Introduction to Programming

Department of Computer Science and Information Systems

Lecturers: Tingting Han and Steve Maybank

sjmaybank@dcs.bbk.ac.uk

Autumn 2019 and Spring 2020

Week 11: Lists

Mock In Lab Test: FizzBuzz

- In the `main()` function, there are 3 steps:
  - 1. Call `getEndInteger` to obtain an integer `end`.
  - 2. Call `numFizzBuzz` with argument `end` to get another integer `numFB`, which is the number of "FizzBuzz".
  - 3. Call `printFizzBuzz` with argument `numFB`

```
end = getEndInteger()

numFB = numFizzBuzz(end)

printFizzBuzz(numFB)
```

Mock In Lab Test: getEndInteger

- Define the function `getEndInteger()`
  - prints the prompt "Please enter the ending integer of the sequence (>=1):"
  - If the integer is less than 1
    - Print error message "Error: a number greater than or equal to 1 is required. Try Again."
    - Repeat until a valid input is entered
  - Return the valid integer

```
def getEndInteger() :
        end = int(input("Please enter the ending integer of the sequence (>=1):"))
        while end < 1:
                print("Error: a number greater than or equal to 1 is required. Try Again.")
                end = int(input("Please enter an integer greater than or equal to 1: "))
        return end
```

Mock In Lab Test: numFizzBuzz

- Define the function `numFizzBuzz(endNumber)`
  - It iterates through the integers from 1 to endNumber
  - For multiples of 15, print "FizzBuzz"
  - For multiples of 3 but not of 5, print "Fizz"
  - For multiples of 5 but not of 3, print "Buzz"
  - In all other cases, print the number itself
  - Return the number of appearances of "FizzBuzz"

```
def numFizzBuzz(endNumber) :
    numFB = 0
    for i in range(1, endNumber+1):
        if i % 15 == 0 :
            print("FizzBuzz")
            numFB = numFB + 1
        elif i % 3 == 0 :
            print("Fizz")
        elif i % 5 == 0 :
            print("Buzz")
        else :
            print(i)
    return numFB
```

Mock In Lab Test: printFizzBuzz

- Define the function `printFizzBuzz(num)`
  - The function prints out `num` times of "FizzBuzz"
    - If `num` is 0, print out "No FizzBuzz found"
    - Otherwise, print out a field width of at least 30 characters

```
def printFizzBuzz(num) :
    if num == 0 :
        print("No FizzBuzz found")
    else:
        string = "FizzBuzz" * num
        print("%30s"  % string)
```

Mock In Lab Test: main

- Define the function `main()`

```
def main():
        #Step 1
        end = getEndInteger()
        #Step 2
        numFB = numFizzBuzz(end)
        #Step 3
        printFizzBuzz(numFB)
```

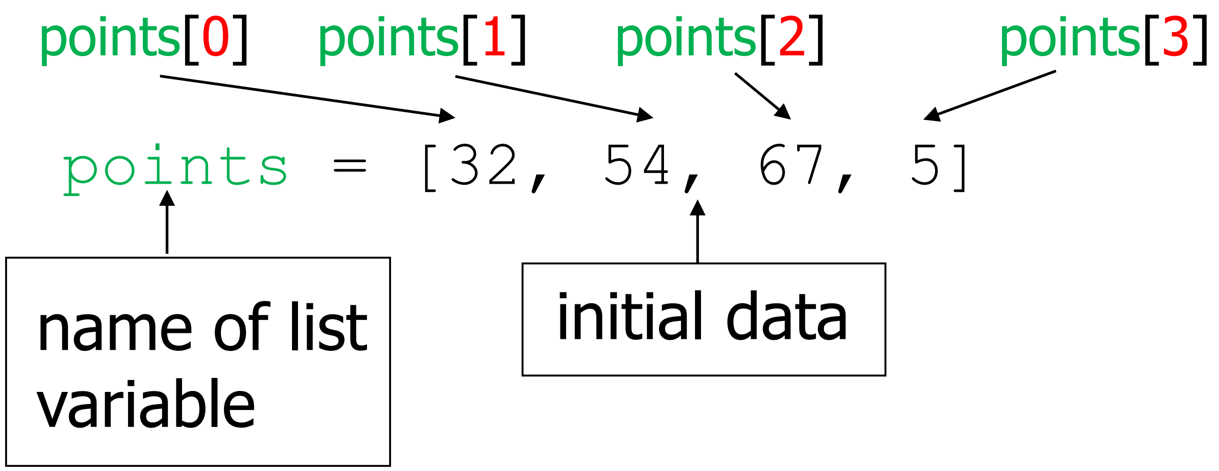
- Make three calls of `main()`

```
main()  #enter 14  – no FizzBuzz
main()  #enter 20 – one FizzBuzz, leaving space
main()  #enter 70 – Four FizzBuzz
```

Lists

- A mechanism for collecting together multiple values.
- A way of allocating names to multiple variables.

Creation of a List



```
points = [32, 54, 67, 5]
```

Names of variables: points[0], points[1], points[2], points[3]
The numbers 0, 1, 2, 3 are indices

```
print(points[2])
```
# prints 67
```
print(points)
```
# prints entire list [32, 54, 67, 5]

Indices and Length

```
points = [32, 54, 67, 5]
points[2] = 10
print(points[2])     #print 67 or 10?
# prints 10
print(len(points))     #print 3 or 4?
# there are 4 values, prints 4 as the list’s length
print(points[-1])
# prints 5
```

Allowed negative indices are

-1 to -len(points),

i.e. -1, -2, -3, -4

Lists and for Loops

- Both these loops have the same effect

```
for i in range(len(points)) :
    print(points[i])

for element in points :
    print(element)
```

# the variable name element can be changed, e.g.

```
for eachGame in points :
    print(eachGames)
```

Bounds Error

```
points = [32, 54, 67, 5]
points[len(points)] = 3
```

# bounds error

# A bounds error causes a run time exception.

# The error is not detected at compile time.

List References

```
scores = [10, 9, 7, 4, 5]
points = scores
scores[3] = 8
print(points[3])
```

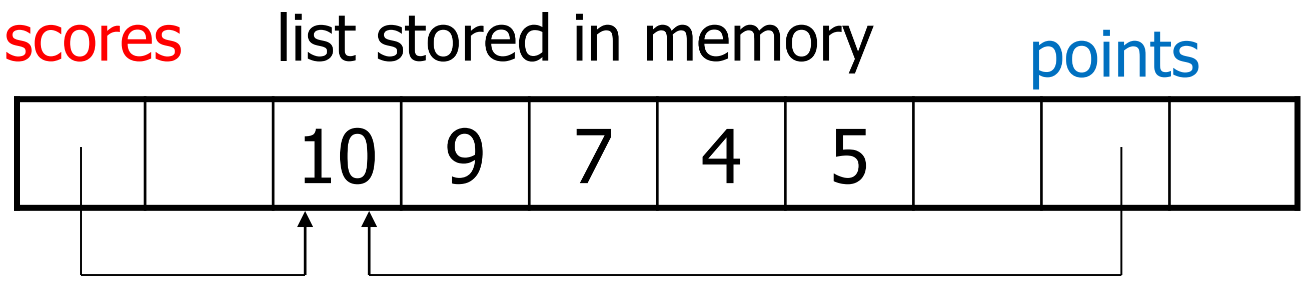
# prints 8!

# A list variable such as points is a pointer to the place
# in memory where the list is stored.

# points and scores both reference the same list of
# numbers in memory.

The List Variable as a Pointer

```
scores = [10, 9, 7, 4, 5]
points = scores
```



The value of the variable scores is a pointer to the list.
The value of the variable points is a pointer to the same list.

Example

```
things = [1, 2, "text", range]
```

# Correct but not recommended. Where possible,
# list elements should have the same type.

Appending an Element

```
friends = []    # empty list
friends.append("Emily")
friends.append("Bob")
print(friends)
# prints ["Emily", "Bob"]
```

Inserting an Element

```
friends = ["Harry", "Bob"]
friends.insert(1, "Cindy")
print(friends)
```

# prints ["Harry", "Cindy", "Bob"]

```
friends.insert(i, "Emily")
```

# i = 0, 1, 2: insert "Emily" before the element with index i

# i = 3: insert "Emily" after "Bob" (same as append)

Finding an Element

```
if "Cindy" in friends :
    print("She's a friend")
```

```
friends = ["Harry", "Emily", "Emily"]
n = friends.index("Emily")
```

# index of first occurrence: 1

```
n = friends.index("Tom")
```

# error, run time exception

Removing an Element

```
friends = ["Harry", "Cindy", "Emily", "Bob"]
name = friends.pop(1)
print(name)
# prints "Cindy"
print(friends)
# prints ["Harry", "Emily", "Bob"]

friends.pop()  # remove the last element "Bob"
print(friends)
# prints ["Harry", "Emily"]
```

Removing Matches

- Remove all strings of length < 4 from the list words

```
words = ['elephant', 'cat', 'ox', 'dolphin', 'bee']
i = 0
while i < len(words) :    # len(words) is the length of the list words
   word = words[i]
   if len(word) < 4 :  # len(word) is the length of the string word
       words.pop(i)
   else :
       i = i+1
```

Removing Matches 2

- Remove all strings of length < 4 from the list words

```
words = ['elephant', 'cat', 'ox', 'dolphin', 'bee']
for i in range(len(words)):
   word = words[i]
   if len(word) < 4 :
       words.pop(i)
```

# This code fails but why?

Reading Input

```
points = []
print("Please enter points, Q to quit: ")
userInput = input("")
while userInput != "Q" :
    points.append(float(userInput))
    userInput = input("")
```

# The Shell looks like this

```
Please enter points, Q to quit:
32
29
67.5
Q
```

Quiz Score

- A final quiz score is computed by adding all the scores except for the lowest two.
- For example, if the scores are

  8, 4, 7, 8.5, 9.5, 7, 5, 10

  then the final score is 50.
- Write a program to compute the final score in this way.

Solution

```
def calScoreSum(scores):

    if len(scores) < 3 :  #check whether there are at least three scores
        print("Too few scores. Please enter at least two scores.")
    else:
        scoreSum = 0       #sum of scores of ALL scores
        low1 = 101         #the lowest score, initially exceeding 100 (the max quiz score)
        low2 = 101         #the second lowest score

        for i in range(0, len(scores)):
            scoreSum = scoreSum + scores[i]  #adding up all scores
            if scores[i] < low1:   #replacing the lowest and second lowest when needed
                low2 = low1
                low1 = scores[i]
            elif scores[i] < low2:
                low2 = scores[i]

        scoreSum = scoreSum - low1 - low2
        print("The sum of scores is", scoreSum)
```

Testing

```
scores = [8,4,7,8.5,9.5,7,5,10]
calScoreSum(scores)

scores = [8,4,4,4,4]
calScoreSum(scores)

scores = [8,4]
calScoreSum(scores)

scores = [9]
calScoreSum(scores)

scores = []
calScoreSum(scores)
```

Insert

- Suppose that points is a sorted list of integers. Write a function to insert a new value into its proper position.

Solution

```
## sortInsert inserts a new value into the proper position in a sorted list.
#@param sortedIntList: a list of integers. We assume this list of integers is sorted in
#                                  an ascending order.
#@param newInt: a new integer to be inserted
#@return: the list of integers with the new value inserted.
#Author: T. Han
#Date: 8.12.2017

def sortInsert(sortedIntList, newInt):

    for i in range(len(sortedIntList)):
        if newInt < sortedIntList[i]:  # the newInt is smaller than the ith element
            sortedIntList.insert(i,newInt)
            return sortedIntList
    sortedIntList.insert(len(sortedIntList),newInt)  # the newInt is the largest
    return sortedIntList
```

Testing

```
sortedIntList = [0,2,4,6]

print(sortInsert(sortedIntList,0))
print(sortInsert(sortedIntList,1))
print(sortInsert(sortedIntList,3))
print(sortInsert(sortedIntList,4))
print(sortInsert(sortedIntList,6))
print(sortInsert(sortedIntList,7))

print(sortInsert([0,2,4,6],0))
print(sortInsert([0,2,4,6],1))
print(sortInsert([0,2,4,6],3))
print(sortInsert([0,2,4,6],4))
print(sortInsert([0,2,4,6],6))
print(sortInsert([0,2,4,6],7))
```