CPS 109 - Lab 2

Agenda

1 - Housekeeping
2 - Recap of Last Week
3 - Operators
4 - Lists
5 - Loops

Housekeeping

We're going to briefly walk you through forwarding .cs emails to your regular Ryerson email (please follow along).

Step 1: Open cmd.exe (Windows) or Terminal (Mac/Linux)
Step 2: Type: ssh username@moon.scs.ryerson.ca (username is your Ryerson CS username)
Step 3: Enter your password (remember it won't show)
Step 4: Type: nano .forward
Step 5: Type: username@ryerson.ca
Step 6: Input: Ctrl + o (to save), then Enter (to confirm)
Step 7: Input: Ctrl + x to exit
Step 8: Type: chmod 600 .forward
Step 9: Exit via Ctrl + d or Type: exit

Recap of Last Week

Remember last week we went over a lot of new stuff! Not remembering all of it is completely fine.

I would recommend rereading if/elif/else

Some Syntax

Let's talk about this statement:

```
5   if __name__ == "__main__":
```

What is this line, exactly? In short, this tells the computer what code to execute when the main file in a program is run.

Operators

Operators are, in essence, the backbone of programming. They’re instructions that the computer recognizes and uses to manipulate or check data.

Anything from boolean comparisons, to mathematical operations, to comparisons.

Common Operator Examples

Boolean Comparisons:

and, or, not

Mathematical Operations:

+, -, *, /, //, **, %

Comparators:

Greater than / or equal to	> / >=
Less than / or equal to	< / <=
Equal to / not equal to	== / !=

Lists

A list is simply a collection of data/variables/objects denoted with a set of [] and separated by commas, like so:

```
this_list = ["this", "is", "our", "list"]
```

Introduction to Loops

Do you want to repeat a bunch of code until you achieve something specific? Maybe until you reach a certain number of iterations or you find what you're looking for in a list?

You need a loop for that.

Introduction to Loops

What is a loop?

A loop is a certain set of instructions (syntax) that the computer knows to repeat until a condition is met.

Kinds of loops: while, for, nested

Loops: For

A for loop will repeat a specific set of instructions (written by you) *for* the amount of time/number of iterations specified.

Simply put: the computer will do something x amount of times.

Loops: For

```
5    def for_loop_here(num):
6        x = 1
7        for i in range(num):
8            x+=i
9        return x
```

Loops: For Part 2

```
2  teletubbies = ["Tinky Winky", "Dipsy", "Laa-Laa", "Po"]
3  # Do kids nowadays know what the teletubbies are?
4
5  for creature in teletubbies:
6      print(creature + " terrifies me...")
```

You can also use for loops to iterate over a list, or the characters in a string.

```
Tinky Winky terrifies me...
Dipsy terrifies me...
Laa-Laa terrifies me...
Po terrifies me...
[Finished in 54ms]
```

Loops: While

A while loop will follow a set of instructions *while* its boolean condition holds true.

Simply put: the computer will do x instructions until y is false.

Loops: While

```
 9  if __name__ == "__main__":
10      x = 0
11
12      while x < 10:
13          print(x)
14          x+=1
```

Loops: Nested

Nesting a loop is just a fancy way of saying putting a loop inside of another loop.

You can do this as many times as you like as long as you don’t care about your computer catching fire.

Loops: Nested

You can put a for loop within a for loop, a while loop within a while loop.

You can also get really spicy and put a for loop in a while loop, or a while loop in a for loop.

Loops: Nested

```
11  if __name__ == "__main__":
12      strOne = "Gibberish?"
13      strTwo = "Perish the thought!"
14
15      for i in strOne:
16          for j in strTwo:
17              print(i + j, end = '')
18
```