# **Python Concepts**

**Keywords**

Keywords are the reserved words in Python and can’t be used as an identifier.

In [3]: print(keyword.kwlist) *# List all Python Keywords*

In [4]: ['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'cl ass', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'fr om', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

Out[4]: len(keyword.kwlist) *# Python contains 35 keywords*

Out[4]: 35

**Identifiers**

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

In[13]: 1var = 10 *# Identifier can't start with a digit*

File "", line 1 1var = 10 # Identifier can't start with a digit ^ Syntax Error: invalid syntax

In[14]: val2@ = 35 *# Identifier can't use special symbols*

File "", line 1 val2@ = 35 # Identifier can't use special symbols ^ Syntax Error: invalid syntax

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

File "", line 1 import = 125 # Keywords can't be used as identifiers ^ Syntax Error: invalid syntax

In[16]: """ Correct way of defining an identifier (Identifiers can be a combination of letters in lowercase (a to z) or uppercase ( A to Z)"""

val2 = 10

In[17]: val\_ = 99

**Comments in Python**

Comments can be used to explain the code for more readability

In[18]: # Single line comment

val1 = 10

IN[19]: # Multiple

# line

# comment

val1 = 10

In[20]: ‘’’

Multiple

line

comment

‘’’

val1 = 10

In[21]: """

Multiple

Line

Comment

"""

val1 = 10

**Statements**

Instructions that a python interpreter can execute

In[26]: *# Multi line statement*

p1 = 20 + 30 \

+ 40 + 50 +\

+ 70 + 80

p1

Out[26]: 290

In[27]: *# Single line statement*

p1 = 10 + 20

p1

Out[27]: 30

In[28]: *# Single line statement*

p2 = ['a' , 'b' , 'c' , 'd']

p2

Out[28]: [‘a’, ‘b’, ‘c’, ‘d’]

In[29]: *# Multiple line statement*

p2 = ['a' , 'b' , 'c' , 'd' ]

p2

Out[29]: ['a', 'b', 'c', 'd']

**Indentation**

Indentation refers to the spaces at the beginning of a code line. It is very important as Python uses indentation to indicate a block of code. If the indentation is not correct, we will end up with the IndentationError error.

In[37]: p = 10

if p == 10:

print ('P is equal to 10') *# correct indentation*

**Docstrings**

* Docstring provide a convenient way of associating documentation with functions, classes, methods or modules
* They appear right after the definition of a function, method, class or module.

In[49]: def square(num):

‘’’Square Function :- This function will return the square of a number’’’

return num\*\*2

In[51]: square(2)

Out[51]: 4

In[52]: square.\_\_doc\_\_ *# We can access the Docstring using \_\_doc\_\_ method*

Out[52]: 'Square Function :- This function will return the square of a number'

In[53]: def evenodd(num):

‘’’evenodd Function: This function will test whether a number is even or odd’’’

If num % 2 == 0:

Print(‘Even Numbner’)

Else:

Print(‘Odd Number’)

In[54]: evenodd(3)

Odd Number

In[55]: evenodd(2)

Even Number

In[56]: evenodd.\_\_doc\_\_

Out[56]: 'evenodd Function :- This function will test whether a number is Even or Odd'

**Variables**

A python variable is a reserved memory location to store value. A variable is created the moment you first assign a value to it.

In[75]: p = 30

In[76]: ‘’’ id() function returns the “identity” of the object.

The identity of an object - Is an integer - Guaranteed to be unique - Constant for this object during its lifetime.’’’

id(p)

Out[76]: 140735029552432

In[77]: hex(id(p)) *# Memory address of the variable*

Out[78]: '0x7fff6d71a530'

**Variable Assignment**

In[100]: intvar = 10 *# Integer variable*

floatvar = 2.57 *# Float Variable*

strvar = "Python Language" *# String variable*

print(intvar) -> 10

print(floatvar) -> 2.57

print(strvar) -> Python Language

**Multiple Assignment**

In[102]: intvar , floatvar , strvar = 10, 2.57, "Python Language" *# Using commas to separate*

print(intvar) -> 10

print(floatvar) -> 2.57

print(strvar) -> Python Language

In[105]: p1 = p2 = p3 = p4 = 44 *# All variables pointing to same value*

print(p1,p2,p3,p4) -> 44 44 44 44

**Data Types**

**Numeric**

In[135]: 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 c*

10

<class int>

28

10 is Integer? True

In[136]: 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

In[137]: 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

In[119]: sys.getsizeof(int()) *# size of integer object in bytes*

Out[119]: 24

In[120]: sys.getsizeof(float()) *# size of float object in bytes*

Out[120]: 24

In[138]: sys.getsizeof(complex()) *# size of complex object in bytes*

Out[138]: 32

**Boolean**

Boolean data type can have only two data types true or false

In[139]: bool1 = True

In[140]: bool2 = False

IN[143]: print(type(bool1)) -> <class ‘bool’>

IN[144]: print(type(bool2)) -> <class ‘bool’>

In[148]: isinstance(bool1, bool) -> True

In[235]: bool(0) -> False

In[236]: bool(1) -> True

In[237]: bool(None) -> False

In[238]: bool(False) -> False

**Strings**

**String Creation**

In[193]: str1 = “HELLO PYTHON”

Prinst(str1)

HELLO PYTHON

In[194]: mystr = ‘Hello Wrld’ *# Define string using single quotes*

Print(mystr)

Hello World

In[195]: mystr = "Hello World" *# Define string using double quotes*

print(mystr)

Hello World

In[196]: mystr = ‘’’Hello

World’’’ *# Define string using triple quotes*

print(mystr)

Hello

World

In[197]: mystr = “””Hello

World””” *# Define string using triple quotes*

print(mystr)

Hello

World

In[198]: mystr = ('Happy '

'Monday '

'Everyone')

print(mystr)

Happy Monday Everyone

In[199]: mystr2 = 'Woohoo '

mystr2 = mystr2\*5

mystr2

Out[199]: ‘Woohoo Woohoo Woohoo Woohoo Woohoo’

In[200]: len(mystr2) *# Length of string*

Out[200]: 35

**String Indexing**

