# COMP10001 Foundations of Computing Project 1 Review; Mid-semester Test Preparations

Semester 2, 2016

Chris Leckie

September 4, 2016

# Mid-Semester Test Outline I

- Everything up to the end of week 5 for the lectures, and week 6 for the workshops is examinable for the mid-semester test
- All Python coding questions
- 45 minutes in duration

# Mid-Semester Test Outline II

- Pen-and-paper test (no calculators, no Grok, mobile phones, ..., no "cheat sheet")
- Make sure to bring your **student card** and come on time (the test will start at 16:20 **sharp**)
- Please place your student card on the desk so we can check it
- You can use pencil, but please write clearly
- The test will be printed one-sided so you can use the blank sides for scribble, but we will only mark material in the lined sections

# Common Mistakes

We are quite tough in marking:

- Code should be syntactically correct
- Make sure **types** are correct
- Make sure order of values in lists are correct
- Make sure **strings** are quoted (and non-strings are not quoted)
- Get range limits correct
- Ensure variables are initialised
- Don't confuse `return` and `print`

# Question 1 (sample test)

Evaluate the following expressions, and provide the output in each case.

```
1/2 + 1
```

# Question 1 (sample test)

Evaluate the following expressions, and provide the output in each case.

```
1/2 + 1
```

```
1.5
```

# Question 1

Evaluate the following expressions, and provide the output in each case.

```
sorted({'b':2, 'a':1}.items())
```

# Question 1

Evaluate the following expressions, and provide the output in each case.

```
sorted({'b':2, 'a':1}.items())
```

```
[('a', 1), ('b', 2)]
```

# Question 1

Evaluate the following expressions, and provide the output in each case.

```
bool("winnie" and "balloon" or False)
```

# Question 1

Evaluate the following expressions, and provide the output in each case.

```
bool("winnie" and "balloon" or False)
```

```
True
```

# Question 1

Evaluate the following expressions, and provide the output in each case.

```
['a','man','a','pan'][1][-1]
```

# Question 1

Evaluate the following expressions, and provide the output in each case.

```
['a','man','a','pan'][1][-1]
```

```
'n'
```

# More of the Same

- `list("banana")[3:5]`

# More of the Same

- ```
  list("banana")[3:5]
  ```

  ```
  ['a', 'n']
  ```
  ```

# More of the Same

- ```
  list("banana")[3:5]
  ```

  ```
  ['a', 'n']
  ```

- ```
  bool(1 and "the same")
  ```

# More of the Same

- ```
  list("banana")[3:5]
  ```

  ```
  ['a', 'n']
  ```

- ```
  bool(1 and "the same")
  ```

  ```
  True
  ```

# More of the Same

- ```
  list("banana")[3:5]
  ```

  ```
  ['a', 'n']
  ```

- ```
  bool(1 and "the same")
  ```

  ```
  True
  ```

- ```
  sorted({1:"one", 2:"two"}.values())[-1]
  ```
  ```

# More of the Same

- ```
  list("banana")[3:5]
  ```

  ```
  ['a', 'n']
  ```

- ```
  bool(1 and "the same")
  ```

  ```
  True
  ```

- ```
  sorted({1:"one", 2:"two"}.values())[-1]
  ```

  ```
  'two'
  ```
  ```

# Question 2

Answer the following questions based on the following function, and the indicated line numbers:

```
1 def fun(a):
2     for i in range(len(a)):
3         if a[i] < 'a' or a[i] > 'z':
4             return False
5         if a[i] in a[:i]:
6             return False
7     return True
```

(a) What is line 3 testing for?

# Question 2

Answer the following questions based on the following function, and the indicated line numbers:

```
1 def fun(a):
2     for i in range(len(a)):
3         if a[i] < 'a' or a[i] > 'z':
4             return False
5         if a[i] in a[:i]:
6             return False
7     return True
```

(a) What is line 3 testing for?

*Whether the 'i'th letter of 'a' is not a lower-case letter of the alphabet*

# Question 2

Answer the following questions based on the following function, and the indicated line numbers:

```
1 def fun(a):
2     for i in range(len(a)):
3         if a[i] < 'a' or a[i] > 'z':
4             return False
5         if a[i] in a[:i]:
6             return False
7     return True
```

(b) What is line 5 testing for?

# Question 2

Answer the following questions based on the following function, and the indicated line numbers:

```
1 def fun(a):
2     for i in range(len(a)):
3         if a[i] < 'a' or a[i] > 'z':
4             return False
5         if a[i] in a[:i]:
6             return False
7     return True
```

(b) What is line 5 testing for?

*Whether the 'i'th letter of 'a' also occurs in the letters preceding it*

# Question 2

Answer the following questions based on the following function, and the indicated line numbers:

```
1 def fun(a):
2     for i in range(len(a)):
3         if a[i] < 'a' or a[i] > 'z':
4             return False
5         if a[i] in a[:i]:
6             return False
7     return True
```

(c) What is the overall purpose of the function?

# Question 2

Answer the following questions based on the following function, and the indicated line numbers:

```
1  def fun(a):
2      for i in range(len(a)):
3          if a[i] < 'a' or a[i] > 'z':
4              return False
5          if a[i] in a[:i]:
6              return False
7      return True
```

(c) What is the overall purpose of the function?

*The function tests whether 'a' is made up of all lower-case letters, without repeated letters*

# A Variant on the Same Theme

On completion of execution of the following code:

```
nums = [1,3,5]
i = j = 0
while nums:
    j += nums.pop()
    i += 1
j /= i
```

What is the value of each of the following variables:

# A Variant on the Same Theme

On completion of execution of the following code:

```
nums = [1,3,5]
i = j = 0
while nums:
    j += nums.pop()
    i += 1
j /= i
```

What is the value of each of the following variables:

- `i`

# A Variant on the Same Theme

On completion of execution of the following code:

```
nums = [1,3,5]
i = j = 0
while nums:
    j += nums.pop()
    i += 1
j /= i
```

What is the value of each of the following variables:

- `i = 3`

# A Variant on the Same Theme

On completion of execution of the following code:

```
nums = [1,3,5]
i = j = 0
while nums:
    j += nums.pop()
    i += 1
j /= i
```

What is the value of each of the following variables:

- `i = 3`
- `j`

# A Variant on the Same Theme

On completion of execution of the following code:

```
nums = [1,3,5]
i = j = 0
while nums:
    j += nums.pop()
    i += 1
j /= i
```

What is the value of each of the following variables:

- `i = 3`
- `j = 3`

# A Variant on the Same Theme

On completion of execution of the following code:

```
nums = [1,3,5]
i = j = 0
while nums:
    j += nums.pop()
    i += 1
j /= i
```

What is the value of each of the following variables:

- `i = 3`
- `j = 3`
- `nums`

# A Variant on the Same Theme

On completion of execution of the following code:

```
nums = [1,3,5]
i = j = 0
while nums:
    j += nums.pop()
    i += 1
j /= i
```

What is the value of each of the following variables:

- `i = 3`
- `j = 3`
- `nums = []`

# Question 3

What is the output of the following code:

```
battleships = [['0','p','0','s'],
               ['0','p','0','s'],
               ['p','p','0','s'],
               ['0','0','0','0']]

def fun(a,b,bships):
    c = len(bships)
    return bships[c-b][a-1]

print(fun(1,1,battleships))
print(fun(1,2,battleships))
```

# Question 3

What is the output of the following code:

```
battleships = [['0','p','0','s'],
               ['0','p','0','s'],
               ['p','p','0','s'],
               ['0','0','0','0']]

def fun(a,b,bships):
    c = len(bships)
    return bships[c-b][a-1]

print(fun(1,1,battleships))
print(fun(1,2,battleships))
```

0

p

# Question 4

`caverage` takes a single argument `intlist` in the form of a list of integers, and returns the average of the values, excluding the largest and smallest values in the list. You may assume that the list contains at least three integers. Example outputs of the function are:

```
>>> caverage([1,3,2,100])
2.5
>>> caverage([1,1,3,100])
2.0
>>> caverage([1,1,100])
1.0
```

# Question 4

Supply code fragments to insert in each of the numbered boxes to complete the function `caverage`:

```
def caverage( 1 ):
    max = intlist[0]
    min = intlist[0]
    sum = max
    for i in 2 :
        3
        if i < min:
            4
        elif i > max:
            max = i
    5
```

# Question 4

Supply code fragments to insert in each of the numbered boxes to complete the function `caverage`:

```
def caverage( [ intlist ] ):
    max = intlist[0]
    min = intlist[0]
    sum = max
    for i in [ 2 ]:
        [ 3 ]
        if i < min:
            [ 4 ]
        elif i > max:
            max = i
    [ 5 ]
```

# Question 4

Supply code fragments to insert in each of the numbered boxes to complete the function `caverage`:

```
def caverage( [      intlist      ] ):
    max = intlist[0]
    min = intlist[0]
    sum = max
    for i in [      intlist[1:]      ]:
        [            3            ]
        if i < min:
            [            4            ]
        elif i > max:
            max = i
    [            5            ]
```

# Question 4

Supply code fragments to insert in each of the numbered boxes to complete the function `caverage`:

```
def caverage( [      intlist      ] ):
    max = intlist[0]
    min = intlist[0]
    sum = max
    for i in [      intlist[1:]      ]:
        [ sum += i ]
        if i < min:
            [          4          ]
        elif i > max:
            max = i
    [          5          ]
```

# Question 4

Supply code fragments to insert in each of the numbered boxes to complete the function `caverage`:

```
def caverage( [      intlist      ] ):
    max = intlist[0]
    min = intlist[0]
    sum = max
    for i in [      intlist[1:]      ]:
        [ sum += i ]
        if i < min:
            [ min = i ]
        elif i > max:
            max = i
    [          5          ]
```

# Question 4

Supply code fragments to insert in each of the numbered boxes to complete the function `caverage`:

```
def caverage( [ intlist ] ):
    max = intlist[0]
    min = intlist[0]
    sum = max
    for i in [ intlist[1:] ]:
        [ sum += i ]
        if i < min:
            [ min = i ]
        elif i > max:
            max = i
    [ return (sum-min-max)/(len(intlist)-2) ]
```

# Question 5

Write code to implement each of the following functions:

(a) `substr`, which takes string arguments `sup` and `sub`, and returns `True` if `sub` occurs in `sup`, and `False` otherwise.

# Question 5

Write code to implement each of the following functions:

(a) `substr`, which takes string arguments `sup` and `sub`, and returns `True` if `sub` occurs in `sup`, and `False` otherwise.

```
def substr(sup,sub):
    return sub in sup
```

# Question 5

Write code to implement each of the following functions:

(b) `substrn`, which takes string arguments `sup` and `sub`, and returns the integer count of the (possibly overlapping) number of times `sub` can be found in `sup`.

# Question 5

Write code to implement each of the following functions:

(b) `substrn`, which takes string arguments `sup` and `sub`, and returns the integer count of the (possibly overlapping) number of times `sub` can be found in `sup`.

```
def substrn(sup,sub):
    sublen = len(sub)
    n = 0
    for i in range(len(sup)-sublen+1):
        if sup[i:i+sublen] == sub:
            n += 1
    return n
```