
Hello
            World
```

[126]:
```
mystr = """Hello
            World""" #Define string using triple quotes
print(mystr)
```

```
Hello
            World
```

[136]:
```
mystr = ('Happy '
         'Monday '
        'Everyone')
print(mystr)
```

```
Happy Monday Everyone
```

[138]:
```
mystr2 = 'Woohoo '
mystr2 = mystr2*5
mystr2
```

[138]: 'Woohoo Woohoo Woohoo Woohoo Woohoo '

[140]:
```
len(mystr2) #Length of string
```

[140]: 35

String Indexing

[143]:
```
str1
```

[143]: 'Hello Python'

[145]:
```
str1[0] #First character in string "str1"
```

[145]: 'H'

[147]:
```
str1[len(str1)-1] # Last character in string using len function
```

[147]: 'n'

[149]:
```
str1[-1] # Last character in string
```

[149]: 'n'

[151]:
```
str1[6] #Fetch 7th element of the string
```

[151]: 'P'

[153]:
```
str1[5]
```

[153]:
```
' '
```

String Slicing

[158]:
```
str1[0:5] #String slicing - Fetch all characters from 0 to 5 index location
```

[158]:
```
'Hello'
```

[160]:
```
str1[6:12] # String slicing - Retreive all characters between 6 - 12 index loc e
```

[160]:
```
'Python'
```

[162]:
```
str1[-4:] # Retreive last four characters of the string
```

[162]:
```
'thon'
```

[170]:
```
str1[-6:] # Retreive last six characters of the string
```

[170]:
```
'Python'
```

[166]:
```
str1[:4] # Retreive first four characters of the string
```

[166]:
```
'Hell'
```

Update & Delete String

[172]:
```
str1
```

[172]:
```
'Hello Python'
```

[168]:
```
str1[:6] # Retreive first six characters of the string
```

[168]:
```
'Hello '
```

[174]:
```
#Strings are immutable which means elements of a string cannot be changed once t
str1[0:5] = 'HOLAA'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[174], line 2
      1 #Strings are immutable which means elements of a string cannot be␣
 ↪changed once t
----> 2 str1[0:5] = 'HOLAA'

TypeError: 'str' object does not support item assignment
```

[178]:
```
del str1 # Delete a string
print(srt1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[178], line 2
      1 del str1 # Delete a string
----> 2 print(srt1)

NameError: name 'srt1' is not defined
```

String Concatenation

[195]:
```
#String concatenation
s1 = "Hello"
s2 = "***"
s3 = "World"
s4 = s1 + s2 + s3
print(s4)
```

```
Hello***World
```

.lstrip() Return a copy of the string with leading whitespace removed.

[5]:
```
txt = "        abc efg sda    "
txt.lstrip()
```

[5]:
```
'abc efg sda    '
```

Using Escape Character Using double quotes in the string is not allowed.

[9]:
```
mystr = "My favourite TV Series is "Game of Thrones""
```

```
  Cell In[9], line 1
    mystr = "My favourite TV Series is "Game of Thrones""
                                        ^

SyntaxError: invalid syntax
```

You can use escape character to allow illegal characters

[18]:
```
abc = "My name is \"Mandeep\""
print(abc)
```

```
My name is "Mandeep"
```