அலகு III

பாடம் 9

(List, Tuples, Set மற்றும் Dictionary) தொகுப்பு தரவினங்கள்

ETJX83

கற்றலின் நோக்கங்கள்

இந்த பாடத்தை கற்ற பிறகு மாணவர்கள் அறிந்து கொள்வது,

- List, Tuples, Set மற்றும் Dictionary போன்ற பல வகையான தொகுப்பு தரவினங்களின் அடிப்படை கருத்துருக்களை புரிந்து கொள்ளுதல்.
- பல்வேறு செயற்கூறுகளைப் பயன்படுத்தி List, Tuples, Set மற்றும் Dictionary - ல் வேலை செய்தல்.
- List, Tuples, Set மற்றும் Dictionary யைப் பயன்படுத்தி பைத்தான் நிரல்களை எழுதுதல்.
- List, Tuples, Set மற்றும் Dictionary - களுக்கு இடையேயான உறவுகளை புரிந்து கொள்ளுதல்.

9.1 List ஓர் அறிமுகம்

பைத்தான் நிரலாக்க மொழியில் நான்கு வகையான தொகுப்பு தரவினங்கள் உள்ளன. அவை List, Tuples, Set மற்றும் Dictionary ஆகும். பைத்தானில் உள்ள List சரத்தைப் போன்றே “வரிசைமுறை தரவினம்” ஆகும். இது சதுர அடைப்புக் குறிக்குள்[ ] அடைக்கப்பட்ட மதிப்புகளின் வரிசைப்படுத்தப்பட்ட தொகுப்பாகும். List-ல் உள்ள ஒவ்வொரு மதிப்பும் உறுப்பு (element) என்றழைக்கப்படுகிறது. இது எண்கள், எழுத்துகள், சரநிலையுருக்கள் மற்றும் பின்னலான List போன்ற எந்த வகையாகவும் இருக்கலாம். உறுப்புகள் மாறும் தன்மையுடையன. அதாவது உறுப்புகளை மாற்றியமைக்கவும், சேர்க்கவும் அல்லது நீக்கவும் முடியும். ஒவ்வொரு உறுப்பும் List-ல் ஒரு குறிப்பிட்ட இடத்தில் அமைந்திருக்கும். ஒரு உறுப்பின் இருப்பிட குறியீட்டென் அல்லது சுட்டெண் சுழியம் என்ற தொடக்க எண்ணால் குறிப்பிடப்படுகிறது. இந்த சுட்டெண் குறிப்பிட்ட உறுப்பை கண்டறிவதற்கும், அதனை கையாளுவதற்கும் பயன்படுகிறது. இதனால் List என்பது பதினொன்றாம் வகுப்பில் நீங்கள் பயின்ற அணிகளை ஒத்தது என்பதை நினைவில் கொள்க.

9.1.1 பைத்தானில் List–ஐ உருவாக்குதல்:

பைத்தானில், List சதுர அடைப்புக்குறியைப் பயன்படுத்தி எளிமையாக உருவாக்கப்படுகின்றன. List ன் உறுப்புகளை கட்டயமாக சதுர அடைப்புக்குறிக்குள் குறிப்பிட வேண்டும். பின்வரும் தொடரியல் List உருவாக்கும் விதத்தை விளக்குகிறது.

தொடரியல்:

Variable = [element-1, element-2, element-3 …… element-n]

எடுத்துக்காட்டு:

Marks = [10, 23, 41, 75]
Fruits = [“Apple”, “Orange”, “Mango”, “Banana”]
MyList = [ ]

மேலே கொடுக்கப்பட்டுள்ள எடுத்துக்காட்டில், Marks என்ற List நான்கு முழு எண் (integer) உறுப்புகளையும், இரண்டாவது List, Fruits-ல் நான்கு சரநிலையுரு உறுப்புகளையும் கொண்டுள்ளது. மூன்றாவது List ஒரு வெற்று List ஆகும். List-ன் உறுப்புகள் ஒரே தரவினமாக இருக்க வேண்டிய அவசியமில்லை. பின்வரும் List பல வகை உறுப்புகளைக் கொண்டுள்ளது.

MyList = [“Welcome”, 3.14, 10, [2, 4, 6]]

மேலே உள்ள எடுத்துக்காட்டில் , **My**List என்ற List மற்றொரு List-டை உறுப்பாகக் கொண்டுள்ளது. இந்த வகை List “பின்னலான List (**Nested** List)” என்று அழைக்கப்படுகிறது.

பின்னலான List என்பது மற்றொரு List-ஐ ஒரு உறுப்பாக கொண்டுள்ள List ஆகும்.

9.1.2 List உறுப்புகளை அணுகுதல்

பைத்தான், List-ன் ஒவ்வொரு உறுப்புக்கும் சுழியத்திலிருந்து தொடங்குகின்ற தானமைவு சுட்டெண் மதிப்பை இருத்துகிறது. சுட்டெண் மதிப்பு, பட்டியலின் ஒரு உறுப்பை அணுகுவதற்கு பயன்படுகிறது. பைத்தானில், சுட்டெண் மதிப்பு நேர்மறை அல்லது எதிர்மறை முழு எண் மதிப்பாக இருக்கலாம்.

எடுத்துக்காட்டு:

Marks = [10, 23, 41, 75]

Marks	10	23	41	75
நேர்மறை சுட்டெண் (Positive)	0	1	2	3
எதிர்மறை சுட்டெண் (Negative)	-4	-3	-2	-1

நேர்மறை சுட்டெண் மதிப்பு List-ன் தொடக்கத்திலிருந்தும் எதிர்மறை சுட்டெண் மதிப்பு இறுதியிலிருந்து பின் வரிசையில் (அதாவது பின்னோக்கு வரிசை) உறுப்புகளை கணக்கிடுகிறது.

List-லிருந்து ஒரு உறுப்பை அணுகுவதற்கு, சதுர அடைப்புக் குறிக்குள் கொடுக்கப்பட்ட உறுப்பின் சுட்டெண்ணை பின்னொட்டாகக் கொண்ட List-ன் பெயரை எழுதவும்.

தொடரியல்

List_Variable = [E1, E2, E3 …… En]

print (List_Variable[index of a element])

எடுத்துக்காட்டு (ஒற்றை உறுப்பை அணுகுதல்):

```
>>> Marks = [10, 23, 41, 75]
>>> print (Marks[0])
10
```

மேலே உள்ள எடுத்துக்காட்டில், சுழியம் என்ற சுட்டெண்ணின் மதிப்பு 10 ஆக உள்ளதால், print கட்டளை 10 என்ற வெளியீட்டை அச்சிடுகிறது.

எடுத்துக்காட்டு (உறுப்புகளை பின்னோக்கு வரிசையில் அணுகுதல்):

```
>>> Marks = [10, 23, 41, 75]
>>> print (Marks[-1])
  75
```

குறிப்பு

எதிர்மறை சுட்டெண், உறுப்புகளை பின்னோக்கு வரிசையில் அணுகுவதற்கு பயன்படுகிறது.

9.1.2.1 List ன் அனைத்து உறுப்புகளையும் அணுகுதல்

List-ன் அனைத்து உறுப்புகளையும் அணுகுவதற்கு மடக்குகள் பயன்படுகின்றன. மடக்கின் தொடக்க மதிப்பு சுழியமாக இருக்க வேண்டும். List-ன் தொடக்க சுட்டெண் மதிப்பு சுழியமாகும்.

எடுத்துக்காட்டு

```
Marks = [10, 23, 41, 75]
i = 0
while i < 4:
        print (Marks[i])
        i = i + 1
```

வெளியீடு

```
        10
        23
        41
        75
```

மேலே உள்ள எடுத்துக்காட்டில் marks என்ற List 10,23,41,75 என்ற நான்கு முழு எண் உறுப்புகளைக் கொண்டுள்ளது. ஒவ்வொரு உறுப்புகளும், சுழியத்திலிருந்து தொடங்குகின்ற சுட்டெண் மதிப்பைப் பெற்றுள்ளது. உறுப்புகளின் சுட்டெண் மதிப்பு முறையே 0,1,2,3 என்பதாகும். இங்கு while மடக்கு List-ன் அனைத்து உறுப்புகளையும் அச்சிடுவதற்கு பயன்படுகிறது. மடக்கின் தொடக்க மதிப்பு சுழியமாகவும், நிபந்தனை சோதிப்பு i < 4 ஆகவும் உள்ளது. நிபந்தனை சோதிப்பு சரி எனில், மடக்கு செயல்படுத்தப்பட்டு தொடர்புடைய வெளியீடு அச்சிடப்படும்.

முதல் சுழற்சியின் போது, i ன் மதிப்பு சுழியமாகும். கொடுக்கப்பட்ட நிபந்தனை மதிப்பு மெய், எனவே தொடர்ந்து வரும் **print (Marks[i])** செயல்படுத்தப்பட்டு, Marks[0] என்ற உறுப்பின் 10 என்ற மதிப்பு அச்சிடப்படும்.

அடுத்த கூற்று **i = i + 1, i** ன் மதிப்பை சுழியத்திலிருந்து, 1-என மிகுக்கிறது. இப்போது, பாய்வுக் கட்டுப்பாடு, நிபந்தனை சோதிப்பை சரிபார்க்க While கூற்றுக்கு மாற்றப்படுகிறது. கொடுக்கப்பட்ட, while மடக்கின் நிபந்தனை சோதிப்பு பொய் என வரும் வரை **Marks** லிஸ்ட்ன் அனைத்து உறுப்புகளையும் அச்சிடும் செயல் மீண்டும் மீண்டும் செயல்படுத்தப்படும். மீதமுள்ள உறுப்புகளை அச்சிட மீண்டும் செய்யப்படுகிறது.

பின்வரும் அட்டவணை, மடக்கு செயல்படும் விதம் மற்றும் அச்சிடப்பட வேண்டிய மதிப்பைக் காண்பிக்கிறது.

சுழற்சி	i	while i < 4	print (Marks[i])	i = i + 1
1	0	0 < 4 True	Marks [0] = 10	0 + 1 = 1
2	1	1 < 4 True	Marks [1] = 23	1 + 1 = 2
3	2	2 < 4 True	Marks [2] = 41	2 + 1 = 3
4	3	3 < 4 True	Marks [3] = 75	3 + 1 = 4
5	4	4 < 4 False	--	--

9.1.2.2 பின்னோக்கு சுட்டு (Reverse Indexing)

பைத்தான், List உறுப்புகளுக்கு பின்னோக்கு அல்லது எதிர்மறை, சுட்டெண்களை வழங்குகிறது. இதனால் பைத்தான், சுட்டெண்களை எதிர் வரிசையில் பட்டியலிடுகிறது. பைத்தான், List–ன் கடைசி உறுப்பிற்கு -1 முந்தைய உறுப்பிற்கு -2 என்ற சுட்டெண் மதிப்புகளையும் இருத்துகிறது. இதுவே பின்னோக்கு சுட்டு என அழைக்கப்படுகிறது.

எடுத்துக்காட்டு

```
Marks = [10, 23, 41, 75]
i = -1
while i >= -4:
        print (Marks[i])
        i = i + -1
```

வெளியீடு

```
75
41
23
10
```

பின்வரும் அட்டவணை மேலே உள்ள பைத்தான் குறிமுறை வேலை செய்யும் விதத்தை காண்பிக்கிறது.

சுழற்சி	i	while i >= -4	print (Marks[i])	i = i + -1
1	-1	-1 >= -4 True	Marks[-1] = 75	-1 + (-1) = -2
2	-2	-2 >= -4 True	Marks[-2] = 41	-2 + (-1) = -3
3	-3	-3 >= -4 True	Marks[-3] = 23	-3 + (-1) = -4
4	-4	-4 >= -4 True	Marks[-4] = 10	-4 + (-1) = -5
5	-5	-5 >= -4 False	--	--

9.1.3 List –ன் நீளம்

பைத்தானில் உள்ள len( ) செயற்கூறு, List-ன் நீளத்தை கண்டறிய பயன்படுகிறது. (அதாவது, List-ல் உள்ள உறுப்புகளின் எண்ணிக்கை). பொதுவாக, len( ) செயற்கூறு List-ல் அனைத்து உறுப்புகளையும் அணுகுவதற்கான மடக்கின் உச்ச வரம்பை நிர்ணயிக்க பயன்படுகிறது. ஒரு List-ல் மற்றொரு List-ஐ ஒரு உறுப்பாக கொண்டிருந்தால், len( ) செயற்கூறு உட்புற List-ஐ ஒரு உறுப்பாகவே எடுத்துக்கொள்ளும்.

எடுத்துக்காட்டு :ஒற்றை உறுப்பை அணுகுதல்

```
>>> MySubject = [“Tamil”, “English”, “Comp. Science”, “Maths”]
>>> len(MySubject)
     4
```

எடுத்துக்காட்டு : மடக்கை பயன்படுத்தி ஒரு List–ல் உள்ள உறுப்புகளை அச்சிடும் நிரல்

```
MySubject = ["Tamil", "English", "Comp. Science", "Maths"]
i = 0
while i < len(MySubject):
        print (MySubject[i])
        i = i + 1
```

வெளியீடு

```
        Tamil
        English
        Comp. Science
        Maths
```

9.1.4 for மடக்கை பயன்படுத்தி உறுப்புகளை அணுகுதல்

பைத்தானில் உள்ள for மடக்கு, List-ல் உள்ள அனைத்து உறுப்புகளையும் ஒவ்வொன்றாக அணுகுவதற்கு பயன்படுகிறது. இது, C++ போன்ற பிற நிரலாக்க மொழிகளில் உள்ள for மடக்கை போன்றது.

தொடரியல்:

```
for index_var in List:
    print (index_var)
```

இங்கு, index_var என்பது List-ல் உள்ள ஒவ்வொரு உறுப்பின் சுட்டு மதிப்பைக் குறிக்கிறது. பைத்தான் ஆங்கில மொழியில் படிப்பது போன்றே, “for” கூற்றை கையாள்கிறது: “For (every) element in (the List of) List and print (the name of the) List items”

எடுத்துக்காட்டு

```
Marks=[23, 45, 67, 78, 98]
for x in Marks:
        print( x )
```

வெளியீடு

```
        23
        45
        67
        78
        98
```

மேலே உள்ள எடுத்துக்காட்டில், Marks List ஐந்து உறுப்புகளைக் கொண்டுள்ளது; ஒவ்வொரு உறுப்பிற்கும் 0 முதல் 4 வரை சுட்டெண் இருத்தப்படுகிறது. பைத்தான், ஆங்கில மொழியில் வாசிப்பதை போன்றே for மடக்கு மற்றும் **print** கூற்றுகளை படிக்கிறது: “For (every) element (represented as x) in (the List of) Marks and print (the values of the) elements”.

9.1.5 List உறுப்புகளை மாற்றம் செய்தல்

பைத்தானில், List-கள் மாறும் தன்மையுடையவை, அதாவது List உறுப்புகள் மாற்றப்படலாம். List உறுப்பு அல்லது தொடர் உறுப்புகளை எளிய மதிப்பிருத்து செயற்குறியை (=) பயன்படுத்தி மாற்றலாம்.

தொடரியல்:

List_Variable [index of an element] = மாற்றப்படும் மதிப்பு

List_Variable [index from : index to] = மாற்றப்படும் மதிப்பு

இங்கு, index from என்பது தொடரின் தொடக்க சுட்டெண் ஆகும். index to என்பது தொடரின் உச்ச வரம்பாகும். கொடுக்கப்படும் உச்ச மதிப்பு தொடரின் உச்ச மதிப்பாக சேர்க்கப்பட்டாது. எடுத்துக்காட்டாக, தொடர் [0:5] என்று நீங்கள் பொருத்தினால், பைத்தான் 0 முதல் 4 என்ற உறுப்பு சுட்டெண்களை மட்டுமே எடுத்துக் கொள்ளும். எனவே, நீங்கள் 1 முதல் 4 தொடர் உறுப்புகளை மாற்றம் செய்ய விரும்பினால், கட்டாயமாக, [1:5] என்று குறிப்பிட வேண்டும்.

எடுத்துக்காட்டு 9.1: ஒற்றை மதிப்பை மாற்றுவதற்கான பைத்தான் நிரல்

```
MyList = [2, 4, 5, 8, 10]
print ("MyList elements before update... ")
for x in MyList:
      print (x)

MyList[2] = 6
print ("MyList elements after updation... ")
for y in MyList:
      print (y)
```

வெளியீடு

```
MyList elements before update...
                    2
                    4
                    5
                    8
                    10
MyList elements after updation...
                    2
                    4
                    6
                    8
                    10
```

எடுத்துக்காட்டு 9.2: தொடர் மதிப்புகளை மாற்றுவதற்கான பைத்தான் நிரல்

```
MyList = [1, 3, 5, 7, 9]
print ("List Odd numbers... ")
for x in MyList
          print (x)

MyList [0:5] = 2,4,6,8,10
print ("List Even numbers... ")
for y in MyList:
          print (y)
```

வெளியீடு

```
List Odd numbers...
                    1
                    3
                    5
                    7
                    9
List Even numbers...
                    2
                    4
                    6
                    8
                    10
```

9.1.6 List–ல் உறுப்பகளை சேர்த்தல்

பைத்தானில், **append** ( ) செயற்கூறு ஒரு உறுப்பையும் **extend**( ) செயற்கூறு ஒன்றுக்கும் மேற்பட்ட உறுப்புகளையும் ஏற்கனவே உள்ள List-ல் சேர்க்க பயன்படுகின்றது.

தொடரியல்:

List.*append (element to be added)*

List..*extend ([elements to be added])*

extend( ) செயற்கூறில், பல உறுப்புகளை சதுர அடைப்புக் குறிக்குள் செயற்கூறின் செயலுருபுகளைப் போலவே குறிப்பிட வேண்டும்.

எடுத்துக்காட்டு

```
>>> MyList=[34, 45, 48]
>>> MyList.append(90)
>>> print(MyList)
        [34, 45, 48, 90]
```

மேலே உள்ள எடுத்துக்காட்டில், My List என்பது மூன்று உறுப்புகளுடன் உருவாக்கப்பட்டுள்ளது. >>> MyList.append(90) என்ற கூற்றின் மூலம் 90 என்ற கூடுதல் மதிப்பு ஏற்கனவே உள்ள List-ல் கடைசி உறுப்பாக சேர்க்கப்படுகிறது. Print கூற்று MyList என்ற லிஸ்ட்-ல் உள்ள அனைத்து உறுப்புகளையும் காண்பிக்கிறது.

எடுத்துக்காட்டு

```
>>> MyList.extend ([71, 32, 29])
>>> print(MyList)
        [34, 45, 48, 90, 71, 32, 29]
```

மேலே உள்ள குறிமுறையில், extend( ) செயற்கூறு பல உறுப்புகளை சேர்ப்பதற்கு பயன்படுகிறது. Print கூற்று, கூடுதல் உறுப்புகள் சேர்க்கப்பட்ட பிறகு List-ல் உள்ள அனைத்து உறுப்புகளையும் காண்பிக்கிறது.

9.1.7 List உறுப்புகளை செருகுதல்

நீங்கள் ஏற்கனவே படித்தபடி, பைத்தானில் உள்ள append( ) ஒரு உறுப்பை சேர்க்கப் பயன்படுகிறது. ஆனால், இது List-ன் இறுதியில் உறுப்பை சேர்க்கிறது. நீங்கள் உங்களுக்கு விருப்பமான இடத்தில் ஒரு உறுப்பை சேர்க்க விரும்பினால், insert ( ) செயற்கூறை பயன்படுத்த வேண்டும். insert ( ) செயற்கூறு, List-ன் எந்தவொரு இடத்திலும் ஒரு உறுப்பை சேர்க்கப் பயன்படுகிறது.

தொடரியல்:

List.*insert (position index, element)*

எடுத்துக்காட்டு

```
>>> MyList=[34,98,47,'Kannan', 'Gowrisankar', 'Lenin', 'Sreenivasan' ]
>>> print(MyList)
        [34, 98, 47, 'Kannan', 'Gowrisankar', 'Lenin', 'Sreenivasan']
>>> MyList.insert(3, 'Ramakrishnan')
>>> print(MyList)
[34, 98, 47, 'Ramakrishnan', 'Kannan', 'Gowrisankar', 'Lenin', 'Sreenivasan']
```

மேலே உள்ள எடுத்துக்காட்டில், insert( ) செயற்கூறு, ‘Ramakrishnan’ என்ற புதிய உறுப்பை மூன்றாவது சுட்டெண்ணில் அதாவது நான்காவது இடத்தில் சேர்க்கிறது. புதிய உறுப்பை ஏற்கனவே உள்ள உறுப்புகளின் இடையில் ஒரு குறிப்பிட்ட இடத்தில் சேர்க்கும் போது, ஏற்கனவே உள்ள உறுப்புகள் வலது பக்கமாக ஒரு இடம் நகர்த்தப்படும்.

9.1.8 Listலிருந்து உறுப்புகளை நீக்குதல்

List-ல் இருந்து ஒரு உறுப்பை நீக்குவதற்கு இரண்டு வழிகள் உள்ளன. அவை del கூற்று மற்றும் remove( ) செயற்கூறு ஆகும். del கூற்று தெரிந்த உறுப்புகளை நீக்குவதற்கு பயன்படுகிறது. ஆனால், remove( ) செயற்கூறு சுட்டெண் தெரியாத உறுப்புகளை List-லிருந்து நீக்குவதற்கு பயன்படுகிறது. del கூற்று முழு List-ஐ நீக்குவதற்கும் பயன்படுகிறது.

தொடரியல்:

```
del List [index of an element]
# ஒரு குறிப்பிட்ட உறுப்பை நீக்க
del List [index from : index to]
# ஒன்றுக்கும் மேற்பட்ட உறுப்புகளை நீக்க
del List
# ஒரு List-யை நீக்க
```

எடுத்துக்காட்டு

```
>>> MySubjects = ['Tamil', 'Hindi', 'Telugu', 'Maths']
>>> print (MySubjects)
        ['Tamil', 'Hindi', 'Telugu', 'Maths']
>>> del MySubjects[1]
>>> print (MySubjects)
        ['Tamil', 'Telugu', 'Maths']
```

மேலே உள்ள எடுத்துக்காட்டில், **MySubjects** என்ற List நான்கு உறுப்புகளுடன் உருவாக்கப்பட்டுள்ளது. Print கூற்று List-ன் அனைத்து உறுப்புகளையும் காண்பிக்கிறது. >>> del MySubjects[1] கூற்று, சுட்டெண் 1 என்ற உறுப்பை நீக்குகிறது. அதன்பின் வருகின்ற print List-ன் மீதமுள்ள உறுப்புகளை காண்பிக்கிறது.

எடுத்துக்காட்டு

```
>>> del MySubjects[1:3]
>>> print(MySubjects)
        ['Tamil']
```

மேலே உள்ள குறிமுறையில், >>> **del MySubjects[1:3]** என்பது இரண்டு மற்றும் மூன்றாவது உறுப்புகளை List-லிருந்து நீக்குகிறது. சதுர அடைப்புக்குறிக்குள் குறிப்பிடப்பட்டுள்ள சுட்டெண்ணின் உச்ச மதிப்பு -1 என எடுத்துக்கொள்ளும்.

எடுத்துக்காட்டு

```
>>> del MySubjects
>>> print(MySubjects)
Traceback (most recent call last):
File "<pyshell#9>", line 1, in <module>
print(MySubjects)
NameError: name 'MySubjects' is not defined
```

இங்கு, >>> del MySubjects, என்பது MySubjects என்ற List-ஐ முழுவதுமாக நீக்குகிறது. இப்போது உறுப்புகளை அச்சிட முயன்றால், பைத்தான் லிஸ்ட் is not defined என்ற பிழையை காண்பிக்கும். அதாவது, my Subjects என்ற List முழுவதுமாக நீக்கப்பட்டுவிட்டது என்று பொருள்.

ஏற்கனவே குறிப்பிட்டபடி, remove( ) செயற்கூறு சுட்டெண் தெரியாத ஒன்று அல்லது அதற்கு மேற்பட்ட உறுப்புகளை நீக்குவதற்கு பயன்படுகிறது. remove( ) செயற்கூறு மட்டுமல்லாமல், pop( ) செயற்கூறும் கொடுக்கப்பட்ட சுட்டெண்களை பயன்படுத்தி, ஒரு உறுப்பை நீக்க பயன்படுகிறது. pop( ) செயற்கூறு சுட்டெண் கொடுக்கப்படாத போது List-ன் கடைசி உறுப்பை நீக்கி அதை காண்பிக்கிறது.

clear( ) செயற்கூறு List-ன் அனைத்து உறுப்புகளையும் நீக்க பயன்படுகிறது. இது உறுப்புகளை மட்டுமே நீக்கி, List-ஐ தொடர்ந்து வைத்திருக்கிறது. Del கூற்று முழு லிஸ்ட்-ஐ நீக்குகிறது என்பதை நினைவில் கொள்க.

தொடரியல்:

List.*remove(element)* # ஒரு குறிப்பிட்ட உறுப்பை நீக்க
List.*pop(index of an element)*
List.*clear( )*

எடுத்துக்காட்டு

```
>>> MyList=[12,89,34,'Kannan', 'Gowrisankar', 'Lenin']
>>> print(MyList)
        [12, 89, 34, 'Kannan', 'Gowrisankar', 'Lenin']
>>> MyList.remove(89)
>>> print(MyList)
        [12, 34, 'Kannan', 'Gowrisankar', 'Lenin']
```

மேலே உள்ள எடுத்துக்காட்டில், MyList மூன்று முழு எண்கள் மற்றும் மூன்று சரநிலையுரு உறுப்புகளுடன் உருவாக்கப்பட்டுள்ளது. அதன்பின் உள்ள print கூற்று List-ல் உள்ள அனைத்து உறுப்புகளையும் காண்பிக்கிறது. >>> MyList.remove(89) கூற்று, List-ல் இருந்து 89 என்ற உறுப்பை நீக்குகிறது. பின்வரும் print கூற்று மீதமுள்ள உறுப்புகளை காண்பிக்கிறது.

எடுத்துக்காட்டு

```
>>> MyList.pop(1)
      34
>>> print(MyList)
      [12, 'Kannan', 'Gowrisankar', 'Lenin']
```

மேலே உள்ள குறிமுறையில், pop( ) செயற்கூறு குறிப்பிட்ட உறுப்பை அதன் சுட்டெண்ணை பயன்படுத்தி நீக்குவதற்கு பயன்படுகிறது. உறுப்பு நீக்கப்பட்டவுடன் pop( ) செயற்கூறு நீக்கப்பட்ட உறுப்பை காண்பிக்கிறது. pop( ) செயற்கூறு லிஸ்ட்ல் இருந்து ஒரு உறுப்பை மட்டும் நீக்கப்பயன்படுகிறது. Del கூற்று பல உறுப்புகளை நீக்குகிறது என்பதை நினைவில் கொள்க.

எடுத்துக்காட்டு

```
>>> MyList.clear( )
>>> print(MyList)
          [ ]
```

மேலே உள்ள குறிமுறையில், clear( ) செயற்கூறு உறுப்புகளை மட்டும் நீக்கி List-ஐ தொடர்ந்து வைத்திருக்கப் பயன்படுகிறது. உறுப்புகள் நீக்கப்பட்ட List-ஐ அச்சிடுவதற்கு நீங்கள் முயன்றால், எந்தவொரு உறுப்பும் இல்லாத காலியான சதுர அடைப்புக் குறி தோன்றும். அதாவது லிஸ்ட் காலியாக இருக்கும்.

9.1.9 List மற்றும் range ( ) செயற்கூறுகள்

range( ) என்பது பைத்தானில் தொடர் மதிப்புகளை உருவாக்கப் பயன்படும் செயற்கூறாகும். range () செயற்கூறை பயன்படுத்தி நீங்கள் தொடர் மதிப்புகளுடன் List-ஐ உருவாக்கலாம். range( ) செயற்கூறு மூன்று செயலுருபுகளைக் கொண்டுள்ளது.

range ( ) செயற்கூறின் தொடரியல்:

range (start value, end value, step value)

இங்கு,

- **start value** – தொடரின் தொடக்க மதிப்பு. சுழியம் தானமைவு தொடக்க மதிப்பாகும்.
- **end value** – தொடரின் உச்ச வரம்பு. பைத்தான் இறுதி மதிப்பை உச்ச வரம்பு-1 என எடுத்துக் கொள்கிறது.
- **step value** – இது ஒரு விருப்ப செயலுருபு (கொடுக்க வேண்டியது கட்டாயமில்லை) இது வெவ்வேறு இடைவெளிகளில் மதிப்புகளை உருவாக்கப் பயன்படுகிறது.

எடுத்துக்காட்டு 9.3: 10 வரை உள்ள முழு எண்களை உருவாக்குதல்:

```
for x in range (1, 11):
        print(x)
```

வெளியீடு

```
1
2
3
4
5
6
7
8
9
10
```

எடுத்துக்காட்டு 9.4: முதல் 10 இரட்டைப்படை எண்களை உருவாக்குதல்

```
for x in range (2, 11, 2):
        print(x)
```

வெளியீடு

```
2
4
6
8
10
```

9.1.9.1 தொடர் மதிப்புகளுடன் லிஸ்ட்–ஐ உருவாக்குதல்

range( ) செயற்கூறை பயன்படுத்தி நீங்கள் தொடர் மதிப்புகளுடன் கூடிய லிஸ்ட்-ஐ உருவாக்கலாம். range( ) செயற்கூறின் விடையை லிஸ்ட் ஆக மாற்றுவதற்கு, List( ) செயற்கூறு பயன்படுகிறது. List( ) செயற்கூறு range( ) ன் விடையை லிஸ்ட் ஆக உருவாக்குகிறது.

தொடரியல்:

List_*Varibale* = List *(range ( ))*

குறிப்பு

list()செயற்கூறு பைத்தானில் லிஸ்ட் உருவாக்கப்பயன்படுகிறது.

எடுத்துக்காட்டு

```
>>> Even_List = List(range(2,11,2))
>>> print(Even_List)
[2, 4, 6, 8, 10]
```

மேலே உள்ள குறிமுறையில், List( )செயற்கூறு range( ) ன் விடையை Even_List என்ற List உறுப்புகளாக எடுத்துக் கொள்கிறது. எனவே, Even_List என்ற List முதல் ஐந்து இரட்டைப்படை எண்களை உறுப்புகளாக பெற்றிருக்கும்.

இதுபோன்று, எந்த தொடர் எண்களையும் range( ) பயன்படுத்தி உருவாக்கலாம். பின்வரும் எடுத்துக்காட்டு முதல் 10 இயல் எண்களின் 2ன் அடுக்கங்களுடன் கூடிய List-ஐ உருவாக்குதலை விளக்குகிறது.

எடுத்துக்காட்டு 9.5 முதல் 10 இயல் எண்களின் 2-ன் அடுக்குகளை உருவாக்குதல்

```
squares = [ ]
for x in range(1,11):
        s = x ** 2
        squares.append(s)
print (squares)
```

மேலே உள்ள நிரலில், “squares” என்ற பெயருடைய ஒரு வெற்று லிஸ்ட் உருவாக்கப்பட்டுள்ளது. பின்னர், for மடக்கு 1 முதல் 10 இயல் எண்களை range( ) செயற்கூறை பயன்படுத்தி உருவாக்குகிறது. மடக்கின் உள்ளே, **xன்** தற்போதைய மதிப்பு, 2-ன் அடுக்கேற்றத்தால் அதிகப்படுத்தப்பட்ட பின்னர் s என்ற மாறியில் சேமிக்கப்படுகிறது. இறுதியாக, நிரல் பின்வரும் மதிப்புகளை விடையாகக் காண்பிக்கிறது.

வெளியீடு

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

9.1.10 List சுருக்கம் (List comprehensions)

List சுருக்கம் என்பது, கொடுக்கப்பட்ட நிபந்தனைகளுக்கு உட்பட்டு உருவாகும் தொடர் மதிப்புகளை ஏற்கும் List-யை உருவாக்கும் எளிய வழிமுறையாகும்.

தொடரியல்:

List = [*expression for variable in range*]

எடுத்துக்காட்டு 9.6: List சுருக்கத்தைப் பயன்படுத்தி முதல் 10 இயல் எண்களின் 2–ன் அடுக்குகளை உருவாக்குதல்.

```
>>> squares = [ x ** 2 for x in range(1,11) ]
>>> print (squares)
```

வெளியீடு

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

மேலே உள்ள எடுத்துக்காட்டில், x ** 2 என்ற கோவை மடக்குச் செயலின் ஒவ்வொரு சுழற்சியிலும் மதிப்பீடு செய்யப்படுகிறது. இது தொடர் மதிப்புகளை உருவாக்குவதற்கான குறுக்கு வழிமுறையாகும்.

9.1.11 பிற முக்கியமான List செயற்கூறுகள்

<table>
<tr><th>செயற்கூறு</th><th>விளக்கம்</th><th>தொடரியல்</th><th>எடுத்துக்காட்டு</th></tr>
<tr><td>copy ( )</td><td>List-ன் நகலை தரும்.</td><td>List.copy( )</td><td>MyList=[12, 12, 36]
x = MyList.copy()
print(x)

வெளியீடு
[12, 12, 36]</td></tr>
<tr><td>count ( )</td><td>List-ல் உள்ள ஒரே மாதிரியான உறுப்புகளின் எண்ணிக்கையை தரும்.</td><td>List.count(value)</td><td>MyList=[36 ,12 ,12]
x = MyList.count(12)
print(x)

வெளியீடு
2</td></tr>
<tr><td>index ( )</td><td>முதலில் வரும் உறுப்பின் சுட்டெண் மதிப்பை தருகிறது.</td><td>List.index(element)</td><td>MyList=[36 ,12 ,12]
x = MyList.index(12)
print(x)

வெளியீடு
1</td></tr>
<tr><td>reverse ( )</td><td>List-ல் உள்ள உறுப்புகளின் வரிசையை மறுபக்கமாக (தலைகீழாக) திருப்புகிறது,</td><td>List.reverse( )</td><td>MyList=[36 ,23 ,12]
MyList.reverse()
print(MyList)

வெளியீடு
[12 ,23 ,36]</td></tr>
<tr><td>sort ( )</td><td>List-ல் உள்ள உறுப்புகளை வரிசையாக்கம் செய்கிறது.</td><td colspan="2">List.sort(reverse=True|False, key=myFunc)</td></tr>
<tr><td colspan="2">இரண்டு செயலுருபுகளும் கட்டாயமில்லாதவை
• reverse ஐ True என பொருத்தினால் இறங்கு வரிசையில் லிஸ்ட் வரிசையாக்கமாகும்.
• ஏறுவரிசை தானமைவு வரிசையாகமாகும்.
• Key=myFunc; “myFunc” – வரிசையாக்க வரண்முறையைக் குறிப்பிடும் பயனர் வரையறுத்த செயற்கூறின் பெயர்.

குறிப்பு: sort( ) மூல List-யை பாதிக்கும்.</td><td colspan="2">MyList=['Thilothamma', 'Tharani', 'Anitha', 'SaiSree', 'Lavanya']
MyList.sort( )
print(MyList)
MyList.sort(reverse=True)
print(MyList)

வெளியீடு
['Anitha', 'Lavanya', 'SaiSree', 'Tharani', 'Thilothamma']
['Thilothamma', 'Tharani', 'SaiSree', 'Lavanya', 'Anitha']</td></tr>
</table>

max( )	ஒரு List-ன் மதிப்புகளில் உச்ச மதிப்பை தரும்.	max(List)	MyList=[21,76,98,23] print(max(MyList)) வெளியீடு 98
min( )	ஒரு List-ல் உள்ள மதிப்புகளில், மிகக் குறைந்த மதிப்பைத் தரும்.	min(List)	MyList=[21,76,98,23] print(min(MyList)) வெளியீடு 21
sum( )	ஒரு List-டிலுள்ள மதிப்புகளின் கூட்டுத் தொகையை தரும்.	sum(List)	MyList=[21,76,98,23] print(sum(MyList)) வெளியீடு 218

9.1.12 List பயன்படுத்தப்பட்ட நிரல்கள்

நிரல் 1: 1 முதல் 20 வரையான எண்களில் 4-ல் வகுபடும் எண்களை பெறும் List ஒன்றை உருவாக்கும் நிரல் ஒன்றை எழுதுக.

```
divBy4=[ ]
for i in range(21):
        if (i%4==0):
                divBy4.append(i)
print(divBy4)
```

வெளியீடு

```
      [0, 4, 8, 12, 16, 20]
```

நிரல் 2: BRICS ன் உறுப்பு நாடுகளின் List–ஐ வரையறை செய்து, உள்ளீடு செய்யப்படும் நாடு BRICS ன் உறுப்பா (அல்லது) இல்லையா என்பதை பரிசோதிக்கும் நிரலை எழுதுக.

```
country=["India", "Russia", "Srilanka", "China", "Brazil"]
is_member = input("Enter the name of the country: ")
if is_member in country:
        print(is_member, " is the member of BRICS")
else:
        print(is_member, " is not a member of BRICS")
```

வெளியீடு

```
      Enter the name of the country: India
      India  is the member of BRICS
```

வெளியீடு

```
      Enter the name of the country: Japan
      Japan  is not a member of BRICS
```

நிரல் 3: ஆறு பாடங்களின் மதிப்பெண்களை உள்ளீடாகப் பெற்று, ஒவ்வொரு பாடத்திலும் பெற்ற மதிப்பெண்களை அச்சிட்டு மதிப்பெண்களின் கூட்டுத்தொகையை காண்பிக்கும் பைத்தான் நிரலை எழுதுக.

```
marks=[]
subjects=['Tamil', 'English', 'Physics', 'Chemistry', 'Comp. Science', 'Maths']
for i in range(6):
        m=int(input("Enter Mark = "))
        marks.append(m)
for j in range(len(marks)):
        print("{ }. { } Mark = { } ".format(j1+,subjects[j],marks[j]))
print("Total Marks = ", sum(marks))
```

வெளியீடு

```
      Enter Mark = 45
      Enter Mark = 98
      Enter Mark = 76
      Enter Mark = 28
      Enter Mark = 46
      Enter Mark = 15
        1. Tamil Mark = 45
        2. English Mark = 98
        3. Physics Mark = 76
        4. Chemistry Mark = 28
        5. Comp. Science Mark = 46
        6. Maths Mark = 15
        Total Marks =  308
```

நிரல் 4: 5 பொருள்களின் விலையை List–ல் பெற்று, அனைத்து விலைகளின் கூட்டுத்தொகை, பெருக்கற்தொகை மற்றும் சராசரியைக் கண்டறியும் பைத்தான் நிரலை எழுதுக.

```
items=[]
prod=1
for i in range(5):
      print ("Enter price for item { } : ".format(i+1))
      p=int(input())
      items.append(p)
for j in range(len(items)):
      print("Price for item { } = Rs. { }".format(j+1,items[j]))
      prod = prod * items[j]
print("Sum of all prices = Rs.", sum(items))
print("Product of all prices = Rs.", prod)
print("Average of all prices = Rs.",sum(items)/len(items))
```

வெளியீடு

```
Enter price for item  1 :
5
Enter price for item  2 :
10
Enter price for item  3 :
15
Enter price for item  4 :
20
Enter price for item  5 :
25
Price for item  1 = Rs. 5
Price for item  2 = Rs. 10
Price for item  3 = Rs. 15
Price for item  4 = Rs. 20
Price for item  5 = Rs. 25
Sum of all prices = Rs. 75
Product of all prices = Rs. 375000
Average of all prices = Rs. 15.0
```

நிரல் 5: ஆண்டுக்கு 1 இலட்சத்திற்கு மேல் ஊதியம் பெறும் பணியாளர்களின் எண்ணிக்கையை கணக்கிடும் பைத்தான் நிரல் n எண்ணிக்கையிலான பணியாளர்களின் மாத ஊதியத்தை உள்ளீடாக பெற வேண்டும்.

```
count=0
n=int(input("Enter no. of employees: "))
print("No. of Employees",n)
salary=[]
for i in range(n):
        print("Enter Monthly Salary of Employee { } Rs.: ".format(i+1))
        s=int(input())
        salary.append(s)

for j in range(len(salary)):
        annual_salary = salary[j] * 12
        print ("Annual Salary of Employee { } is:Rs. { }".format(j+1,annual_salary))
        if annual_salary >= 100000:
                count = count + 1
print("{ } Employees out of { } employees are earning more than Rs. 1 Lakh per annum".
format(count, n))
```

வெளியீடு

```
Enter no. of employees: 5
No. of Employees 5
Enter Monthly Salary of Employee 1 Rs.:
3000
Enter Monthly Salary of Employee 2 Rs.:
9500
Enter Monthly Salary of Employee 3 Rs.:
12500
Enter Monthly Salary of Employee 4 Rs.:
5750
Enter Monthly Salary of Employee 5 Rs.:
8000
Annual Salary of Employee 1 is:Rs. 36000
Annual Salary of Employee 2 is:Rs. 114000
Annual Salary of Employee 3 is:Rs. 150000
Annual Salary of Employee 4 is:Rs. 69000
Annual Salary of Employee 5 is:Rs. 96000
2 Employees out of 5 employees are earning more than Rs. 1 Lakh per annum
```

நிரல் 6: 1 முதல் 10 வரையுள்ள தொடர் எண்களை ஏற்கும் List ஒன்றை உருவாக்குவதற்கான நிரலை எழுதுக. பிறகு அனைத்து இரட்டைப்படை எண்களையும் நீக்கவிட்டு இறுதிப்பட்டியலை அச்சிடுக.

```
Num = []
for x in range(1,11):
        Num.append(x)
print("The List of numbers from 1 to 10 = ", Num)

for index, i in enumerate(Num):
        if(i%2==0):
                del Num[index]
print("The List after deleting even numbers = ", Num)
```

வெளியீடு

```
The List of numbers from 1 to 10 =  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
The List after deleting even numbers =  [1, 3, 5, 7, 9]
```

நிரல் 7: பைபோனாசி வரிசையை உருவாக்கி அதை List –ல் சேமிப்பதற்கான நிரலை எழுதுக. பிறகு அனைத்து மதிப்புகளின் கூட்டுத்தொகையைக் கண்டறிக.

```
a=-1
b=1
n=int(input("Enter no. of terms: "))
i=0
sum=0
Fibo=[]
while i<n:
        s = a + b
        Fibo.append(s)
        sum+=s
        a = b
        b = s
        i+=1
print("Fibonacci series upto "+ str(n) +" terms is : " + str(Fibo))
print("The sum of Fibonacci series: ",sum)
```

வெளியீடு

```
        Enter no. of terms: 10
        Fibonacci series upto 10 terms is : [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
        The sum of Fibonacci series:  88
```

9.2 Tuples

Tuples அறிமுகம்:

Tuples காற்புள்ளியால் பிரிக்கப்பட்ட பல மதிப்புகளை வளைவு அடைப்புக் குறிக்குள் கொண்ட தரவினமாகும். டப்புல்ஸ் List-க்கு இணையானதாகும். ஆனால் List-ல் மதிப்புகளை மாற்ற முடியும். Tuples -ல் மாற்ற முடியாது.

உங்களுக்குத் தெரியுமா?

Tuples என்ற வார்த்தை லத்தீன் மொழியிலிருந்து தோற்றுவிக்கப்பட்டது. இந்த லத்தீன் வார்த்தை எண் தொடர்களைக் குறிக்கிறது:

single(1), double(2), triple(3), quadruple(4), quinTuples(5), sexTuples(6), sepTuples(7), ocTuples(8), ..., n-Tuples, ...,

9.2.1 List–க்கு மேலான Tuples–ன் நன்மைகள்

1. List-ன் உறுப்புகளை மாற்றலாம் ஆனால் Tuples-ன் உறுப்புகளை மாற்ற முடியாது. இதுவே List மற்றும் Tuples-க்கு இடையே உள்ள முக்கியமான வேறுபாடு ஆகும்.
2. List-ன் உறுப்புகள் சதுர அடைப்புக் குறிக்குள் அடைக்கப்பட்டிருக்கும். ஆனால், Tuple -ன் உறுப்புகள் வளைவு குறிக்குள் அடைக்கப்பட்டிருக்கும்
3. Tuples-ன் மடக்குச் செயல் List-ஐ காட்டிலும் விரைவானது.

9.2.2 Tuples உருவாக்குதல்

List-ஐ போன்றே Tuples உருவாக்கப்படும். List-ல் உறுப்புகள் சதுர அடைப்புக்குறிக்குள் அடைக்கப்பட்டு வரையறை செய்யப்படுகின்றன. ஆனால் டப்புள்ஸ்ல் அவைகள் வளைந்த அடைப்புக்குறிக்குள் அடைக்கப்பட்டிருக்கலாம். Tuples-ன் உறுப்புகளை வளைந்த அடைப்புக்குறி இல்லாமலும் வரையறுக்க முடியும். எவ்வாறு பயன்படுத்தினாலும் Tuples-ன் வேலை செய்யும் விதம் ஒன்றாகவே இருக்கும்.

தொடரியல்:

வெற்று Tuples

Tuples_***Name*** = ***( )***

n எண்ணிக்கை உறுப்புகளுடன் Tuples

Tuples_***Name*** = ***(E1, E2, E2 ……. En)***

அடைப்புக்குறி இல்லாத Tuple உறுப்புகள்

Tuples_***Name*** = ***E1, E2, E3 ….. En***

எடுத்துக்காட்டு

```
>>> MyTup1 = (23, 56, 89, 'A', 'E', 'I', "Tamil")
>>> print(MyTup1)
          (23, 56, 89, 'A', 'E', 'I', 'Tamil')

>>> MyTup2 = 23, 56, 89, 'A', 'E', 'I', "Tamil"
>>> print (MyTup2)
          (23, 56, 89, 'A', 'E', 'I', 'Tamil')
```

9.2.2.1 செயற்கூறை பயன்படுத்தி Tuples உருவாக்குதல்

Tuples ( ) செயற்கூறு லிஸ்ட்டிலிருந்து Tuples- ஐ உருவாக்கவும் பயன்படுகிறது. லிஸ்ட்-ல் இருந்து Tuples-ஐ உருவாக்கும் போது, உறுப்புகள் சதுர அடைப்புக்குறிக்குள் அடைக்கப்பட்டிருக்க வேண்டும்.

தொடரியல்:

Tuples_*Name* = Tuples *([List elements])*

எடுத்துக்காட்டு

```
>>> MyTup3 = tuple( [23, 45, 90] )
>>> print(MyTup3)
          (23, 45, 90)
>>> type (MyTup3)
          <class ‘Tuples’>
```

குறிப்பு

பைத்தானில் ஒரு பொருளின் தரவினத்தை அறிய type() செயற்கூறு பயன்படுகிறது.

9.2.2.2 ஒற்றை உறுப்பு கொண்ட Tuples உருவாக்குதல்

Tuples-ஐ ஒற்றை உறுப்புடன் உருவாக்கும் போது, உறுப்பின் இறுதியில் காற்புள்ளியை சேர்க்க வேண்டும். காற்புள்ளி இல்லையெனில், பைத்தான் அந்த உறுப்பை ஒரு சாதாரண தரவினம் போல எடுத்துக் கொள்ளும், Tuples ஆக கருதாது. ஒரு உறுப்புடன் Tuples- ஐ உருவாக்குவது “சிங்கிள்டன் (singleton)” Tuples என அழைக்கப்படுகிறது.

எடுத்துக்காட்டு

```
>>> MyTup4 = (10)
>>> type(MyTup4)
        <class 'int'>

>>> MyTup5 = (10,)
>>> type(MyTup5)
        <class 'Tuples'>
```

9.2.3 Tuples–ன் மதிப்புகளை அணுகுதல்

List-ஐ போலவே, Tuples உறுப்புகளும் சுழியத்தில் தொடங்கும் சுட்டெண்களை பெற்றிருக்கும். Tuples-ன் உறுப்புகளை சுட்டெண்ணை பயன்படுத்தி எளிமையாக அணுகலாம்.

எடுத்துக்காட்டு

```
>>> Tup1 = (12, 78, 91, “Tamil”, “Telugu”, 3.14, 69.48)

# Tuples-ன் அனைத்து உறுப்புகளையும் அணுக.
>>> print(Tup1)
        (12, 78, 91, 'Tamil', 'Telugu', 3.14, 69.48)

#சுட்டெண்ணை பயன்படுத்தி தேர்வு செய்யப்பட்ட உறுப்புகளை அணுகுதல்.
>>> print(Tup1[2:5])
        (91, 'Tamil', 'Telugu')

#முதல் உறுப்பிலிருந்து குறிப்பிடப்பட்ட சுட்டெண் வரை அணுகுதல்.
>>> print(Tup1[:5])
        (12, 78, 91, 'Tamil', 'Telugu')

# குறிப்பிடப்பட்ட உறுப்பிலிருந்து இறுதி உறுப்பு வரை அணுகுதல்.
>>> print(Tup1[4:])
        ('Telugu', 3.14, 69.48)

# முதல் உறுப்பிலிருந்து இறுதி உறுப்பு வரை அணுகுதல்
>>> print(Tup1[:])
        (12, 78, 91, 'Tamil', 'Telugu', 3.14, 69.48)
```

9.2.4 Tuples–ஐ புதுப்பித்தல் மற்றும் நீக்குதல்:

நீங்கள் அறிந்தபடி, Tuples-ஐ மாற்ற முடியாது என அறிவோம். அதாவது Tuples-ன் உறுப்புகளை மாற்றம் (புதுப்பிக்க) செய்ய முடியாது . Tuples-ன் மதிப்புகளை மாற்றுவதற்கு பதிலாக, இரண்டு Tuples- ஐ இணைக்கவும் அல்லது முழு Tuples-ஐ நீக்கவும் முடியும்.

எடுத்துக்காட்டு

```
# இரண்டு Tuples-களை இணைக்கும் நிரல்
Tup1 = (2,4,6,8,10)
Tup2 = (1,3,5,7,9)
Tup3 = Tup1 + Tup2
print(Tup3)
```

வெளியீடு

```
(2, 4, 6, 8, 10, 1, 3, 5, 7, 9)
```

முழு Tuples-ஐ நீக்குவதற்கு, del கட்டளை பயன்படுகிறது.

தொடரியல்:

del Tuples_*name*

எடுத்துக்காட்டு

```
Tup1 = (2,4,6,8,10)
print("The elements of Tup1 is ", Tup1)
del Tup1
print (Tup1)
```

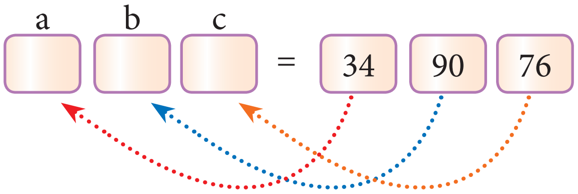
வெளியீடு

```
The elements of Tup1 is  (2, 4, 6, 8, 10)
Traceback (most recent call last):
File "D:/Python/Tuple Examp 1.py", line 4, in <module>
print (Tup1)
NameError: name 'Tup1' is not defined
```

குறிப்பு: மேலே உள்ள குறிமுறையில் print கூற்று உறுப்புகளை அச்சிடுகிறது. பின்னர், del கூற்று முழு Tuples-ஐ நீக்குகிறது. நீக்கப்பட்ட Tuples-ஐ நீங்கள் அச்சிட முயன்றால், பைத்தான் பிழையைக் காண்பிக்கும்.

9.2.5 Tuples-க்கு மதிப்பிருத்துதல்

Tuples மதிப்பிருத்தல் என்பது பைத்தானில் ஆற்றல் மிக்க சிறப்பியல்பாகும். இது, மதிப்பிருத்து செயற்குறியின் இடது பக்கத்தில் உள்ள மாறிகளில் மதிப்பிருத்த செயற்குறியின் வலது பக்கத்தில் உள்ள மதிப்புகளை இருத்த அனுமதிக்கிறது. ஒவ்வொரு மதிப்பும் அதற்குரிய மாறியில் இருத்தப்படுகிறது.



எடுத்துக்காட்டு

```
>>> (a, b, c) = (34, 90, 76)
>>> print(a,b,c)
        34 90 76
# மதிப்பிருத்துவதற்கு முன் கோவைகள் மதிப்பீடு செய்யப்படுகின்றன
>>> (x, y, z, p) = (2**2, 5/3+4, 15%2, 34>65)
>>> print(x,y,z,p)
        4 5.666666666666667 1 False
```

குறிப்பு: நீங்கள், Tuples- ல் மதிப்புகளை இருத்தும் போது, மதிப்பிருத்து செயற்குறியின் இரு பக்கங்களிலும் உள்ள மதிப்புகளின் எண்ணிக்கை சமமாக உள்ளதா என்பதை உறுதி செய்து கொள்ள வேண்டும். இல்லையெனில், பைத்தானில் பிழை உருவாக்கப்படும்.

9.2.6 Tuples–ல் ஒன்றுக்கும் மேற்பட்ட மதிப்புகளை திருப்புதல்

ஒரு செயற்கூறு ஒரு நேரத்தில் ஒரு மதிப்பை மட்டுமே திருப்பி அனுப்ப முடியும். ஆனால் பைத்தான், செயற்கூறிலிருந்து ஒன்றுக்கு மேற்பட்ட மதிப்புகளை திருப்பி அனுப்புகிறது. பைத்தான் பல மதிப்புகளை தொகுத்து, அவைகளை ஒன்றாக திருப்பி அனுப்புகிறது.

எடுத்துக்காட்டு 9.7: List அதிகபட்ச மற்றும் குறைந்தபட்ச மதிப்புகளை திருப்பி அனுப்பும் நிரல்

```
def Min_Max(n):
      a = max(n)
      b = min(n)
      return(a, b)
Num = (12, 65, 84, 1, 18, 85, 99)
(Max_Num, Min_Num) = Min_Max(Num)
print("Maximum value = ", Max_Num)
print("Minimum value = ", Min_Num)
```

வெளியீடு

```
      Maximum value =  99
      Minimum value =  1
```

9.2.7 பின்னலான Tuples

பைத்தானில், ஒரு Tuples-ஐ மற்றொரு Tuples-க்குள் வரையறை செய்வதை பின்னலான Tuples என்கிறோம். பின்னலான Tuples-ல் ஒவ்வொரு Tuples-ல் ஒரு உறுப்பாக கருதப்படுகிறது. for மடக்கு பின்னலான Tuples-ன் அனைத்து உறுப்புகளை அணுகுவதற்கு பயன்படுகிறது.

எடுத்துக்காட்டு

```
Toppers = (("Vinodini", "XII-F", 98.7), ("Soundarya", "XII-H", 97.5),
                    ("Tharani", "XII-F", 95.3), ("Saisri", "XII-G", 93.8))
for i in Toppers:
          print(i)
```

வெளியீடு

```
('Vinodini', 'XII-F', 98.7)
('Soundarya', 'XII-H', 97.5)
('Tharani', 'XII-F', 95.3)
('Saisri', 'XII-G', 93.8)
```

குறிப்பு

லிஸ்ட்ல் பயன்படுத்திய அனைத்து செயற்கூறுகளும் Tuples-க்கும் பொருந்தும்

9.2.8 Tuples-ஸ் பயன்படுத்தப்பட்ட நிரல்கள்

நிரல் 1: Tuples-ஸ் மதிப்பிருத்தலை பயன்படுத்தி இரண்டு மதிப்புகளை இடமாற்றுவதற்கான நிரலை எழுதுக.

```
a = int(input("Enter value of A: "))
b = int(input("Enter value of B: "))
print("Value of A = ", a, "\n Value of B = ", b)
(a, b) = (b, a)
print("Value of A = ", a, "\n Value of B = ", b)
```

வெளியீடு

```
        Enter value of A: 54
        Enter value of B: 38
        Value of A =  54
         Value of B =  38
        Value of A =  38
         Value of B =  54
```

நிரல் 2: செயற்கூறைப் பயன்படுத்தி, வட்டத்தின் ஆரத்தை செயற்கூறுக்கு அளபுருவாக அனுப்பி, வட்டத்தின் பரப்பு மற்றும் சுற்றளவை திருப்பி அனுப்புவதற்கான நிரலை எழுதுக.

```
pi = 3.14
def Circle(r):
        return (pi*r*r, 2*pi*r)

radius = float(input("Enter the Radius: "))
(area, circum) = Circle(radius)
print ("Area of the circle = ", area)
print ("Circumference of the circle = ", circum)
```

வெளியீடு

```
Enter the Radius: 5
Area of the circle =  78.5
Circumference of the circle =  31.400000000000002
```

நிரல் 3: நேர்மறை மற்றும் எதிர்மறை எண்களின் List–ஐ கொண்ட நிரலை எழுதுக. லிஸ்ட்ல் இருந்து நேர்ம எண்களை மட்டும் பெறுகின்ற புதிய Tuples உருவாக்குக.

```
Numbers = (5, -8, 6, 8, -4, 3, 1)
Positive = ( )
for i in Numbers:
        if i > 0:
                Positive += (i, )
print("Positive Numbers: ", Positive)
```

வெளியீடு

```
Positive Numbers:  (5, 6, 8, 3, 1)
```

9.3 SET

அறிமுகம்

பைத்தானில், Set என்பது தரவின தொகுப்பின் மற்றொரு வகையாகும். Set என்பது மாறக்கூடிய மற்றும் நகல்கள் இல்லாத வரிசைப்படுத்தப்படாத உறுப்புகளின் தொகுப்பாகும். அதாவது, Set-ல் உள்ள உறுப்புகள் மீண்டும் இடம்பெற முடியாது. இந்த சிறப்பியல்பு உறுப்பு சோதனையை சேர்க்கவும் மற்றும் நகல் உறுப்புகளை நீக்கவும் பயன்படுகிறது.

9.3.1 Set உருவாக்குதல்

நெளிவு அடைப்புக்குறிக்குள் காற்புள்ளியால் பிரிக்கப்பட்ட அனைத்து உறுப்புகளையும் இருத்துவதன் மூலம் Set உருவாக்கப்படுகிறது. Set() செயற்கூறு, பைதானில் Set-ஐ உருவாக்கப் பயன்படுகிறது.

தொடரியல்:

Set_Variable = {E1, E2, E3 …….. En}

TF1B1L

எடுத்துக்காட்டு

```
>>> S1={1,2,3,'A',3.14}
>>> print(S1)
        {1, 2, 3, 3.14, 'A'}

>>> S2={1,2,2,'A',3.14}
>>> print(S2)
        {1, 2, 'A', 3.14}
```

மேலே கொடுக்கப்பட்ட எடுத்துக்காட்டில், Set S1 என்பது நகல்கள் இல்லாத பல்வேறு வகை உறுப்புகளுடன் உருவாக்கப்பட்டுள்ளது. Set S2 என்பது நகல் மதிப்புகளுடன் உருவாக்கப்பட்டுள்ளது. ஆனால் பைத்தான், நகல் மதிப்புகளிலிருந்து ஒரேயொரு உறுப்பை மட்டுமே எடுத்துக் கொள்ளும். அதாவது, பைத்தான் நகல் மதிப்புகளை நீக்கிவிடும் ஏனெனில் பைத்தானில் Set நகல் உறுப்புகளை பெற்றிருக்காது.

குறிப்பு

Set-ல் இருந்து உறுப்புகளை அச்சிடும் பொழுது, பைத்தான் மதிப்புகளை வெவ்வேறு வரிசையில் காண்பிக்கும்.

9.3.2 List அல்லது Tuple ஐ பயன்படுத்தி Set உருவாக்குதல்

ஒரு List அல்லது Tuple ஐ Set() செயற்கூறைப் பயன்படுத்தி Set ஆக மாற்ற முடியும். இது மிகவும் எளிய முறையாகும். List அல்லது Tuple ஐ உருவாக்கிய பின் அதன் மாறியை Set() செயற்கூறினுள் அளபுருவாக கொடுக்க வேண்டும்.

எடுத்துக்காட்டு

```
MyList=[2,4,6,8,10]
MySet=set(MyList)
print(MySet)
```

வெளியீடு

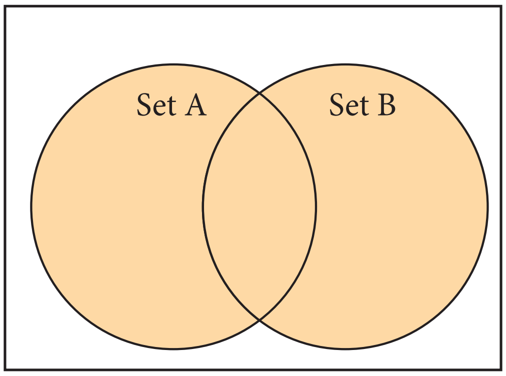
```
        {2, 4, 6, 8, 10}
```

9.3.3 Set செயற்பாடுகள்

கணிதத்தில் கற்ற Set செயற்பாடுகள் ஆகிய ஒட்டு (Union), வெட்டு (intersection)வேறுபாடு (difference) சமச்சீரான வேறுபாடு (Symmetric difference), போன்றவற்றை பைத்தானிலும் பயன்படுத்தலாம்.

9.3.3.1 ஒட்டு (Union):

இது இரண்டு அல்லது அதற்கு மேற்பட்ட Set-களின் அனைத்து உறுப்புகளையும் உள்ளடக்கும்.



பைத்தானில் | என்ற செயற்குறி இரண்டு **Set** களின் ஒட்டை உருவாக்கப்பயன்படுகிறது. **Union** செயற்கூறும் பைத்தானில் இரண்டு **Set** களை இணைக்கப் பயன்படுகிறது.

எடுத்துக்காட்டு: ஒட்டு (Union) செயற்குறியைப் பயன்படுத்தி இரண்டு Set களை இணைப்பதற்கான நிரல்

```
set_A={2,4,6,8}
set_B={'A', 'B', 'C', 'D'}
U_set=set_A|set_B
print(U_set)
```

வெளியீடு

```
{2, 4, 6, 8, 'A', 'D', 'C', 'B'}
```

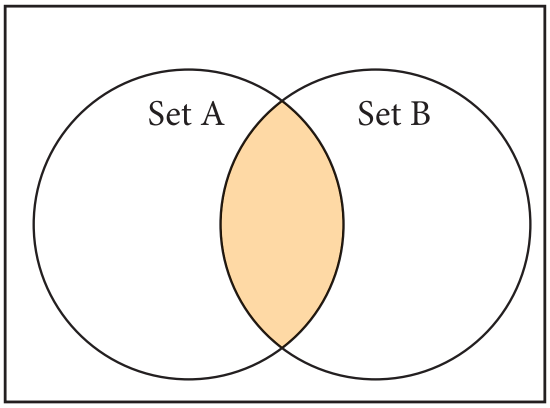
எடுத்துக்காட்டு: ஒட்டு செயற்கூறுவைப் பயன்படுத்தி இரண்டு Setகளை இணைப்பதற்கான நிரல்

```
set_A={2,4,6,8}
set_B={'A', 'B', 'C', 'D'}
set_U=set_A.union(set_B)
print(set_U)
```

வெளியீடு

```
{'D', 2, 4, 6, 8, 'B', 'C', 'A'}
```

9.3.3.2 வெட்டு (intersection)இது இரண்டு Setகளின் பொதுவான உறுப்புகளை உள்ளடக்கியது



பைத்தானில் & செயற்குறி இரண்டு Set களை வெட்டுவதற்கு பயன்படுகிறது. வெட்டு (intersection()) செயற்கூறும் பைத்தானில் இரண்டு Set களை வெட்டுவதற்கு பயன்படுகிறது.

எடுத்துக்காட்டு: வெட்டு intersection()செயற்குறியைப் பயன்படுத்தி இரண்டு Setகளை வெட்டுவதற்கான நிரல்

```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A & set_B)
```

வெளியீடு

```
{'A', 'D'}
```

எடுத்துக்காட்டு: வெட்டு intersection()செயற்குறியைப் பயன்படுத்தி இரண்டு Setகளை வெட்டுவதற்கான நிரல்

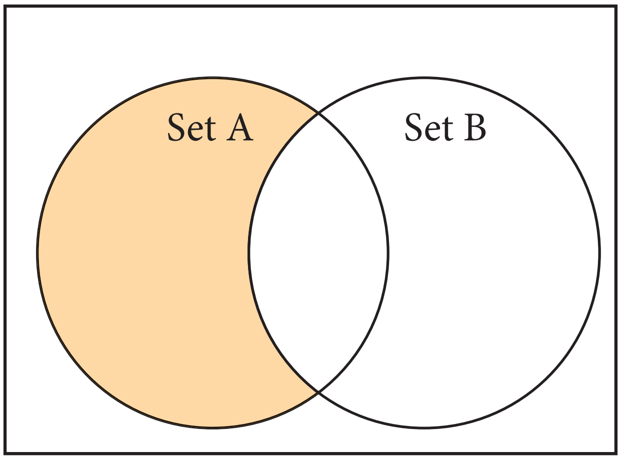
```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A.intersection(set_B))
```

வெளியீடு

```
{'A', 'D'}
```

9.3.3.3 வேறுபாடு Difference

இது முதல் Set(A) ல் உள்ள அனைத்து உறுப்புகளையும் உள்ளடக்கியது. இது இரண்டாவது Set-ஐ தவிர்க்கிறது.



பைத்தானில் –(minus) செயற்குறி Set செயற்பாட்டின் வேறுபாட்டைக் கண்டறிய பயன்படுகிறது. difference() செயற்கூறும் வேறுபாட்டு செயற்பாட்டிற்காக பயன்படுகிறது.

எடுத்துக்காட்டு: செயற்குறியைப் பயன்படுத்தி இரண்டு Setகளின் வேற்றுமைக்கான நிரல்

```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A - set_B)
```

வெளியீடு

```
{2, 4}
```

எடுத்துக்காட்டு: difference செயற்கூறுவைப் பயன்படுத்தி இரண்டு Setகளின் வேற்றுமைக்கான நிரல்

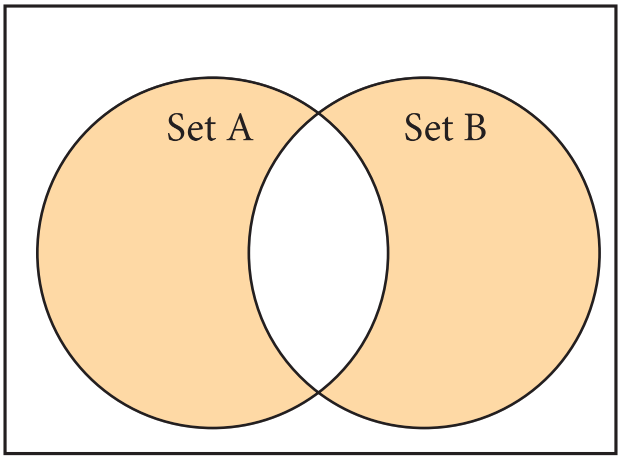
```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A.difference(set_B))
```

வெளியீடு

```
{2, 4}
```

9.3.3.4 சமச்சீரான வேறுபாடு (Symmetric difference)

இது இரண்டு Set-ல் உள்ள பொதுவான உறுப்புகளை மட்டும் தவிர்த்து மற்ற அனைத்து உறுப்புகளையும் உள்ளடக்கியது.



Caret(^) செயற்குறி பைத்தானில் சமச்சீரான வேறுபாட்டை கண்டறிய பயன்படுகிறது. Symmetric difference() செயற்கூறும் அதே செயலை செய்ய பயன்படுகிறது.

எடுத்துக்காட்டு: Caret(^) செயற்குறியைப் பயன்படுத்தி சமச்சீரான வேறுபாட்டை கண்டறியும் நிரல்.

```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A ^ set_B)
```

வெளியீடு

```
{2, 4, 'B', 'C'}
```

எடுத்துக்காட்டு: Symmetric difference செயற்கூறைப் பயன்படுத்தி சமச்சீரான வேறுபாட்டை கண்டறியும் நிரல்

```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A.symmetric_difference(set_B))
```

வெளியீடு

```
{2, 4, 'B', 'C'}
```

9.3.4 setகளைப் பயன்படுத்தி நிரல்

நிரல் **1:** பகா எண்களைக் கொண்ட ஒரு **Set,** இரட்டைப்படை எண்களை கொண்ட மற்றொரு **Set** உருவாக்குவதற்கான நிரல் (ஒட்டு, வெட்டு, வேறுபாடு, சமச்சீரான வேறுபாடு போன்ற செயற்பாடுகளுக்கான விடையை நிரூபிக்கவும்).

எடுத்துக்காட்டு

```
even=set([x*2 for x in range(1,11)])
primes=set()
for i in range(2,20):
        j=2
        f=0
        while j<=i/2:
                if i%j==0:
                        f=1
                j+=1
        if f==0:
                primes.add(i)
print("Even Numbers: ", even)
print("Prime Numbers: ", primes)
print("Union: ", even.union(primes))
print("Intersection: ", even.intersection(primes))
print("Difference: ", even.difference(primes))
print("Symmetric Difference: ", even.symmetric_difference(primes))
```

வெளியீடு

```
Even Numbers: {2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
Prime Numbers: {2, 3, 5, 7, 11, 13, 17, 19}
Union: {2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20}
Intersection: {2, 4}
Difference: {6, 8, 10, 12, 14, 16, 18, 20}
Symmetric Difference: {3, 5, 6, 7, 8, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20}
```

9.4 Dictionaries

அறிமுகம்

பைத்தானில், Dictionary என்பது பல்வேறு வகையான உறுப்புகளின் தொகுப்பாகும். பிற தரவினங்களாகிய List அல்லது Tuples போன்று இல்லாமல், Dictionary வகை உறுப்புகளுடன் அதற்கான திறவு கோலையும் (இணைப்புப் பெயர்) சேமிக்கிறது. பைத்தான் Dictionary உள்ள திறவு கோல்கள் முக்காற்புள்ளியாலும் (:) உறுப்புகள் காற்புள்ளியாலும் (,) பிரிக்கப்பட வேண்டும். திறவுகோல் மற்றும் மதிப்புகள் நெளிவு அடைப்புக்குறிக்குள் { } வரையறுக்கப்படும்.

Dictionaryஐ வரையறுப்பதற்கான தொடரியல்:

```
Dictionary_Name = { Key_1: Value_1,
                    Key_2:Value_2,
                    ……..
                    Key_n:Value_n
                    }
```

Dictionary உள்ள திறவுகோல் தனித்தன்மை வாய்ந்த ஒரே வகையான (Case Sensitive) மற்றும் பைத்தானில் எந்தவொரு தகுதிவாய்ந்த தரவினமாகவும் இருக்கலாம்.

9.3.1 Dictionaryஐ உருவாக்குதல்

```
# காலியான வெற்று Dictionary
    Dict1 = { }

# திறவுகோலைக் கொண்ட Dictionary
    Dict_Stud = { 'RollNo': 1234, 'Name':'Murali', 'Class':'XII', 'Marks':451}
```

9.3.2 Dictionary சுருக்கம்

பைத்தானில், சுருக்கம் என்பது Dictionary ஐ உருவாக்குவதற்கான மற்றொரு வழியாகும். இந்த வகை Dictionary ஐ உருவாக்குவதற்கான தொடரியல் பின்வருமாறு கொடுக்கப்பட்டுள்ளது.

தொடரியல்:

Dict = { expression for variable in sequence [if condition] }

if நிபந்தனை கட்டாயமல்ல. If நிபந்தனையை குறிப்பிட்டால், நிபந்தனை பூர்த்தி செய்யும் மதிப்புகள் மட்டுமே கோவையைப் பயன்படுத்தி மதிப்பீடு செய்யப்படுகின்றன.

எடுத்துக்காட்டு

*Dict = { x : 2 * x for x in range(1,10)}*

மேலே உள்ள குறிமுறையின் வெளியீடு

{1: 2, 2: 4, 3: 6, 4: 8, 5: 10, 6: 12, 7: 14, 8: 16, 9: 18}

9.3.3 Dictionary உறுப்புகளை அணுகுதல், சேர்த்தல், மாற்றியிடுதல் மற்றும் நீக்குதல்

Dictionary ன் அனைத்து உறுப்புகளையும் அணுகுதல் List மற்றும் Tuples-ஐ போன்றே உள்ளது. எளிய print செயற்கூறு அனைத்து உறுப்புகளையும் அணுகப் பயன்படுகிறது. நீங்கள், ஒரு குறிப்பிட்ட உறுப்பை அணுக விரும்பினால், திறவுகோலுடன் சதுர அடைப்புக்குறி பயன்படுத்தப்பட வேண்டும்.

எடுத்துக்காட்டு 9.8: Dictionary சேமிக்கப்பட்ட அனைத்து மதிப்புகளையும் அணுகுவதற்கான நிரல்.

```
MyDict = { 'Reg_No': '1221',
           'Name' : 'Tamilselvi',
           'School' : 'CGHSS',
           'Address' : 'Rotler St., Chennai 112' }
print(MyDict)
print("Register Number: ", MyDict['Reg_No'])
print("Name of the Student: ", MyDict['Name'])
print("School: ", MyDict['School'])
print("Address: ", MyDict['Address'])
```

வெளியீடு

```
    {'Reg_No': '1221', 'Name': 'Tamilselvi', 'School': 'CGHSS', 'Address': 'Rotler St., Chennai 112'}
    Register Number:  1221
    Name of the Student:  Tamilselvi
    School:  CGHSS
    Address:  Rotler St., Chennai 112
```

குறிப்பு: முதல் print கூற்று Dictionary–ல் உள்ள அனைத்த மதிப்புகளையும் அச்சிடுகிறது. மற்ற கூற்றுகள், சதுர அடைப்புக்குறிக்குள் உள்ள குறிப்பிட்ட மதிப்புகளை மட்டுமே அச்சிடுகின்றன.

ஏற்கனவே உள்ள Dictionary-ல் மதிப்பை அதன் திறவுகோலுடன் இருத்துவதன் வழியாக பல மதிப்புகளை சேர்க்கலாம். பின்வரும் தொடரியல் Dictionary பல உறுப்புகள் சேர்த்தலை புரிந்து கொள்ள பயன்படுகிறது.

dictionary_name [key] = value/element

எடுத்துக்காட்டு 9.9: Dictionary புதிய மதிப்பை சேர்ப்பதற்கான நிரல்

```
MyDict = { 'Reg_No': '1221',
                'Name' : 'Tamilselvi',
                'School' : 'CGHSS', 'Address' : '
                Rotler St., Chennai 112'}
print(MyDict)
print("Register Number: ", MyDict['Reg_No'])
print("Name of the Student: ", MyDict['Name'])
MyDict['Class'] = 'XII - A'                # Adding new value
print("Class: ", MyDict['Class'])          # Printing newly added value
print("School: ", MyDict['School'])
print("Address: ", MyDict['Address'])
```

Dictionary உள்ள மதிப்பை மாற்றியிடுதல், உறுப்புகளை சேர்ப்பது போன்றதே ஆகும். நீங்கள் திறவுகோலுக்கு ஒரு மதிப்பை இருத்தும் போது, அது பழைய மதிப்பின் மேல் எழுதப்படுகிறது.

பைத்தான் Dictionary, del சிறப்புச் சொல் ஒரு குறிப்பிட்ட உறுப்பை நீக்குவதற்கு பயன்படுகிறது. clear( ) செயற்கூறு Dictionary-ன் அனைத்து உறுப்புகளையும் நீக்குவதற்கு பயன்படுகிறது. Dictionary யை நீக்குவதற்கு del சிறப்புச் சொல்லுடன் Dictionary ன் பெயரையும் பயன்படுத்த வேண்டும்.

தொடரியல்:

\# குறிப்பிட்ட உறுப்பை நீக்க.
del Dictionary_name[key]

\# அனைத்து உறுப்புகளையும் நீக்க
Dictionary***_name.clear( )***

\# முழு Dictionary நீக்க
del Dictionary***_name***

எடுத்துக்காட்டு 9.10: Dictionary ல் இருந்து உறுப்புகளை நீக்குதல் மற்றும் இறுதியாக டிக்ஷ்னரிஜ நீக்குவதற்கான நிரல்.

```
Dict = {'Roll No' : 12001, 'SName' : 'Meena', 'Mark1' : 98, 'Marl2' : 86}
print("Dictionary elements before deletion: \n", Dict)
del Dict['Mark1']                    # குறிப்பிட்ட உறுப்பை நீக்குதல்
print("Dictionary elements after deletion of a element: \n", Dict)
Dict.clear()                         # அனைத்து உறுப்புகளையும் நீக்குதல்
print("Dictionary after deletion of all elements: \n", Dict)
del Dict
print(Dict)                          # முழு Dictionary ஐ நீக்குதல்
```

வெளியீடு

```
Dictionary elements before deletion:
        {'Roll No': 12001, 'SName': 'Meena', 'Mark1': 98, 'Marl2': 86}
Dictionary elements after deletion of a element:
        {'Roll No': 12001, 'SName': 'Meena', 'Marl2': 86}
Dictionary after deletion of all elements:
        { }
Traceback (most recent call last):
        File "E:/Python/Dict_Test_02.py", line 8, in <module>
        print(Dict)
NameError: name 'Dict' is not defined
```

9.3.4 List மற்றும் Dictionary இடையேயான வேறுபாடு

(1) List என்பது வரிசைப்படுத்திய உறுப்புகளின் தொகுப்பாகும். ஆனால், Dictionary ஒரு உறுப்பை (திறவுகோல்) மற்றொரு உறுப்புடன் (மதிப்பு) பொருத்தப் பயன்படும் தரவு அமைப்பாகும்.

(2) List-ன் சுட்டெண்கள் குறிப்பிட்ட உறுப்பை அணுகுவதற்குப் பயன்படுகின்றன. ஆனால், Dictionary-ல் திறவுகோல் சுட்டெண்ணைக் குறிக்கிறது. ஒரு சரத்தின் எண்ணாகவும், திறவுகோல் இருக்கலாம் என்பதை நினைவில் கொள்க.

(3) List-ன் மதிப்பை பார்த்துக் கொள்ளப்பயன்படுகிறது. Dictionary ஒரு மதிப்பை எடுத்துக் கொண்டு மற்றொரு மதிப்பை பார்த்துக் கொள்ள பயன்படுகிறது.

நினைவில் கொள்க:

- பைத்தான் நிரலாக்க மொழியில் நான்கு வகையான தரவினங்கள் உள்ளன. அவை List, Tuples, Set மற்றும் Dictionary ஆகும்.
- List "வரிசைமுறை தரவினம்" என்றும் அழைக்கப்படுகிறது. List-ல் உள்ள ஒவ்வொரு மதிப்பும் உறுப்பு என்றழைக்கப்படுகிறது.
- List உள்ள உறுப்புகள் சதுர அடைப்புக்குறிக்குள் அடைக்கப்பட்டிருக்க வேண்டும்.
- ஒவ்வொரு உறுப்பும் பூஜ்ஜியத்திலிருந்து தொடங்குகின்ற தனித்தன்மை வாய்ந்த சுட்டெண்ணை கொண்டுள்ளது.
- பைத்தான் நேர்மறை மற்றும் எதிர்மறை எண்களை சுட்டெண்ணாக அனுமதிக்கிறது.
- மடக்குகள், List இருந்து அனைத்து உறுப்புகளையும் அணுகப் பயன்படுகின்றன.
- "for" மடக்கு அனைத்து உறுப்புகளையும் ஒன்றன்பின் ஒன்றாக அணுகுவதற்கு பொருத்தமான மடக்காகும்.
- append( ), extend( ) மற்றும் insert( ) செயற்கூறுகள் List அதிக உறுப்புகளை சேர்க்கப் பயன்படுகின்றன.

நினைவில் கொள்க:

- del, remove( ) மற்றும் pop( ) List இருந்து உறுப்புகளை நீக்கப்பயன்படுகின்றன.
- Tuples காற்புள்ளியால் பிரிக்கப்பட்ட பல மதிப்புகளை வளைவு அடைப்புக்குறிக்குள் கொண்டுள்ளது.
- Tuples-ன் மடக்குச் செயல் List-ஐ காட்டிலும் விரைவானது.
- Tuples( ) செயற்கூறு List-ல் இருந்து Tuples-ஐ உருவாக்கவும் பயன்படுகிறது.
- ஒரு உறுப்புடன் Tuples-ஐ உருவாக்குவது “ஒற்றை” Tuples என அழைக்கப்படுகிறது.
- Dictionary என்பது பல்வேறு வகையான உறுப்புகளின் தொகுப்பாகும்.

செய்முறைப் பயிற்சி

1. List-ல் இருந்து நகர்த்துவதற்கான நிரலை எழுதுக.
2. Tuples-ல் இருந்து மிகப் பெரிய மதிப்பை அச்சிடும் நிரலை எழுதுக.
3. While மடக்கைப் பயன்படுத்தி Tuples-ல் உள்ள அனைத்து எண்களின் கூட்டுத்தொகை கண்டறியும் நிரலை எழுதுக.
4. List உள்ள அனைத்து இரட்டைப்படை எண்களின் கூட்டுத்தொகை கண்டறியும் நிரலை எழுதுக.
5. மடக்கைப் பயன்படுத்தி List-ஐ தலைகீழாக மாற்றுவதற்கான நிரலை எழுதுக.
6. List ஒரு மதிப்பை குறிப்பிட்ட இடத்தில் செருகுவதற்கான நிரலை எழுதுக.
7. 3 அல்லது 6 ஆல் வகுபடக்கூடிய, 1 முதல் 50 எண்களின் List-ஐ உருவாக்கும் நிரலை எழுதுக.
8. 1 முதல் 20 தொடர் எண்களைக் கொண்ட List-ஐ உருவாக்குவதற்கான நிரலை எழுதுக. பின்னர் 3 ஆல் வகுபடக்கூடிய அனைத்து எண்களையும் லிஸ்ட்ல் இருந்து நீக்குக.
9. List ஒரு மதிப்பு எத்தனை முறை தோன்றுகிறது என்பதை கணக்கிடும் நிரலை எழுதுக. மடக்கை பயன்படுத்தவும்.
10. Dictionary உள்ள அதிகபட்ச மற்றும் குறைந்தபட்ச மதிப்பை அச்சிடும் நிரலை எழுதுக.

மதிப்பீடு

Q1C8TB

பகுதி–அ

சரியான விடையை தேர்ந்தெடுத்து எழுதுக **(1 மதிப்பெண்)**

1. தரவினத் தொகுதியின் தொடர்பில்லாத ஒன்றைத் தேர்வு செய்க

   (a) List (b) Tuple (c) Dictionary (d) Loop

2. Let list1=[2,4,6,8,10], எனில் print(List1[-2]) ன் விடை

   (a) 10 (b) 8 (c) 4 (d) 6

3. பின்வரும் எந்த செயற்கூறு List-ல் உள்ள உறுப்புகளின் எண்ணிக்கையைக் கணக்கிட பயன்படுகிறது?

   (a) count() (b) find() (c)len() (d) index()

4. If List=[10,20,30,40,50] எனில் List[2]=35 ன் விடை

   (a) [35,10,20,30,40,50] (b) [10,20,30,40,50,35]

   (c) [10,20,35,40,50] (d) [10,35,30,40,50]

5. If List=[17,23,41,10] எனில் List.append(32) ன் விடை

   (a) [32,17,23,41,10] (b) [17,23,41,10,32]

   (c) [10,17,23,32,41] (d) [41,32,23,17,10]

6. பின்வரும் எந்த பைத்தான் செயற்கூறு ஏற்கனவே உள்ள List-ல் ஒன்றுக்கும் மேற்பட்ட உறுப்புகளை சேர்க்கப் பயன்படுகிறது?

   (a) append() (b) append_more() (c)extend() (d) more()

7. பின்வரும் பைத்தான் குறிமுறையின் விடை என்ன?

```
S=[x**2 for x in range(5)]

print(S)
```

   (a) [0,1,2,4,5] (b) [0,1,4,9,16] (c) [0,1,4,9,16,25] (d) [1,4,9,16,25]

8. பைத்தானில் type() செயற்கூறின் பயன் என்ன?

   (a) Tuple உருவாக்க

   (b) Tuple உள்ள உறுப்புகளின் வகையைக் கண்டறிய

   (c) பைத்தான் பொருளின் தரவினத்தை கண்டறிய

   (d) பட்டியலை உருவாக்க

9. பின்வரும் எந்த கூற்று சரியானது அல்ல?

(a) List மாற்றம் செய்யலாம்

(b) Tuples மாற்றம் செய்ய முடியாது.

(c) Append() செயற்கூறு, ஒரு உறுப்பை சேர்க்கப் பயன்படுகிறது.

(d) Extend() செயற்கூறு லிஸ்ட்ல் உறுப்புகளை சேர்க்க Tuples-ல் பயன்படுகிறது

10. SetA={3,6,9}, setB={1,3,9}. எனில், பின்வரும் நிரலின் வெளியீடு என்ன?

print(setA|setB)

(a) {3,6,9,1,3,9}(b) {3,9} (c) {1} (d) {1,3,6,9}

11. பின்வரும் எந்த set செயல்பாடு, இரண்டு set-களுக்கும் போதுவான உறுப்புகள் நீங்கலாக மற்ற அனைத்து உறுப்புகளையும் உள்ளடக்கியது?

(a) சமச்சீரான வேறுபாடு (b) வேறுபாடு

(c) வெட்டு (d) ஒட்டு

12. பைத்தான், Dictionary-ல் திறவுகோல்கள் எதனால் குறிப்பிடப்படுகின்றன

(a) = (b) ; (c)+ (d) :

பகுதி–ஆ

அனைத்து வினாக்களுக்கும் விடையளி **(2 மதிப்பெண்)**

1. பைத்தானில் List என்றால் என்ன?
2. List உறுப்புகளை பின்னோக்கு வரிசையில் தலைகீழாக எவ்வாறு அணுகுவாய்
3. பின்வரும் பைத்தான் குறிமுறையில் x ன் மதிப்பு என்ன?

List1=[2,4,6[1,3,5]]

x=len(List1)

4. List-ன் del மற்றும் remove( ) செயற்கூறின் வேறுபாடுகள் யாவை?
5. ஒரு Tuples n எண்ணிக்கை உறுப்புகளுடன் உருவாக்குவதற்கான தொடரியலை எழுதுக.
6. பைத்தானில் set என்றால என்ன?

பகுதி–இ

அனைத்து வினாக்களுக்கும் விடையளி **(3 மதிப்பெண்)**

1. List -ஐ விட மேலான Tuples-ன் நன்மைகளை எழுதுக
2. Sort( ) பற்றி சிறுகுறிப்பு எழுதுக.
3. பின்வரும் குறிமுறையின் வெளியீடு என்ன?

```
list = [2**x for x in range(5)]

print(list)
```

4. del மற்றும் clear( ) செயற்கூறுகளுக்கு இடையேயான வேறுபாடுகளை எடுத்துக்காட்டுடன் விளக்குக.
5. பைத்தானின் set செயல்பாடுகளை பட்டியலிடுக.
6. List மற்றும் Dictionary இடையேயான வேறுபாடுகள் யாவை?

பகுதி–ஈ

அனைத்து வினாக்களுக்கும் விடையளி **(5 மதிப்பெண்)**

1. List-ல் ஒரு உறுப்பை சேர்ப்பதற்கான பல்வேறு வழிகள் யாவை? பொருத்தமான எடுத்துக்காட்டுடன் விளக்குக.
2. range( )ன் நோக்கம் என்ன? எடுத்துக்காட்டுடன் விளக்குக.
3. பின்னலான Tuple என்றால் என்ன? எடுத்துக்காட்டுடன் விளக்குக.
4. பைத்தானிலுள்ள பல்வேறு set செயல்பாடுகளை பொருத்தமான எடுத்துக்காட்டுகளுடன் விளக்குக.

References

1. https://docs.python.org/3/tutorial/index.html
2. https://www.techbeamers.com/python-tutorial-step-by-step/#tutorial-list
3. Python programming using problem solving approach – Reema Thareja – Oxford University press.
4. Python Crash Course – Eric Matthes – No starch press, San Francisco.