பாடம் 9.

(LIST, TUPLES, SET மற்றும் DICTIONARY)

தொகுப்பு தரவினங்கள்

Prepared by,

J. KAVITHA, B.Sc,B.Ed,M.C.A,M.Phil.,

Computer Instructor Gr - I,

GHSS, S.S.KULAM,

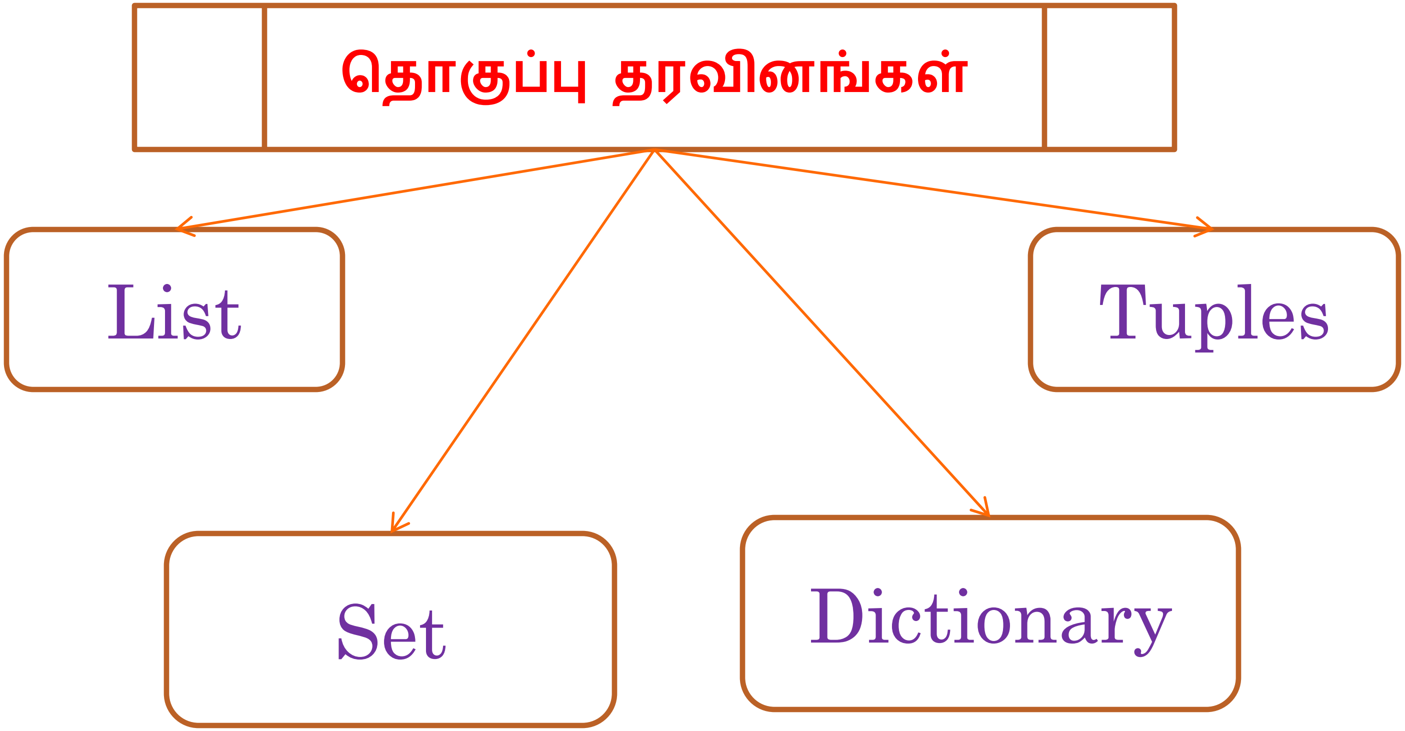
COIMBATORE – 641107.

கற்றலின் நோக்கங்கள்

- 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.

தொகுப்பு தரவினங்கள்

- பைத்தான் நிரலாக்க மொழியில், நான்கு வகையான தொகுப்பு தரவினங்கள் உள்ளன.



தொகுப்பு தரவினங்கள்

List

Tuples

Set

Dictionary

LIST - ஓர் அறிமுகம்

- பைத்தானில் உள்ள List, சரத்தைப் போன்றே “வரிசை முறை தரவினம்” ஆகும்.
- இது சதுர அடைப்புக் குறிக்குள் [ ] அடைக்கப்பட்ட மதிப்புகளின் வரிசைப்படுத்தப்பட்ட தொகுப்பாகும்.
- List-ல் உள்ள ஒவ்வொரு மதிப்பும், உறுப்பு (element) என்று அழைக்கப்படுகிறது.
- List-ல் உள்ள உறுப்புகள்,
  - எண்கள்,
  - எழுத்துகள்,
  - சரநிலையுருக்கள் மற்றும்
  - பின்னலான List போன்ற எந்த வகையாகவும் இருக்கலாம்.

பைத்தானில் LIST-ஐ உருவாக்குதல்:

- பைத்தானில், List சதுர அடைப்புக்குறியைப் பயன்படுத்தி எளிமையாக உருவாக்கப்படுகின்றன.

தொடரியல்:

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

எடுத்துக்காட்டு:

```
Marks = [10, 23, 41, 75]
Fruits = [“Apple”, “Orange”, “Mango”, “Banana”]
MyList = [ ]
MyList = [ “Welcome”, 3.14, 10, [2, 4, 6] ]
```

- List மற்றொரு List-டை உறுப்பாகக் கொண்டுள்ளது.
- இந்த வகை List “பின்னலான List (Nested List)” என்று அழைக்கப்படுகிறது.

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

LIST – ல் ஒரு உறுப்பை அணுகுதல்

- 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])
```

வெளியீடு: 10

LIST – ல் ஒரு உறுப்பை பின்னோக்கு வரிசையில் அணுகுதல்

- List-ல் உறுப்புகளை பின்னோக்கு வரிசையில் அணுகுவதற்கு எதிர்மறை சுட்டெண் பயன்படுகிறது.

எடுத்துக்காட்டு:
உறுப்புகளை பின்னோக்கு வரிசையில் அணுகுதல்

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

வெளியீடு:

```
75
```

List ன் அனைத்து உறுப்புகளையும் அணுகுதல்

- List-ல் 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
```

மடக்கு செயல்படும் விதம்

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	--	--

பின்னோக்கு சுட்டு (REVERSE INDEXING)

- பைத்தான், List உறுப்புகளுக்கு பின்னோக்கு அல்லது எதிர்மறை , சுட்டெண்களை வழங்குகிறது.
- பைத்தான் , List–ன் கடைசி உறுப்பிற்கு -1 முந்தைய உறுப்பிற்கு -2 என்ற சுட்டெண் மதிப்புகளை 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
```

மடக்கு செயல்படும் விதம்

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	--	--

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)
```

வெளியீடு: 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)
```

எடுத்துக்காட்டு:

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

வெளியீடு:

```
23
45
67
78
98
```

List உறுப்புகளை மாற்றம் செய்தல்

- பைத்தானில், List-ன் உறுப்புகள் மாறும் தன்மையுடையவை,.
- 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 சேர்க்கப்படாது.
- எடுத்துக்காட்டாக, தொடர் [0:5] என்று பொருத்தினால், பைத்தான் 0 முதல் 4 என்ற உறுப்பு சுட்டெண்களை மட்டுமே எடுத்துக் கொள்ளும்.

எடுத்துக்காட்டு

ஒற்றை மதிப்பை மாற்றுவதற்கான பைத்தான் நிரல்

```
MyList = [2, 4, 5, 8, 10]
print ("MyList elements before update... ")
for x in MyList:
      print (x, end =‘ ‘)
MyList[2] = 6
print (“\nMyList elements after updation... ")
for y in MyList:
      print (y, end = ‘ ‘)
```

வெளியீடு:

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

எடுத்துக்காட்டு

தொடர் மதிப்புகளை மாற்றுவதற்கான பைத்தான் நிரல்

```
MyList = [1, 3, 5, 7, 9]
print ("List Odd numbers... ")
for x in MyList:
   print (x, end=' ')
MyList [0:5] = 2,4,6,8,10
print ("\nList Even numbers... ")
for y in MyList:
   print (y, end = ' ')
```

வெளியீடு:

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

LIST-ல் உறுப்புகளை சேர்த்தல்

append() செயற்கூறு:

- பைத்தானில், ஏற்கனவே உள்ள list – ன் இறுதியில் ஒரு உறுப்பை சேர்ப்பதற்கு **append()** செயற்கூறு பயன்படுகிறது

தொடரியல்:

```
List.append(element to be added)
```

எடுத்துக்காட்டு

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

வெளியீடு:

```
[34, 45, 48, 90]
```

LIST-ல் உறுப்புகளை சேர்த்தல்

extend() செயற்கூறு :

- பைத்தானில், ஏற்கனவே உள்ள list – ன் இறுதியில் ஒன்றுக்கும் மேற்பட்ட உறுப்புகளை சேர்ப்பதற்கு extend() செயற்கூறு பயன்படுகிறது.

தொடரியல்:

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

எடுத்துக்காட்டு

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

வெளியீடு:

```
[34, 45, 48, 90, 71, 32, 29]
```

LIST-ல் உறுப்புகளை சேர்த்தல்

Insert() செயற்கூறு:

- பைத்தானில், ஏற்கனவே உள்ள list – ன் எந்தவொரு இடத்திலும் ஒரு உறுப்பை சேர்ப்பதற்கு Insert() செயற்கூறு பயன்படுகிறது.

தொடரியல்:

```
List.insert (position index, element)
```

எடுத்துக்காட்டு

```
>>> MyList=[34,98,47,'Kannan', 'Gowri', 'Lenin', 'Sreeni' ]
>>> MyList.insert(3, 'Raman')
>>> print(MyList)
```

வெளியீடு:

```
[34, 98, 47, 'Raman', 'Kannan', 'Gowri', 'Lenin', 'Sreeni']
```

List லிருந்து உறுப்புகளை நீக்குதல்

- List-ல் இருந்து ஒரு உறுப்பை நீக்குவதற்கு இரண்டு வழிகள் உள்ளன. அவை del கூற்று மற்றும் remove( ) செயற்கூறு ஆகும்.

del கூற்று:

- சுட்டெண் தெரிந்த உறுப்புகளை நீக்குவதற்கு del கூற்று பயன்படுகிறது.
- மேலும், முழு List-ஐ நீக்குவதற்கும் பயன்படுகிறது

தொடரியல்:	எடுத்துக்காட்டு

# ஒரு குறிப்பிட்ட உறுப்பை நீக்க 	
del List [index of an element]
# ஒன்றுக்கும் மேற்பட்ட உறுப்புகளை நீக்க 	
del List [index from : index to]
# ஒரு List-யை நீக்க 	
del List
	>>> MySubjects = ['Tamil', ‘English', ‘ 'Maths']
	>>> print (MySubjects)
	வெளியீடு: ['Tamil', ‘English', ‘ 'Maths']
	>>> del MySubjects[1]
	>>> print (MySubjects)
	வெளியீடு: ['Tamil', 'Maths']

LIST லிருந்து உறுப்புகளை நீக்குதல்

remove( ) செயற்கூறு:

- சுட்டெண் தெரியாத உறுப்புகளை List-லிருந்து நீக்குவதற்கு remove( ) செயற்கூறு பயன்படுகிறது.

தொடரியல்:

```
# ஒரு குறிப்பிட்ட உறுப்பை நீக்க
        List.remove(element)
```

எடுத்துக்காட்டு

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

LIST லிருந்து உறுப்புகளை நீக்குதல்

clear() செயற்கூறு:

- List - ன் அனைத்து உறுப்புகளையும் நீக்கி List – ஐ தொடர்ந்து வைத்திருக்க பயனபடுகிறது.

தொடரியல்:

```
List.clear( )
```

எடுத்துக்காட்டு

எ.கா:
```
>>> MySubjects = ['Tamil', 'English']
>>> MySubjects.clear()
>>> print (MySubjects)
```

வெளியீடு: [ ]

List லிருந்து உறுப்புகளை நீக்குதல்

pop( ) செயற்கூறு:

- குறிப்பிட்ட உறுப்பை அதன் சுட்டெண்ணை பயன்படுத்தி நீக்குவதற்கு pop( ) செயற்கூறு பயன்படுகிறது.
- உறுப்பு நீக்கப்பட்டவுடன் pop( ) செயற்கூறு நீக்கப்பட்ட உறுப்பை காண்பிக்கிறது.
- pop( ) செயற்கூறு லிஸ்ட்ல் இருந்து ஒரு உறுப்பை மட்டும் நீக்கப்பயன்படுகிறது.

தொடரியல்:

```
List.pop(index of an element)
```

எடுத்துக்காட்டு:

```
>>> MyList=[12,89,34,'Kannan', 'Gowrisankar', 'Lenin']
>>> MyList.pop(1)
      89
>>> print(MyList)
வெளியீடு: [12,34, 'Kannan', 'Gowrisankar', 'Lenin']
```

LIST மற்றும் RANGE ( ) செயற்கூறுகள்

- Range() செயற்கூறு பைத்தானில் தொடர் மதிப்புகளை உருவாக்க பயன்படுகிறது. Range() செயற்கூறு மூன்று செயலுருபுகளைக் கொண்டுள்ளது.

தொடரியல்:
range (start value, end value, step value)

- start value – தொடரின் தொடக்க மதிப்பு.
- end value – தொடரின் உச்ச வரம்பு.
- step value – இது ஒரு விருப்ப செயலுருபு , இது வெவ்வேறு இடைவெளிகளில் மதிப்புகளை உருவாக்கப் பயன்படுகிறது.

எடுத்துக்காட்டு

 # 10 வரை உள்ள முழு எண்களை உருவாக்குதல்:

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

வெளியீடு: 1 2 3 4 5 6 7 8 9 10

 # முதல் 10 இரட்டைப்படை எண்களை உருவாக்குதல்

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

வெளியீடு:   2 4 6 8 10

தொடர் மதிப்புகளுடன் லிஸ்ட்-ஐ உருவாக்குதல்

- list()செயற்கூறு பைத்தானில் லிஸ்ட் உருவாக்கப் பயன்படுகிறது.
- List( )செயற்கூறு range( ) ன் விடையை லிஸ்ட் ஆக உருவாக்குகிறது.

தொடரியல்:

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

எடுத்துக்காட்டு:

```
>>> Even_List = List(range(2,11,2))
>>> print(Even_List)
```

வெளியீடு: [2, 4, 6, 8, 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:

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]
```

பிற முக்கியமான LIST செயற்கூறுகள்

தொடரியல்	விளக்கம்	எடுத்துக்காட்டு
List.copy( )	List-ன் நகலை தரும்.	MyList=[12, 12, 36] x = MyList.copy() print(x) வெளியீடு: [12, 12, 36]
List.count(value)	**List-ல் உள்ள ஒரே மாதிரியான உறுப்புகளின் எண்ணிக்கையை தரும்.**	**MyList=[36 ,12 ,12] x = MyList.count(12) print(x) வெளியீடு: 2**
List.reverse( )	**List-ல் உள்ள உறுப்புகளின் வரிசையை தலைகீழாக திருப்புகிறது.**	**MyList=[36 ,23 ,12] MyList.reverse() print(MyList) வெளியீடு: [12 ,23 ,36]**

பிற முக்கியமான LIST செயற்கூறுகள்

Sort() செயற்கூறு:

- List - ல் உள்ள உறுப்புகளை வரிசையாக்கம் செய்ய பயன்படுகிறது.
- **Reverse – ஐ True என பொருத்தினால் இறங்கு வரிசையில் List வரிசையாக்கம் செய்யும். ஏறுவரிசை தானமைவு வரிசையாக்கம் ஆகும்.**

தொடரியல்

```
List.sort(reverse=True/False,key=myFunc)
```

எடுத்துக்காட்டு

```
MyList=['Thilothamma', 'Tharani', 'Anitha', 'SaiSree','Lavanya']
MyList.sort( )
print(MyList)
MyList.sort(reverse=True)
print(MyList)
```

வெளியீடு:

```
['Anitha', 'Lavanya', 'SaiSree', 'Tharani’]
['Tharani', 'SaiSree', 'Lavanya','Anitha']
```

பிற முக்கியமான List செயற்கூறுகள்

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

TUPLES அறிமுகம்:

- Tuples காற்புள்ளியால் பிரிக்கப்பட்ட பல மதிப்புகளை வளைவு அடைப்புக் குறிக்குள் கொண்ட தரவினமாகும். இது List-க்கு இணையானதாகும்.

List மற்றும் ஒப்பீடு:

List	Tuples
List-ன் உறுப்புகளை மாற்றலாம்.	ஆனால் Tuples-ன் உறுப்புகளை மாற்ற முடியாது.
List-ன் உறுப்புகள் சதுர அடைப்புக் குறிக்குள் அடைக்கப்பட்டிருக்கும்.	ஆனால், Tuple -ன் உறுப்புகள் வளைவு குறிக்குள் அடைக்கப்பட்டிருக்கும். Tuples-ன் மடக்குச் செயல் List-ஐ காட்டிலும் விரை வானது.

TUPLES உருவாக்குதல்

- **List-ஐ போன்றே Tuples உருவாக்கப்படும்.**
- **டப்புள்ஸ்ல் வளைந்த அடைப்புக் குறிக்குள் 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)
```

வெளியீடு: **(23, 56, 89, 'A', 'E', 'I', 'Tamil')**

செயற்கூறை பயன்படுத்தி TUPLES உருவாக்குதல்

- Tuples ( ) செயற்கூறு லிஸ்ட்டிலிருந்து Tuples- ஐ உருவாக்க 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)
வெளியீடு:      (23, 45, 90)
>>> type (MyTup3)
வெளியீடு:    <class 'Tuples'>
```

- பைத்தானில் ஒரு பொருளின் தரவினத்தை அறிய type() செயற்கூறு பயன்படுகிறது

பின்னலான Tuples

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

எடுத்துக்காட்டு:

```
Toppers = (("Vino", “XII-F", 98), ("Sara", “XII-H", 97),(“Rani", “XII-F", 95))
for i in Toppers:
        print(i)
```

வெளியீடு:

```
('Vino', 'XII-F', 98)
(‘Sara', 'XII-H', 97)
(‘Rani', 'XII-F', 95)
```

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.

SET உருவாக்குதல்

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

தொடரியல்:

Set_Variable = {E1, E2, E3 ….. En}

எடுத்துக்காட்டு:

```
>>> S1={1,2,3,'A',3.14}
>>> print(S1)
வெளியீடு:    {1, 2, 3, 3.14, 'A'}
>>> S2={1,2,2,'A',3.14}
>>> print(S2)
வெளியீடு:    {1, 2, 'A', 3.14}
```

- Set S1 என்பது நகல்கள் இல்லாத பல்வேறு வகை உறுப்புகளுடன் உருவாக்கப்பட்டுள்ளது.
- Set S2 என்பது நகல் மதிப்புகளுடன் உருவாக்கப்பட்டுள்ளது. ஆனால் பைத்தான், நகல் மதிப்புகளிலிருந்து ஒரேயொரு உறுப்பை மட்டுமே எடுத்துக் கொள்ளும்.

Set செயல்பாடுகள்

சேர்ப்பு:

- இரண்டு அல்லது அதற்கு மேற்பட்ட set களின் அனைத்து உறுப்புகளையும் உள்ளடக்கும்.
- பைத்தானில், |, Union செயற்கூறுகள் set சேர்ப்புக்காக பயன்படுத்தப்படுகிறது.

எடுத்துக்காட்டு:

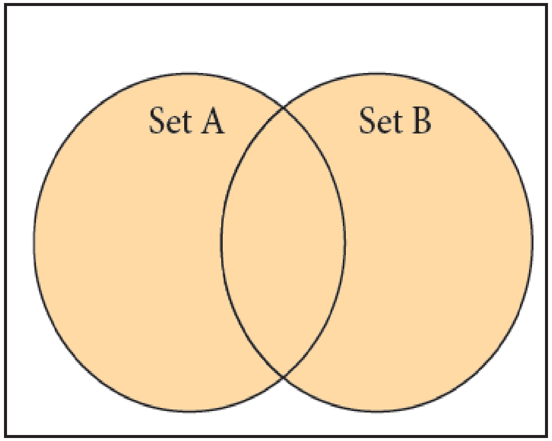
```
set_A = {2,4,6,8}

set_B = {'A', 'B', 'C', 'D'}

U_set = set_A | set_B

print(U_set)
```

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



Set செயல்பாடுகள்

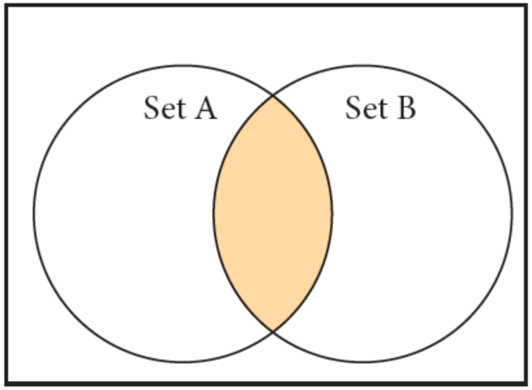
வெட்டு:

- இது இரண்டு set களின் பொதுவான உறுப்புகளை உள்ளடக்கியது.
- பைத்தானில், &, intersection செயற்கூறுகள் set களை வெட்டுவதற்கு பயன்படுகிறது.

எடுத்துக்காட்டு:

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

வெளியீடு: {'A', 'D'}



Set செயல்பாடுகள்

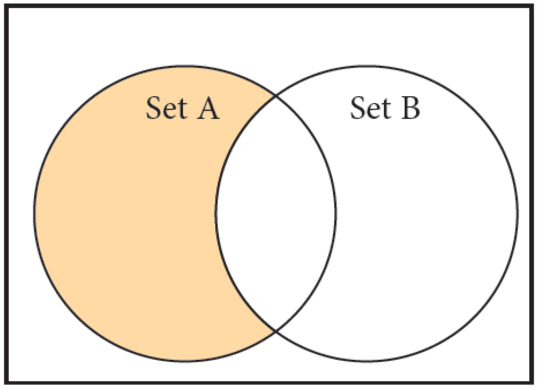
வேறுபாடு:

- இது முதல் set–ல் உள்ள உறுப்புகள் இரண்டாவது set–ல் இருந்தால் அதை தவிர்த்து மற்ற உறுப்புகளை உள்ளடக்கியது.
- பைத்தானில், -(minus), Difference செயற்கூறுகள் வேறுபாடு செயற்கூறாக பயன்படுகிறது.

எடுத்துக்காட்டு:

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

வெளியீடு: {2, 4}



Set செயல்பாடுகள்

சமச்சீரான வேறுபாடு:

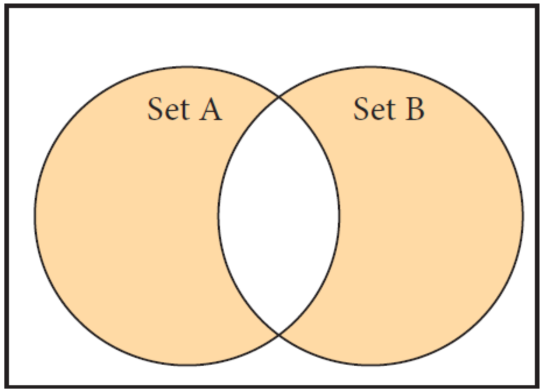
- இது இரண்டு set களில் உள்ள பொதுவான உறுப்புகளை மட்டும் தவிர்த்து மற்ற அனைத்து உறுப்புகளையும் உள்ளடக்கியது.
- பைத்தானில், ^(carret), Symmetric difference செயற்கூறுகள் சமச்சீரான வேறுபாடு கண்டறிய பயன்படுகிறது

எடுத்துக்காட்டு:

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

வெளியீடு:

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



DICTIONARIES

- பிற தரவினங்களாகிய List அல்லது Tuples போன்று இல்லாமல், Dictionary வகை உறுப்புகளுடன் அதற்கான திறவுகோலையும் (இணைப்புப் பெயர்) சேமிக்கிறது.
- பைத்தான் Dictionary உள்ள திறவுகோல்கள் முக்காற் புள்ளியாலும் (:) உறுப்புகள் காற்புள்ளியாலும் (,) பிரிக்கப்பட வேண்டும். திறவுகோல் மற்றும் மதிப்புகள் நெளிவு அடைப்புக்குறிக்குள் { } வரையறுக்கப்படும்.

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

DICTIONARIES - ஐ உருவாக்குதல்

- காலியான வெற்று Dictionary :

```
Dict1 = { }
```

- திறவுகோலைக்கொண்ட Dictionary :

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

DICTIONARY சுருக்கம்

- பைத்தானில், சுருக்கம் என்பது Dictionary ஐ உருவாக்குவதற்கான மற்றொரு வழியாகும்.
- இந்தவகை Dictionary ஐ உருவாக்குவதற்கான தொடரியல்:

தொடரியல்:

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

எடுத்துக்காட்டு

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

வெளியீடு:

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

List மற்றும் Dictionary இடையேயான வேறுபாடுகள்

List	Dictionary
List என்பது வரிசைபடுத்தப்பட்ட உறுப்புகளின் தொகுப்பாகும்.	Dictionary ஒரு உறுப்பை மற்றொரு உறுப்புடன் பொருத்த பயன்படும் தரவு அமைப்பாகும்.
List - ன் சுட்டெண் குறிப்பிட்ட உறுப்பை அணுகுவதற்கு பயன்படுகிறது.	Dictionary - ல் திறவுகோல் சுட்டெண்ணை குறிக்கிறது.
List - ன் மதிப்பை பார்த்துக் கொள்ள பயன்படுகிறது.	Dictionary - ல் ஒரு மதிப்பை எடுத்துக்கொண்டு மற்றொரு மதிப்பை பார்த்துக்கொள்ள பயன்படுகிறது.

மதிப்பீடு

1. தரவினத் தொகுதியின் தொடர்பில்லாத ஒன்றைத் தேர்வு செய்க.

   அ) list ஆ) tuple இ) dictionary ஈ) Loop

   Loop

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

   அ) 10 ஆ) 8 இ) 4 ஈ) 6

   8

3. பின்வரும் எந்த செயற்கூறு list – ல் உள்ள உறுப்புகளின் எண்ணிக்கையைக் கணக்கிட பயன்படுகிறது?

   அ) count( ) ஆ) find( ) இ) len( ) ஈ) index( )

   len( )

4. பின்வரும் எந்த பைத்தான் செயற்கூறு ஏற்கனவே உள்ள list-ல் ஒன்றுக்கும் மேற்பட்ட உறுப்புகளை சேர்க்கப் பயன்படுகிறது?

   அ) Append( ) ஆ) append_more( ) இ) extend( ) ஈ) more( )

   extend( )

மதிப்பீடு

5. பின்வரும் பைத்தான் குறிமுறையின் விடை என்ன?

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

அ) [0,1,2,4,5]
ஆ) [0,1,4,9,16]
இ) [0,1,4,9,16,25]
ஈ)[1,4,9,16,25]

[0,1,4,9,16]

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

அ) [35,10,20,30,40,50]
ஆ) [10,20,30,40,50,35]
இ) [10,20,35,40,50]
ஈ) [10,35,30,40,50]

[10,20,35,40,50]

7. if list=[17,23,41,10] எனில் list.append (32) ன் விடை

அ) [32,17,23,41,10]
ஆ) [17,23,41,10,32]
இ) [10,17,23,32,41]
ஈ) [41,32,23,17,10]

[17,23,41,10,32]

மதிப்பீடு

8. பைத்தானில் type( ) செயற்கூறின் பயன் என்ன?

அ) tuple உருவாக்க ஆ) பட்டியலை உருவாக்க

இ) பைத்தான் பொருளின் தரவினத்தை கண்டறிய

ஈ) tuple உள்ள உறுப்புகளின் வகையைக் கண்டறிய

பைத்தான் பொருளின் தரவினத்தை கண்டறிய

9. பின்வரும் எந்த கூற்று சரியானது?

அ) list மாற்றம் செய்யலாம் ஆ) tuples மாற்றம் செய்யலாம்

இ) append( ) செயற்கூறு, ஒரு உறுப்பை சேர்க்கப் பயன்படுகிறது

ஈ) extend() செயற்கூறு list-ல் உறுப்புகளை சேர்க்க tuples–ல் பயன்படுகிறது.

extend() செயற்கூறு list-ல் உறுப்புகளை சேர்க்க tuples–ல் பயன்படுகிறது.

10. Set A={3,6,9}, set B={1,3,9} எனில், பின்வரும் நிரலின் வெளியீடு என்ன?

Print(set A|set B)

அ) {3,6,9,1,3,9} ஆ) {3,9} இ) {1} ஈ) {1,3,6,9}

{1,3,6,9}

மதிப்பீடு

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

அ) சமச்சீரான வேறுபாடு ஆ) வேறுபாடு இ) வெட்டு ஈ) சேர்ப்பு

சமச்சீரான வேறுபாடு

12. பைத்தான், dictionary- ல் திறவுகோல்கள் எதனால் குறிப்பிடப்படுகின்றன.

அ) = ஆ) ; இ) + ஈ) :

:

13. பின்வரும் பைத்தான் குறிமுறையில் x - ன் மதிப்பு என்ன?

```
List1 = [2,4,6,[1,3,5]]
x = len(List1)
```

x - ன் மதிப்பு – 4

14. பின்வரும் குறிமுறையின் வெளியீடு என்ன?

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

வெளியீடு: [1,2,4,8,16]

முக்கிய வினாக்கள்

1. பைத்தானில் List என்றால் என்ன?
2. List உறுப்புகளை பின்னோக்கு வரிசையில் தலைகீழாக எவ்வாறு அணுகுவாய்?
3. List – ன் del மற்றும் remove() செயற்கூறின் வேறுபாடுகள் யாவை?
4. பைத்தானில் set என்றால் என்ன?
5. List மற்றும் Tuples – ஒப்பிடுக.
6. பைத்தானின் set செயல்பாடுகளை பட்டியலிடுக.
7. List மற்றும் Dictionary இடையேயான வேறுபாடுகள் யாவை?
8. List – ல் ஒரு உறுப்பை சேர்ப்பதற்கான பல்வேறு வழிகள் யாவை? பொருத்தமான எடுத்துக்காட்டுடன் விளக்குக.
9. Range() ன் நோக்கம் என்ன? எடுத்துக்காட்டுடன் விளக்குக.
10. பின்னலான Tuples என்றால் என்ன? எடுத்துக்காட்டுடன் விளக்குக.

நன்றி!

கல்வி என்பது கடல்.
அதை கற்றுக் கொடுப்பது
தொழில் அல்ல தவம்.
விருப்பம் பல கொண்டு
விரைவுடன் நீ படித்தால்
வாழ்வில் மேன்மை
பெறலாம்..

வாழ்த்துக்கள்.

ஜெ. கவிதா B.Sc, B.Ed, M.C.A, M.Phil.,
கணினி பயிற்றுநர் நிலை - I
அரசு மேல்நிலைப்பள்ளி,
சர்க்கார்சாமக்குளம்,
கோயம்புத்தூர் – 641107.