SSE 691

Engineering Data Visualization

Project #1

by

Jason Payne

September 21, 2015

TABLE OF CONTENTS

Topics Covered	Topic Examples
Python Programming Language v3.4.3	• Core types • Collections • Objects & Classes • Modules • Other language features

1. Python 3 Tutorial

The following sections demonstrate the basic features of the Python programming language by exercising the provided examples from the official Python 3 tutorial and select chapters from the *Learning Python* (Lutz, 2013) text that will be utilized for this course. The format follows a basic construct where the provided example is exercised and demonstrated followed by a custom user-defined example highlighted in yellow. Efforts were taken to capture the formatting as it was displayed within the IPython console, but unfortunately the indentions were lost in the translation. However, certain examples include the proper indentations to emphasize the proper scope of certain aspects of the code attributes and variables. To see the properly formatted code as it was captured in html files from the IPython console, access the files in the zip file provided in the ftp folder.

1.1 Informal Introduction to Python

1.1.1 Numbers

```
In [1]: 2+2
Out[1]: 4
In [2]: 50-5*6
Out[2]: 20
In [3]: (50-5*6)/4
Out[3]: 5.0
In [4]: 8/5 # division always returns a floating point number
Out[4]: 1.6

In [5]: 1+2-3*4/5
Out[5]: 0.6000000000000001
In [6]: (1+2-3)*4/5
Out[6]: 0.0

In [7]: 17/3 # classic division returns a float
Out[7]: 5.666666666666667
In [8]: 17//3 # floor division discards the fractional part
Out[8]: 5
In [9]: 17%3 # the % operator returns the remainder of the division
Out[9]: 2
In [10]: 5*3+2 # result * divisor + remainder
Out[10]: 17

In [11]: 2/1
Out[11]: 2.0
In [12]: 1/2
Out[12]: 0.5
In [13]: 1//2
Out[13]: 0
In [14]: 1%2
Out[14]: 1
In [15]: 3/2
Out[15]: 1.5
In [16]: 3//2
Out[16]: 1
In [17]: 3%2
```

```
Out[17]: 1

In [18]: 5**2 # 5 squared
Out[18]: 25
In [19]: 2**7 # 2 to the power of 7
Out[19]: 128

In [20]: 2**-1
Out[20]: 0.5
In [21]: 2**-3
Out[21]: 0.125
In [22]: 4**-1/2
Out[22]: 0.125
In [23]: 3**3
Out[23]: 27
In [24]: 10**6
Out[24]: 1000000
In [25]: 4**(-1/2)
Out[25]: 0.5

In [26]: width=20
In [27]: height=5*9
In [28]: width*height
Out[28]: 900

In [29]: a=1
In [30]: b=2
In [31]: c=3
In [32]: a+b+c
Out[32]: 6

In [33]: n # try to access an undefined variable
Traceback (most recent call last):
File "<ipython-input-33-ad45dea5b6a7>", line 1, in <module>
n # try to access an undefined variable
NameError: name 'n' is not defined

In [34]: z # 'z' has not been defined yet
Traceback (most recent call last):
File "<ipython-input-34-66346fc216f9>", line 1, in <module>
z # 'z' has not been defined yet
NameError: name 'z' is not defined

In [35]: 3*3.75/1.5 # arithmetic with mixed type operands (ints, floats)
Out[35]: 7.5
In [36]: 7.0/2
Out[36]: 3.5

In [37]: 1*2.5+3 # int + float + int = floating number
Out[37]: 5.5

In [38]: # '_' (underscore) variable
In [39]: tax=12.5/100
In [40]: price=100.50
In [41]: price*tax
```

```
Out[41]: 12.5625
In [42]: price + _ # 100.5 + 12.5625 = 113.0625
Out[42]: 113.0625
In [43]: round(_,2) # round 113.0625 to 2 significant digits
Out[43]: 113.06

In [44]: a=1
In [45]: b=2
In [46]: 3
Out[46]: 3
In [47]: a+b+_ #1+2+3 = 6
Out[47]: 6
In [48]: a=10
In [49]: 30
Out[49]: 30
In [50]: b=20
In [51]: a+b+_ # 10+20+30=60
Out[51]: 60

In [1]: # Decimals: fixed precision
In [2]: import decimal
In [3]: d=decimal.Decimal('3.141')
In [4]: d+1
Out[4]: Decimal('4.141')
In [5]: decimal.getcontext().prec=2
In [6]: decimal.Decimal('1.00') / decimal.Decimal('3.00')
Out[6]: Decimal('0.33')

In [7]: d=decimal.Decimal()
In [9]: d+1
Out[9]: Decimal('1')
In [10]: decimal.getcontext().prec=4
In [13]: decimal.Decimal('1.00') / decimal.Decimal('3.0')
Out[13]: Decimal('0.3333')

In [14]: # Fractions: numerators & denominators
In [15]: from fractions import Fraction
In [16]: f=Fraction(2,3)
In [17]: f+1
Out[17]: Fraction(5, 3)
In [18]: f+Fraction(1,2)
Out[18]: Fraction(7, 6)

In [19]: Fraction(1,16)+Fraction(1,16)
Out[19]: Fraction(1, 8)
In [20]: Fraction(1,16)+Fraction(1,16) # 2/16 factored to lowest common denominator
Out[20]: Fraction(1, 8)
```

1.1.2 Strings

```
In [53]: 'spam eggs' # single quotes
Out[53]: 'spam eggs'
In [54]: 'doesn\'t' # use \' to escape the single quote...
Out[54]: "doesn't"
```

```
In [55]: "doesn't" # ...or use double quotes instead
Out[55]: "doesn't"
In [56]: '"Yes," he said.'
Out[56]: '"Yes," he said.'
In [57]: "\"Yes,\" he said."
Out[57]: '"Yes," he said.'
In [58]: '"Isn\'t," she said.'
Out[58]: '"Isn\'t," she said.'

In [59]: '0123'
Out[59]: '0123'
In [60]: "0123"
Out[60]: '0123'
In [61]: '01 "2" 3'
Out[61]: '01 "2" 3'
In [62]: "01 \"2\" 3"
Out[62]: '01 "2" 3'
In [63]: '01 \'2\' 3'
Out[63]: "01 '2' 3"

In [64]: '"Isn\'t," she said.'
Out[64]: '"Isn\'t," she said.'
In [65]: print('"Isn\'t," she said.')
"Isn't," she said.
In [66]: s='First line. \nSecond line.' # \n means newline
In [67]: s # without print(), '\n' is included in the output
Out[67]: 'First line. \nSecond line.'
In [68]: print(s) # with print(), '\n' produces a new line
First line.
Second line.

In [69]: '01 "\'2\'" 3' # '\' characters will be displayed since they are enclosed by
double quotes
Out[69]: '01 "\'2\'" 3'
In [70]: print('01 "\'2\'" 3') # '\' character will NOT be displayed since they are part
of the argument for the print() function
01 "'2'" 3
In [71]: "01\n23"
Out[71]: '01\n23'
In [72]: '01\n23'
Out[72]: '01\n23'
In [73]: s='01\n23'
In [74]: print(s)
01
23

In [75]: print('C:\some\name') # here, '\n' means newline!
C:\some
ame
In [76]: print(r'C:\some\name') # note the 'r' before the quote
C:\some\name

In [77]: print('C:\name1\name2\name3') # here, '\n' means newline!
C:
ame1
```

```
ame2
ame3
In [78]: print(r'C:\name1\name2\name3') # note the 'r' before the quote
C:\name1\name2\name3

In [79]: print("""\
    ...: Usage: thingy [OPTIONS]
    ...: -h Display this usage message
    ...: -H hostname Hostname to connect to
    ...: """)
Usage: thingy [OPTIONS]
-h Display this usage message
-H hostname Hostname to connect to

In [80]: print('''\
    ...: Line 1
    ...: Line '2'
    ...: Line "3"
    ...: ''')
Line 1
Line '2'
Line "3"

In [81]: # 3 times 'un', followed by 'ium'
In [82]: 3*'un' + 'ium'
Out[82]: 'ununium'

In [83]: 4*'-duck' +'-GOOSE!' # -duck-duck-duck-duck-GOOSE!
Out[83]: '-duck-duck-duck-duck-GOOSE!'

In [84]: 'Py' 'thon' # consecutive string literals are automatically concatenated
Out[84]: 'Python'

In [85]: 'Marco' '...' 'Polo' '...' 'Marco'
Out[85]: 'Marco...Polo...Marco'

In [86]: # Automatic concatenation of strings only works with string liters, not variables
or expressions
In [87]: prefix='Py'
In [88]: prefix 'thon' # can't concatenate a variable and a string literal
File "<ipython-input-88-8a4072fe3911>", line 1
prefix 'thon' # can't concatenate a variable and a string literal
^
SyntaxError: invalid syntax
In [89]: ('un' *3) 'ium'
File "<ipython-input-89-fead5d02adc5>", line 1
('un' *3) 'ium'
^
SyntaxError: invalid syntax
In [90]: # using '+' allows string literals to be concatenated with variables and/or
expressions
In [91]: prefix+'thon'
Out[91]: 'Python'

In [92]: duck='-duck'
```

```
In [93]: (3*duck)+'-GOOSE!'
Out[93]: '-duck-duck-duck-GOOSE!'

In [94]: text=('Put several strings within parenthesis '
   ...: 'to have them joined together.')
In [95]: text
Out[95]: 'Put several strings within parenthesis to have them joined together.'

In [96]: text=('Line 1'
   ...: 'Line 2'
   ...: 'Line 3')
In [97]: text
Out[97]: 'Line 1Line 2Line 3'

In [98]: word='Python'
In [99]: word[0] # character in position 0
Out[99]: 'P'
In [100]: word[5] # character in position 5
Out[100]: 'n'

In [101]: word[0]+word[1]+word[2]+word[3]+word[4]+word[5]
Out[101]: 'Python'

In [102]: word[-1] # Last character
Out[102]: 'n'
In [103]: word[-2] # Next-to-last character
Out[103]: 'o'
In [104]: word[-6] # First character
Out[104]: 'P'

In [105]: word[-6]+word[-5]+word[-4]+word[-3]+word[-2]+word[-1]
Out[105]: 'Python'

In [106]: word[0:2] # characters from position 0 (included) to 2 (excluded)
Out[106]: 'Py'
In [107]: word[2:5] # characters from position 2 (included) to 5 (excluded)
Out[107]: 'tho'

In [108]: word[0:6]
Out[108]: 'Python'

In [109]: word[:2]+word[2:]
Out[109]: 'Python'
In [110]: word[:4]+word[4:]
Out[110]: 'Python'

In [111]: word[2:]
Out[111]: 'thon'
In [112]: word[:2]
Out[112]: 'Py'
In [113]: word[:2]+word[2:]
Out[113]: 'Python'
In [114]: word[:2]
Out[114]: 'Py'
```

```
In [115]: word[:2] # character from the beginning to position 2 (excluded)
Out[115]: 'Py'
In [116]: word[4:] # characters from position 4 (included) to the end
Out[116]: 'on'
In [117]: word[-2:] # character from the next-to-last (included) to the end
Out[117]: 'on'

In [118]: word[6:]
Out[118]: ''
In [119]: word[-6:]
Out[119]: 'Python'

In [120]: word[42] # the word only has 6 characters
Traceback (most recent call last):
File "<ipython-input-120-e6388d1a40a1>", line 1, in <module>
word[42] # the word only has 6 characters
IndexError: string index out of range
In [121]: word[4:42]
Out[121]: 'on'
In [122]: word[42:]
Out[122]: ''

In [123]: word[-2:42]
Out[123]: 'on'
In [124]: word[-10:]
Out[124]: 'Python'

In [125]: word[0]='J'
Traceback (most recent call last):
File "<ipython-input-125-ea9bbb9211d6>", line 1, in <module>
word[0]='J'
TypeError: 'str' object does not support item assignment
In [126]: word[2:]='py'
Traceback (most recent call last):
File "<ipython-input-126-d303a1256087>", line 1, in <module>
word[2:]='py'
TypeError: 'str' object does not support item assignment
In [127]: 'J'+word[1:]
Out[127]: 'Jython'
In [128]: word[:2]+'py'
Out[128]: 'Pypy'

In [129]: iPython='i'+word
In [130]: print(iPython)
iPython
In [131]: iPython
Out[131]: 'iPython'

In [132]: s='supercalifragilisticexpialidocious'
In [133]: len(s)
Out[133]: 34

In [134]: len(word) # size of 'word' is 6
Out[134]: 6
```

```
In [1]: # Immutability of Strings
In [2]: S='Spam'
In [3]: S[0]='z' # Immutable objects cannot be changed
Traceback (most recent call last):
File "<ipython-input-3-22dd60f14af0>", line 1, in <module>
S[0]='z' # Immutable objects cannot be changed
TypeError: 'str' object does not support item assignment
In [4]: S='z'+S[1:] # But we can run expressions to make new objects
In [5]: S
Out[5]: 'zpam'

In [6]: name='SSE6_1'
In [7]: name[-2]='9'
Traceback (most recent call last):
File "<ipython-input-7-5ebf8d1da556>", line 1, in <module>
name[-2]='9'
TypeError: 'str' object does not support item assignment
In [8]: newName=name.replace('_','9')
In [9]: newName
Out[9]: 'SSE691'

In [10]: S='shrubbery'
In [11]: L=list(S) # Expand to a list: [...]
In [12]: L
Out[12]: ['s', 'h', 'r', 'u', 'b', 'b', 'e', 'r', 'y']
In [13]: L[1]='c' # Change it in place
In [14]: ''.join(L) # Join with empty delimiter
Out[14]: 'scrubbery'

In [15]: nameList=list(name)
In [16]: nameList
Out[16]: ['S', 'S', 'E', '6', '_', '1']
In [17]: nameList[-2]='9'
In [18]: ''.join(nameList)
Out[18]: 'SSE691'


In [19]: # Pattern Matching
In [20]: import re
In [21]: match = re.match('Hello[ \t]*(.*)world', 'Hello\tPython world')
In [22]: match.group(1)
Out[22]: 'Python '

In [23]: re.split('a', 'Heartland')
Out[23]: ['He', 'rtl', 'nd']
In [24]: re.findall('a.', 'Heartland')
Out[24]: ['ar', 'an']
In [25]: re.findall('.?a', 'Heartland')
Out[25]: ['ea', 'la']
In [26]: re.findall('a.*', 'Heartland')
Out[26]: ['artland']

In [27]: match = re.match('[/:](.*)[/:](.*)[/:](.*)', '/usr/home:lumberjack')
In [28]: match.groups()
Out[28]: ('usr', 'home', 'lumberjack')
In [29]: re.split('[/:]', '/usr/home:lumberjack')
```

```
Out[29]: ['', 'usr', 'home', 'lumberjack']

In [30]: re.findall('^.*a', 'Heartland')
Out[30]: ['Heartla']
In [31]: re.findall('^.*?a', 'Heartland')
Out[31]: ['Hea']
In [32]: re.findall('a.*$', 'Heartland')
Out[32]: ['artland']
```

1.1.3 Lists

```
In [1]: squares=[1,4,9,16,25]
In [2]: squares
Out[2]: [1, 4, 9, 16, 25]

In [3]: squares=['one','four','nine']
In [4]: squares
Out[4]: ['one', 'four', 'nine']

In [5]: squares=[1,4,9,16,25]
In [6]: squares[0] # indexing returns the item
Out[6]: 1
In [7]: squares[-1]
Out[7]: 25
In [8]: squares[-3:] # slicing returns a new list
Out[8]: [9, 16, 25]

In [9]: squares[2:5]
Out[9]: [9, 16, 25]

In [10]: squares[:]
Out[10]: [1, 4, 9, 16, 25]
In [11]: squares+[36,49,64,81,100]
Out[11]: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

In [12]: [1,2,3]+[8,9,10]
Out[12]: [1, 2, 3, 8, 9, 10]

In [13]: cubes=[1,8,27,65,125] # something is wrong here!
In [14]: 4**3 # the cube of 4 is 64, not 65!
Out[14]: 64
In [15]: cubes[3]=64 # replace the wrong value
In [16]: cubes
Out[16]: [1, 8, 27, 64, 125]

In [17]: cubes[0]=10**3
In [18]: cubes
Out[18]: [1000, 8, 27, 64, 125]
In [19]: cubes[0]=1
In [20]: cubes
Out[20]: [1, 8, 27, 64, 125]

In [21]: cubes.append(216) # add the cube of 6
In [22]: cubes.append(7**3) # add the cube of 7
```

```
In [23]: cubes
Out[23]: [1, 8, 27, 64, 125, 216, 343]

In [24]: cubes.append(10**3) # add the cube of 10
In [25]: cubes
Out[25]: [1, 8, 27, 64, 125, 216, 343, 1000]

In [26]: letters=['a','b','c','d','e','f','g']
In [27]: letters
Out[27]: ['a', 'b', 'c', 'd', 'e', 'f', 'g']
In [28]: # replace some values
In [29]: letters[2:5]=['C','D','E']
In [30]: letters
Out[30]: ['a', 'b', 'C', 'D', 'E', 'f', 'g']
In [31]: # now remove them
In [32]: letters[2:5]=[]
In [33]: letters
Out[33]: ['a', 'b', 'f', 'g']
In [34]: # clear the list by replacing all the elements with an empty list
In [35]: letters[:]=[]
In [36]: letters
Out[36]: []

In [37]: test=[5,4,3,2,1]
In [38]: test
Out[38]: [5, 4, 3, 2, 1]
In [39]: test[-2:-5]=[0,0,0]
In [40]: test
Out[40]: [5, 4, 3, 0, 0, 0, 2, 1]
In [41]: test[3:6]=[]
In [42]: test
Out[42]: [5, 4, 3, 2, 1]
In [43]: test[-2:-5]=[0,0,0]
In [44]: test
Out[44]: [5, 4, 3, 0, 0, 0, 2, 1]

In [45]: len(letters)
Out[45]: 0

In [46]: len(test)
Out[46]: 8

In [47]: a=['a','b','c']
In [48]: n=[1,2,3]
In [49]: x=[a,n]
In [50]: x
Out[50]: [['a', 'b', 'c'], [1, 2, 3]]
In [51]: x[0]
Out[51]: ['a', 'b', 'c']
In [52]: x[0][1]
Out[52]: 'b'

In [53]: listA=['R','G','B']
In [54]: listB=[0,1,2]
In [55]: test=[listB,listA]
```

```
In [56]: test
Out[56]: [[0, 1, 2], ['R', 'G', 'B']]
In [57]: test[0]
Out[57]: [0, 1, 2]
In [58]: test[1]
Out[58]: ['R', 'G', 'B']
In [59]: test[0][1]
Out[59]: 1
In [60]: test[1][0]
Out[60]: 'R'
```

1.1.4 First Steps Towards Programming

```
In [1]: # Fibonacci series:
In [2]: # the sum of two elements defines the next
In [3]: a,b=0,1
In [4]: while b<10:
   ...: print(b)
   ...: a,b=b,a+b
   ...:
1
1
2
3
5
8

In [12]: root,square=4,16
In [13]: while root>0:
   ...: print('@ '+str(root)+', square = '+str(square))
   ...: root,square=root-1,(root-1)**2
   ...:
@ 4, square = 16
@ 3, square = 9
@ 2, square = 4
@ 1, square = 1

In [14]: i=256*256
In [15]: print('The value of i is',i)
The value of i is 65536

In [16]: root,square=4,16
In [17]: while root>0:
   ...: print('@',root,', square = ',square)
   ...: root,square=root-1,(root-1)**2
   ...:
@ 4 , square = 16
@ 3 , square = 9
@ 2 , square = 4
@ 1 , square = 1

In [18]: a,b=0,1
In [19]: while b<1000:
   ...: print(b, end=',')
```

```
   ...: a,b=b,a+b
   ...:
1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,

In [20]: root,square=4,16
In [21]: while root>0:
   ...: print('@',root,', square = ',square,end='...')
   ...: root,square=root-1,(root-1)**2
   ...:
@ 4 , square = 16...@ 3 , square = 9...@ 2 , square = 4...@ 1 , square = 1...
```

1.2 More Control Flow Tools

1.2.1 if Statements

```
In [1]: # if Statements
In [2]: x=int(input("Please enter an integer: "))
Please enter an integer: 42
In [3]: if x<0:
   ...: x=0
   ...: print('Negative changed to zero')
   ...: elif x==0:
   ...: print('Zero')
   ...: elif x==1:
   ...: print('Single')
   ...: else:
   ...: print('More')
   ...:
More

In [4]: choice=int(input("Select from the following:"\
   ...: "1 - Red"\
   ...: "2 - Green"\
   ...: "3 - Blue"))
Select from the following:1 - Red2 - Green3 - Blue
2
In [8]: if choice<1:
   ...: choice=1
   ...: print('Invalid selection, defaulting to Red')
   ...: elif choice==1:
   ...: print('User chose Red')
   ...: elif choice==2:
   ...: print('User chose Green')
   ...: elif choice==3:
   ...: print('User chose Blue')
   ...: else:
   ...: print('Defaulting to Red')
   ...:
User chose Green
```

1.2.2 for Statements

```
In [9]: # for Statements
In [11]: # Measure some strings:
In [12]: words=['cat','window','defenestrate']
In [13]: for w in words:
   ...: print(w, len(w))
   ...:
cat 3
window 6
defenestrate 12

In [14]: for ch in "Super Word":
   ...: print(ch)
   ...:
S
U
P
E
R

W
O
R
d

In [15]: for w in words[:]: # Loop over a slice copy of the entire list.
   ...: if len(w) > 6:
   ...: words.insert(0,w)
   ...:
In [16]: words
Out[16]: ['defenestrate', 'cat', 'window', 'defenestrate']

In [17]: test="Super Word"
In [18]: for ch in test[:]:
   ...: if ch==' ':
   ...: test='_'+test
   ...:
In [19]: test
Out[19]: '_Super Word'
```

1.2.3 The range() Function

```
In [20]: # 4.3 The range() Function
In [21]: for i in range(5):
   ...: print(i)
   ...:
0
1
2
3
4

In [22]: for i in range(10):
   ...: print(i)
```

```
   ...:
0
1
2
3
4
5
6
7
8
9

In [23]: range(5,10)
Out[23]: range(5, 10)
In [24]: for i in range(5,10):
   ...: print(i)
   ...:
5
6
7
8
9
In [25]: for i in range(0,10,3):
   ...: print(i)
   ...:
0
3
6
9
In [27]: for i in range(-10,-100,-30):
   ...: print(i)
   ...:
-10
-40
-70

In [28]: for i in range(2,21,2):
   ...: print(i)
   ...:
2
4
6
8
10
12
14
16
18
20

In [29]: # Iterate over indices of a sequence
In [30]: a=['Mary', 'had', 'a', 'little', 'lamb']
In [31]: for i in range(len(a)):
   ...: print(i,a[i])
   ...:
```

```
0 Mary
1 had
2 a
3 little
4 lamb

In [32]: test="test"
In [33]: for i in range(len(test)):
   ...: print(i, test[i])
   ...:
0 t
1 e
2 s
3 t

In [34]: print(range(10))
range(0, 10)
In [35]: list(range(5))
Out[35]: [0, 1, 2, 3, 4]

In [36]: list(range(2,21,2))
Out[36]: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

1.2.4 break and continue Statements, and else Clauses on Loops

```
In [1]: # break and continue Statements, and else Clauses on Loops
In [3]: for n in range(2,10):
   ...: for x in range(2,n):
   ...: if n%x==0:
   ...: print(n,'equals',x,'*',n//x)
   ...: break
   ...: else:
   ...: # Loop fell through without finding a factor
   ...: print(n, 'is a prime number')
   ...:
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3

In [4]: rows=[['X','X','X'],['X','-','X'],['X','X','X']]
In [5]: for r in range(len(rows)):
   ...: for c in rows[r]:
   ...: if c=='-':
   ...: print('Seats available in row ',r)
   ...: break
   ...: else:
   ...: print('No seats available in row ',r)
   ...:
No seats available in row 0
```

```
Seats available in row 1
No seats available in row 2

In [6]: for num in range(2,10):
   ...: if num%2==0:
   ...: print("Found an even number",num)
   ...: continue
   ...: print("Found a number",num)
   ...:
Found an even number 2
Found a number 3
Found an even number 4
Found a number 5
Found an even number 6
Found a number 7
Found an even number 8
Found a number 9

In [7]: # using continue statement to find seats
In [8]: for r in range(len(rows)):
   ...: for c in rows[r]:
   ...: if c=='X':
   ...: continue
   ...: print('Seats available in row',r)
   ...:
Seats available in row 1
```

1.2.5 pass Statements

```
In [9]: # pass Statements
   ...: while True:
   ...: pass # Busy-wait for keyboard interrupt (Ctrl+C)
   ...:
Traceback (most recent call last):
File "<ipython-input-9-e92b95378185>", line 3, in <module>
pass # Busy-wait for keyboard interrupt (Ctrl+C)
KeyboardInterrupt

In [10]: choice = int(input("Please make your choice:\n"\
   ...: "1. <Placeholder for future use>\n"\
   ...: "2. Proceed with option 2.\n"))
   ...: if choice==1:
   ...: pass
   ...: elif choice==2:
   ...: print("Do something")
   ...:
Please make your choice:
1. <Placeholder for future use>
2. Proceed with option 2.
1
In [11]: choice = int(input("Please make your choice:\n"\
   ...: "1. <Placeholder for future use>\n"\
   ...: "2. Proceed with option 2.\n"))
```

```
   ...: if choice==1:
   ...: pass
   ...: elif choice==2:
   ...: print("Do something")
   ...:
Please make your choice:
1. <Placeholder for future use>
2. Proceed with option 2.
2 Do something
```

1.2.6 Defining Functions

```
In [12]: # Defining Functions
   ...: def fib(n): # write Fibonacci series up to n
   ...: """Print a Fibonacci series up to n."""
   ...: a,b=0,1
   ...: while a<n:
   ...: print(a,end=' ')
   ...: a,b=b,a+b
   ...: print()
   ...:
In [13]: # Now call the function we just defined:
In [14]: fib(2000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
In [15]: fib
Out[15]: <function __main__.fib>
In [16]: f=fib
In [17]: f(100)
0 1 1 2 3 5 8 13 21 34 55 89

In [18]: def sum(a,b): # print the sum of two numbers
   ...: """Displays the sum of two numbers."""
   ...: print(a+b)
   ...:
In [19]: sum(1,2)
3
In [20]: sum
Out[20]: <function __main__.sum>
In [21]: add2=sum
In [22]: add2(1,2)
3
In [23]: add2
Out[23]: <function __main__.sum>
In [24]: print(sum(1,2))
3
None

In [36]: def fib2(n): # return Fibonacci series up to n
   ...: """Return a List containing the Fibonacci series up to n."""
   ...: result=[]
   ...: a,b=0,1
   ...: while a<n:
   ...: result.append(a) # see below
   ...: a,b=b,a+b
```

```
   ...: return result
   ...:
In [38]: f100=fib2(100) # call it
In [39]: f100 # write the result
Out[39]: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

In [40]: def sum2(a,b): # return the sum of two numbers
   ...: """Returns the sum of two numbers."""
   ...: result=a+b
   ...: return result
   ...:
In [41]: value=sum2(1,2)
In [42]: value
Out[42]: 3
```

1.2.7 DefaultArgument Values

```
In [43]: # More on Defining Functions
In [44]: # Default Argument Values
In [45]: def ask_ok(prompt, retries=4,complaint='Yes or no, please!'):
   ...: while True:
   ...: ok=input(prompt)
   ...: if ok in ('y','ye','yes'):
   ...: return True
   ...: if ok in ('n','no','nop','nope'):
   ...: return False
   ...: retries=retries-1
   ...: if retries < 0:
   ...: raise OSError('uncooperative user')
   ...: print(complaint)
   ...:
In [46]: ask_ok('Do you really want to quit?')
Do you really want to quit?yes
Out[46]: True
In [47]: ask_ok('OK to overwrite the file?',2)
OK to overwrite the file?n
Out[47]: False
In [48]: ask_ok('OK to overwrite the file?',2)
OK to overwrite the file?k
Yes or no, please!
OK to overwrite the file?k
Yes or no, please!
OK to overwrite the file?k
Traceback (most recent call last):
File "<ipython-input-48-137b4cc7daa7>", line 1, in <module>
ask_ok('OK to overwrite the file?',2)
File "<ipython-input-45-f0b61b7b4f16>", line 10, in ask_ok
raise OSError('uncooperative user')
OSError: uncooperative user
In [49]: ask_ok('OK to overwrite the file?',2,'Come on, only yes or no!')
OK to overwrite the file?k
Come on, only yes or no!
OK to overwrite the file?y
Out[49]: True
```

```
In [50]: def sum3(a,b=0): # return the sum of two numbers
   ...: """Returns the sum of two numbers."""
   ...: result=a+b
   ...: return result
   ...:
In [51]: print(sum3(1,2))
3
In [52]: print(sum3(1))
1

In [53]: i=5
In [54]: def f(arg=i):
   ...: print(arg)
   ...:
In [55]: i=6
In [56]: f()
5

In [57]: defaultAddend=99
In [58]: def sum4(a,b=defaultAddend):
   ...: return a+b
   ...:
In [59]: defaultAddend=1
In [60]: sum4(1)
Out[60]: 100
```

1.2.8 Keyword Arguments

```
In [61]: # Keyword Arguments
In [63]: def parrot(voltage,state='a stiff',action='voom',type='Norwegian Blue'):
   ...: print("-- This parrot wouldn't",action,end=' ')
   ...: print("if you put",voltage,"volts through it.")
   ...: print("-- Lovely plumage, the",type)
   ...: print("-- It's",state,"!")
   ...:
In [64]: parrot(1000) # 1 positional argument
-- This parrot wouldn't voom if you put 1000 volts through it.
-- Lovely plumage, the Norwegian Blue
-- It's a stiff !
In [65]: parrot(voltage=1000) # 1 keyword argument
-- This parrot wouldn't voom if you put 1000 volts through it.
-- Lovely plumage, the Norwegian Blue
-- It's a stiff !
In [66]: parrot(voltage=1000000,action='VOOOOOM') # 2 keyword arguments
-- This parrot wouldn't VOOOOOM if you put 1000000 volts through it.
-- Lovely plumage, the Norwegian Blue
-- It's a stiff !
In [67]: parrot(action='VOOOOOM',voltage=1000000) # 2 keyword arguments
-- This parrot wouldn't VOOOOOM if you put 1000000 volts through it.
-- Lovely plumage, the Norwegian Blue
-- It's a stiff !
In [68]: parrot('a million', 'bereft of life', 'jump') # 3 positional arguments
-- This parrot wouldn't jump if you put a million volts through it.
```

```
-- Lovely plumage, the Norwegian Blue
-- It's bereft of life !
In [69]: parrot('a thousand', state='pushing up the daisies') # 1 positional, 1 keyword
-- This parrot wouldn't voom if you put a thousand volts through it.
-- Lovely plumage, the Norwegian Blue
-- It's pushing up the daisies !
In [70]: parrot() # required argument missing
Traceback (most recent call last):
File "<ipython-input-70-29cd9b53cc1d>", line 1, in <module>
parrot() # required argument missing
TypeError: parrot() missing 1 required positional argument: 'voltage'
In [71]: parrot(voltage=5.0,'dead') # non-keyword argument after a keyword argument
File "<ipython-input-71-8d29551fb915>", line 1
parrot(voltage=5.0,'dead') # non-keyword argument after a keyword argument
^
SyntaxError: non-keyword arg after keyword arg
In [72]: parrot(110,voltage=220) # duplicate value for the same argument
Traceback (most recent call last):
File "<ipython-input-72-967951ae2fad>", line 1, in <module>
parrot(110,voltage=220) # duplicate value for the same argument
TypeError: parrot() got multiple values for argument 'voltage'
In [73]: parrot(actor='John Cleese') # unknown keyword argument
Traceback (most recent call last):
File "<ipython-input-73-ac0c1a281d99>", line 1, in <module>
parrot(actor='John Cleese') # unknown keyword argument
TypeError: parrot() got an unexpected keyword argument 'actor'

In [74]: def sumThree(a,b=1,c=2):
   ...: result=a+b+c
   ...: print(result)
   ...:
In [75]: sumThree(1)
4
In [76]: sumThree(1,0,0)
1
In [77]: sumThree(1,b=0)
3
In [78]: sumThree(1,c=0)
2
In [79]: sumThree(a='SSE',b='691',c='Test')
SSE691Test
In [81]: sumThree(99,a=1)
Traceback (most recent call last):
File "<ipython-input-81-78b87432477d>", line 1, in <module>
sumThree(99,a=1)
TypeError: sumThree() got multiple values for argument 'a'
In [82]: sumThree(c=99,a=1)
101

In [83]: def cheeseshop(kind,*arguments,**keywords):
   ...: print("-- Do you have any",kind,"?")
   ...: print("-- I'm sorry, we're all out of",kind)
   ...: for arg in arguments:
   ...: print(arg)
   ...: print("-"*40)
```

```
   ...: keys=sorted(keywords.keys())
   ...: for kw in keys:
   ...: print(kw,":",keywords[kw])
   ...:
In [84]: cheeseshop("Limburger","It's very runny, sir.","It's really very, VERY runny,
sir.",shopkeeper="Michael Palin",client="John Cleese",sketch="Cheese Shop Sketch")
-- Do you have any Limburger ?
-- I'm sorry, we're all out of Limburger
It's very runny, sir.
It's really very, VERY runny, sir.
----------------------------------------
client : John Cleese
shopkeeper : Michael Palin
sketch : Cheese Shop Sketch

In [88]: def sumWithMessage(a, *addends, **messages):
   ...: msgKeys=messages.keys()
   ...: for km in msgKeys:
   ...: print(messages[km])
   ...: result=a
   ...: resultStr=str(a)
   ...: for num in addends:
   ...: result=result+num
   ...: resultStr=resultStr+" + "+str(num)
   ...: resultStr=resultStr+" = "+str(result)
   ...: print(resultStr)
   ...:
In [89]: sumWithMessage(0,1,2,3,welcomeMsg="Welcome to the Calculator!!",sumMsg="This
function will display the sum of the provided values.")
Welcome to the Calculator!!
This function will display the sum of the provided values.
0 + 1 + 2 + 3 = 6
```

1.2.9 Arbitrary Argument Lists

```
In [91]: # Arbitrary Argument Lists
In [92]: def write_multiple_items(file,separator,*args):
   ...: file.write(separator.join(args))
   ...:
In [93]: def concat(*args,sep="/"):
   ...: return sep.join(args)
   ...:
In [94]: concat("earth", "mars", "venus")
Out[94]: 'earth/mars/venus'
In [95]: concat("earth", "mars", "venus", sep=".")
Out[95]: 'earth.mars.venus'

In [96]: def sumWithMessage(a, *addends, msg="Welcome!"):
   ...: print(msg)
   ...: result=a
   ...: resultStr=str(a)
   ...: for num in addends:
   ...: result=result+num
   ...: resultStr=resultStr+" + "+str(num)
```

```
   ...: resultStr=resultStr+" = "+str(result)
   ...: print(resultStr)
   ...:
In [97]: sumWithMessage(99)
Welcome!
99 = 99
In [98]: sumWithMessage(1,2,99)
Welcome!
1 + 2 + 99 = 102
In [99]: sumWithMessage(1,2,99,501,msg="Your result is:")
Your result is:
1 + 2 + 99 + 501 = 603
```

1.2.10 Unpacking Argument Lists

```
In [100]: # Unpacking Argument Lists
In [101]: list(range(3,6)) # normal call with separate arguments
Out[101]: [3, 4, 5]
In [102]: args=[3,6]
In [103]: list(range(*args)) # call with arguments unpacked from a list
Out[103]: [3, 4, 5]
In [104]: def parrot(voltage, state='a stiff', action='voom'):
   ...: print("-- This parrot wouldn't",action,end=' ')
   ...: print("if you put",voltage,"volts through it.",end=' ')
   ...: print("E's",state,"!")
   ...:
In [105]: d={"voltage":"four million","state":"bleedin' demised","action":"VOOM"}
In [106]: parrot(**d)
-- This parrot wouldn't VOOM if you put four million volts through it. E's bleedin'
demised !

In [107]: def student_info(name,address='N/A',email='N/A'):
   ...: print(name)
   ...: print(address)
   ...: print(email)
   ...:
In [108]: student_info('John Smith')
John Smith
N/A
N/A
In [109]: info={"name":"John Smith","address":"123 American Way, Macon, GA,
12345","email":"john.smith@live.mercer.edu"}
In [110]: student_info(**info)
John Smith
123 American Way, Macon, GA, 12345
john.smith@live.mercer.edu
```

1.2.11 Lambda Expressions

```
In [111]: # Lambda Expressions
In [112]: def make_incrementor(n):
   ...: return lambda x: x+n
   ...:
```

```
In [113]: f=make_incrementor(42)
In [114]: f(0)
Out[114]: 42
In [115]: f(1)
Out[115]: 43
In [116]: f(5)
Out[116]: 47
In [117]: pairs=[(1,'one'),(2,'two'),(3,'three'),(4,'four')]
In [118]: pairs.sort(key=lambda pair: pair[1])
In [119]: pairs
Out[119]: [(4, 'four'), (1, 'one'), (3, 'three'), (2, 'two')]

In [120]: complexExpression=lambda a,b,c:a+b-(c**2)
In [121]: complexExpression(1,2,3)
Out[121]: -6
```

1.2.12 Documentation Strings

```
In [122]: # Documentation Strings
In [123]: def my_function():
   ...:     """Do nothing, but document it.
   ...:
   ...:     No, really, it doesn't do anything.
   ...:     """
   ...:     pass
   ...:
In [124]: print(my_function.__doc__)
Do nothing, but document it.
    No, really, it doesn't do anything.

In [125]: def sse691_function():
   ...:     """Short description goes here.
   ...:
   ...:     Details including calling conventions, side effects, etc. goes here.
   ...:     """
   ...:
In [126]: print(sse691_function.__doc__)
Short description goes here.
    Details including calling conventions, side effects, etc. goes here.
```

1.2.13 Function Annotations

```
In [127]: # Function Annotations
In [128]: def f(ham:str, eggs:str = 'eggs') -> str:
   ...:     print("Annotations:",f.__annotations__)
   ...:     print("Arguments:",ham,eggs)
   ...:     return ham+' and '+eggs
   ...:
In [129]: f('spam')
Annotations: {'return': <class 'str'>, 'ham': <class 'str'>, 'eggs': <class 'str'>}
Arguments: spam eggs
Out[129]: 'spam and eggs'
```

```
In [132]: def sum2(a:int,b:int=0,msg:str="Welcome!") -> int:
   ...: print("Annotations:",sum2.__annotations__)
   ...: print()
   ...: print(msg)
   ...: print(a+b)
   ...:
In [133]: sum2(1)
Annotations: {'return': <class 'int'>, 'b': <class 'int'>, 'msg': <class 'str'>, 'a':
<class 'int'>}
Welcome!
1
In [134]: sum2(1,2,"Adding 1 + 2...")
Annotations: {'return': <class 'int'>, 'b': <class 'int'>, 'msg': <class 'str'>, 'a':
<class 'int'>}
Adding 1 + 2...
3
```

1.3 Data Structures

1.3.1 More on Lists

```
In [2]: # More on Lists
In [3]: a=[66.25,333,333,1,1234.5]
In [4]: print(a.count(333),a.count(66.25),a.count('x'))
2 1 0
In [5]: a.insert(2,-1)
In [6]: a.append(333)
In [7]: a
Out[7]: [66.25, 333, -1, 333, 1, 1234.5, 333]
In [8]: a.index(333)
Out[8]: 1
In [9]: a.remove(333)
In [10]: a
Out[10]: [66.25, -1, 333, 1, 1234.5, 333]
In [11]: a.reverse()
In [12]: a
Out[12]: [333, 1234.5, 1, 333, -1, 66.25]
In [13]: a.sort()
In [14]: a
Out[14]: [-1, 1, 66.25, 333, 333, 1234.5]
In [15]: a.pop()
Out[15]: 1234.5
In [16]: a
Out[16]: [-1, 1, 66.25, 333, 333]

In [17]: myList=[]
In [18]: myList.append(3)
In [19]: myList
Out[19]: [3]
In [20]: myList.extend([2,-1,3,0,1])
In [21]: myList
Out[21]: [3, 2, -1, 3, 0, 1]
In [22]: myList.insert(1,-2)
```

```
In [23]: myList
Out[23]: [3, -2, 2, -1, 3, 0, 1]
In [24]: myList.remove(5)
Traceback (most recent call last):
File "<ipython-input-24-dadfb9f60967>", line 1, in <module>
myList.remove(5)
ValueError: list.remove(x): x not in list
In [25]: myList.remove(-2)
In [26]: myList
Out[26]: [3, 2, -1, 3, 0, 1]
In [27]: myList.pop()
Out[27]: 1
In [28]: myList
Out[28]: [3, 2, -1, 3, 0]
In [29]: myList.pop(2)
Out[29]: -1
In [30]: myList
Out[30]: [3, 2, 3, 0]
In [31]: myList.clear()
In [32]: myList
Out[32]: []
In [33]: myList.extend([1,2,-1,3,0])
In [34]: myList
Out[34]: [1, 2, -1, 3, 0]
In [35]: myList.index(5)
Traceback (most recent call last):
File "<ipython-input-35-48427c957685>", line 1, in <module>
myList.index(5)
ValueError: 5 is not in list
In [36]: myList.index(0)
Out[36]: 4
In [37]: myList.append(0)
In [38]: myList
Out[38]: [1, 2, -1, 3, 0, 0]
In [39]: myList.count(-1)
Out[39]: 1
In [40]: myList.count(0)
Out[40]: 2
In [41]: myList.sort()
In [42]: myList
Out[42]: [-1, 0, 0, 1, 2, 3]
In [43]: myList.reverse()
In [44]: myList
Out[44]: [3, 2, 1, 0, 0, -1]
In [45]: myList.copy()
Out[45]: [3, 2, 1, 0, 0, -1]

In [46]: # Using Lists as Stacks
In [47]: stack=[3,4,5]
In [48]: stack.append(6)
In [49]: stack.append(7)
In [50]: stack
Out[50]: [3, 4, 5, 6, 7]
In [51]: stack.pop()
Out[51]: 7
```

```
In [52]: stack
Out[52]: [3, 4, 5, 6]
In [53]: stack.pop()
Out[53]: 6
In [54]: stack.pop()
Out[54]: 5
In [55]: stack
Out[55]: [3, 4]

In [56]: myStack=[]
In [57]: myStack.extend(['s','t','a','c','k'])
In [58]: myStack
Out[58]: ['s', 't', 'a', 'c', 'k']
In [59]: myStack.append('y')
In [60]: myStack
Out[60]: ['s', 't', 'a', 'c', 'k', 'y']
In [61]: myStack.pop()
Out[61]: 'y'
In [62]: myStack.pop()
Out[62]: 'k'
In [63]: myStack
Out[63]: ['s', 't', 'a', 'c']

In [64]: # Using Lists as Queues
In [65]: from collections import deque
In [66]: queue=deque(["Eric","John","Michael"])
In [67]: queue.append("Terry") # Terry arrives
In [68]: queue.append("Graham") # Graham arrives
In [69]: queue.popleft() # The first to arrive now leaves
Out[69]: 'Eric'
In [70]: queue.popleft() # The second to arrive now leaves
Out[70]: 'John'
In [71]: queue
Out[71]: deque(['Michael', 'Terry', 'Graham'])

In [72]: myQ=deque(["Monday","Wednesday","Friday"])
In [73]: myQ.append("Saturday")
In [74]: myQ.popleft() # pop Monday
Out[74]: 'Monday'
In [75]: myQ.popleft() # pop Wednesday
Out[75]: 'Wednesday'
In [76]: myQ
Out[76]: deque(['Friday', 'Saturday'])

In [1]: # List Comprehensions
In [2]: squares=[]
In [3]: for x in range(10):
   ...:     squares.append(x**2)
   ...:
In [4]: squares
Out[4]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
In [5]: squares=list(map(lambda x:x**2,range(10)))
In [6]: squares
Out[6]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
In [7]: squares=[x**2 for x in range(10)]
```

```
In [8]: squares
Out[8]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
In [9]: [(x,y) for x in [1,2,3] for y in [3,1,4] if x!=y]
Out[9]: [(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
In [10]: combs=[]
In [11]: for x in [1,2,3]:
   ...: for y in [3,1,4]:
   ...: if x!=y:
   ...: combs.append((x,y))
   ...:
In [12]: combs
Out[12]: [(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
In [13]: vec=[-4,-2,0,2,4]
In [14]: # create a new List with the values doubled
In [15]: [x*2 for x in vec]
Out[15]: [-8, -4, 0, 4, 8]
In [16]: # filter the list to exclude negative numbers
In [17]: [x for x in vec if x>=0]
Out[17]: [0, 2, 4]
In [18]: # apply a function to all the elements
In [19]: [abs(x) for x in vec]
Out[19]: [4, 2, 0, 2, 4]
In [20]: # call a method on each element
In [21]: freshfruit=[" banana"," loganberry","passion fruit "]
In [22]: [weapon.strip() for weapon in freshfruit]
Out[22]: ['banana', 'loganberry', 'passion fruit']
In [23]: # create a list of 2-tuples like (number, square)
In [24]: [(x,x**2) for x in range(6)]
Out[24]: [(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]
In [25]: # the tuple must be parenthesized, otherwise an error is raised
In [26]: [x,x**2 for x in range(6)]
File "<ipython-input-26-8d6940458683>", line 1
[x,x**2 for x in range(6)]
^
SyntaxError: invalid syntax
In [27]: # flatten a list using a listcomp with two 'for'
In [28]: vec=[[1,2,3],[4,5,6],[7,8,9]]
In [29]: [num for elem in vec for num in elem]
Out[29]: [1, 2, 3, 4, 5, 6, 7, 8, 9]

In [30]: # create a new list with altered values
In [31]: myList=['a','b','c']
In [32]: ['_'+ch for ch in myList]
Out[32]: ['_a', '_b', '_c']
In [33]: # filter the list
In [34]: [x for x in myList if x != 'a']
Out[34]: ['b', 'c']
In [35]: # apply a function to all the elements
In [36]: [len(x) for x in myList]
Out[36]: [1, 1, 1]
In [37]: # call a method on each element
In [39]: [ch.upper() for ch in myList]
Out[39]: ['A', 'B', 'C']
In [40]: # create a list of 2-tuples like (lower, upper)
In [41]: [(ch, ch.upper()) for ch in myList]
```

```
Out[41]: [('a', 'A'), ('b', 'B'), ('c', 'C')]
In [42]: people=[["Bob",40],["Amy",35],["PJ",18]]
In [43]: [x for person in people for x in person]
Out[43]: ['Bob', 40, 'Amy', 35, 'PJ', 18]

In [44]: from math import pi
In [45]: [str(round(pi,i)) for i in range(1,6)]
Out[45]: ['3.1', '3.14', '3.142', '3.1416', '3.14159']

In [46]: def complex_function(num):
   ...:  return 1+2-(num**2)
   ...:
In [47]: [complex_function(x) for x in range(4)]
Out[47]: [3, 2, -1, -6]

In [48]: # Nested List Comprehensions
In [49]: matrix=[
   ...: [1,2,3,4],
   ...: [5,6,7,8],
   ...: [9,10,11,12],
   ...: ]
In [50]: [[row[i] for row in matrix] for i in range(4)]
Out[50]: [[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]

In [61]: [[matrix[-row][-col] for row in range(1,len(matrix)+1)] for col in range(1,5)]
Out[61]: [[12, 8, 4], [11, 7, 3], [10, 6, 2], [9, 5, 1]]
```

1.3.2 The del statement

```
In [62]: # The del statement
In [63]: a = [-1,1,66.25,333,333,1234.5]
In [64]: del a[0]
In [65]: a
Out[65]: [1, 66.25, 333, 333, 1234.5]
In [66]: del a[2:4]
In [67]: a
Out[67]: [1, 66.25, 1234.5]
In [68]: del a[:]
In [69]: a
Out[69]: []
In [71]: del a
In [72]: a
Traceback (most recent call last):
File "<ipython-input-72-60b725f10c9c>", line 1, in <module>
a
NameError: name 'a' is not defined

In [76]: name=['J','a','s','o','n']
In [77]: name
Out[77]: ['J', 'a', 's', 'o', 'n']
In [78]: del name[0]
In [79]: name
Out[79]: ['a', 's', 'o', 'n']
In [80]: del name[2:]
```

```
In [81]: name
Out[81]: ['a', 's']
In [82]: del name
In [83]: name
Traceback (most recent call last):
File "<ipython-input-83-18697449d7c4>", line 1, in <module>
name
NameError: name 'name' is not defined
```

1.3.3 Tuples and Sequences

```
In [1]: t=12345,54321,'hello!'
In [2]: t[0]
Out[2]: 12345
In [3]: t
Out[3]: (12345, 54321, 'hello!')
In [4]: # Tuples may be nested:
In [5]: u=t,(1,2,3,4,5)
In [6]: u
Out[6]: ((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
In [7]: # Tuples are immutable:
In [8]: t[0]=88888
Traceback (most recent call last):
File "<ipython-input-8-186f923d18bf>", line 1, in <module>
t[0]=88888
TypeError: 'tuple' object does not support item assignment
In [9]: # but they can contain mutable objects:
In [10]: v=([1,2,3],[3,2,1])
In [11]: v
Out[11]: ([1, 2, 3], [3, 2, 1])

In [12]: myTuple=(1, "Bob", 0.75, 'Amy')
In [13]: myTuple
Out[13]: (1, 'Bob', 0.75, 'Amy')
In [14]: myTuple[3]
Out[14]: 'Amy'
In [15]: otherTuple=(123.45,999),myTuple
In [16]: otherTuple
Out[16]: ((123.45, 999), (1, 'Bob', 0.75, 'Amy'))
In [17]: myTuple[0]="Teddy"
Traceback (most recent call last):
File "<ipython-input-17-325e4f646ff0>", line 1, in <module>
myTuple[0]="Teddy"
TypeError: 'tuple' object does not support item assignment

In [18]: empty=()
In [19]: singleton='hello', # <-- note trailing comma
In [20]: len(empty)
Out[20]: 0
In [21]: len(singleton)
Out[21]: 1
In [22]: singleton
Out[22]: ('hello',)
In [23]: t
```

```
Out[23]: (12345, 54321, 'hello!')
In [24]: x,y,z=t
In [25]: t
Out[25]: (12345, 54321, 'hello!')
In [26]: x
Out[26]: 12345
In [27]: y
Out[27]: 54321
In [28]: z
Out[28]: 'hello!'

In [29]: myTuple
Out[29]: (1, 'Bob', 0.75, 'Amy')
In [30]: otherTuple
Out[30]: ((123.45, 999), (1, 'Bob', 0.75, 'Amy'))
In [31]: money, people=otherTuple
In [32]: money
Out[32]: (123.45, 999)
In [33]: people
Out[33]: (1, 'Bob', 0.75, 'Amy')
```

1.3.4 Sets

```
In [34]: # Sets
In [35]: basket={'apple','orange','apple','pear','orange','banana'}
In [36]: print(basket) # show that duplicates have been removed
{'orange', 'banana', 'apple', 'pear'}
In [37]: 'orange' in basket
Out[37]: True
In [38]: 'crabgrass' in basket
Out[38]: False
In [39]: # Demonstrate set operations on unique letters from two words
In [40]: a=set('abracadabra')
In [41]: b=set('alacazam')
In [42]: a # unique letters in a
Out[42]: {'a', 'b', 'c', 'd', 'r'}
In [43]: a-b # letters in a but not in b
Out[43]: {'b', 'd', 'r'}
In [44]: a | b # letters in either a or b
Out[44]: {'a', 'b', 'c', 'd', 'l', 'm', 'r', 'z'}
In [45]: a & b # letters in both a and b
Out[45]: {'a', 'c'}
In [46]: a ^ b # letters in a or b but not both
Out[46]: {'b', 'd', 'l', 'm', 'r', 'z'}

In [47]: setA=set('abc')
In [48]: setB=set('cba')
In [49]: setA
Out[49]: {'a', 'b', 'c'}
In [50]: setA-setB
Out[50]: set()
In [51]: setA | setB
Out[51]: {'a', 'b', 'c'}
In [52]: setA & setB
```

```
Out[52]: {'a', 'b', 'c'}
In [53]: setA ^ setB
Out[53]: set()

In [54]: a={x for x in 'abracadabra' if x not in 'abc'}
In [55]: a
Out[55]: {'d', 'r'}

In [56]: mySet = {0, 9.98, 5.04, 0.01, 0.9,0.095,-3.5}
In [57]: mySet
Out[57]: {-3.5, 0, 0.01, 0.095, 0.9, 5.04, 9.98}
In [58]: mySet = {0, 9.98, 5.04, 0.01, 0.9,0.095,-3.5,0,5.04}
In [59]: {x for x in mySet if x > 0.1}
Out[59]: {0.9, 5.04, 9.98}
```

1.3.5 Dictionaries

```
In [60]: # Dictionaries
In [61]: tel={'jack':4098, 'sape':4139}
In [62]: tel['guido']=4127
In [63]: tel
Out[63]: {'guido': 4127, 'jack': 4098, 'sape': 4139}
In [64]: tel['jack']
Out[64]: 4098
In [65]: del tel['sape']
In [66]: tel['irv']=4127
In [67]: tel
Out[67]: {'guido': 4127, 'irv': 4127, 'jack': 4098}
In [68]: list(tel.keys())
Out[68]: ['jack', 'guido', 'irv']
In [69]: sorted(tel.keys())
Out[69]: ['guido', 'irv', 'jack']
In [70]: 'guido' in tel
Out[70]: True
In [71]: 'jack' not in tel
Out[71]: False

In [72]: colorMap={'red':10,'green':20}
In [73]: colorMap['blue']=30
In [74]: colorMap
Out[74]: {'blue': 30, 'green': 20, 'red': 10}
In [75]: colorMap['blue']
Out[75]: 30
In [76]: del colorMap['red']
In [77]: colorMap['yellow']=50
In [78]: colorMap
Out[78]: {'blue': 30, 'green': 20, 'yellow': 50}
In [79]: list(colorMap.keys())
Out[79]: ['green', 'blue', 'yellow']
In [80]: sorted(colorMap.keys())
Out[80]: ['blue', 'green', 'yellow']
In [81]: 'yellow' in colorMap
Out[81]: True
In [82]: 'red' not in colorMap
```

```
Out[82]: True

In [84]: # dict() constructor
In [86]: dict([('sape',4139),('guido',4127),('jack',4098)])
Out[86]: {'guido': 4127, 'jack': 4098, 'sape': 4139}

In [87]: colorMap=dict()
In [88]: colorMap
Out[88]: {}
In [89]: colorMap=dict([('red',10),('blue',20),('green',30)])
In [90]: colorMap
Out[90]: {'blue': 20, 'green': 30, 'red': 10}

In [91]: {x:x**2 for x in (2,4,6)}
Out[91]: {2: 4, 4: 16, 6: 36}

In [92]: {color:(colorMap[color]/100) for color in colorMap}
Out[92]: {'blue': 0.2, 'green': 0.3, 'red': 0.1}

In [93]: dict(sape=4139,guido=4127,jack=4098)
Out[93]: {'guido': 4127, 'jack': 4098, 'sape': 4139}

In [94]: colorMap=dict()
In [96]: colorMap
Out[96]: {}
In [97]: colorMap=dict(red=10,green=20,blue=30)
In [98]: colorMap
Out[98]: {'blue': 30, 'green': 20, 'red': 10}
```

1.3.6 Looping Techniques

```
In [99]: # Looping Techniques
In [100]: knights={'gallahad':'the pure', 'robin':'the brave'}
In [101]: for k,v in knights.items():
   ...: print(k,v)
   ...:
gallahad the pure
robin the brave

In [104]: positions=dict(QB='Quarterback',RB='Running Back',DT='Defensive
Tackle',LB='Linebacker')
In [105]: for k,v in positions.items():
   ...: print(k,' - ',v)
   ...:
RB - Running Back
DT - Defensive Tackle
LB - Linebacker
QB - Quarterback

In [106]: for i,v in enumerate(['tic','tac','toe']):
   ...: print(i,v)
   ...:
0 tic
1 tac
```

```
2 toe

In [107]: for i,v in enumerate(myTuple):
   ...: print(i,v)
   ...:
0 1
1 Bob
2 0.75
3 Amy
In [108]: for i,v in enumerate(positions):
   ...: print(i,v)
   ...:
0 RB
1 DT
2 LB
3 QB
In [109]: for i,v in enumerate(positions.items()):
   ...: print(i,v)
   ...:
0 ('RB', 'Running Back')
1 ('DT', 'Defensive Tackle')
2 ('LB', 'Linebacker')
3 ('QB', 'Quarterback')

In [110]: questions=['name','quest','favorite color']
In [111]: answers=['lancelot','the holy grail','blue']
In [112]: for q,a in zip(questions,answers):
   ...: print("What is your {0}? It is {1}.".format(q,a))
   ...:
What is your name? It is lancelot.
What is your quest? It is the holy grail.
What is your favorite color? It is blue.

In [113]: abbreviations=["QB","LB"]
In [114]: positions=["Quarterback","Linebacker"]
In [115]: for a,p in zip(abbreviations,positions):
   ...: print(a,'-',p)
   ...:
QB - Quarterback
LB - Linebacker

In [116]: for i in reversed(range(1,10,2)):
   ...: print(i)
   ...:
97531

In [118]: for c in reversed(myTuple):
   ...: print(c)
   ...:
Amy
0.75
Bob
1
In [119]: myTuple
Out[119]: (1, 'Bob', 0.75, 'Amy')
```

```
In [120]: basket=['apple','orange','apple','pear','orange','banana']
In [121]: for f in sorted(set(basket)):
   ...: print(f)
   ...:
apple
banana
orange
pear

In [122]: for p in sorted(abbreviations):
   ...: print(p)
   ...:
LB
QB
In [123]: abbreviations
Out[123]: ['QB', 'LB']

In [124]: words=['cat','window','defenestrate']
In [125]: for w in words[:]: # Loop over a slice copy of the entire list.
   ...: if len(w) > 6:
   ...: words.insert(0,w)
   ...:
In [126]: words
Out[126]: ['defenestrate', 'cat', 'window', 'defenestrate']

In [128]: selectedItems=[True,True,True]
In [129]: selectedItems
Out[129]: [True, True, True]
In [135]: for si in selectedItems[:]:
   ...: if si:
   ...: selectedItems.append(False)
   ...:
In [136]: selectedItems
Out[136]: [True, True, True, False, False, False]
```

1.3.7 More on Conditions

```
In [137]: # More on Conditions
In [138]: string1, string2, string3='','Trondheim','Hammer Dance'
In [139]: non_null=string1 or string2 or string3
In [140]: non_null
Out[140]: 'Trondheim'

In [141]: int1, int2, int3 = 0, 1, 2
In [142]: non_null=int1 or int2 or int3
In [143]: non_null
Out[143]: 1
In [144]: int1, int2, int3 = 0, 5, 10
In [145]: non_null=int1 or int2 or int3
In [146]: non_null
Out[146]: 5
In [147]: non_null=int1 and int2 or int3
In [148]: non_null
```

```
Out[148]: 10
In [149]: non_null=int1 and int2 and int3
In [150]: non_null
Out[150]: 0
In [153]: 'A' < 'C' < 'Pa' < 'Py'
Out[153]: True
In [154]: (1,2,3,4) < (1,2,4)
Out[154]: True
In [155]: (1,2,3,4) < (1,2,3)
Out[155]: False
In [156]: (1,2,3,4) > (1,2,3)
Out[156]: True
```

1.3.8 Comparing Sequences and Other Types

```
In [157]: # Comparing Sequences and Other Types
In [158]: (1,2,3) < (1,2,4)
Out[158]: True
In [159]: [1,2,3] < [1,2,4]
Out[159]: True
In [160]: 'ABC' < 'C' < 'Pascal' < 'Python'
Out[160]: True
In [161]: (1,2,3,4) < (1,2,4)
Out[161]: True
In [162]: (1,2) < (1,2,-1)
Out[162]: True
In [163]: (1,2,3) == (1.0,2.0,3.0)
Out[163]: True
In [164]: (1,2,('aa','ab')) < (1,2,('abc','a'),4)
Out[164]: True

In [165]: 'A' < 'C' < 'Pa' < 'Py'
Out[165]: True
In [166]: (1,2,3,4) > (1,2,3)
Out[166]: True
In [167]: (1,2,(10,5)) > (1,2,(5,7,8))
Out[167]: True
```

1.4 Modules

fibo.py

```
# Fibonacci numbers module

def fib(n):     # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()

def fib2(n):    # return Fibonacci series up to n
    result = []
```

```
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
```

```
In [1]: import fibo
In [2]: fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
In [3]: fibo.fib2(100)
Out[3]: [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
In [4]: fibo.__name__
Out[4]: 'fibo'
In [5]: fib=fibo.fib
In [6]: fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

sum_up.py

```
# Summing (to n) module

def sum_up_to(n):
    msg = "Summing the numbers up to " + str(n) + " = "
    if n == 0:
        msg = msg + "0"
    elif n == 1:
        msg = msg + "1"
    else:
        result = 0
        for i in range(n+1):
            result = result + i
        msg = msg + str(result)
    print(msg)
```

```
In [7]: import sum_up
In [8]: sum_up.sum_up_to(5)
Summing the numbers up to 5 = 15
In [9]: sum_up.__name__
Out[9]: 'sum_up'
In [10]: sum_to=sum_up.sum_up_to
In [11]: sum_to(5)
Summing the numbers up to 5 = 15
In [12]: sum_to.__name__
Out[12]: 'sum_up_to'
```

1.4.1 More on Modules

```
In [1]: # More on Modules
In [2]: from fibo import fib,fib2
In [3]: fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377

In [4]: from sum_up import sum_up_to
In [5]: sum_up_to(5)
Summing the numbers up to 5 = 15
```

```
In [1]: from fibo import *
In [2]: fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377

In [3]: from sum_up import *
In [4]: sum_up_to(5)
Summing the numbers up to 5 = 15
```

fibo.py

```
# Fibonacci numbers module

def fib(n):     # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()

def fib2(n):    # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result

if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

```
In [1]: # Executing modules as scripts
In [2]: run fibo.py 50
1 1 2 3 5 8 13 21 34
In [3]: import fibo # script will not be executed
```

sum_up.py

```
# Summing (to n) module

def sum_up_to(n):
    msg = "Summing the numbers up to " + str(n) + " = "
    if n == 0:
        msg = msg + "0"
    elif n == 1:
        msg = msg + "1"
    else:
        result = 0
        for i in range(n+1):
            result = result + i
        msg = msg + str(result)
    print(msg)

if __name__ == "__main__":
    import sys
    sum_up_to(int(sys.argv[1]))
```

```
In [4]: run sum_up.py 5
Summing the numbers up to 5 = 15
In [5]: import sum_up # script will not be executed
```

1.4.2 The dir() Function

```
In [1]: # The dir() Function
In [2]: import fibo, sys
In [3]: dir(fibo)
Out[3]:
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__package__', '__spec__', 'fib', 'fib2']
In [4]: dir(sys)
Out[4]:
['__displayhook__', '__doc__', '__excepthook__', '__interactivehook__', '__loader__',
'__name__', '__package__', '__spec__', '__stderr__', '__stdin__', '__stdout__',
'_clear_type_cache', '_current_frames', '_debugmallocstats', '_getframe', '_home',
'_mercurial', '_xoptions', 'api_version', 'argv', 'base_exec_prefix', 'base_prefix',
'builtin_module_names', 'byteorder', 'call_tracing', 'callstats', 'copyright',
'displayhook', 'dllhandle', 'dont_write_bytecode', 'exc_info', 'excepthook',
'exec_prefix', 'executable', 'exit', 'flags', 'float_info', 'float_repr_style',
'getallocatedblocks', 'getcheckinterval', 'getdefaultencoding', 'getfilesystemencoding',
'getprofile', 'getrecursionlimit', 'getrefcount', 'getsizeof', 'getswitchinterval',
'gettrace', 'getwindowsversion', 'hash_info', 'hexversion', 'implementation', 'int_info',
'intern', 'maxsize', 'maxunicode', 'meta_path', 'modules', 'path', 'path_hooks',
'path_importer_cache', 'platform', 'prefix', 'ps1', 'ps2', 'ps3', 'setcheckinterval',
'setprofile', 'setrecursionlimit', 'setswitchinterval', 'settrace', 'stderr', 'stdin',
'stdout', 'thread_info', 'version', 'version_info', 'warnoptions', 'winver']

In [5]: import sum_up
In [6]: dir(sum_up)
Out[6]:
2
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__package__', '__spec__','sum_up_to']

In [1]: a=[1,2,3,4,5]
In [2]: import fibo
In [3]: fib=fibo.fib
In [4]: dir()
Out[4]:
['In', 'Out', '_', '__', '___', '__builtin__', '__builtins__', '__doc__', '__loader__',
'__name__', '__package__', '__spec__', '_dh', '_i', '_i1', '_i2', '_i3', '_i4', '_ih',
'_ii', '_iii', '_oh', '_sh', 'a', 'exit', 'fib', 'fibo', 'get_ipython', 'quit']

In [5]: import sum_up
In [6]: my_special_function=sum_up.sum_up_to
In [7]: dir()
Out[7]:
['In', 'Out', '_', '_4', '__', '___', '__builtin__', '__builtins__', '__doc__',
'__loader__', '__name__', '__package__', '__spec__', '_dh', '_i', '_i1', '_i2', '_i3',
'_i4', '_i5', '_i6', '_i7', '_ih', '_ii', '_iii', '_oh', '_sh', 'a', 'exit', 'fib',
'fibo', 'get_ipython', 'my_special_function', 'quit', 'sum_up']
```

```
In [8]: import builtins
In [9]: dir(builtins)
Out[9]:
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException',
'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError',
'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError',
'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError',
'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPointError',
'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning',
'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError',
'KeyboardInterrupt', 'LookupError', 'MemoryError', 'NameError', 'None',
'NotADirectoryError', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError',
'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'ReferenceError',
'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopIteration', 'SyntaxError',
'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True',
'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError',
'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError',
'Warning', 'WindowsError', 'ZeroDivisionError', '__IPYTHON__', '__IPYTHON__active',
'__build_class__', '__debug__', '__doc__', '__import__', '__loader__', '__name__',
'__package__', '__spec__', 'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'bytearray',
'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits',
'debugfile', 'delattr', 'dict', 'dir', 'divmod', 'dreload', 'enumerate', 'eval', 'evalsc',
'exec', 'filter', 'float', 'format', 'frozenset', 'get_ipython', 'getattr', 'globals',
'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass',
'iter', 'len', 'license', 'list', 'locals', 'map', 'max', 'memoryview', 'min', 'next',
'object', 'oct', 'open', 'open_in_spyder', 'ord', 'pow', 'print', 'property', 'range',
'repr', 'reversed', 'round', 'runfile', 'set', 'setattr', 'slice', 'sorted',
'staticmethod', 'str', 'sum','super', 'tuple', 'type', 'vars', 'zip']
```

1.5 Input and Output

1.5.1 Fancier Output Formatting

```
In [1]: # Input and Output
In [2]: # Fancier Output Formatting
In [3]: s='Hello, world.'
In [4]: str(s)
Out[4]: 'Hello, world.'
In [5]: repr(s)
Out[5]: "'Hello, world.'"
In [6]: str(1/7)
Out[6]: '0.14285714285714285'
In [7]: x=10*3.25
In [8]: y=200*200
In [9]: s='The value of x is '+repr(x)+', and y is '+repr(y)+'...'
In [10]: print(s)
The value of x is 32.5, and y is 40000...
In [11]: # The repr() of a string add string quotes and backslashes:
In [12]: hello='hello, world\n'
In [13]: hellos=repr(hello)
In [14]: print(hellos)
'hello, world\n'
In [15]: # The argument to repr() may be any Python object:
```

```
In [16]: repr((x,y,('spam','eggs')))
Out[16]: "(32.5, 40000, ('spam', 'eggs'))"

In [17]: c='SSE 691 - Engineering Data Visualization'
In [18]: str(c)
Out[18]: 'SSE 691 - Engineering Data Visualization'
In [19]: str(2/3.845)
Out[19]: '0.5201560468140441'
In [20]: a=2*11.05
In [21]: b=200*300
In [22]: c='The value of a is '+repr(a)+', and y is '+repr(b)+'...'
In [23]: print(c)
The value of a is 22.1, and y is 60000...
In [24]: c='goodbye, world\n'
In [25]: print(c)
goodbye, world
In [26]: cXform=repr(c)
In [27]: print(cXform)
'goodbye, world\n'
In [28]: repr((a,b,3/4,('whole','wheat')))
Out[28]: "(22.1, 60000, 0.75, ('whole', 'wheat'))"

In [29]: for x in range(1,11):
   ...: print(repr(x).rjust(2), repr(x*x).rjust(3),end=' ')
   ...: # Note use of 'end' on previous line
   ...: print(repr(x*x*x).rjust(4))
   ...:
1 1 1
2 4 8
3 9 27
4 16 64
5 25 125
6 36 216
7 49 343
8 64 512
9 81 729
10 100 1000
In [30]: for x in range(1,11):
   ...: print('{0:2d} {1:3d} {2:4d}'.format(x,x*x,x*x*x))
   ...:
1 1 1
2 4 8
3 9 27
4 16 64
5 25 125
6 36 216
7 49 343
8 64 512
9 81 729
10 100 1000

In [33]: for a in range(1,11):
   ...: print(repr(a).rjust(2),repr(10*a).rjust(3),end='~')
   ...: print(repr(100*a).rjust(4))
   ...:
```

```
1 10~ 100
2 20~ 200
3 30~ 300
4 40~ 400
5 50~ 500
6 60~ 600
7 70~ 700
8 80~ 800
9 90~ 900
10 100~1000
In [34]: for a in range(1,11):
    ...:     print('{0:2d} {1:3d}~{2:4d}'.format(a,10*a,100*a))
    ...:
1 10~ 100
2 20~ 200
3 30~ 300
4 40~ 400
5 50~ 500
6 60~ 600
7 70~ 700
8 80~ 800
9 90~ 900
10 100~1000

In [36]: '12'.zfill(5)
Out[36]: '00012'
In [37]: '-3.14'.zfill(7)
Out[37]: '-003.14'
In [38]: '3.14159265359'.zfill(5)
Out[38]: '3.14159265359'
In [39]: print('We are the {} who say "{}!"'.format('knights','Ni'))
We are the knights who say "Ni!"

In [40]: print('My kids names are: {}, {}, {}, {}, and
{}'.format('PJ','Teddy','Gabe','Charlie','Toby'))
My kids names are: PJ, Teddy, Gabe, Charlie, and Toby

In [41]: print('{0} and {1}'.format('spam','eggs'))
spam and eggs
In [42]: print('{1} and {0}'.format('spam','eggs'))
eggs and spam

In [43]: print('{0}, {1}, {2}'.format('first','second','third'))
first, second, third
In [44]: print('{1}, {2}, {0}'.format('first','second','third'))
second, third, first

In [45]: print('This {food} is {adjective}.'.format(food='spam',adjective='absolutely
horrible'))
This spam is absolutely horrible.

In [46]: print('I love {sport} during each
{season}.'.format(sport='football',season='fall'))
I love football during each fall.
```

```
In [47]: print('The story of {0}, {1}, and
{other}.'.format('Bill','Manfred',other='Georg'))
The story of Bill, Manfred, and Georg.

In [48]: print('My favorite sport is {0}, but I also like
{others}.'.format('football',others='basketball, baseball, softball, and hockey'))
My favorite sport is football, but I also like basketball, baseball, softball, and
hockey.

In [49]: import math
In [50]: print('The value of PI is approximately {}.'.format(math.pi))
The value of PI is approximately 3.141592653589793.
In [51]: print('The value of PI is approximately {!r}.'.format(math.pi))
The value of PI is approximately 3.141592653589793.
In [55]: print('The value of PI is approximately {0:.3f}.'.format(math.pi))
The value of PI is approximately 3.142.

In [57]: print('I made ${0:.2f} last night at work!'.format(9.5*20.87))
I made $198.27 last night at work!

In [58]: table={'Sjoerd':4127,'Jack':4098,'Dcab':7678}
In [59]: for name,phone in table.items():
   ...: print('{0:10} ==> {1:10d}'.format(name,phone))
   ...:
Dcab ==> 7678
Jack ==> 4098
Sjoerd ==> 4127

In [60]: ids={'Bob':123456,'Amy':987654,'PJ':321789}
In [61]: for name,iid in ids.items():
   ...: print('{0:8} -> {1:7}'.format(name,iid))
   ...:
PJ -> 321789
Amy -> 987654
Bob -> 123456

In [64]: table={'Sjoerd':4127,'Jack':4098,'Dcab':8637678}
In [65]: print('Jack: {0[Jack]:d}; Sjoerd: {0[Sjoerd]:d}; Dcab:
{0[Dcab]:d}'.format(table))
Jack: 4098; Sjoerd: 4127; Dcab: 8637678
In [66]: print('Jack: {Jack:d}; Sjoerd: {Sjoerd:d}; Dcab: {Dcab:d}'.format(**table))
Jack: 4098; Sjoerd: 4127; Dcab: 8637678

In [67]: print('Bob: {0[Bob]}; Amy: {0[Amy]}; PJ: {0[PJ]}'.format(ids))
Bob: 123456; Amy: 987654; PJ: 321789
In [68]: print('Bob: {Bob}; Amy: {Amy}; PJ: {PJ}'.format(**ids))
Bob: 123456; Amy: 987654; PJ: 321789
```

1.5.2 Reading and Writing Files

```
In [1]: # Reading and Writing Files
In [2]: # Methods of File Objects
In [8]: f=open('workfile.txt')
In [9]: f.read()
```

```
Out[9]: 'This is the entire file.\n'
In [10]: f.read()
Out[10]: ''
In [11]: f=open('workfile2.txt')
In [12]: f.readline()
Out[12]: 'This is the first line of the file.\n'
In [13]: f.readline()
Out[13]: 'Second line of the file\n'
In [14]: f.readline()
Out[14]: ''
In [15]: f=open('workfile2.txt')
In [16]: for line in f:
   ...: print(line,end='')
   ...:
This is the first line of the file.
Second line of the file

In [17]: # Read all the lines of the file
In [18]: f=open('workfile2.txt')
In [19]: f.readlines()
Out[19]: ['This is the first line of the file.\n', 'Second line of the file\n']
In [20]: list(f)
Out[20]: []
In [21]: f=open('workfile2.txt')
In [22]: list(f)
Out[22]: ['This is the first line of the file.\n', 'Second line of the file\n']
In [23]: f.write('This is a test\n')
Traceback (most recent call last):
File "<ipython-input-23-9c582d31bee7>", line 1, in <module>
f.write('This is a test\n')
UnsupportedOperation: not writable

In [27]: f=open('workfile2.txt','r+')
In [28]: f.write('This is a test\n')
Out[28]: 15
In [29]: f=open('workfile2.txt','r+')
In [30]: value=('the answer',42)
In [31]: s=str(value)
In [32]: f.write(s)
Out[32]: 18
In [33]: f=open('workfile3.txt','rb+')
In [34]: f.write(b'0123456789abcdef')
Out[34]: 16
In [35]: f.seek(5) # Go to the 6th byte in the file
Out[35]: 5
In [36]: f.read(1)
Out[36]: b'5'
In [37]: f.seek(-3,2) # Go to the 3rd byte before the end
Out[37]: 13
In [38]: f.read(1)
Out[38]: b'd'
In [39]: f.close()
In [40]: f.read()
Traceback (most recent call last):
File "<ipython-input-40-bacd0e0f09a3>", line 1, in <module>
```

```
f.read()
ValueError: read of closed file
In [41]: with open('workfile3.txt','r') as f:
   ...: read_data=f.read()
   ...: f.closed
   ...:
Out[41]: True

In [42]: # Saving structured data with json
In [44]: import json
In [45]: json.dumps([1,'simple','list'])
Out[45]: '[1, "simple", "list"]'
In [49]: f=open('workfile.txt','w')
In [50]: json.dump(x,f)
In [51]: f=open('workfile.txt')
In [52]: xx=json.load(f)
In [53]: xx
Out[53]: [1, 'simple', 'list']
```

1.6 Errors and Exceptions

1.6.1 Handling Exceptions

```
In [1]: # Errors and Exceptions
In [4]: # Handling Exceptions
In [6]: while True:
   ...: try:
   ...: x=int(input("Please enter a number: "))
   ...: break
   ...: except ValueError:
   ...: print("Oops! That was no valid number. Try again...")
   ...:
Please enter a number: a
Oops! That was no valid number. Try again...
Please enter a number: b
Oops! That was no valid number. Try again...
Please enter a number: 2

In [7]: a=[0,1,2]
In [13]: while True:
   ...: try:
   ...: print("Select any number from ",a," : ",end='')
   ...: sel=int(input())
   ...: n=a[sel]
   ...: print("You selected {0}".format(n))
   ...: break
   ...: except IndexError:
   ...: print("Not a valid number, try again...")
   ...:
Select any number from [0, 1, 2] :
3
Not a valid number, try again...
Select any number from [0, 1, 2] :
```

```
2
You selected 2

In [14]: import sys
In [15]: try:
    ...: f=open('myfile.txt')
    ...: s=f.readline()
    ...: i=int(s.strip())
    ...: except OSError as err:
    ...: print("OS error: {0}".format(err))
    ...: except ValueError:
    ...: print("Could not convert data to an integer.")
    ...: except:
    ...: print("Unexpected error:", sys.exc_info()[0])
    ...: raise
    ...:
OS error: [Errno 2] No such file or directory: 'myfile.txt'

In [16]: while True:
    ...: try:
    ...: print("Select any number from ",a," : ",end='')
    ...: sel=int(input())
    ...: n=a[sel]
    ...: print("You selected {0}".format(n))
    ...: break
    ...: except IndexError:
    ...: print("Not a valid number, try again...")
    ...: except ValueError:
    ...: print("Not a number, enter a number and try again...")
    ...:
Select any number from [0, 1, 2] :
--
Not a number, enter a number and try again...
Select any number from [0, 1, 2] :
p
Not a number, enter a number and try again...
Select any number from [0, 1, 2] :
9
Not a valid number, try again...
Select any number from [0, 1, 2] :
1
You selected 1
In [19]: for arg in ['workfile.txt', 'badFile.txt']:
    ...: try:
    ...: f=open(arg,'r')
    ...: except IOError:
    ...: print('cannot open', arg)
    ...: else:
    ...: print(arg,'has',len(f.readlines()),'lines')
    ...: f.close()
    ...:
workfile.txt has 1 lines
cannot open badFile.txt

In [20]: try:
```

```
   ...: raise Exception('spam','eggs')
   ...: except Exception as inst:
   ...: print(type(inst)) # the exception instance
   ...: print(inst.args) # arguments stored in .args
   ...: print(inst) # __str__ allows args to be printed directly,
   ...: # but may be overridden in exception subclasses
   ...: x,y=inst.args # unpack args
   ...: print('x =',x)
   ...: print('y =',y)
   ...:
<class 'Exception'>
('spam', 'eggs')
('spam', 'eggs')
x = spam
y = eggs
In [21]: def this_fails():
   ...: x=1/0
   ...:
In [22]: try:
   ...: this_fails()
   ...: except ZeroDivisionError as err:
   ...: print('Handling run-time error:',err)
   ...:
Handling run-time error: division by zero
In [23]: try:
   ...: this_fails()
   ...: except ZeroDivisionError as err:
   ...: print('Handling run-time error:',err)
   ...: print(err.args)
   ...:
Handling run-time error: division by zero
('division by zero',)
```

1.6.2 Raising Exceptions

```
In [24]: # Raising Exceptions
In [25]: raise NameError('HiThere')
Traceback (most recent call last):
File "<ipython-input-25-93385ba972b1>", line 1, in <module>
raise NameError('HiThere')
NameError: HiThere

In [26]: raise EnvironmentError('Something bad happened!!')
Traceback (most recent call last):
File "<ipython-input-26-810f9ed8dfea>", line 1, in <module>
raise EnvironmentError('Something bad happened!!')
OSError: Something bad happened!!

In [27]: try:
   ...: raise NameError('HiThere')
   ...: except NameError:
   ...: print('An exception flew by!')
   ...: raise
   ...:
```

```
Traceback (most recent call last):
File "<ipython-input-27-5d1f4d009277>", line 2, in <module>
raise NameError('HiThere')
NameError: HiThere
An exception flew by!

In [28]: try:
   ...: raise BrokenPipeError('Pipes have broken!!')
   ...: except BrokenPipeError:
   ...: print('Meh, let someone else fix the broken pipes.')
   ...: raise
   ...:
Traceback (most recent call last):
File "<ipython-input-28-aa0dff477b95>", line 2, in <module>
raise BrokenPipeError('Pipes have broken!!')
BrokenPipeError: Pipes have broken!!
Meh, let someone else fix the broken pipes.
```

1.6.3 User-defined Exceptions

```
In [29]: # User-defined Exceptions
In [30]: class MyError(Exception):
   ...: def __init__(self,value):
   ...: self.value=value
   ...: def __str__(self):
   ...: return repr(self.value)
   ...:
In [31]: try:
   ...: raise MyError(2*2)
   ...: except MyError as e:
   ...: print('My exception occurred, value:',e.value)
   ...:
My exception occurred, value: 4
In [32]: class Error(Exception):
   ...: """Base class for exceptions in this module."""
   ...: pass
   ...:
In [33]: class InputError(Error):
   ...: """Exception raised for errors in the input.
   ...:
   ...: Attributes:
   ...: expression -- input expression in which the error occurred
   ...: message -- explanation of the error
   ...: """
   ...: def __init__(self,expression,message):
   ...: self.expression=expression
   ...: self.message=message
   ...:
In [34]: class TransitionError(Error):
   ...: """Raised when an operation attempts a state transition that's not allowed.
   ...:
   ...: Attributes:
   ...: previous -- state at beginning of transition
   ...: next -- attempted new state
```

```
   ...: message -- explanation of why the specific transition is not allowed
   ...: """
   ...: def __init__(self,previous,next,message):
   ...: self.previous=previous
   ...: self.next=next
   ...: self.message=message
   ...:

In [35]: class SSE691Error(Error):
   ...: """Example exception class used for SSE691.
   ...:
   ...: Attributes:
   ...: color -- color of the error (red = error, yellow = warning)
   ...: message -- explanation of why the error has occurred
   ...: """
   ...: def __init__(self,color,message):
   ...: self.color=color
   ...: self.message=message
   ...:
In [36]: try:
   ...: raise SSE691Error('yellow',"Uh-oh! You failed!")
   ...: except SSE691Error as err:
   ...: print("The error color is {0}. -> {1}".format(err.color,err.message))
   ...:
The error color is yellow. -> Uh-oh! You failed!
```

1.6.4 Defining Clean-up Actions

```
In [37]: # Defining Clean-up Actions
In [38]: try:
   ...: raise KeyboardInterrupt
   ...: finally:
   ...: print('Goodbye, world!')
   ...:
Goodbye, world!
Traceback (most recent call last):
File "<ipython-input-38-b3375e85a6fd>", line 2, in <module>
raise KeyboardInterrupt
KeyboardInterrupt
In [39]: def divide(x,y):
   ...: try:
   ...: result=x/y
   ...: except ZeroDivisionError:
   ...: print("division by zero!")
   ...: else:
   ...: print("result is", result)
   ...: finally:
   ...: print("executing finally clause")
   ...:
In [40]: divide(2,1)
result is 2.0
executing finally clause
In [41]: divide(2,0)
division by zero!
```

```
executing finally clause
In [42]: divide("2","1")
executing finally clause
Traceback (most recent call last):
File "<ipython-input-42-da42c9ab7899>", line 1, in <module>
divide("2","1")
File "<ipython-input-39-3725a54eb46b>", line 3, in divide
result=x/y
TypeError: unsupported operand type(s) for /: 'str' and 'str'

In [43]: def pick_color(color):
   ...: try:
   ...: if (color == 'crimson') or (color == 'white'):
   ...: pass
   ...: elif color == 'black':
   ...: raise SSE691Error(color,'not supported')
   ...: else:
   ...: raise Exception
   ...: except SSE691Error as e:
   ...: print("'{0}' is a bad color! -> {1}".format(e.color,e))
   ...: finally:
   ...: print("The color entered was '{0}'".format(color))
   ...:
In [44]: pick_color('crimson')
The color entered was 'crimson'
In [45]: pick_color('black')
'black' is a bad color! -> ('black', 'not supported')
The color entered was 'black'
In [46]: pick_color('yellow')
Traceback (most recent call last):
File "<ipython-input-46-03a2990ab53e>", line 1, in <module>
pick_color('yellow')
File "<ipython-input-43-f1c41a07beaf>", line 8, in pick_color
raise Exception
Exception
The color entered was 'yellow'
```

1.7 Classes

1.7.1 Python Scopes and Namespaces

```
In [1]: # Classes
In [2]: # Scopes and Namespaces Example
In [3]: def scope_test():
   ...:   def do_local():
   ...:       spam="local spam"
   ...:   def do_nonlocal():
   ...:       nonlocal spam
   ...:       spam="nonlocal spam"
   ...:   def do_global():
   ...:       global spam
   ...:       spam="global spam"
   ...:   spam="test spam"
```

```
   ...:   do_local()
   ...:   print("After local assignment:",spam)
   ...:   do_nonlocal()
   ...:   print("After nonlocal assignment:",spam)
   ...:   do_global()
   ...:   print("After global assignment:",spam)
   ...: scope_test()
   ...: print("In global scope:",spam)
   ...:
After local assignment: test spam
After nonlocal assignment: nonlocal spam
After global assignment: nonlocal spam
In global scope: global spam

In [15]: def color_scope_test():
    ...:     def update_color_local():
    ...:         color='blue'
    ...:     def update_color_nonlocal():
    ...:         nonlocal color
    ...:         color='red'
    ...:     def update_color_global():
    ...:         global color
    ...:         color='green'
    ...:     color='yellow' # “this” scoped color is defined as yellow
    ...:     update_color_global() # color remains yellow, but global color changed to
                                   green
    ...:     print(color) # prints yellow
    ...:     update_color_local() # color remains yellow
    ...:     print(color) # prints yellow
    ...:     update_color_nonlocal() # color changes to red
    ...:     print(color) # prints red
    ...: # Switch to global scope
    ...: color_scope_test() # run scope test
    ...: print(color) # prints green
    ...:
yellow
yellow
red
green
```

1.7.2 First Look at Classes

```
In [4]: # A First Look at Classes
In [5]: # Class Objects
In [6]: class MyClass:
   ...: """A simple example class"""
   ...: i=12345
   ...: def f(self):
   ...: return 'hello world'
   ...:
In [7]: x=MyClass()
In [8]: x.i
Out[8]: 12345
In [9]: x.f
```

```
Out[9]: <bound method MyClass.f of <__main__.MyClass object at 0x0000000008ECF470>>
In [10]: print(x.f)
<bound method MyClass.f of <__main__.MyClass object at 0x0000000008ECF470>>
In [11]: x.f()
Out[11]: 'hello world'
In [12]: class MyClass:
   ...: """A simple example class"""
   ...: i=12345
   ...: def f(self):
   ...: return 'hello world'
   ...: def __init__(self):
   ...: self.data=[]
   ...:
In [13]: x.data
Traceback (most recent call last):
File "<ipython-input-13-5b202a7f83ba>", line 1, in <module>
x.data
AttributeError: 'MyClass' object has no attribute 'data'
In [14]: x=MyClass()
In [15]: x.data
Out[15]: []
In [16]: class Complex:
   ...: def __init__(self,realpart,imagpart):
   ...: self.r=realpart
   ...: self.i=imagpart
   ...:
In [17]: x=Complex(3.0, -4.5)
In [18]: x.r,x.i
Out[18]: (3.0, -4.5)

In [19]: class Student:
   ...: """A simple class representing a student."""
   ...: def __init__(self,name,id_num,phone,balance):
   ...: self.name=name
   ...: self.id=id_num
   ...: self.phone_num=phone
   ...: self.amount_owed=balance
   ...:
In [20]: student=Student('Bob',100005,'123-456-7890',10.25)
In [21]: student.id, student.name, student.phone_num, student.amount_owed
Out[21]: (100005, 'Bob', '123-456-7890', 10.25)

In [22]: # Instance Objects
In [23]: x.counter=1
In [24]: while x.counter < 10:
   ...: x.counter=x.counter*2
   ...: print(x.counter)
   ...:
16
In [25]: del x.counter
In [26]: x.counter
Traceback (most recent call last):
File "<ipython-input-26-f3fafb390b1e>", line 1, in <module>
x.counter
AttributeError: 'Complex' object has no attribute 'counter'
```

```
In [27]: # Method Objects
In [28]: x.f()
Traceback (most recent call last):
File "<ipython-input-28-9cdd69494efb>", line 1, in <module>
x.f()
AttributeError: 'Complex' object has no attribute 'f'
In [29]: x=MyClass()
In [30]: x.f()
Out[30]: 'hello world'
In [31]: xf=x.f
In [32]: while True:
   ...: print(xf())
   ...: break
   ...:
hello world

In [33]: # Class and Instance Variables
In [36]: class Dog:
   ...: kind='canine' # class variable shared by all instances
   ...: def __init__(self,name):
   ...: self.name=name # instance variable unique to each instance
   ...:
In [37]: d=Dog('Fido')
In [38]: e=Dog('Buddy')
In [39]: d.kind # shared by all dogs
Out[39]: 'canine'
In [40]: e.kind # shared by all dogs
Out[40]: 'canine'
In [41]: d.name # unique to d
Out[41]: 'Fido'
In [42]: e.name # unique to e
Out[42]: 'Buddy'

In [43]: del Student
In [44]: class Student:
   ...: """A simple class representing a student."""
   ...: university='Mercer' # class variable shared by all Student objects
   ...: def __init__(self,name,id_num,phone,balance):
   ...: self.name=name
   ...: self.id=id_num
   ...: self.phone_num=phone
   ...: self.amount_owed=balance
   ...:
In [45]: student
Out[45]: <__main__.Student at 0x8f5b438>
In [46]: bob=Student('Bob',100005,'123-456-7890',10.25)
In [47]: amy=Student('Amy',100006,'123-456-7891',0)
In [48]: bob.university
Out[48]: 'Mercer'
In [49]: amy.university
Out[49]: 'Mercer'
In [50]: bob.name
Out[50]: 'Bob'
In [51]: amy.name
```

```
Out[51]: 'Amy'

In [52]: class Dog:
   ...: tricks=[] # mistaken use of a class variable
   ...: def __init__(self,name):
   ...: self.name=name # instance variable unique to each instance
   ...: def add_trick(self,trick):
   ...: self.tricks.append(trick)
   ...:
In [53]: d=Dog('Fido')
In [54]: e=Dog('Buddy')
In [55]: d.add_trick('roll over')
In [56]: e.add_trick('play dead')
In [57]: d.tricks # unexpectedly shared by all dogs
Out[57]: ['roll over', 'play dead']
In [58]: # Correct design of Dog class
In [59]: class Dog:
   ...: def __init__(self,name):
   ...: self.name=name
   ...: self.tricks=[] # creates a new empty list for each dog
   ...: def add_trick(self,trick):
   ...: self.tricks.append(trick)
   ...:
In [60]: d=Dog('Fido')
In [61]: e=Dog('Buddy')
In [62]: d.add_trick('roll over')
In [63]: e.add_trick('play dead')
In [64]: d.tricks
Out[64]: ['roll over']
In [65]: e.tricks
Out[65]: ['play dead']
```

1.7.3 Inheritance

```
In [1]: # Inheritance
In [2]: class FirstClass:
   ...: def setdata(self,value):
   ...: self.data=value
   ...: def display(self):
   ...: print(self.data)
   ...:
In [3]: class SecondClass(FirstClass): # SecondClass inherits from FirstClass
   ...: def display(self): # Customize the display() function for SecondClass by
redefining the function
   ...: print('Current value = "%s"' % self.data)
   ...:
In [4]: first=FirstClass()
In [5]: first.setdata(5) # Create a FirstClass object and set its data
In [6]: first.display() # Calls FirstClass.display() function
5
In [7]: z=SecondClass() # Create a SecondClass object
In [8]: z.setdata(42) # Finds setdata in FirstClass
In [9]: z.display() # Finds overridden method in SecondClass
Current value = "42"
```

```
In [10]: class Animal:
   ...: def __init__(self,name):
   ...: self.name=name
   ...: def speak(self):
   ...: print('I am an Animal')
   ...:
In [11]: class Dog(Animal):
   ...: def speak(self):
   ...: print('I am a dog and I like to bark!')
   ...:
In [12]: animal=Animal('Lion') # Create an Animal object
In [13]: animal.speak() # Calls the Animal.speak() method
I am an Animal
In [14]: dog=Dog('Dog') # Create a Dog object using the Animal constructor
In [15]: dog.name # Access to the name attribute is provided through the Animal class
Out[15]: 'Dog'
In [16]: dog.speak() # Finds the overridden method in Dog class
I am a dog and I like to bark!
```

1.7.4 Iterators

```
In [1]: s='abc'
In [2]: it=iter(s)
In [3]: it
Out[3]: <str_iterator at 0x88993c8>
In [4]: next(it)
Out[4]: 'a'
In [5]: next(it)
Out[5]: 'b'
In [6]: next(it)
Out[6]: 'c'
In [7]: next(it)
Traceback (most recent call last):
File "<ipython-input-7-2cdb14c0d4d6>", line 1, in <module>
next(it)
StopIteration

In [8]: info=('Bob',45,'123 American Way',9.8)
In [9]: personInfo=iter(info)
In [10]: next(personInfo)
Out[10]: 'Bob'
In [11]: next(personInfo)
Out[11]: 45
In [12]: next(personInfo)
Out[12]: '123 American Way'
In [13]: next(personInfo)
Out[13]: 9.8
In [14]: next(personInfo)
Traceback (most recent call last):
File "<ipython-input-14-70e1878630e6>", line 1, in <module>
next(personInfo)
StopIteration
```

```
In [9]: class Reverse:
   ...: """Iterator for looping over a sequence backwards."""
   ...: def __init__(self,data):
   ...: self.data=data
   ...: self.index=len(data)
   ...: def __iter__(self):
   ...: return self
   ...: def __next__(self):
   ...: if self.index == 0:
   ...: raise StopIteration
   ...: self.index=self.index-1
   ...: return self.data[self.index]
   ...:
In [10]: rev=Reverse('spam')
In [11]: iter(rev)
Out[11]: <__main__.Reverse at 0x8c7ecc0>
In [12]: for char in rev:
   ...: print(char)
   ...:
Maps

In [13]: class ReverseByTwo(Reverse):
   ...: def __next__(self):
   ...: if (self.index == 0) or (self.index < 0):
   ...: raise StopIteration
   ...: self.index=self.index-2
   ...: if self.index < 0:
   ...: raise StopIteration
   ...: return self.data[self.index]
   ...:
In [14]: rev_by_two=ReverseByTwo('_!_s_s_e_c_c_u_s_')
In [15]: iter(rev_by_two)
Out[15]: <__main__.ReverseByTwo at 0x8d092e8>
In [16]: for char in rev_by_two:
   ...: print(char)
   ...:
success!
```

1.7.5 Generators

```
In [18]: # Generators
In [19]: def reverse(data):
   ...: for index in range(len(data)-1,-1,-1):
   ...: yield data[index]
   ...:
In [20]: for char in reverse('golf'):
   ...: print(char)
   ...:
flog

In [24]: def rev_by_two(data):
   ...: for i in range(len(data)-2,-1,-2):
   ...: if i < 0:
   ...: raise StopIteration
```

```
   ...: yield data[i]
   ...:
In [25]: for c in rev_by_two('_!_s_s_e_c_c_u_s_'):
   ...: print(c)
   ...:
success!
```

1.8 Standard Library

1.8.1 Operating System Interface

```
In [1]: # Brief Tour of the Standard Library
In [2]: # Operating System Interface
In [3]: import os
In [4]: os.getcwd() # Return the current working directory
Out[4]: 'Y:\\git\\MercerU\\SSE691\\Project_1\\Source\\Python 3 Tutorial'
In [5]: os.system('mkdir today') # Run the command mkdir in the system shell
Out[5]: 0
In [6]: os.chdir('./today') # Change current working directory
In [7]: os.getcwd()
Out[7]: 'Y:\\git\\MercerU\\SSE691\\Project_1\\Source\\Python 3 Tutorial\\today'

In [8]: os.chdir('../')
In [9]: os.getcwd()
Out[9]: 'Y:\\git\\MercerU\\SSE691\\Project_1\\Source\\Python 3 Tutorial'

In [10]: dir(os) # Returns a list of all module functions
Out[10]:
['F_OK', 'MutableMapping', 'O_APPEND', 'O_BINARY', 'O_CREAT', 'O_EXCL', 'O_NOINHERIT',
'O_RANDOM', 'O_RDONLY', 'O_RDWR', 'O_SEQUENTIAL', 'O_SHORT_LIVED', 'O_TEMPORARY',
'O_TEXT', 'O_TRUNC', 'O_WRONLY', 'P_DETACH', 'P_NOWAIT', 'P_NOWAITO', 'P_OVERLAY',
'P_WAIT', 'R_OK', 'SEEK_CUR', 'SEEK_END', 'SEEK_SET', 'TMP_MAX', 'W_OK', 'X_OK',
'_Environ', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__',
'__name__', '__package__', '__spec__', '_execvpe', '_exists', '_exit',
'_get_exports_list', '_putenv', '_unsetenv', '_wrap_close', 'abort', 'access', 'altsep',
'chdir', 'chmod', 'close', 'closerange', 'cpu_count', 'curdir', 'defpath',
'device_encoding', 'devnull', 'dup', 'dup2', 'environ', 'errno', 'error', 'execl',
'execle', 'execlp', 'execlpe', 'execv', 'execve', 'execvp', 'execvpe', 'extsep', 'fdopen',
'fsdecode', 'fsencode', 'fstat', 'fsync', 'get_exec_path', 'get_handle_inheritable',
'get_inheritable', 'get_terminal_size', 'getcwd', 'getcwdb', 'getenv', 'getlogin',
'getpid', 'getppid', 'isatty', 'kill', 'linesep', 'link', 'listdir', 'lseek', 'lstat',
'makedirs', 'mkdir', 'name', 'open', 'pardir', 'path', 'pathsep', 'pipe', 'popen',
'putenv', 'read', 'readlink', 'remove', 'removedirs', 'rename', 'renames', 'replace',
'rmdir', 'sep', 'set_handle_inheritable', 'set_inheritable', 'spawnl', 'spawnle',
'spawnv', 'spawnve', 'st', 'startfile', 'stat', 'stat_float_times', 'stat_result',
'statvfs_result', 'strerror', 'supports_bytes_environ', 'supports_dir_fd',
'supports_effective_ids', 'supports_fd', 'supports_follow_symlinks', 'symlink', 'sys',
'system', 'terminal_size', 'times', 'times_result', 'umask', 'uname_result', 'unlink',
'urandom', 'utime', 'waitpid', 'walk', 'write']
In [11]: help(os) # Returns an extensive manual page created from the module's docstrings
Help on module os:
NAME
os - OS routines for NT or Posix depending on what system we're on.
```

```
DESCRIPTION
This exports:
- all functions from posix, nt or ce, e.g. unlink, stat, etc.
- os.path is either posixpath or ntpath
- os.name is either 'posix', 'nt' or 'ce'.
- os.curdir is a string representing the current directory ('.' or ':')
- os.pardir is a string representing the parent directory ('..' or '::')
- os.sep is the (or a most common) pathname separator ('/' or ':' or '\\')
- os.extsep is the extension separator (always '.')
- os.altsep is the alternate pathname separator (None or '/')
- os.pathsep is the component separator used in $PATH etc
- os.linesep is the line separator in text files ('\r' or '\n' or '\r\n')
- os.defpath is the default search path for executables
- os.devnull is the file path of the null device ('/dev/null', etc.)
Programs that import and use 'os' stand a better chance of being
portable between different platforms. Of course, they must then
only use functions that are defined by all platforms (e.g., unlink
and opendir), and leave all pathname manipulation to os.path
(e.g., split and join).
CLASSES
builtins.Exception(builtins.BaseException)
builtins.OSError
builtins.tuple(builtins.object)
nt.times_result
nt.uname_result
etc.
etc.
...

In [12]: os.path
Out[12]: <module 'ntpath' from 'C:\\Program Files\\Anaconda3\\lib\\ntpath.py'>
In [13]: os.name
Out[13]: 'nt'
In [14]: os.cpu_count
Out[14]: <function nt.cpu_count>
In [15]: os.cpu_count()
Out[15]: 2
In [17]: os.getcwd
Out[17]: <function nt.getcwd>
In [18]: os.getcwd()
Out[18]: 'Y:\\git\\MercerU\\SSE691\\Project_1\\Source\\Python 3 Tutorial'
In [20]: os.system('rmdir today')
Out[20]: 0
```

1.8.2 File Wildcards

```
In [21]: # File Wildcards
In [22]: import glob
In [23]: glob.glob('*.py')
Out[23]: ['fibo.py', 'sum_up.py']

In [24]: glob.glob('*.txt')
Out[24]: ['workfile.txt', 'workfile2.txt', 'workfile3.txt']
```

1.8.3 Internet Access

```
In [25]: # Internet Access
In [26]: from urllib.request import urlopen
In [27]: with urlopen('http://tycho.usno.navy.mil/cgi-bin/timer.pl') as response:
   ...: for line in response:
   ...: line=line.decode('utf-8') # Decoding the binary data to text.
   ...: if 'EST' in line or 'EDT' in line: # Look for Eastern Time
   ...: print(line)
   ...:
Traceback (most recent call last):
File "<ipython-input-27-3b4e62eeb849>", line 1, in <module>
with urlopen('http://tycho.usno.navy.mil/cgi-bin/timer.pl') as response:
File "C:\Program Files\Anaconda3\lib\urllib\request.py", line 161, in urlopen
return opener.open(url, data, timeout)
File "C:\Program Files\Anaconda3\lib\urllib\request.py", line 463, in open
response = self._open(req, data)
File "C:\Program Files\Anaconda3\lib\urllib\request.py", line 481, in _open
'_open', req)
File "C:\Program Files\Anaconda3\lib\urllib\request.py", line 441, in _call_chain
result = func(*args)
File "C:\Program Files\Anaconda3\lib\urllib\request.py", line 1210, in http_open
return self.do_open(http.client.HTTPConnection, req)
File "C:\Program Files\Anaconda3\lib\urllib\request.py", line 1184, in do_open
raise URLError(err)
URLError: <urlopen error [WinError 10060] A connection attempt failed because the
connected party did not properly respond after a period of time, or established connection
failed because connected host has failed to respond>
In [28]: with urlopen('http://www.usno.navy.mil/USNO/time/display-clocks/simpletime') as
response:
   ...: for line in response:
   ...: line=line.decode('utf-8') # Decoding the binary data to text.
   ...: if 'EST' in line or 'EDT' in line: # Look for Eastern Time
   ...: print(line)
   ...:
if ( DST=="EDT"){
document.getElementById("eastern").innerHTML=tempString.substring(0,
tempString.lastIndexOf(" "))+" EDT";
document.getElementById("eastern").innerHTML=tempString.substring(0,
tempString.lastIndexOf(" "))+" EST";

In [33]: with urlopen('http://www.python.org/') as f:
   ...: print(f.read(300))
   ...:
b'<!doctype html>\n<!--[if lt IE 7]> <html class="no-js ie6 lt-ie7 lt-ie8 lt-ie9"> <!
[endif]-->\n<!--[if IE 7]> <html class="no-js ie7 lt-ie8 lt-ie9"> <!
[endif]-->\n<!--[if IE 8]> <html class="no-js ie8 lt-ie9"> <!
[endif]-->\n<!--[if gt IE 8]><!--><html class="no-js"'
```

1.8.4 Dates and Times

```
In [34]: # Dates and Times
In [35]: # dates are easily constructed and formatted
In [36]: from datetime import date
In [37]: now=date.today()
```

```
In [38]: now
Out[38]: datetime.date(2015, 9, 16)

In [40]: year=date.year
In [41]: year
Out[41]: <attribute 'year' of 'datetime.date' objects>
In [45]: print(date.min)
0001-01-01
In [46]: print(date.max)
9999-12-31
```

1.8.5 Multi-threading

```
In [1]: # Multi-threading
In [2]: import threading,zipfile
In [3]: def run_async_zip_test():
   ...: class AsyncZip(threading.Thread):
   ...: def __init__(self,infile,outfile):
   ...: threading.Thread.__init__(self)
   ...: self.infile=infile
   ...: self.outfile=outfile
   ...: def run(self):
   ...: f=zipfile.ZipFile(self.outfile,'w',zipfile.ZIP_DEFLATED)
   ...: f.write(self.infile)
   ...: f.close()
   ...: print('Finished background zip of:',self.infile)
   ...: background=AsyncZip('mydata.txt','myarchive.zip')
   ...: background.start()
   ...: print('The main program continues to run in foreground.')
   ...: background.join() # Wait for the background task to finish
   ...: print('Main program waited until background was done.')
   ...:
In [4]: run_async_zip_test()
The main program continues to run in foreground.
Finished background zip of: mydata.txt
Main program waited until background was done.

In [5]: import time
In [13]: def run_async_test(seed):
   ...: class AsyncTask(threading.Thread):
   ...: def __init__(self,num):
   ...: threading.Thread.__init__(self)
   ...: self.num=num
   ...: def run(self):
   ...: for x in range(self.num):
   ...: if (x%5)==0:
   ...: print(str(x),end=' ')
   ...: time.sleep(0.005)
   ...: task=AsyncTask(seed)
   ...: task.start()
   ...: print('Main is executing...')
   ...: task.join()
   ...: print()
   ...: print('Main waited for task to complete.')
```

```
   ...:
In [14]: run_async_test(100)
0 Main is executing...
5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95
Main waited for task to complete.
```

Non-Direct Activity Report

Date	Duration (minutes)	Specific Task / Activity
20-Aug-2015	38	Work on project #1
25-Aug-2015	99	Work on project #1
27-Aug-2015	92	Work on project #1
30-Aug-2015	130	Work on project #1
31-Aug-2015	36	Work on project #1
1-Sep-2015	146	Work on project #1
8-Sep-2015	162	Work on project #1
9-Sep-2015	315	Work on project #1
10-Sep-2015	143	Work on project #1
11-Sep-2015	360	Work on project #1
12-Sep-2015	78	Work on project #1
14-Sep-2015	130	Work on project #1
15-Sep-2015	270	Work on project #1
16-Sep-2015	120	Work on project #1
18-Sep-2015	220	Work on project #1
20-Sep-2015	210	Work on project #1
21-Sep-2015	60	Work on project #1
Sum for Report #1	**2609**	**/ 1500 (5 weeks @ 300/wk)**
Sum for Report #2		**/ 1500 (5 weeks @ 300/wk)**
Sum for Report #3		**/ 1500 (5 weeks @ 300/wk)**
Sum for Class	**2609**	**/ 4500 (15 weeks @ 300/wk)**