basic-code-task1-2

August 20, 2024

1 Basic Code

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

1.1 Keywords:Keywords are the reserved words in Python and can't be used as an identifier

[7]:
```
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']
```

[9]:
```
len(keyword.kwlist) # Python contains 35 keywords
```

[9]: 35

1.2 Identifiers:An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another

[11]:
```
1var = 10 # Identifier can't start with a digit
```

```
  Cell In[11], line 1
    1var = 10 # Identifier can't start with a digit
    ^
SyntaxError: invalid decimal literal
```

[13]:
```
val2@ = 35 # Identifier can't use special symbol
```

```
  Cell In[13], line 1
    val2@ = 35 # Identifier can't use special symbol
         ^
SyntaxError: invalid syntax
```

[15]:
```
import = 125 # Keywords can't be used as identifiers
```

```
  Cell In[15], line 1
    import = 125 # Keywords can't be used as identifiers
           ^
SyntaxError: invalid syntax
```

Correct way of defining an identifier(Identifiers can be a combination of letters in lowercase (a to z) or uppercase

[19]:
```
val_ = 99
```

[ ]:
```
val2 = 10
```

1.3 Comments in Python:Comments can be used to explain the code for more readabilty.

[21]:
```
val1 = 10# Single line comment
```

[23]:
```
val1 = 10# Multiple  # line  # comment
```

[27]:
```
val1 = 10 '''Multiple  line   comment'''
```

```
  Cell In[27], line 1
    val1 = 10 '''Multiple  line   comment'''
              ^
SyntaxError: invalid syntax
```

[29]:
```
val1 = 10 """Multiple  line   comment"""
```

```
  Cell In[29], line 1
    val1 = 10 """Multiple  line   comment"""
              ^
SyntaxError: invalid syntax
```

1.4 Statements:Instructions that a Python interpreter can execute

[35]:
```
p = 20 #Creates an integer object with value 20 and assigns the variable p to p
q = 20 # Create new reference q which will point to value 20. p & q will be poi
r = q # variable r will also point to the same location where p & q are pointin
p , type(p), hex(id(p)) # Variable P is pointing to memory location '0x7fff6d71a
```

[35]:
```
(20, int, '0x7ffe2da83c18')
```

[39]:
```
q , type(q), hex(id(q))
```

[39]:
```
(20, int, '0x7ffe2da83c18')
```

[41]:
```
r , type(r), hex(id(r))
```

[41]:
```
(20, int, '0x7ffe2da83c18')
```

[43]:
```
p = 20
p = p + 10 # Variable Overwriting
p
```

[43]:
```
30
```

1.4.1 Variable Assigment

[47]:
```
intvar = 10 # Integer variable
floatvar = 2.57 # Float Variable
strvar = "Python Language" # String variable
print(intvar)
print(floatvar)
print(strvar)
```

```
10
2.57
Python Language
```

1.4.2 Multiple Assignments

[49]:
```
p1 = p2 = p3 = p4 = 44 # All variables pointing to same value
print(p1,p2,p3,p4)
```

```
44 44 44 44
```

2 Data Types

2.1 Numeric

[51]:
```
val1 = 10 # Integer data type
print(val1)
print(type(val1)) # type of object
print(sys.getsizeof(val1)) # size 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
```

[53]:
```
val2 = 92.78 # Float data type
print(val2)
print(type(val2)) # type of object
print(sys.getsizeof(val2)) # size of float object in bytes
print(val2, " is float?", isinstance(val2, float)) # Val2 is an instance of␣
 ↪float
```

```
92.78
<class 'float'>
24
92.78  is float? True
```

[55]:
```
val3 = 25 + 10j # Complex data type
print(val3)
print(type(val3)) # type of object
print(sys.getsizeof(val3)) # size of float object in bytes
print(val3, " is complex?", isinstance(val3, complex)) # val3 is an instance of␣
 ↪complex
```

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

[57]:
```
sys.getsizeof(int()) # size of integer object in bytes
```

[57]:
```
28
```

[59]:
```
sys.getsizeof(float()) # size of float object in bytes
```

[59]:
```
24
```

[61]:
```
sys.getsizeof(complex())# size of complex object in bytes
```

[61]:
```
32
```

2.2 Boolean:Boolean data type can have only two possible values true or false.

[63]:
```
bool1= True
```

[65]:
```
bool2= False
```

[67]:
```
print(type(bool1))
```

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

[69]:
```
print(type(bool2))
```

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

[71]:
```
isinstance(bool1, bool)
```

[71]:
```
True
```

[73]:
```
bool(0)
```

[73]:
```
False
```

[75]:
```
bool(1)
```

[75]:
```
True
```

[77]:
```
bool(None)
```

[77]:
```
False
```

[79]:
```
bool(False)
```

[79]:
```
False
```

2.3 Strings:String Creation

[81]:
```
str1 = "HELLO PYTHON"
print(str1)
```

```
HELLO PYTHON
```

[83]:
```
mystr = 'Hello World' # Define string using single quotes
print(mystr)
```

```
Hello World
```

[85]:
```
mystr = "Hello World" # Define string using double quotes
print(mystr)
```

```
Hello World
```

[87]:
```
mystr ="""Hello World""" # Define string using double quotes
print(mystr)
```

```
Hello World
```

[89]:
```
mystr = ('Happy '
'Monday '
'Everyone')
print(mystr)
```

```
Happy Monday Everyone
```

[91]:
```
mystr2 = 'Woohoo '
mystr2 = mystr2*5
mystr2
```

[91]: 'Woohoo Woohoo Woohoo Woohoo Woohoo '

[93]:
```
len(mystr2) # Length of string
```

[93]: 35

2.4 String Indexing

[95]:
```
str1
```

[95]: 'HELLO PYTHON'

[99]:
```
str1[0]# First character in string "str1"
```

[99]: 'H'

[101]:
```
str1[len(str1)-1] # Last character in string using len function
```

[101]: 'N'

[103]:
```
str1[-1] # Last character in string
```

[103]: 'N'

[105]:
```
str1[6] #Fetch 7th element of the string
```

[105]: 'P'

[107]:
```
str1[5]
```

[107]: ' '

2.5 String Slicing

[109]:
```
str1[0:5] # String slicing - Fetch all characters from 0 to 5 index location␣
↪excact
```

[109]: 'HELLO'

[111]:
```
str1[6:12] # String slicing - Retreive all characters between 6 - 12 index loc␣
↪excact
```

[111]: 'PYTHON'

[113]:
```
str1[-4:] # Retreive last four characters of the string
```

[113]: 'THON'

[115]:
```
str1[-6:] # Retreive last six characters of the string
```

[115]: 'PYTHON'

[117]:
```
str1[:4] # Retreive first four characters of the string
```

[117]: 'HELL'

[119]:
```
str1[:6] # Retreive first six characters of the string
```

[119]: 'HELLO '

2.6 Update & Delete String

[121]:
```
str1
```

[121]: 'HELLO PYTHON'

[123]:
```
str1[0:5] = 'HOLAA'#Strings are immutable which means elements of a string␣
↪cannot be changed once t
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[123], line 1
----> 1 str1[0:5] = 'HOLAA'
```

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

[125]:
```
del str1 # Delete a string
print(srt1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[125], line 2
      1 del str1 # Delete a string
----> 2 print(srt1)

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

2.7 String concatenation

[127]:
```
s1 = "Hello"
s2 = "Asif"
s3 = s1 + s2
print(s3)
```

```
HelloAsif
```

[ ]:
```

```