Quiz Section 4

trees, while loops, and functions

2019-04-25

Python - while loops and functions

while loops

```
while <test>:
  statement1
  statement2
```

"While some logical test is true, continue executing the block of statements"

What does this program do?

```
sum = 0
count = 1

while count < 3:
  sum = sum + count
  count = count + 1
print(count)
print(sum)
```

What does this program do?

```
sum = 0
count = 1

while count < 3:
  sum = sum + count
  count = count + 1
print(count)
print(sum)
```

```
sum = 0
count = 1
count < 3 TRUE
  sum = 0 + 1
  count = 1 + 1
```

```
sum = 1
count = 2
```

```
count < 3 TRUE
  sum = 1 + 2
  count = 2 + 1
sum = 3
count = 3
count < 3 FALSE
```

`for` vs. `while` loops

- `for` loops: *determinate* number of loops (probably more common)
- `while` loops: *indeterminate* number of loops

Examples of `for` loops

```
for base in sequence:
  <do something with each base>

for sequence in database:
  <do something with each sequence>

for index in range(5, 200):
  <do something with each index>
```

Examples of `while` loops

```
while error > 0.05:
  <do something to reduce error>

while score > 0:
  <traceback through DP matrix>
```

Example of infinite loop

You can set up a while loop such that it will run forever

```
count = 0

while count <=0:
  count = count + 1
```

What's a function?

> reusable pieces of code that take zero or more arguments, perform some actions, and returns one or more values.

```
len('AGT')
```

- arguments: string or list

- actions: count number of elements
- return: integer

You can write your own functions

```
def my_function(datapoint):
  result = datapoint * 100
  return result
l = [1, 2, 3]
for i in l:
  print(my_function(i))
```

You need to define your function *then* call the function

Why custom functions?

Avoid repetition

- easier to look at
- avoid bugs

Self-documenting code

```
def upgma_iteration(data):
    dist_matrix = calculate_distance_matrix(data)
    smallest_pair  = find_smallest_dist(dist_matrix)
    merged_node_data = merge_nodes(smallest_pair, data)
    return merged_node_data
```