COMP10001 Foundations of Computing
Semester 1, 2019

Tutorial Questions: Week 4

— VERSION: 1490, DATE: MARCH 25, 2019 —

Discussion

1. What is “Boolean”? What values does it store? Can other types be converted to it?

   A: *Boolean is a data type which stores a truth value: either True or False. Any other type can be converted to it: for numbers, zero converts to False and any non-zero number converts to True; and for strings and lists an empty sequence converts to False and any non-empty sequence converts to True.*

2. For each of the following, identify whether it is: (a) a Boolean value; (b) a relational operator; or (c) a logical operator.

   A: *The relational operators compare two values to produce a truth (boolean) result. They include less than (<), greater than (>), less than or equal to (<=), greater than or equal to (>=), equal (==) and not equal (! =) The* `in` *operator is also a kind of relational operator.*
   The logical operators combine Boolean values to return a single truth value. They are `and`*,* `or` *and* `not`*.*
   Order of precedence is Relational operators then not, and, and finally or. Brackets can clarify order of operations and you are encouraged to use them.

`==`	Relational	`>`	Relational	`False`	Boolean
`!=`	Relational	`and`	Logical	`<=`	Relational
`or`	Logical	`>=`	Relational	`not`	Logical
`True`	Boolean	`<`	Relational		

3. How do we use an if statement? What are the variants? How do we know what is contained inside it and what is after?

   A: *An if statement is of the form* `if <condition>:` *where* `condition` *can be formed of relational and logical operators, or anything else you like. If the condition expression evaluates to True, the code “inside” (indented after) the if statement is run. If not, it is skipped. Variants include* `elif` *and* `else` *statements following if, which catch further conditions if the first conditional statement is not fulfilled. Indentation tells us what code belongs inside an if statement and what code follows it.*

   Now try Exercises 1 - 4

4. What is a “Sequence”? What sequences have we seen so far?

   A: *A sequence is a data type which allows us to store multiple pieces of information in an organised way. Strings store sequences of characters while lists and tuples store sequences of any object.*

5. What is indexing? How can you do it?

   A: *Indexing means accessing the item stored in a particular (integer) position in a sequence. You index using square brackets containing the index* `[i]` *at the end of the variable name or object literal. You can index with positive integers where 0 corresponds to the first item or negative integers where -1 corresponds to the last item.*

6. What is slicing? How can you do it?

   A: *Slicing is like taking an index but rather than only item, a slice will give you a subsection of the sequence consisting typically of more than one item (though you can have an empty slice). Slicing works with two indices* `[i:j]` *and takes items from the first to one before the second.*

7. **Bonus question:** How do you change the “step size” of a slice?

   A: *A slice can take three integers: start, end and step [i:j:k] The step integer controls whether every character is included in the slice (1), every second character (2), and so on. A negative step makes the slice go backwards.*

   Now try Exercises 5 & 6

Exercises

1. Evaluate the following truth expressions:

   (a) `True or False`
   A: *True*

   (b) `True and False`
   A: *False*

   (c) `False and not False or True`
   A: *True*

   (d) `False and (not False or True)`
   A: *False*

2. For each of the following if statements, give an example of a value for `var` which will trigger it and one which will not.

   (a) `if 10 > var >= 5:`
   A: *A number 5 or greater and lower than 10 will trigger this if statement. Another number will not. A value other than a number will cause an error.*

   (b) `if var in ("VIC", "NSW", "ACT"):`
   A: *Only strings "VIC", "NSW" and "ACT" will trigger this condition. Any other value of var will result in a False result.*

   (c) `if var[0] == "A"and var[-1] == "e":`
   A: *A string which begins with "A" and ends with "e", for example Apple, Antelope or Algae (with that capitalisation). Other strings and sequences will not pass, and any string or sequence which does not have at least one one item will give an IndexError.*

   (d) `if var:`
   A: *This condition will convert var into a boolean value, so if it is non-zero/non-empty, the if statement will be triggered.*

3. What's wrong with this code? How can you fix it?

```
letter = input("Enter␣a␣letter:␣")
if letter == 'a' or 'e' or 'i' or 'o' or 'u':
    print("vowel")
else:
    print("consonant")
```

A: *Logical operators separate conditions, so `letter == 'a'` is separate from `'e'` and the rest of the conditions. This expression will always evaluate to True because non-empty strings such as 'e' evaluate to True. This can be fixed by writing out `letter ==`for every condition or using the `in` operator as `if letter in 'aeiou:'`.*

4. What's wrong with this code? How can you fix it?

```
eggs == 3
if eggs = 5:
    print("spam")
else:
    print("not␣spam")
```

A: *This programmer has confused the assignment (=) and equality (==) operators. This can be fixed by swapping them (`eggs = 3`, `if eggs == 5:`).*

5. Evaluate the following given the assignment `s = "pythonisation"`

   (a) `s[1]`
   A: `'y'`
   (Indexing starts from 0)

   (b) `s[-1]`
   A: `'n'`
   (Negative indexing starts from -1)

   (c) `s[2:4] + s[6:8]`
   A: `'this'`

   (d) `s[25]`
   A: `IndexError: string index out of range`

   (e) `s[25:]`
   A: `''` *(Empty String)*

   (f) `s[-7:-3]`
   A: `'isat'` *(Note that even with negative indices, the index of the leftmost character is first)*

   (g) `s[:-3]`
   A: `'pythonisat'`

   (h) `s[::2]`
   A: `'ptoiain'`
   (skips every second letter)

   (i) `s[::-1]`
   A: `'noitasinohtyp'`
   (backwards)

A: *Note that both d and e ask for a portion of the string which doesn't exist, but while d (indexing) gave an error, e (slicing) returned an empty sequence.*

6. Evaluate the following given the assignment `lst = [4, ("green", "eggs", "ham"), False]`

   (a) `lst[2]`
   A:`False`

   (b) `lst[1][1]`
   A:`'eggs'`

   (c) `lst[1][1][:3]`
   A:`'egg'`

Problems

1. Write a program which asks the user for two numbers and an operator out of +, -, / and * and performs that operation on the two numbers, printing the result.

   A:

```python
num_1 = float(input("Enter␣the␣first␣number:␣"))
num_2 = float(input("Enter␣the␣second␣number:␣"))
op = input("Enter␣operator␣(+␣-␣*␣/)␣")
if op == "+":
    print(num_1, "+", num_2, "=", num_1 + num_2)
elif op == "-":
    print(num_1, "-", num_2, "=", num_1 - num_2)
elif op == "*":
    print(num_1, "*", num_2, "=", num_1 * num_2)
else:
    print(num_1, "/", num_2, "=", num_1 / num_2)
```

2. Write a program which asks the user for their name and prints a shortened version consisting of the first three letters and then every second letter in the rest of the word.

   A:

```python
name = input("Enter␣your␣name:␣")
short_name = name[:3] + name[3::2]
print("Your␣shortened␣name␣is:␣" + short_name)
```

3. Write a program which asks the user to write a sentence and checks whether the first letter is capitalised and the last character is a full stop to decide whether it's a correct sentence.

   A:

```python
sentence = input("Enter␣your␣sentence:␣")
if "A" <= sentence[0] <= "Z" and sentence[-1] == ".":
    print("You␣typed␣a␣correct␣sentence")
else:
    print("Your␣sentence␣is␣incorrect")
```