Practical 2
Strings and Lists

Learning Objectives

1. Define and use more complex datatypes (strings and lists) and variations on control structures
2. Use slicing and indexing to access elements in a list
3. Use a supplied Python package to provide random number options
4. Understand and implement simple Monte Carlo algorithms

Overview

In this practical we will enter and modify programs to work with and explore strings and lists. We will then use random numbers to select list items without replacement.

The final two tasks will implement two Monte Carlo algorithms: calculating Pi and tossing coins.

Tasks

1. Setting up for Practical 2

Login to the computers as in Practical 1. Within your home directory (/home/student_id) you should have the following structure:

- FOP
  - Prac1
  - **Prac2**
  - Prac3
  - Prac4
  - Prac5
  - Prac6
  - Prac7
  - Prac8
  - Prac9
  - Prac10
  - Prac11

Type `ls FOP` from your home directory to check your directory structure.

We will be working in the Prac2 directory today. If you do not have the directory structure correct, your tutor can help you to rearrange it.

Copy the README file from your Prac1 directory to your Prac2 directory. Use `cp ../Prac1/README .` from within the Prac2 directory, them `vim README` to edit. Update the README to refer to Prac 2 and include the correct date.

2. Everybody loves string(s)!

Strings are very important in Python and the language provides powerful options to manipulate strings. The code below shows three different approaches to printing out a string in reverse: while, for and split. Type this code in and test it out…

```
#
# strings1.py: Read in a string and print it in reverse
#              using a loop and a method call

instring = input('Enter a string... ')
# *** add upper and duplicating code here

# reversing with a while loop

print('Reversed string is : ', end='')
index = len(instring)-1
while index >= 0:
    print(instring[index], end='')
    index = index - 1
print()

# reversing with a range loop

print('Reversed string is : ', end='')
for index in range(len(instring)-1, -1, -1):
    print(instring[index], end='')
print()

# reversing with slicing

print('Reversed string is :', instring[::-1])
```

Next, copy strings1.py to strings2.py and make the following changes…

1. Change the start, stop and step values in each section to print the string forwards (instead of in reverse).

Add code after the *** comment to do the following:

2. Convert the string to upper case (so that ‘abcd’ becomes ‘ABCD’)
3. Duplicate the string (ABCD becomes ABCDABCD)

Modify each of the three approaches (while, range and slicing) in strings2.py to:

4. Print out every second character (ABCDABCD becomes ACAC)
5. Print out every second character, excluding the first and last (ABCDABCD -> BDB)

3. Bucket List

The Bucket List is a movie from 2007 where two men work through a list of things they want to do in life. This task will have you work with their bucket list.

You will define bucket1 directly as a list, then append three values. Deleting a value uses the index of the item in the list (e.g. del bucket1[5]) We then create a second list and make a new list bucket fomr bucket1 + bucket2. Finally we insert a new item and print out the buckets.

```
#
# bucket1.py - use a python list for items in a bucket list
#

print('\nBUCKET LIST\n')

bucket1 = ['Witness something truly majestic',
           'Help a complete stranger',
           'Laugh until I cry','Drive a Shelby Mustang']

bucket1.append('Kiss the most beautiful girl in the world')
bucket1.append('Get a tattoo')
bucket1.append('Skydiving')
del bucket1[5]

bucket2 = ['Visit Stonehenge',
            'Drive a motorcycle on the Great Wall of China',
            'Go on a Safari','Visit the Taj Mahal',
            'Sit on the Great Egyptian Pyramids',
            'Find the Joy in your life']

print('Bucket 1: ', bucket1)
print('Bucket 2: ', bucket2)

bucket = bucket1 + bucket2
bucket.insert(5, 'Get a tattoo')

print('Joined buckets: ', bucket)

print('\nNicer formatting....\n')

for item in bucket:
    print(item)
```

We will now create a bucket list builder to interactively create a new bucket list…

```
#
# bucket2.py - bucket list builder
#
```

```
print('\nBUCKET LIST BUILDER\n')

bucket = []

choice = input('Enter selection: e(X)it, (A)dd, (L)ist...')

while choice[0] != 'X':
    if choice[0] == 'A':
        print('Enter list item... ')
        newitem = input()
        bucket.append(newitem)
    elif choice[0] == 'L':
        for item in bucket:
            print(item)
    else:
        print('Invalid selection.')
    choice = input('Enter selection: e(X)it, (A)dd, (L)ist..')

print('\nGOODBYE!\n')
```

Modify the code to:

1. Accept lowercase as well as uppercase letters as choices (hint: `upper()`)
2. Provide an option for deleting items (hint: `del bucket[int(delitem)]`)

4. Assorted Creams

This program generates a list of items then prints out each selected item before deleting it. We are using a “without replacement” approach as the selected items are no longer part of the pool to be selected.

```
#
# assorted.py - selecting random biscuits from a pack
#

import random

biscuits = []

biscuits.extend(['Monte Carlo']*7)
biscuits.extend(['Shortbread Cream']*7)
biscuits.extend(['Delta Cream']*6)
biscuits.extend(['Orange Slice']*6)
biscuits.extend(['Kingston']*5)

print('\nASSORTED CREAMS\n')

print('There are ', len(biscuits), ' biscuits in the pack.')

print('\n', biscuits, '\n')
```

```
more = input('\nWould you like a biscuit (Y/N)... ')

while more != 'N':
    choice = random.randint(0,len(biscuits)-1)
    print('Your biscuit is : ', biscuits[choice])
    del biscuits[choice]
    more = input('\nWould you like a biscuit (Y/N)...')

print('\nThere are ', len(biscuits), ' biscuits left.')
print('\n', biscuits, '\n')
```

Modify the code to check if the pack is empty before continuing the loop.

5. Method of Darts

The lecture slides included code for calculating Pi using the Method of Darts. Type the code in as darts.py and explore the accuracy you can achieve.

6. Tossing Coins

In this program we will toss a coin 1000 times and see how many heads or tails we count.

```
#
# cointoss.py - simulate tossing a coin multiple times
#

import random

coin = ['heads','tails']
heads = 0
tails = 0
trials = 1000

print('\nCOIN TOSS\n')

for index in range(trials):
    if random.choice(coin) == 'heads':
        heads = heads + 1
    else:
        tails = tails + 1

print('\nThere were ', heads, ' heads & ', tails, ' tails.\n')
```

Modify the code to ask the user to enter the number of tosses.

Submission

All of your work for this week's practical should be submitted via Blackboard using the link on the Assessment page. This should be done as a single "zipped" file (see Prac 1 for directions). This is the file to submit through Blackboard. There are no direct marks for these submissions, but they will be taken into account when finalising your mark.

NOTE: when you leave the class you should "logout" through the menus. Do not shut down the machine! Also, lock the machine if you need to leave it.

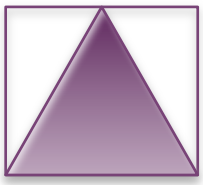
Reflection

1. Knowledge: What is the difference between append and extend when working with lists? Use Google and/or the Python documentation to find the answer.
2. Comprehension: What random methods would we use to: 1) generate floating point numbers, 2) generate integers, 3) choose between items in a list?
3. Application: How would you set up assorted.py to hold values to represent a small box of Smarties? There are 24 Smarties in a box, colours are yellow, green, pink, orange, blue, red, purple and brown. Each equally likely.
4. Analysis: Why did we test against choice[0] in bucket2.py? (as opposed not choice)
5. Synthesis: How would you modify the assorted.py code to make it "with replacement"?
6. Evaluation: Which of the three approaches for reversing a string do you recommend, and why?

Extension

For those who want to explore a bit more of the topics covered in this practical. Note that the challenges are not assessed but may form part of the mid-semester test, prac tests or exam.

1. Modify cointoss.py to model a six-sided dice being thrown.
2. Modify assorted.py to have the list represent a pack of playing cards (instead of biscuits). Select and print out two 5-card hands.
3. Modify darts.py to calculate the area of a triangle: coords (0,0), (1,0), (0.5,1)



4. Consider the image of a bingo card that is attached to the practical. Write a program to generate a bingo card. (Note: this wont be easy!)