Midsemester Test Preparation

COMP10001 Foundations of Computing

Alex Cummaudo

April 3, 2018

Vocabulary

1. Write a one sentence definition for each of the following terms.

- Literal
- Variable
- Statements
- Assignment
- Program State
- Types
- Type Conversion
- Conditions
- Comparison
- Boolean
- Python Operators
- Comments
- String
- Indexing
- Slicing
- Splitting
- Functions
- Function call
- Methods
- Parameters
- Arguments
- `for` loop
- `while` loop
- `break` keyword
- `continue` keyword
- Tuples
- Lists
- Dictionaries
- Mutability
- Objects
- Namespaces
- PEP8
- Docstrings
- List Comprehensions

Code Execution

2. (4 points) **MCQ:** What are the final values of the variables in the code snippets below?

   (a) (1 point) Snippet A

```
1  p = 0
2  q = 1
3  r = 3
4  s = 4
5
6  if p < q:
7    q = q + 3
8    if q > r:
9      s = 0
10   elif q > s:
11     s = 1
12   else:
13     s = 2
14 else:
15   p = 0
```

   ◯ `p = 0; q = 1; r = 3; s = 4`
   ◯ `p = 0; q = 4; r = 3; s = 1`
   ◯ `p = 0; q = 1; r = 3; s = 2`
   ◯ `p = 0; q = 4; r = 3; s = 2`

   (b) (1 point) Snippet B

```
1  count = 0
2  balance = 0
3  deposit = 10
4
5  while count < 4:
6    balance = balance + deposit
7    count = count + 1
```

   ◯ `deposit = 10; balance = 30; count = 3`
   ◯ `deposit = 10; balance = 40; count = 4`
   ◯ `deposit = 10; balance = 40; count = 3`
   ◯ `deposit = 10; balance = 30; count = 4`

(c) (1 point) Snippet C

```
1 name = "Freddy"
2 surname = "McKenzie"
3 full_name = name[:3] + surname[2:]
```

- ◯ `name = "Freddy"; surname = "McKenzie"; full_name = "FredKenzie"`
- ◯ `name = "Fred"; surname = "Kenzie"; full_name = "FredKenzie"`
- ◯ `name = "Fred"; surname = "Kenzie"; full_name = "FredMc"`
- ◯ `name = "dy"; surname = "Kenzie"; full_name = "dyMc"`

(d) (1 point) Snippet D

```
1 items = ["Apple", "Banana", "Avocado"]
2 price = [65000, 1.50, 2.50]
3 merge = {}
4 index = 0
5
6 for item in items:
7   merge[item] = sorted(price)[index]
8   index += 1
```

- ◯ `items = ["Apple", "Banana", "Avocado"]`
  `price = [65000, 1.50, 2.50]`
  `merge = {"Apple" : 65000; "Banana": 1.50; "Avocado": 2.50}`
  `index = 3`
- ◯ `items = ["Apple", "Banana", "Avocado"]`
  `price = [65000, 1.50, 2.50]`
  `merge = {"Apple" : 1.50; "Banana": 2.50; "Avocado": 65000}`
  `index = 4`
- ◯ `items = ["Apple", "Avocado", "Banana"]`
  `price = [65000, 1.50, 2.50]`
  `merge = {"Apple" : 65000; "Banana": 1.50; "Avocado": 2.50}`
  `index = 3`
- ◯ `items = ["Apple", "Banana", "Avocado"]`
  `price = [65000, 1.50, 2.50]`
  `merge = {"Apple" : 1.50; "Banana": 2.50; "Avocado": 65000}`
  `index = 3`

3. (8 points) What is the program output of the following code snippets?

   (a) (1 point) Snippet A

```
1 pizza = 10
2 pasta = 20
3 meals = pizza + pasta
4 print(meals)
5 meals = str(meals) + " meals on order"
6 print(meals)
```

   (b) (2 points) Snippet B

```
1  items = [6, 10, 12, 16]
2  x = min(items)
3  y = max(items)
4  z = len(items)
5  print(x)
6  print(y)
7  x = x / z
8  y = y / z
9  print(x)
10 print(y)
```

(c) (2 points) Snippet C[1]

```
1  def is_even(num):
2    return num % 2 == 0
3
4  for i in range(1,10):
5    if is_even(i):
6      print(i * 3)
7    else:
8      print(i + 3)
```

(d) (4 points) Snippet D[2]

```
1  my_str = "2148"
2  my_num = 0
3  i = 0
4
5  while i < len(my_str):
6    num = int(my_str[i])
7    exponent = len(my_str) - 1 - i
8    my_num += num * (10 ** exponent)
9    i += 1
10   print(my_num)
```

[1] *Hint:* The `%` operator means 'modulus'.
[2] *Hint:* The `**` operator means 'power to', i.e., `x ** y` == x^y.

Code Debugging

4. (7 points) Review the following Python code:

```
1  def count_if(items, value):
2    '''
3    Counts the number of "value" items present in the list "items".
4    E.g., count_if([1,2,2,3,6,2], 2) == 2
5    '''
6    for item in items:
7      if item == value:
8        count += 1
```

(a) (1 point) Determine the output of the code[3], *as is*, for the following input data:

items = [5, 6, 5]
value = 10

(a) _____

(b) (2 points) Identify two bugs in the code. Fix the code, and write the fixed code below *without* comments.

(c) (4 points) Determine the output of the code after you have fixed it in part (b) for the following input data:

i. **items** = [5, 6, 5]
value = 10

i. _____

ii. **items** = ['A', 'B', 'C']
value = 'B'

ii. _____

[3]That is, what is *returned* from the function?

iii. **items** = [[4, 3, 3], [4, 5, 6], [2, 2, 6], [4, 5, 6]]
value = [4, 5, 6]

iii. _____

iv. **items** = [ ]
value = `None`

iv. _____

5. (5 points) Review the following Python code:

```
1  def maximum(items):
2    '''
3    Calculates and returns the maximum value in items, a list.
4    E.g., maximum([1,2,3]) == 3
5    '''
6    result = 0
7    index = len(items) - 1
8    while index > 0:
9      if items[index] > result:
10       result = items[index]
11       index -= 1
12   return result
```

(a) (1 point) Determine the output of the code, *as is*, for the following input data:

items = [12, 4, 8, 5]

(a) _____

(b) (2 points) Identify two bugs in the code. Fix the code, and write the fixed code below *without* comments.

_____

_____

_____

_____

_____

_____

(c) (2 points) Determine the output of the code after you have fixed it in part (b) for the following input data:

i. **items** = [0, 5, 0, 6, 7]

i. _____

ii. **items** = ['C', 'Z', 'X']

ii. _____

Code Writing

6. (4 points) Write a function, `sum_positive`, that calculates and returns the sum of all positive numbers in a list of integers.

   E.g., `sum_positive([-3, 3, 7, -1, 0]) == 10`

7. (5 points) Write a function, `count_vowels`, that calculates and returns the number of vowels in a given string. Remember to consider case insensitivity.

   E.g., `count_vowels("Banana") == 3`
   E.g., `count_vowels("BANANA") == 3`