python™

Week 7

Lists

Special thanks to Roy McElmurry, John Kurkowski, Scott Shawcroft, Ryan Tucker, Paul Beck for their work.

Lists

- **list**: Python's equivalent to Java's array (but cooler)
  - Declaring:

    name = [**value**, **value**, ..., **value**] or,

    name = [**value**] * **length**

  - Accessing/modifying elements: (same as Java)

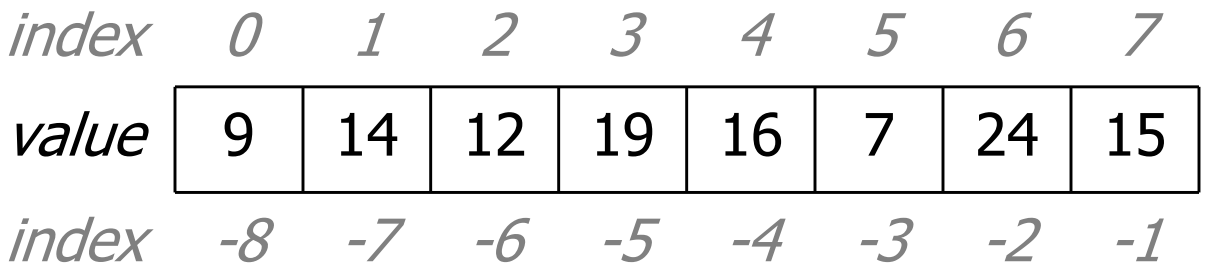
    name[**index**] = **value**

```
>>> scores = [9, 14, 18, 19, 16]
[9, 14, 18, 19, 16]
>>> counts = [0] * 4
[0, 0, 0, 0]
>>> scores[0] + scores[4]
25
```

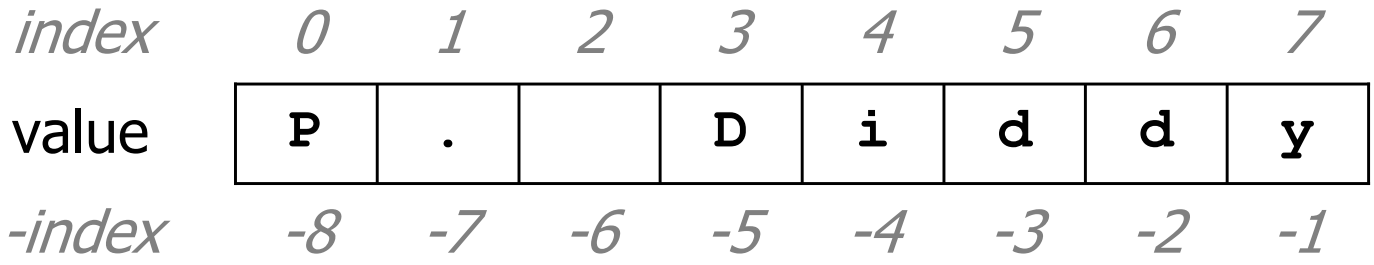
Indexing

- Lists can be indexed using positive or negative numbers:

```
>>> scores = [9, 14, 12, 19, 16, 7, 24, 15]
[9, 14, 12, 19, 16, 7, 24, 15]
>>> scores[3]
19
>>> scores[-3]
7
```



Recall: Strings



- Accessing character(s):

  variable [**index**]

  variable [**index1** : **index2**]

  - **index2** exclusive
  - **index1** or **index2** can be omitted (goes to end of string)

```
>>> name = "P. Diddy"
>>> name[0]
'P'
>>> name[7]
'y'
>>> name[-1]
'y'
>>> name[3:6]
'Did'
>>> name[3:]
'Diddy'
>>> name[:-2]
'P. Did'
```

python™

Slicing

- **slice**: A sub-list created by specifying start/end indexes

```
name[start:end]          # end is exclusive
name[start:]             # to end of list
name[:end]               # from start of list
name[start:end:step]     # every step'th value
```

```
>>> scores = [9, 14, 12, 19, 16, 18, 24, 15]
>>> scores[2:5]
[12, 19, 16]
>>> scores[3:]
[19, 16, 18, 24, 15]
>>> scores[:3]
[9, 14, 12]
>>> scores[-3:]
[18, 24, 15]
```

index	0	1	2	3	4	5	6	7
value	9	14	12	19	16	18	24	15
index	-8	-7	-6	-5	-4	-3	-2	-1

Other List Abilities

- Lists can be printed (or converted to string with `str()`).
- Find out a list's length by passing it to the `len` function.
- Loop over the elements of a list using a `for ... in` loop.

```
>>> scores = [9, 14, 18, 19]
>>> print("My scores are", scores)
My scores are [9, 14, 18, 19]
>>> len(scores)
4
>>> total = 0
>>> for score in scores:
...     print("next score:", score)
...     total += score
next score: 9
next score: 14
next score: 18
next score: 19
>>> total
60
```

Ranges, Strings, and Lists

- The `range` function returns a list.

```
>>> nums = range(5)
>>> nums
[0, 1, 2, 3, 4]
>>> nums[-2:]
[3, 4]
>>> len(nums)
5
```

- Strings behave like lists of characters:
  - `len`
  - indexing and slicing
  - `for` ... `in` loops

String Splitting

- `split` breaks a string into a list of tokens.

  name`.split()` `# break by whitespace`

  name`.split(`**delimiter**`)` `# break by delimiter`

- `join` performs the opposite of a `split`

  delimiter`.join(`**list**`)`

```
>>> name = "Brave Sir Robin"
>>> name[-5:]
'Robin'
>>> tokens = name.split()
['Brave', 'Sir', 'Robin']
>>> name.split("r")
['B', 'ave Si', ' Robin']
>>> "||".join(tokens)
'Brave||Sir||Robin'
```

python

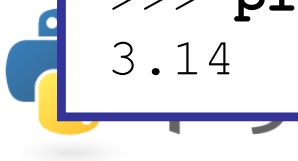
Tokenizing File Input

- Use `split` to tokenize line contents when reading files.
  - You may want to type-cast tokens: **type**(**value**)

```
>>> f = open("example.txt")
>>> line = f.readline()
>>> line
'hello world 42 3.14\n'

>>> tokens = line.split()
>>> tokens
['hello', 'world', '42', '3.14']

>>> word = tokens[0]
'hello'
>>> answer = int(tokens[2])
42
>>> pi = float(tokens[3])
3.14
```



Exercise

- Recall `hours.txt`. Suppose the # of days can vary:

```
123 Susan 12.5 8.1 7.6 3.2
456 Brad 4.0 11.6 6.5 2.7 12
789 Jenn 8.0 8.0 8.0 8.0 7.5
```

- Compute each worker's total hours and hours/day.
  - Should work no matter how many days a person works.

```
Suzy ID 123 worked 31.4 hours: 6.3 / day
Brad ID 456 worked 36.8 hours: 7.36 / day
Jenn ID 789 worked 39.5 hours: 7.9 / day
```

python™

Exercise Answer

hours.py

```
 1  file = open("hours.txt")
 2  for line in file:
 3      tokens = line.split()
 4      id = tokens[0]
 5      name = tokens[1]
 6
 7      # cumulative sum of this employee's hours
 8      hours = 0.0
 9      days = 0
10      for token in tokens[2:]:
11          hours += float(token)
12          days += 1
13
14      print(name, "ID", id, "worked", \
15            hours, "hours:", hours / days, "/ day")
```

Exercise

- Suppose we have a file of midterm scores, `scores.txt`:

```
76
89
76
72
68
```

- Create a histogram of the scores as follows:

```
75: *
76: *****
79: **
81: ********
82: ******
84: **********
```

python™

Exercise

- Suppose we have Internet Movie Database (IMDb) data:

```
1 9.1 196376 The Shawshank Redemption (1994)
2 9.0 139085 The Godfather: Part II (1974)
3 8.8 81507 Casablanca (1942)
```

- Write a program to search for all films with a given phrase:

```
Search word? part
Rank     Votes    Rating   Title
2        139085   9.0      The Godfather: Part II (1974)
40       129172   8.5      The Departed (2006)
95       20401    8.2      The Apartment (1960)
192      30587    8.0      Spartacus (1960)
4 matches.
```

Exercise Answer

movies.py

```
search_word = input("Search word? ")
matches = 0
file = open("imdb.txt")
for line in file:
    tokens = line.split()
    rank = int(tokens[0])
    rating = float(tokens[1])
    votes = int(tokens[2])
    title = " ".join(tokens[3:])

    # does title contain search_word?
    if search_word.lower() in title.lower():
        matches += 1
        print(rank, "\t", votes, "\t", rating, "\t", title)

print(matches, "matches.")
```