COMP10001 Foundations of Computing
Mid-semester Test Preparations: DON'T PANIC

Semester 1, 2019
Tim Baldwin, Nic Geard, Farah Khan, and Marion Zalk

— VERSION: 1515, DATE: APRIL 2, 2019 —

Reminders

- Live(ish) Help starts from **tomorrow** and will run **every teaching day** (= Mon–Fri except for public holidays and the mid-semester break) for the remainder of semester, for the following times:
  - 1pm–2pm
  - 7pm–8pm

  Use the Tutor Messaging facility on Grok as per usual, but expect realish-time responses during these times
- Office hours will also start up from this week, as follows:
  - Tue 1-2pm, Doug McDonell 322
  - Wed 2-3pm, Doug McDonell 902

Lecture Outline

1. Mid-semester Test Logistics
2. Mid-semester Test Solutions

Mid-Semester Test Outline

- Will be held across 4 venues:
  - **Wilson Hall**
  - **Kwong Lee Dow Building** (level 1, 234 Queensberry Street)
  - **Union Hall** (level 2, Union House)
  - **B101, Arts West** (the usual lecture theatre)

  at the same time as the usual Fri lectures; seats are pre-allocated, and will be listed on the LMS
- Make sure to come at the **right time** and to the **right place**, and armed with knowledge of your **seat number**
- Everything up to the end of week 4 for the lectures is examinable for the mid-semester test

Mid-Semester Test Outline

- All Python (v3.6) coding questions
- **40 minutes** in duration
- Pen-and-paper test (no calculators, no Grok, no mobile phones, ... in fact **no allowed materials** at all)
- Bags must be left outside the exam venue (in the foyer)
- Make sure to bring your **student card** and come on time (the test will start at 11:15 and 12:10 **sharp**)

Lecture Outline

1. Mid-semester Test Logistics
2. Mid-semester Test Solutions

Question 1

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

```
["apples", "bananas", "cantaloupes"][-1][:3]
```

Question 1

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

```
["apples", "bananas", "cantaloupes"][-1][:3]
```

```
'can'
```

Question 1

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

```
bool([1,3,5] and "bods")
```

Question 1

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

```
bool([1,3,5] and "bods")
```

```
True
```

Question 1

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

```
f"{4+7} and {11*2} were racehorses"
```

Question 1

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

```
f"{4+7} and {11*2} were racehorses"
```

```
"11 and 22 were racehorses"
```

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
```

- `'bear' in 'there'`

More of the Same

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

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

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

```
True
```

- `'bear' in 'there'`

```
False
```

Question 2

What are the final values of each of these variables:

```python
def fun(i,j):
    return str(j) in str(i)

nums = (1, 1)
r = 3
total = 0
for i in range(3):
    total += fun(nums, r)
    p, q = nums
    nums = (q, p + q)
```

(a) `i`

Question 2

What are the final values of each of these variables:

```python
def fun(i,j):
    return str(j) in str(i)

nums = (1, 1)
r = 3
total = 0
for i in range(3):
    total += fun(nums, r)
    p, q = nums
    nums = (q, p + q)
```

(a) `i = 2`

Question 2

What are the final values of each of these variables:

```python
def fun(i,j):
    return str(j) in str(i)

nums = (1, 1)
r = 3
total = 0
for i in range(3):
    total += fun(nums, r)
    p, q = nums
    nums = (q, p + q)
```

(b) `nums`

Question 2

What are the final values of each of these variables:

```python
def fun(i,j):
    return str(j) in str(i)

nums = (1, 1)
r = 3
total = 0
for i in range(3):
    total += fun(nums, r)
    p, q = nums
    nums = (q, p + q)
```

(b) `nums = (3, 5)`

Question 2

What are the final values of each of these variables:

```python
def fun(i,j):
    return str(j) in str(i)

nums = (1, 1)
r = 3
total = 0
for i in range(3):
    total += fun(nums, r)
    p, q = nums
    nums = (q, p + q)
```

(c) `total`

Question 2

What are the final values of each of these variables:

```python
def fun(i,j):
    return str(j) in str(i)

nums = (1, 1)
r = 3
total = 0
for i in range(3):
    total += fun(nums, r)
    p, q = nums
    nums = (q, p + q)
```

(c) `total = 1`

Question 3

Rewrite the following function, replacing the `for` loop with a `while` loop:

```
def last_letter(word):
    last = 0
    for i in range(1, len(word)):
        if word[i] > word[last]:
            last = i
    return last
```

Question 3

```
def last_letter(word):
    last = 0
    i = 1
    while i < len(word):
        if word[i] > word[last]:
            last = i
        i += 1
    return last
```

Question 4

The following `create_deck` function initialises a deck of 52 playing cards in the form of a list of lists, where each entry is a string representing a card (e.g. `'AS'` is the ace of spades, and `'0D'` is the 10 of diamonds). A second function `replace_with_joker` replaces a single card in the original deck with a joker, at the indicated (zero-offset) position in the deck. When called as shown in the final three lines of code below, the printed output should be equivalent to the following: ...

Question 4

```
VALUES = "A234567890JQK"
SUITS = "SHDC"

def create_deck():
    [          1          ]
    for i in VALUES:
        [          2          ]
        for j in SUITS:
            [          3          ]
        deck.append(row)
    return deck

deck = create_deck()
```

Question 4

```
VALUES = "A234567890JQK"
SUITS = "SHDC"

def create_deck():
    deck = []
    for i in VALUES:
        [          2          ]
        for j in SUITS:
            [          3          ]
        deck.append(row)
    return deck

deck = create_deck()
```

Question 4

```
VALUES = "A234567890JQK"
SUITS = "SHDC"

def create_deck():
    deck = []
    for i in VALUES:
        row = []
        for j in SUITS:
            [          3          ]
        deck.append(row)
    return deck

deck = create_deck()
```

Question 4

```
VALUES = "A234567890JQK"
SUITS = "SHDC"

def create_deck():
    deck = []
    for i in VALUES:
        row = []
        for j in SUITS:
            row.append(i + j)
        deck.append(row)
    return deck

deck = create_deck()
```

Question 4

```
VALUES = "A234567890JQK"
SUITS = "SHDC"

def replace_with_joker(deck, pos):
    row_id = pos//4
    [          4          ]
    deck[row_id][val_id] = "joker"

deck = create_deck()
replace_with_joker(deck, 11)
print(deck)
```

Question 4

```
VALUES = "A234567890JQK"
SUITS = "SHDC"

def replace_with_joker(deck, pos):
    row_id = pos//4
    val_id = pos%4
    deck[row_id][val_id] = "joker"

deck = create_deck()
replace_with_joker(deck, 11)
print(deck)
```

Question 5

Write the function `sum_eq_prod`, which takes a non-negative integer argument `num`, and returns `True` if the sum of the digits of `num` is the same as the product of the digits, and `False` otherwise. For example, the function call `sum_eq_prod(123)` should return `True` as $1 + 2 + 3 = 1 \times 2 \times 3 = 6$, whereas `sum_eq_prod(42)` should return `False` as $4 + 2 \neq 4 \times 2$.

Question 5

```
def sum_eq_prod(num):
    dig_sum = 0
    dig_product = 1
    for digit in str(num):
        digit = int(digit)
        dig_sum += digit
        dig_product *= digit
    return dig_sum == dig_product
```

Summary

- Mid-semester test in **Wilson Hall**, the **Kwong Lee Dow Building**, **Union Hall**, and **B101** at the same time as your normal Fri lecture
- Make sure to come along to the correct session (11:15 vs. 12:10), based on your timetable
- Make sure to come to the **right place**
- Make sure to bring your **student card**
- Make sure to check your **seat number** ahead of time
- Good luck!