1. Exploring basics of python like data types (strings, list, array, dictionaries, set, tuples) and control statements

(i) Numbers

a = 5

print(a, "is of type", type(a))

a = 2.0

print(a, "is of type", type(a))

a = 1+2j

print(a, "is complex number?", isinstance(1+2j,complex))

## (ii) Python List

a = [5,10,15,20,25,30,35,40]

# a[2] = 15

print("a[2] = ", a[2])

# a[0:3] = [5, 10, 15]

print("a[0:3] = ", a[0:3])

# a[5:] = [30, 35, 40]

print("a[5:] = ", a[5:])

## (iii) Python Tuple

t = (5,'program', 1+3j)

# t[1] = 'program'

print("t[1] = ", t[1])

# t[0:3] = (5, 'program', (1+3j))

print("t[0:3] = ", t[0:3])

# Generates error

# Tuples are immutable

t[0] = 10

## (iv) Python Strings

s = "This is a string"

print(s)

s = '''A multiline

string'''

print(s)

s = 'Hello world!'

# s[4] = 'o'

print("s[4] = ", s[4])

# s[6:11] = 'world'

print("s[6:11] = ", s[6:11])

# Generates error

# Strings are immutable in Python

s[5] ='d'

## (v) Python Set

a = {5,2,3,1,4}

# printing set variable

print("a = ", a)

# data type of variable a

print(type(a))

## (vi) Python Dictionary

d = {1:'value','key':2}

print(type(d))

print("d[1] = ", d[1]);

print("d['key'] = ", d['key']);

# Generates error

print("d[2] = ", d[2]);

## (vii) Conversion between data types

We can convert between different data types by using different type conversion functions like int(), float(), str(), etc.

>>> float(5)

5.0

Conversion from float to int will truncate the value (make it closer to zero).

>>> int(10.6)

10

>>> int(-10.6)

-10

Conversion to and from string must contain compatible values.

>>> float('2.5')

2.5

>>> str(25)

'25'

>>> int('1p')

Traceback (most recent call last):

File "<string>", line 301, in runcode

File "<interactive input>", line 1, in <module>

ValueError: invalid literal for int() with base 10: '1p'

We can even convert one sequence to another.

>>> set([1,2,3])

{1, 2, 3}

>>> tuple({5,6,7})

(5, 6, 7)

>>> list('hello')

['h', 'e', 'l', 'l', 'o']

To convert to dictionary, each element must be a pair:

>>> dict([[1,2],[3,4]])

{1: 2, 3: 4}

>>> dict([(3,26),(4,44)])

{3: 26, 4: 44}

**Conditional Statements in Python**

**Syntax**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | if condition1:      statements    elif condition2:      statements    else:      statements |

Consider the example below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | X = 10  Y = 12    if X < Y: print('X is less than Y') elif X > Y:      print('X is greater than Y')  else:      print('X and Y are equal') |

**Output:**X is less than Y

**While Loop**

Here, first the condition is checked and if it’s true, control will move inside the loop and execute the statements inside the loop until the condition becomes false. We use this loop when we are not sure how many times we need to execute a group of statements or you can say that when we are unsure about the number of iterations.

Consider the example:

**Syntax and Usage**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | count = 0  while (count < 10):     print ( count )     count = count + 1    print ("Good bye!") |

**Output** = 0

1

2

3

4

5

6

7

8

9

Good bye!

**For Loop**

Like the While loop, the For loop also allows a code block to be repeated certain number of times. The difference is, in For loop we know the amount of iterations required unlike While loop, where iterations depends on the condition. You will get a better idea about the difference between the two by looking at the syntax:

**Syntax**

|  |  |
| --- | --- |
| 1  2 | for variable in Sequence:      statements |

Notice here, we have specified the range, that means we know the number of times the code block will be executed.

Consider the example:

|  |  |
| --- | --- |
| 1  2  3  4 | fruits = ['Banana', 'Apple',  'Grapes']    for index in range(len(fruits)):     print (fruits[index]) |

**Output:** Banana Apple Grapes

**Nested Loops**

It basically means a loop inside a loop. It can be a For loop inside a While loop and vice-versa. Even a For loop can be inside a For loop or a While loop inside a While loop.

Consider the example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | count = 1  for i in range(10):      print (str(i) \* i)        for j in range(0, i):          count = count +1 |

**Output**

1

22

333

4444

55555

666666

7777777

88888888

999999999

2. Creating functions, classes and objects using python. Demonstrate exception handling and inheritance.

--Code for Class Object and Function

class Person:

"This is a person class"

age = 10

def greet(self):

print('Hello')

# create a new object of Person class

harry = Person()

# Output: <function Person.greet>

print(Person.greet)

# Output: <bound method Person.greet of <\_\_main\_\_.Person object>>

print(harry.greet)

# Calling object's greet() method

# Output: Hello

harry.greet()

Demonstrating exception handling

# import module sys to get the type of exception

import sys

randomList = ['a', 0, 2]

for entry in randomList:

try:

print("The entry is", entry)

r = 1/int(entry)

break

except:

print("Oops!", sys.exc\_info()[0], "occurred.")

print("Next entry.")

print()

print("The reciprocal of", entry, "is", r)

3.a. Python program to append data to existing file and then display the entire file

Code:

1. def file\_read(fname):
2. from itertools import islice
3. with open(fname, "w") as myfile:
4. myfile.write("Python Exercises\n")
5. myfile.write("Java Exercises")
6. txt = open(fname)
7. print(txt.read())
8. file\_read('abc.txt')

3.b. Python program to count number of lines, words and characters in a file

Code:

fname = input("Enter file name: ")

num\_words = 0

**with** open(fname, 'r') **as** f:

**for** line **in** f:

words = line.split()

num\_words += len(words)

**print**("Number of words:")

**print**(num\_words)

3.c. Python program to display file available in current directory

Code:

import os

path ="C:/python"

#we shall store all the file names in this list

filelist = []

for root, dirs, files in os.walk(path):

for file in files:

#append the file name to the list

filelist.append(os.path.join(root,file))

#print all the file names

for name in filelist:

print(name)

Example : Get the list of all files with a specific extension

In this example, we will take a path of a directory and try to list all the files, with a specific extension **.py** here, in the directory and its sub-directories recursively.

Code:

import os

path ="C:\ambika"

for root, dirs, files in os.walk(path):

for file in files:

if(file.endswith(".py")):

print(os.path.join(root,file))