Python Programming

Dr Binu P Chacko

Introduction

- Python (1989) is a high-level, interpreted, general-purpose, dynamic programming language; free and open-source software
- Programming style: object-oriented, functional, procedural, imperative
- Features: (reusable) code is automatically compiled to byte code and executed, scripting language, extend Python with C/C++, built-in data types
- Supports nested code blocks, classes, modules, packages, control statements
- Start Python: Run a script, Use GUI from IDE, Interactive approach
- `>>> 10+5                                  # 15`
- `>>> print(“Welcome to Python”)            # Welcome to Python`
- `>>> ‘Python programming’                  # ‘Python programming’`
- Identifier (case sensitive): *alphabet*, *underscore*, digits

Data Types

Numeric

- >>> n=10
- >>> n # 10
- >>> n=5.8

String

- >>> s='with'
- >>> len(s) # 4
- >>> s.upper() # 'WITH'
- >>> s+'in' # 'within'
- >>> s*2 # 'withwith'
- >>> s[1] # 'i'
- >>> s[0:3] # 'wit' >>> s[:3]
- >>> s[0:3:2] # 'wt'
- >>> s[2:] # 'th' slice

List

- >>> l = [1, 5.3,'cat']
- >>> l # [1, 5.3, 'cat']

String methods
.lower()
.isalpha()
.isdigit()
.isspace()
.find("string")
.replace("old","new")
.count("character")

Cont…

Tuple

```
>>> t=(11, “area”, 4.3)          # read only
>>> t                            # (11, ‘area’, 4.3)
```

Dictionary

```
>>> d={1:”one”, “two”:2}
>>> d                            # {1:’one’,’two’:2}
>>> d.keys()                     # [1, ‘two’]
>>> d.values()                   # [‘one’,2]
```

Boolean

```
>>> b=True
>>> type(b)                              # bool
```

Sets

```
>>> s1=set([1,2,3,2,8])                  # No duplicate values
>>> s2=set([1,3,5,9])
>>> u=s1|s2
>>> I = s1&s2
>>> df = s1 –s2
>>> sym_df = s1 ^ s2
```

Operators

- Arithmetic operators: +,-,*,/,%,** (power),// (no fractional part)
- Relational operators: ==,!=,<,>,<=,>=
- Assignment operators: =,+=,-=,*=,/=,%=,**=, //=
- Bitwise operators: &,|,^ (XOR), ~(complement), <<,>> (right shift)
- Logical operators: and, or, not
- Membership operators
- >>> 1 *in* l   # True
- >>> 6 *not in* l   # True
- Identity operators
- >>> m=5.8
- >>> n *is* m   # True
- >>> n *is not* m   # False

Precedence of operators
not, or, and
in, not in
is, is not
=,+=,-=,*=,/=,%=,//=,**=
!=, ==
^, |
&
<<, >>
+, -
*, /, %, //
** (RL associativity)

Control Statements

-
  ```
  >>> if (n>0):
          print('Positive')          # Positive
  ```

-
  ```
  >>> if (n>0):
          print('Positive')
        else:
          print('Not positive')
  ```

-
  ```
  >>> if (n>0):
          print('Positive')
         elif (n==0):
          pass
         else:
          print('Negative')
  ```

Cont…

-
  ```
  >>> for x in l:
              print(x)
  ```

-
  ```
  >>> for x in t:
              print(x)
      else:
              print('End')
  ```

-
  ```
  >>> for x in range(10,20,2):
              print(x)
  ```
- `>>> n = int(input("enter a number: '))`
- float(), str()

```
>>> while (n > 0):
            print('Count ; ', n)
            n-=1
```

```
>>> while (n>0):
        n-=1
        if (n%2):
            continue
        print(n)
    else:
        print('End')
```

```
>>> while (n>0):
        if (n<5):
            break
        print(n)
        n-=1
```

Program

- Square root of a number
- Swap values of two variables
- Largest of three numbers
- Check for leap year
- Fibonacci sequence for n terms
- Prime numbers within a range
- Find HCF of given numbers # Function
- Display factors of a number
- Find ASCII value of a character # ord()
- Convert decimal number to binary, octal, hexadecimal
- Recursive function to find the sum of first N natural numbers

Function

-
  ```
  >>> def display(name, age):
          print(name, age)
          return
  ```
- `>>> display(‘Raj’, 25)`
- `>>> display(age=20,name=‘Smitha’)`

-
  ```
  >>> def fun1(name, grade=‘A’):                # Default argument
          print(name, grade)
          return
  ```
- `>>> fun1(‘Leo’)`
- `>>> fun1(‘Reema’, ‘A+’)`

-
  ```
  >>> def varpar(*arg):                          # Variable-length arguments
          print(arg)
          return
  ```
- `>>> varpar()`
- `>>> varpar(10)`
- `>>> varpar(5,’Rat’)`

Built-in Functions

Mathematical functions

- `>>> import math` # import math module
- `>>> math.log10(15)`
- `>>> math.log(22.4)`
- `>>> help(math.log)`
- `>>> math.sin(5)`
- `>>> degree=90`
- `>>> angle = degree * 2*math.pi/360`
- `>>> math.sin(angle)`
- `>>> dir(math)`

```
>>> math.floor(n)
>>> math.ceil(n)
>>> round(n)
>>> abs(n)
>>> math.sqrt(n)
```

Date & Time functions

- `>>>import time`
- `>>> time.localtime(time.time())`
- `>>> time.asctime(time.localtime(time.time()))`
- `>>> import calendar`
- `>>> c = calendar.month(2025,6)`
- `>>> print(c)`

```
>>> import datetime
>>> today=datetime.date.today()
>>> today.day
>>> today.month
>>> today.year
>>> today.ctime()
>>> today.timetuple()
>>> today.toordinal()
```

Recursive Function

-
  ```
  >>> def fact(n):
          if (n==1):
              return(1)
          else:
              return(n*fact(n-1))
  ```
- `>>> fact(5)`

Anonymous Function

- `>>> prod = lambda a, b : a * b`
- `>>> prod(25, 5)`

Python Script

- Module: Python code in a file (.py)

Strings

- Compound data type (made up of smaller pieces)
- Immutable - can't change any element of a string
- Traversal – access each element of the string using loop statement
- >>> s='Python'
- >>> s[-1]   # 'n'
- >>> s[:]   # 'Python'
- >>> s[3:2]   # ''
- >>> s[: : -1]   # 'nohtyP'
- Escape character: \\, \', \", \a, \b, \f, \n, \r, \t, \v, \ooo, \xhhh
- >>> print('I can\'t talk')
- Format symbol: %c, %s, %i, %d, %u, %o, %x, %X, %e, %E, %f, %g, %G
- print("%s contains %d characters" %('Python', 6))

Formatting Functions

- String.isalnum(), isalpha(), isdigit(), islower(), isupper(), isnumeric(), isdecimal(), isspace(), istitle(), lower(), upper(), title(), capitalize(), center(width, fillchar), ljust(width, fillchar), rjust(width, fillchar),count(str, beg, end), endswith(str, beg, end), find(str, beg, end), rfind(str, beg, end), index(str, beg, end), rindex(str, beg, end), replace(old, new[, max]), join(str), lstrip(), rstrip(), strip(), split(str, max), splitlines(max), swapcase(), zfill(width)
- max(str), min(str)

Lists

- `>>> a1=[]` # empty list
- `>>> a2=[10, 3.9, ‘Cards’]`
- `>>> a3=a2` # shared the list
- `>>> a2.append(35)`
- `>>> a3=a2[ : ]` # two different lists
- `>>> from copy import copy`
- `>>> a3=copy(a2)`
- `>>> a3[2]=300` # mutable
- `>>> for i in range(3):` # traversing

  ```
      a3[i]=a3[i]+10
  ```
- `>>> x=a2.pop(2)` # Delete Nth element
- `>>> del a2[1]`
- `>>> a2.remove(10)` # Delete the item

Cont…

- >>> a4=a2+a3 #concatenation
- >>> a2.extend(a3)
- >>> a3 * 2 # repetition
- >>> ‘Chess’ in a3 # False
- >>> a2.sort()
- >>> a2.count(10) # 1
- >>> a2.index(3.9) # index of the item
- >>> a2.insert(index, item)
- >>> a2.reverse()
- max(a2), min(a2), len(a2)

? Find duplicate characters in a string

? Remove punctuations in a string

? Search a character in the string

Tuple

- Can’t delete or modify the elements in the tuple
- >>> nst=(1, [2, ‘are’], 3.9)   # nested tuple
- >>> a=(“apple”,)
- >>> print(nst[0])
- >>> nst[: 2]
- >>> std = (2, ‘Smitha’, 19)
- >>> (no, name, age)=std
- >>>print(no, name, age)
- >>> x, y=2, 3
- >>> y, x=x, y

Cont…

-
  ```
  >>> def division(d):                    # accept tuple
              q=d[1]//d[0]
              r=d[1]%d[0]
              return(q, r)                # return tuple
  ```
- `>>> n=(4,44)`
- `>>> division(n)`
- Operators: +, *, in
- Functions: len(t), max(t), min(t), tuple(L)
- Print the values of tuples in two lines
- Print even numbers in the tuple
- Print a tuple containing cubes of numbers from 1 to 15
- Find the sum of N numbers using tuple and function

Dictionary

- Keys are immutable
- Key will be updated if duplicate key is used
- >>> d1={}
- >>> d2=dict({1:’red’, 2:’blue’, 3:’black’})
- >>> d3=dict([(1,’red’), (2,’blue’), (3,’black’)])
- >>> d4=dict(one=1, two=2, three=3)
- >>> d4[‘one’]   # 1
- >>> d4.get(‘two’)   # 2 OR d4.setdefault(‘two’)
- >>> d4[‘three’]=30   # updated
- >>> d4[‘four’]=4   # pair added
- >>> d4.pop(‘three’)
- >>> d4.popitem()   # delete arbitrary item
- >>> del d4[‘two’]
- >>> d4.clear()   # delete all items
- >>> del d3   # delete the dictionary

Cont…

- `>>> for d in d2:` # traversing
  `print(d, d2[d])`
- `>>> 1 in d2` # True – Membership
- `>>> 5 not in d2` # True
- `>>> all(d2)` # True
- `>>> any(d2)`
- `>>> sorted(d2)`
- `>>> str(d2)`
- `>>> d3=d2.copy()`
- `>>> d3.items()` # display all elements
- `>>> d1.update(d2)` # add items from d2 to d1
- `>>> len(d1)`
- Program to print square of numbers up to N
- Program to update dictionary key-value pair
- Find the sum of all items in a dictionary
- Convert key-value list to dictionary

Programs

- Multiplication table

```
n=int(input('Enter a number : '))
for i in range(1,11):
   print(n*i, end=" ")
```

- Cat/Mouse count

```
def cat_mouse(str):
   catc=str.count('cat')
   mousec=str.count('mouse')
   return(catc==mousec)
```

- Print characters at even index

```
str=input('Enter a string : ')
print(str[0::2])
```

- Squares up to N

```
n=int(input('Enter a number : '))
i=1
while(i*i<=n):
   print(i**2,end=" ")
   i+=1
```

- Print up to 0

```
n=int(input('Enter a number : '))
if(n>0):
   for i in range(n,-1,-1):
      print(i,end=" ")
else:
   for i in range(n,1,1):
      print(i,end=" ")
```

Programs

- Nth Fibonacii number

```
def fib(n):
    if(n==1):
        return(0)
    elif(n==2):
        return(1)
    else:
        return(fib(n-1)+fib(n-2))
```

- Split array add to end

```
def arrsplit(arr,n,k):
    b=arr[:k]
    return(arr[k:]+b)
----------------------------------------
ar=[1,3,2,5,6,7,34]
pos=int(input('Split position : '))
l=len(ar)
print(arrsplit(ar,l,pos))
```

- Matrix Multiplication

```
A=[[1,2,3],[4,5,6],[7,8,9]]
B=[[2,5,8,9],[12,8,3,20],[11,4,1,7]]
R=[[0,0,0,0],[0,0,0,0],[0,0,0,0]]
for i in range(len(A)):
    for j in range(len(B[0])):
        for k in range(len(B)):
            R[i][j]+=A[i][k]*B[k][j]
print(R)
```

- Reverse the words in the string

```
str=‘Computer Science’
print(‘ ‘.join(str.split()[::-1]))
```

- Replace a character

```
str.replace(‘p’,’b’)
```

- Sum of a list

```
a=[1,2,3,6,7,3]
print(sum(a))
```

Programs

- Second largest in the list

```
a.sort(reverse=True)
print(a[1])
```

- N largest elements

```
n=int(input('Enter a number'))
a.sort()
print(a[-n:])
```

- Remove multiple elements

```
r=[3,7]
for ele in r:
    if ele in a:
            a.remove(ele)
print(a)
```

•Remove empty list

```
a=[[2,6],[],[12,0,-4]]
n=[]
for ele in a:
    if ele:
        n.append(ele)
print(n)
```

•Find cumulative sum of a list

```
total=0
csum=[]
a [2,6,3,15,8]
for i in a:
    total+=I
    csum.append(total)
print(csum)
```

Programs

- Split list in chunks of size N

```
a = [1, 2, 3, 4, 5, 6, 7, 8]
n = 3
c = []
for i in range(0, len(a), n):
    c.append(a[i:i + n])
print(c)
```

- Check a substring in a string

```
str = “Python programs are easy"
if “are" in str:
     print("Yes! it is present in the string")
 else:
     print("No! it is not present")
```

- Check whether the string contains all vowels

```
s = “Welcome to programming world"
v = 'aeiou‘
if all(i in s.lower() for i in v):
   print("True")
else:
   print("False")
```

- Number of matching characters in a pair

```
s1 = “Electronics"
s2 = “Electric“
m= len(set(s1.lower()).intersection(set(s2.lower())))
print(m)
```

- Remove ith character from the string

```
i = 6
r = s[:i] + s[i+1:]
print(r)
```

Programs

- Check whether string contains special characters

```
n = “non-uniform"
n.split()
c = 0
s = '[@_!#$%^&*()<>?/\|}{~:]’
for i in range(len(n)):
   if n[i] in s:
      c += 1
if c:
   print(“special characters")
else:
   print(“no special characters")
```

- Check for binary string

```
s = "101010000111“
if all(c in '01' for c in s):
   print("Yes")
else:
   print("No")
```

- Uncommon words

```
s1 = “English"
s2 = “English drama“
set1 = set(s1.split())
set2 = set(s2.split())
 # symmetric difference
d= list(set1 ^ set2)
print(d)
```

- Replace multiple words

```
s = “best CS text book“
li = ["best", "CS”]
k = “good"
for word in li:
   s = s.replace(word, k)
print(s)
```

Programs

- Execute a string of code

```
code = "x = 5\ny = 10\nprint(x + y)"
exec(code)
```

- Rotate a string

```
s = “Prose & Drama"
d = 2
left = s[d:] + s[:d]
right = s[-d:] + s[:-d]
print("Left Rotation:", left)
print("Right Rotation:", right)
```

- Find duplicate characters

```
s = “Java Programming"
res = []
for c in set(s):
    if s.count(c) > 1:
        res.append(c)
print(res)
```

•Sum of all items in a dictionary

```
d = {'a': 100, 'b': 200, 'c': 300}
res = sum(d.values())
print(res)
```

•Sort dictionary by value

```
list = [{"name": "Nandini", "age": 20},
       {"name": “Renjith", "age": 20},
       {"name": "Nikhil", "age": 19}]
print("The list printed sorting by age: ")
print(sorted(list, key=lambda i: i['age']))
print("\r")
print("The list printed sorting by age and name: ")
print(sorted(list, key=lambda i: (i['age'], i['name'])))
print("\r")
print("The list printed sorting by age in descending order: ")
print(sorted(list, key=lambda i: i['age'], reverse=True))
```

Programs

- Sort dictionary key and values list

```
test_dict = {'gfg': [7, 6, 3],
             'is': [2, 10, 3],
             'best': [19, 4]}
res = dict()
for key in sorted(test_dict):
    res[key] = sorted(test_dict[key])
print(“The sorted dictionary : “ + str(res))
```

- Handling missing values in dictionary

```
ele = {'a': 5, 'c': 8, 'e': 2}
if “d" in ele:
    print(ele["d"])
else:
    print("Key not found")
```

Dictionary with keys having multiple inputs

```
dict = {}
x, y, z = 10, 20, 30
dict[x, y, z] = x + y - z;
x, y, z = 5, 2, 4
dict[x, y, z] = x + y - z;
print(dict)
```

Remove duplicate words from the string

```
s1 = “the book is on the table"
s2 = s1.split()
s3 = list(set(s2))
s4 = ' '.join(s3)
print(s4)
```

Count the frequencies in a list

```
a = ['apple', 'banana', 'apple', 'orange', 'banana']
b = {}
for c in a:
    if c in b:
        b[c] += 1
    else:
        b[c] = 1
print(b)
```

Programs

- Keys associated with values

```
test_dict = {'grapes' : [1, 2, 3], ‘this' : [1, 4], 'east' :
[4, 2]}
result_dict = {}
for key, val in test_dict.items():
    for ele in val:
        if ele in result_dict:
            result_dict[ele].append(key)
        else:
            result_dict[ele] = [key]
print("The values associated dictionary : "
+ str(result_dict))
```

Maximum and minimum K elements in the tuple

```
test_tup = (5, 20, 3, 7, 6, 8)
K = 2
test_tup = list(test_tup)
temp = sorted(test_tup)
res = tuple(temp[:K] + temp[-K:])
print("The extracted values : " + str(res))
```

Create a list of tuples with numbers and their cubes

```
a = [1, 2, 3, 4, 5]
res = [(n, n**3) for n in a]
print(res)
```

Add tuple to list and vice-versa

```
a = [1, 2, 3]
b = (4, 5)
c = [6, 7]
a.extend(b)
print(a)
d = b + tuple(c)
print(d)
```

Remove tuples of length K

```
test_list = [(4, 5), (4, ), (8, 6, 7), (1, ), (3, 4, 6, 7)]
K = 1
res = [ele for ele in test_list if len(ele) != K]
print("Filtered list : " + str(res))
```

Programs

- Sort a list of tuples by second item

```
a = [(5, 2), (1, 6), (3, 4)]
a.sort(key=lambda x: x[1])
print(a)
```

- Order tuples by list

```
test_list = [('Gas', 3), ('best', 9), ('CS', 10),
('Gear', 2)]
ord_list = ['Gear', 'best', 'CS', 'Gas']
temp = dict(test_list)
res = [(key, temp[key]) for key in ord_list]
print(“The ordered tuple list : “ + str(res))
```

- Flatten tuple of list to tuple

```
tup = ([5, 6], [6, 7, 8, 9], [3])
res = tuple(x for sublist in tup for x in sublist)
print(res)
```

Convert nested tuple to custom key dictionary

```
a = ((4, 'Gas', 10), (3, 'is', 8), (6, 'Best', 10))
res = [{'key': sub[0], 'value': sub[1], 'id':
sub[2]} for sub in a]
print(str(res))
```

Get current time

```
from datetime import datetime
now = datetime.now()
currentTime = now.strftime("%H:%M:%S")
print("Current Time =", currentTime)
```

Current date-time

```
import datetime
current_time = datetime.datetime.now()
print("Time now at greenwich meridian
is:", current_time)
```

Programs

- Yesterday, Today, Tomorrow

```
from datetime import datetime, timedelta
presentday = datetime.now()
# or presentday = datetime.today()
yesterday = presentday - timedelta(1)
tomorrow = presentday + timedelta(1)
print("Yesterday = ", yesterday.strftime('%d-%m-%Y'))
print("Today = ", presentday.strftime('%d-%m-%Y'))
print("Tomorrow = ", tomorrow.strftime('%d-%m-%Y'))
```

- 12 hour time to 24 hour time

```
def convert24(str1):
    if str1[-2:] == "AM" and str1[:2] == "12":
        return "00" + str1[2:-2]
    elif str1[-2:] == "AM":
        return str1[:-2]
    elif str1[-2:] == "PM" and str1[:2] == "12":
        return str1[:-2]
    else:
        return str(int(str1[:2]) + 12) + str1[2:8]
print(convert24("08:05:45 PM"))
```

Time difference

```
def difference(h1, m1, h2, m2):
    t1 = h1 * 60 + m1
    t2 = h2 * 60 + m2
    if (t1 == t2):
        print("Both are same times")
        return
    else:
        diff = t2-t1
    h = (int(diff / 60)) % 24
    m = diff % 60
    print(h, ":", m)
difference(7, 20, 9, 45)
```

Prime factors of a number

```
import math
def primeFactors(n):
    while n % 2 == 0:
        print(2)
        n = n // 2
    for i in range(3,int(math.sqrt(n))+1,2):
        while n % i== 0:
            print(i)
            n = n // i
    if n > 2:
        print(n)
primeFactors(315)
```