Midterm Locations Posted

Midterm Locations posted here:  
http://www.teach.cs.toronto.edu/~csc120h/fall/tests.shtml

• The room assignments listed below are based on the first letter(s) of your last (family) name. It is your responsibility to go to the correct location. All ranges are inclusive.
  • **RW 110**: A - J (map)
  • **SS 1071**: K - Mi (map)
  • **BA 1130**: Mo - V (map)
  • **SS 1084**: W - Z (map)

• Arrive on time and don’t forget your T-Card!
During the test

• Write your name in the front and last page of the test.
• Read the questions carefully. You may want to skim through the entire test once, before you start writing.
• Keep an eye on the clock.
• Write as clearly as possible. Make sure your code’s indentation is clear.
• I will be going around the rooms to see if you have any questions.
• You are NOT permitted to tear any pages.
Take a minute to remember how far you’ve come since early September 😊
Please note that the topics briefly mentioned in the subsequent slides are some highlights and do not provide an exhaustive coverage of all the topics we’ve covered so far in CSC120.
Python as a calculator

- Basic arithmetic operators
  - *Precedence (from high to low)*
    - **
    - - (negation)
    - *, //, /, %
    - +, -

- Python types
  - int, float, etc.

- Assignment statements
  - x = 1 + 2
  - Choose valid, meaningful variable names
Functions

• Defining a function
  
  ```python
  def function name(parameters):
      body
  ```

• Calling a function
def triangle_area(triangle_base, triangle_height):
    """
    (number, number) -> float
    
    Precondition: triangle_base > 0 and triangle_height > 0
    
    Return the area of a triangle with base triangle_base and height triangle_height.
    
    >>> triangle_area(100, 2)
    100.0
    >>> triangle_area(12.5, 34.0)
    212.5
    """
    return triangle_base * triangle_height / 2
If statements

• If statement format:

```python
if condition_1:
    statements_1
elif condition_2:
    statements_2
elif ...
    ...
else:
    statements_N
```

• Can also have nested if-statements
Boolean Logic

• Type `bool` only has two possible values: `True`, `False`

• **Relational (or comparison) operators** are: `>`, `>=`, `<=`, `<`, `==`, `!=`
  • Note: when comparing an int with a float, Python compares values and not types.
  • Thus, `2 == 2.0` evaluates to `True`, even though `type(2) == type(2.0)` evaluates to `False`.

• **Logical (or Boolean) operators** are (in order of precedence): `not`, `and`, `or`
  • We can use multiple operators to write a more complex Boolean expression.
And vs. Or, Lazy evaluation, etc.

• var1 = condition_a and condition_b
  • var1 will refer to True if and only if both condition_a and condition_b evaluate to True

• var2 = condition_a or condition_b
  • var2 will refer to True if at least one of the two conditions (condition_a, condition_b) evaluates to True.
    • Note, “at least one” means it’s OK if both condition_a and condition_b evaluate to True.

• Short-circuiting (Python’s lazy evaluation)
  • For example: x and y – if x is False, y is not evaluated

• Boolean good practice
  • don’t write if (temp < 0) == True:
Precedence (from high to low)

• Arithmetic Operators (Highest precedence)
• Relational (also called Comparison) Operators
• Boolean (also called Logical) Operators (lowest precedence)
Strings

• Operators: +, *, in
• String slicing
  >>> s = 'hello'
  >>> s[0:2]
  'he'
• String comparison
  >>> 'a' < 'b'
  True
Strings cont’d

• Looping over strings

```python
for char in s:
    print(char)
```

• String methods
  • e.g., `lower()`, `replace()`, `count()`

• Strings are immutable
  • Cannot modify a string like this: `s[2] = 'a'`

• Escape sequences: `\n`, `\t`, `\'`, `\`
Lists

• Defining lists
  • e.g., \( L = [1, 2.0, \text{‘}hello\text{‘}] \)

• List indexing
  • e.g. \( L[0] \) is 1

• Lists are mutable
  • Can modify a list like this: \( L[2] = \text{‘}goodbye\text{‘} \)

• List functions
  • e.g., \( \text{len}(), \text{max}(), \text{sum}() \)

• List methods
  • e.g., \( \text{append}(), \text{extend}(), \text{insert}() \)
Lists cont’d

• Two types of for loops:

for item in L:

vs.

for index in range(len(L)):
Lists cont’d

• `range()` function
  • Allows us to iterate over a sequence of values

• Slicing lists
  • `L[0:5]`, `L[2:]`, etc.
Lists cont’d

• Nested lists
  • e.g., $L = [[‘Alice’, 150], [‘Bob’, 165], [‘Charlie’, 180]]$

• Nested loops
  
  ```python
  for i in range(len(L)):
    for j in range(2):
      print(L[i][j])
  
  Alice
  150
  Bob
  <etc>
  ```
Input

- Using input to get info from user
- E.g., `name = input("Enter name: ")`
- Always returns a string
- If it is a number, we can convert it to an integer
While Loops

```python
while some_condition:
    # do something
```

You need to ensure that you do something in the body of the loop that will make this condition eventually evaluate to False. Otherwise, you’ll have an infinite loop.
A note on Fall’s 2016 Midterm Solution

• Question 5, 3rd code snippet:
  
s = 'mississippi'
  print(s[:5])

Answer should NOT include quotes, as we’re printing the string (there is a typo in the posted draft solution). It should be:

  missi