Sequence types - Lists, Strings, Tuples, Sets, Dicts

September 5, 2022

Logistics

Assignment 1 due on today(Sept 5, 2022) at 23:59

Assignment 2 and term projects will be released before the midsem, so you can workout during autumn break

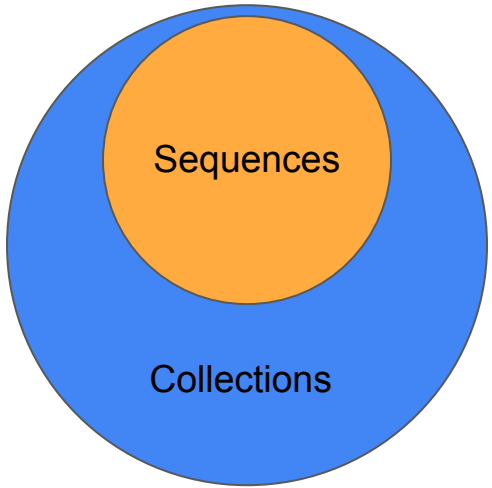
Collections & Sequences

Collection : Collections are Python's container data types

Ex : Lists, Tuples, Dicts, Sets, Strings

Sequence : A subset of Python collections which have positional ordering

Ex : Lists, Tuples, Strings



Sequences in Python

Sequence : Positionally ordered collection of items

Sequence data types : Lists, Tuples, Strings

List Sequence type

Examples of lists :

1. [1, 2, 3]
2. [True, 0, 1, “Hello”]
3. [[1, 2, 3], [1, 2, 3]]

Creating a list : [ ] or list()

```
l = [1, 2, 3, 4, 5] #list with 5 items
```

Accessing through indexing:

```
l = [1, 2, 3, 4, 5]

  0  1  2  3  4      #Normal/Forward indexing

 -5 -4 -3 -2 -1      #Reverese/Backward
```

Accessing elements & slicing :

Accessing individual elements

l[0] → 1

l[4] → 5

l[-1] → 5

l[-5] → 1

Slicing a list

Syntax : listvar[start : stop : step]

l[0:5:1] → [1, 2, 3, 4, 5]

l[: : 2] → [1, 3, 5]

Finding the length of list :

Keyword : len # returns the length of the sequence

len(l) → 5

len(city[: : 2]) → 3

List Sequence type

Mutability: Able to modify existing elements

l = [1, 2, 3, 4, 5] l[2] = 10 → l = [1, 2, 10, 4, 5]

Methods :

l = [1, 2, 3, 4, 5]

Adding a new element at the end

l.append(6) → [1, 2, 3, 4, 5, 6]

Remove an element from the end

l.pop() → [1, 2, 3, 4, 5]

Insert element at an index

l.insert(5, 6) → [1, 2, 3, 4, 5, 6]

Remove element at an index

l.pop(5) → [1, 2, 3, 4, 5]

Extending a list

l.extend([6, 7, 8]) → [1, 2, 3, 4, 5, 6, 7, 8]

Sorting a list

l.sort() → inplace sorting in ascending order

l.sort(reverse=True) → inplace descending order

Reverse a list

l.reverse() → [8, 7, 6, 5, 4, 3, 2, 1]

Others

count(), index(), copy(), clear()

Nested lists

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

l[0][0] → 1

l[-1][-1] → 6

len(l) → 2

List comprehensions

l = [i for i in range(5)] → [0, 1, 2, 3, 4]

l = [x for x in range(5) if x%2==0] → ?

Lists important properties

1. Brackets → ?
2. Ordering → ?
3. Indexing → ?
4. Mutability → ?
5. Duplication → ?

String Sequence type

Examples :

city = “Kharagpur” #type of str

Or anything in *single/double/triple* quotes

Accessing string chars :

city[0] → “K”

city[-1] → “r”

city[: : 2] → “Kaapr”

city[: : -1] → “rupgarahK”

Finding length of the string:

Keyword : len # returns the length of the sequence

len(city) → 9

len(city[: : 2]) → 5

String Sequence type

Mutability : Strings are immutable (can’t modify)

Methods : All string methods are outplace

p = ' Kharagpur 2022 '

Casing a string

p.upper() → ' KHARAGPUR 2022 '

p.lower() → ' kharagpur 2022 '

What's in the string?

p.isalpha() → False

p.isnumeric() → False

p.isalnum() → True

Replace substrings

p.replace('2022', '2023') → ' Kharagpur 2023 '

Find substrings → returns starting index if found else -1

p.find('2022') → 11

Splitting a string

p.split(' ') → ['', 'Kharagpur', '2022', '']

Combining strings

'--'.join(['This', 'is', 'class']) → 'This--is--class'

Concatenating strings

'Hello, ' + 'There' → 'Hello, There'

Stripping spaces in strings

p.strip() → 'Kharagpur 2022'

Strings important properties

1. Brackets → ?
2. Ordering → ?
3. Indexing → ?
4. Mutability → ?
5. Duplication → ?

Tuple sequence type

Imp : Same as lists but immutable, and don't have class methods, parentheses

Examples of tuples:

1. (1, 2, 3)
2. (True, 0, 1, "Hello")
3. ((1, 2, 3), (1, 2, 3))

Creating a tuple: () or tuple()

t = (1, 2, 3) #tuple with 3 items

Accessing through indexing, slicing :

Same as lists

Packing & Unpacking:

a, b, c = t → a = 1, b = 2, c = 3 # unpacking

first, *second = t → first = 1, second = (2, 3)

t = (a, b, c) # packing

Aggregate/Summary methods:

sum(t) → 6

min(t) → 1

max(t) → 3

len(t) → 3

Ordering Methods:

sorted(t, reverse=True) → (3, 2, 1)

Tuple comprehensions:

funstr = 'Coding is Not Fun'

t = (c for c in funstr if c.isupper()) → ?

t = tuple(c for c in funstr if c.isupper()) → ?

Pros :

Fast read access, low on memory

Tuple important properties

1. Brackets → ?
2. Ordering → ?
3. Indexing → ?
4. Mutability → ?
5. Duplication → ?

Dicts

Imp : key-value stores, helpful while working with pairs, curly brackets, immutable keys

Examples

d = {'class_code': 'CS60013', 'student_count': 15}

d = dict([('class_code', 'CS60013'), ('student_count', 15)])

Creation

Using { } or dict()

Accessing with keys

d['class_code'] → 'CS60013'

d['student_count'] → 15

d['student_names'] → KeyError: 'student_names'

Modifying values

D['student_count'] += 1 → 16

Methods

d.get('student_count') → 15

d.items() → [('class_code', 'CS60013'), ('student_count', 15)]

d.keys() → ['class_code', 'student_count']

d.values() → ['CS60013', 15]

d.pop('student_count') → {'class_code': 'CS60013'}

d.update({'instructor' : 'SM'}) → {'class_code': 'CS60013', 'instructor': 'SM'}

d.setdefault('dept', 'SMST') → {'class_code': 'CS60013', 'instructor': 'SM', 'dept': 'SMST'}

Dict comprehensions

d = {x : x*x for x in range(1, 5)} → ?

Pros

Fast read access, pairs

Dict important properties

1. Brackets → ?
2. Ordering → ?
3. Indexing → ?
4. Mutability → ?
5. Duplication → ?

Sets

Imp : unordered, immutable, unique collection

Examples

s1 = set([1, False])

s2 = {1, 'Hi', False, (1,2)}

Remember

Can’t accommodate unhashable items (lists, dicts, sets)

Duplicate items not allowed

No indexing

Methods

Adding & Removing

s2.add(True) → {1, 'Hi', False, (1,2)} ???

s2.remove(False) → {1, 'Hi', (1,2)}

Set operations

s1.union(s2) → {False, 1, (1, 2), 'Hi'}

s1.intersection(s2) → {1}

s2.difference(s1) → {(1, 2), 'Hi'}

s1.issubset(s2) → ?

s2.issuperset(s1) → ?

s1.isdisjoint(s2) → ?

Pros

Existence checking, math venn operations, uniqueness

Set important properties

1. Brackets → ?
2. Ordering → ?
3. Indexing → ?
4. Mutability → ?
5. Duplication → ?