Python 3 Control Structures

Eddie Guo

September 2019

1 Decision Making

1.1 Topics Covered

(i) `if/elif/else`

(ii) `for` loop

(iii) `range`

(iv) `while` loop

1.2 `if/elif/else` Statement

- `if` statement only executes when Boolean expression is True
- `elif` statement executes if `if` statement & above `elif` statements are false; tests series of conditions
- `else` statement only executes if all above condition(s) are false

```
# To compare floats w/o rounding, test the difference & compare to tolerance value
tolerance = 0.001
if abs(float_val_1 - float_val_2) <= tolerance:
    print('equal')
```

2 Loops

2.1 `for` Loops

- `for` loop → when you know how many times to repeat code: *count ctrld*
- `for` loop repeats code for every element in given seq
- Can use for strs, lists, tuples, sets
- `range(start, stop, step)`
  - Up to, but not including, `stop`
  - Must always include `stop`, `start` & `step` are optional

```
# print chrs that come b4 s/S alphabetically
>>> for char in 'CMPUT':
...     if char.lower() < 's':
...         print(char, end='*)
'C*M*P*'
```

2.2 `while` Loops

- `while` loop → when you know how many times to repeat code: *condition ctrld*
- `while condition`: execute fn → condition will be evaluated to true/false
- Repeat action until user enters specific value: a **sentinel** value

```
>>> SENTINEL = 0
>>> total = 0
>>> num = -1 # initialize -> diff from sentinel
>>> while num != SENTINEL:
...     num = int(input('Enter value to add; 0 to stop > '))
...     total += num
... print('Sum of values entered by user is', total)
```

- BEWARE of the ∞ loop; something inside loop should eventually make `while` condition `False`
  - i.e., *at least one thing related to condition expression must be updated each iteration*
- Can also use ctrl value

```
# Should prolly use a for loop for this scenario (more Pythonic)
>>> total = 0
>>> my_list = [10, -2, 3, 24]
>>> i = 0 # INITIALIZE control variable
>>> while i < len(my_list):
...     total += my_list[i]
...     i += 1 # UPDATE control variable
... print('Sum of elements is', total)
Sum of elements is 35
```

2.3 `break`

- Avoid using `break`: better to use Boolean flags b/c code easier to read
- Don't use break in this class

2.4 Nested Loops

```
>>> for row in range(3):
...     print(row, end=": ")
...     for col in range(5):
...         print(col, end=" ")
... print()
0: 0 1 2 3 4
1: 0 1 2 3 4
2: 0 1 2 3 4
```