Introduction to Python
Part 2

COMP 8347

Usama Mir

Usama.Mir@unwindsor.ca

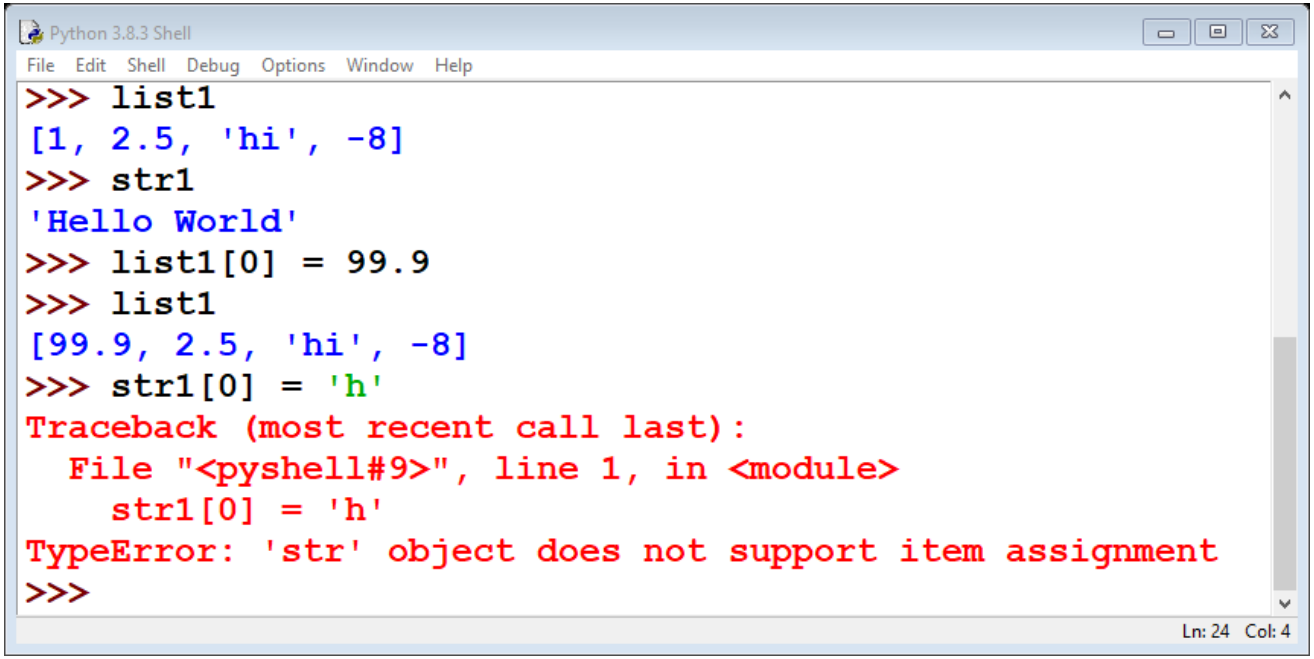
Python Basics

Topics

- Collection Data Types
  - Strings
  - Lists
  - Dicts
- Comparison/Logic Operations

Collection Data Types

- Holds a collection of items, which may or may not be of the same type.
- May be *mutable* (e.g. list, dict) or *immutable* (e.g. tuple, str)
- Lists and strings are examples of *sequence* data types.
- There is another sequence type known as *tuple*
- For more on data types:
  - https://docs.python.org/3/tutorial/datastructures.html



Mutable vs Immutable

Strings

- A collection data type that is ***ordered*** and ***unchangeable*** (***immutable***).
- The string type in Python is called **str**
- String literals delimited by single or double quotes; e.g. 'hello' or "hello"
- Access individual items using the index number and enclosing in square brackets, e.g. str1[0]
- The first element always has index 0

Strings

- Use index([]) and ‘slice’ similar to lists
  - S1 = “A red car ”
  - >>> S1[4]

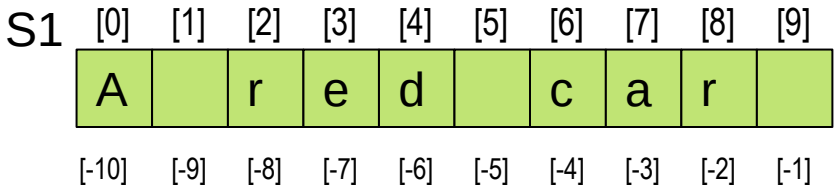
    ‘d’
  - >>> S1[:7]

    ‘A red c’
  - >>> S1[2:-6]

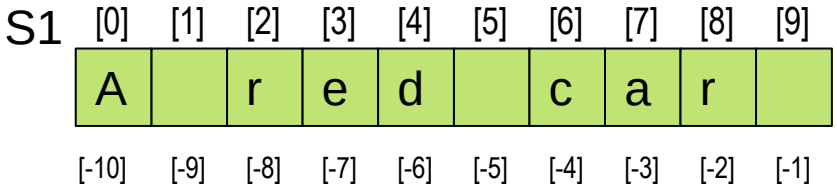
    ‘re’
  - >>> S1[1] = ‘*’

    TypeError: 'str' object does not support item assignment
  - >>> S1 = ‘something else’

    # This is ok



String Methods



- S1.count('r')

2

- S1.split()

['A', 'red', 'car']

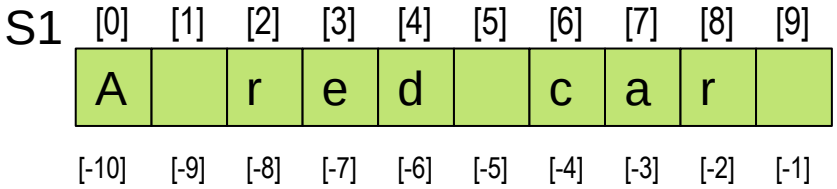
- S1.replace('r', '*')

"A *ed ca* "

- S1.upper()

"A RED CAR "

String Methods



- S1.index('r')

2

- S1.index('x')

ValueError

- S1.find('r')

2

- S1.find('x')

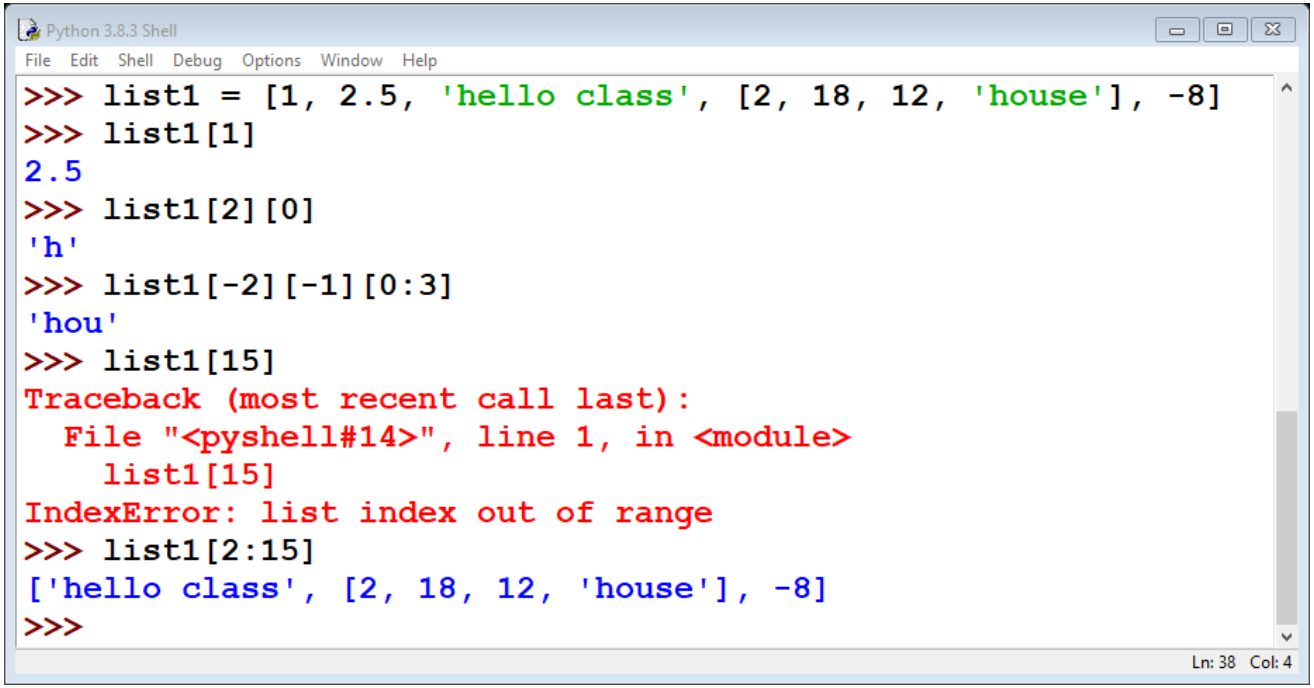
-1

- S1.startswith('A red')

True

Lists

- A collection data type that is ordered and changeable (mutable).
  - e.g. [1], [3, ‘hi’, 4.5, [1,2,3]], []
- Use [] to index items; similar to strings
- Can have multiple indices, e.g. list1[2][0], list1[-2][-1][-3]



Lists (Examples)

List Methods

L = [1,2,3,[4,5,'a','b'], 'hello', '6', 7]

- L.append([8,9])

[[1, 2, 3, [4, 5, 'a', 'b'], 'hello', '6', 7, [8, 9]]

- L += [8,9]

[1, 2, 3, [4, 5, 'a', 'b'], 'hello', '6', 7, 8, 9]

- L.index(3)

2

range(n)

- *range(n)*: produces sequence 0, 1, 2, … *n*-1
- range([*i*,]*stop*[, *k*]): sequence starts at *i* (instead of 0) and incremented by *k* (instead of 1)
- range(5) → produces sequence 0, 1, 2, 3, 4
- range(3, 20, 4) → produces 3, 7, 11, 15, 19

```
for i in range(5):
    for j in range(10):
        print(j)
    print(i)
print('done')
```

Dictionaries

- *dict*: an ***unordered*** collection of key-value pairs
  - ***mutable***
  - unordered → no notion of index positions
  - Similar to "hash maps/associative arrays" in other programming languages
  - Key is the unique identifier and value is the data
  - Example:
    - student = {'name':'john', 'age':'25', 'courses':['8347','8117']}

Dictionaries

```
student = {'name':'john', 'age':'25', 'courses':['8347','8117']}

>>>student['name']
'john'

>>>student['25']
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    student['25']
KeyError: '25'

>>>student['marks']= 97
>>>print(student)
{'name': 'john', 'age': '25', 'courses': ['8347', '8117'], 'marks': 97}
```

Dictionaries

```
>>>del student['age']
>>>print(student)
{'name': 'john', 'courses': ['8347', '8117'], 'marks': 97}

>>>student['name'] = 'Usama'
>>>print(student)
{'name': 'Usama', 'courses': ['8347', '8117'], 'marks': 97}

>>>print(student.keys())
#shows all the keys
>>>print(student.values())
#shows all the values
```

And so on….

Membership Operator

- *in*: tests for membership, returns True or False
- *not in*: tests for non-membership, returns True or False
- L = [1, [2,3], “ab”, -23] S = “Python is great!”
  - [2,3] in L

  True
  - 2 in L

  False
  - 2 not in L

  True
  - “on is” in S

  True
  - “eat” not in S

  False

Comparison Operators

- Basic comparison operators: `<, <=, ==, !=, >=, >`
  - `a = 4,` b=12
  - `a < b` → True
  - `a == b` → False
  - `a >= b, a != b, a <= b` → (False, True, True)
- Can be chained
  - `2 <= a < b <= 20` → True

Logical Operators

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

References

- Programming in Python 3 A complete introduction to the python language (2nd Ed) by Mark Summerfield. Addison Wesley 2010.
- https://www.w3schools.com/python/python_operators.asp
- https://www.youtube.com/watch?v=daefaLgNkw0
- https://www.scaler.com/topics/sequence-data-type-in-python/
- https://www.tutorialspoint.com/What-is-a-sequence-data-type-in-Python
- https://flexiple.com/python/comparison-operators-in-python/
- https://www.w3schools.com/python/python_dictionaries.asp
- Slides from Dr. Arunita and Dr. Saja