Programming cont’d

IC152 Lecture 7
Feb 2021

Strings

- string is a compound type

```
In [4]: x = 'IIT'

In [5]: type(x)
Out[5]: str

In [6]: y = '17'

In [7]: type(y)
Out[7]: str

In [8]: yi = int(y)

In [9]: x_i = int(x)
Traceback (most recent call last):

  File "<ipython-input-9-c06e6edf0465>", line 1, in <module>
    x_i = int(x)

ValueError: invalid literal for int() with base 10: 'IIT'
```

Strings are made up of characters, and are enclosed in ‘ ‘ (or “ ”)
We might want to access its parts, or consider it in whole.

```
In [10]: x
Out[10]: 'IIT'

In [11]: x[0]
Out[11]: 'I'

In [12]: x[1]
Out[12]: 'I'

In [13]: x[2]
Out[13]: 'T'
```


Access the individual elements using an index

Index starts from 0

```
In [15]: fruit = 'banana'

In [16]: len(fruit)
Out[16]: 6

In [17]: fruit[len(fruit)-1]
Out[17]: 'a'
```

`len()` is a built-in function

Strings can be traversed with loops

for loop can also be used

```
 8
 9 fruit = 'banana'
10 index = 0
11 while index < len(fruit):
12     letter = fruit[index]
13     print(letter)
14     index = index + 1
```

while loop

```
17 for c in fruit:
18     print(c)
```

```
b
a
n
a
n
a
```

What will be the value of index?

HW: Accept a string and print it in reverse

```
prefixes = "JKLMNOPQ"
suffix = "ack"

for letter in prefixes:
    print (letter + suffix)
```

```
Jack
Kack
Lack
Mack
Nack
Oack
Pack
Qack
```

Slicing strings

```
In [23]: s = "Peter, Paul, and Mary"

In [24]: print(s[0:5])
Peter

In [25]: print(s[7:11])
Paul

In [26]: print(s[17:21])
Mary
```

```
In [28]: fruit = 'banana'

In [29]: fruit[:3]
Out[29]: 'ban'

In [30]: fruit[3:]
Out[30]: 'ana'

In [31]: fruit[:]
Out[31]: 'banana'
```

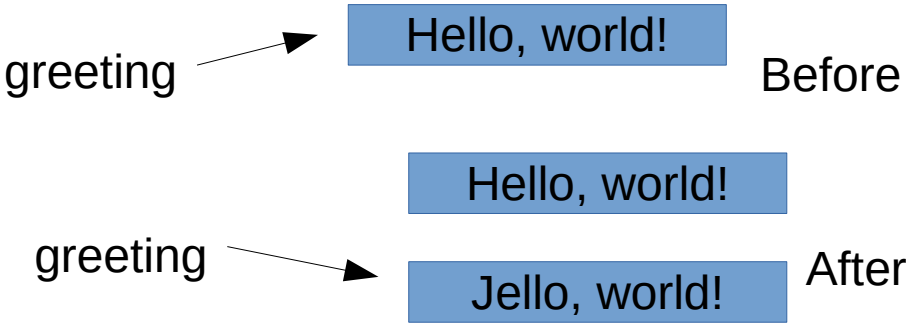
```
In [32]: 'Zebra' < 'Banana'
Out[32]: False

In [33]: 'Zebra' < 'banana'
Out[33]: True
```

Strings can be compared

```
In [34]: x = 'Zebra' < 'Banana'

In [35]: type(x)
Out[35]: bool
```

What type is `x`?

Strings are immutable




```
greeting = "Hello, world!"
greeting[0] = 'J'
```

Error!

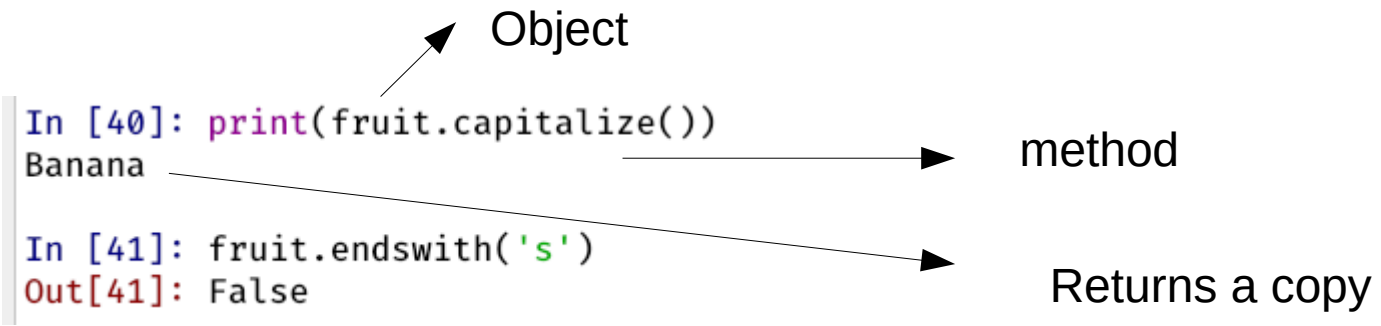
You can create a new string

```
greeting = 'J' + greeting[1:]
print(greeting)
```

Concatenate new character to a slice of the original

Strings are objects

Objects have built-in *methods*



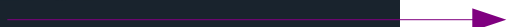
https://docs.python.org/3/library/stdtypes.html

```
In [63]: x='python'

In [64]: x.find('h')
Out[64]: 3

In [65]: y='that is the test'

In [66]: y.find('th')
Out[66]: 0

In [67]: y.find('th',5)
Out[67]: 8
```

```
In [69]: z='3.4a'

In [70]: z.isalpha()
Out[70]: False

In [71]: z.isalnum()
Out[71]: False

In [72]: w='34a'

In [73]: w.isalnum()
Out[73]: True
```

```
In [75]: x = '  once upon a time   '

In [76]: x.strip()
Out[76]: 'once upon a time'

In [77]: x.lstrip()
Out[77]: 'once upon a time   '
```

```
In [82]: name='Rahul Nair'

In [83]: nameParts = name.partition(' ')

In [84]: type(nameParts)
Out[84]: tuple

In [85]: nameParts
Out[85]: ('Rahul', ' ', 'Nair')

In [86]: len(nameParts)
Out[86]: 3

In [87]: nameParts[0]
Out[87]: 'Rahul'
```



tuple type

Lists

- Big data problems:
  - 1000s of students in a university, 10 million accounts in a bank
- Cannot use one variable for each

Sometimes we can manage

```
 9  n = int(input('Enter the number of terms: '))
10  print('Enter the numbers')
11  temp = 0
12  for i in range(n):
13      foo = int(input())
14      temp = temp + foo
15  mean = temp/n
16  print('The mean is', mean)
17
```

Can handle any number of inputs

Array: variable with N entries, with a single name
Arrays are called lists in Python

```
In [94]: s = [3,6,9,11,5]

In [95]: len(s)
Out[95]: 5

In [96]: type(s)
Out[96]: list

In [97]: s[0]
Out[97]: 3

In [98]: s[4]
Out[98]: 5

In [99]: s[-1]
Out[99]: 5
```

Lists are mutable

```
In [100]: s
Out[100]: [3, 6, 9, 11, 5]

In [101]: s[1] = 7

In [102]: s
Out[102]: [3, 7, 9, 11, 5]
```

Contrast with strings

Lists are objects

```
In [105]: s
Out[105]: [3, 7, 9, 11, 5]

In [106]: s.append(11)

In [107]: s
Out[107]: [3, 7, 9, 11, 5, 11]

In [108]: s.count(11)
Out[108]: 2
```

Add to the end

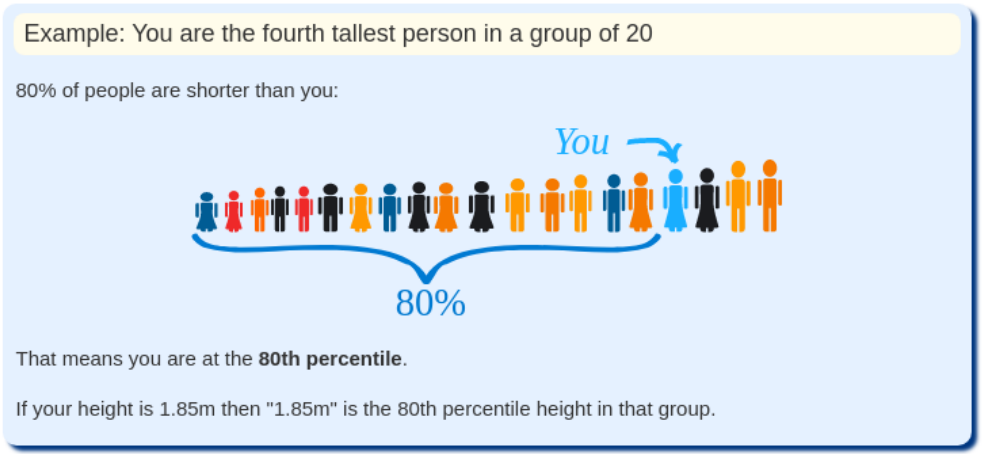
```
In [116]: s.sort()

In [117]: s
Out[117]: [3, 5, 7, 9, 11, 11, 15]
```

```
In [112]: s.insert(3,15)

In [113]: s
Out[113]: [3, 7, 9, 15, 11, 5, 11]

In [114]: s.reverse()

In [115]: s
Out[115]: [11, 5, 11, 15, 9, 7, 3]
```

Reverses in-place

Computing the median

Percentile: value below which a percentage of data falls.



https://www.mathsisfun.com/data/percentiles.html

Median is the 50th percentile
Half the numbers are larger, and half are smaller

Median of [45,1,10,30,25] is 25

[1,10,25,30,45]

If N is even, there are two choices for the median

Purpose of the median

Summarize a set of numbers by a single, typical value

The mean or average can be used

But median has some advantages:

- Always one of the data values
- Less sensitive outliers

Eg, a hundred 1s
A single mistake: 1 -> 100
Median: unaffected

Computing the median

Read the N numbers

Determine the median

Print the median

```
Input N
for i = 1 to N
        read a number into A[i]
```

```
Sort A
Find middle element
```

```
 8
 9  n = int(input('Enter the number of terms: '))
10  print('Enter the numbers')
11  # create empty list
12  a = []
13  for i in range(n):
14      a.append(int(input()))
15  a.sort()
16  print('Debug: sorted a:' + str(a))
17  if (n%2) == 1:
18      # n is odd
19      median = min(a[int(n/2)],a[int(n/2)+1])
20  else:
21      # n is even
22      median = a[int(n/2)]
23  print('The median is',median)
24
```

Homework

1)Find the number of digits in an integer
2)Find the number of digits in a float
3)Determine if a number is prime
4)Accept a polynomial p(x) and evaluate it for a given x.