CMPT 120, Deck 7-2, Midterm Review

In this deck: we will walk through the learning objectives!

Week 1 Concepts

- Know that CS is problem solving
- Understand problem solving by subdividing tasks
- Know what is plagiarism for programming
- Know what pseudocode is
- Able to write Python comments
- Know CMPT 120 header block
- Able to output a "Hello World" using print()
- Able to contrast programming languages with natural languages
- Able to submit code in a .zip or .py file
- Able to run code "locally" and using a cloud service like Colab

Week 1 Examples

```
# Author: Nick V
# Date: Oct 17, 2024
# Purpose: Show snippets from Week 1

print("Hello, World")
print("I have", 1)
# This is a comment
"""
Multi
line
string
"""
```

Week 2 Concepts

- Design algorithms (comments/pseudocode)
- Problem solving strategies (subdivision)
  - used in problems
- Basic input/output in Python
- Variable assignment
- String concatenation
- Variable naming conventions
- Data types (start with String)
- Lists and `random.choice()`
- Module imports and functions
- `if/elif/else`, logical operators
- Boolean expressions, good software characteristics

Week 2 Examples

```
input()
variable = "example", # `=`, assignment
print(variable + " string"), # `" "` for strings and `+` fo
mylist = [1,2,3] # `[ ]` for lists
import random # import
random.choice(my_list) # random.choice()
x = 5
y = 4
if x == y:  # example of condition
    print("if")
elif x == y+1:
    print("elif")
else:
    print("else")
5 > 3
5 >= 3
5 != 3
not 5 == 3
a = True
b = False
a and b
a or b
a and not b
(a and b) or (a or b)
```

Week 3 Concepts

- String methods: `strip()`, `lower()`, `upper()`
- use REPL to test code
- `in` keyword for strings/lists
- Looping over lists (`for`)
- range function (with 1, 2, or 3 arguments)
- Integer and Strings
- Integer to string conversion
- Knows that concatenation is only applicable between 2 strings, not Int and String
- design and implement nested conditionals
- Error types: syntax vs. semantic
- Method chaining

Week 3 Examples

```
my_string = "PYTHON."
my_string.lower()
my_string.upper()
my_string.strip(".")
"substring" in my_string # `in` (strings)
1 in [1,2,3] # `in` (lists)
for i in range(5):  # loop example
    print(i)

str(5)
int("5")

range(4)
range(1,5)
range(1,10,2)
```

Week 4 Concepts

- `range()` with variables
- Knows that a loop is a way to reduce duplication of code
- Integer/Float manipulation
- Accumulator pattern
- Length of list
- Convert str to int
- Knows that division of integers converts type to float
- Able to perform arithmetic operations on numbers
- Can use the accumulator pattern with other arithmetic operators
- Print floats with precision

Week 4 Examples

```
x = 1
x += 1  # Accumulator pattern
print("Number: {:.3f}".format(myfloat))
print(
  type(17)`, `type(0.0)`, `type("xo")`, `type([1,2,3])`, `t
)
print(
  2 ** 2, 3 * 4, 5 - 3, 4 + 4, 5 / 3, 5 // 3, 12 % 5
)
x=1
x = x + 1
x+=1
int(4.3)
float("123.45")
str(12.3)
len(my_list)
range(len(my_list))
print("Number: {:.3f}".format(myfloat))
```

Week 5 Concepts

- Open + read lines from a file and split a string into a list
- Access specific element(s) of a list OR string using indexing/slicing
- Comparisons between numbers, taking into account order of operators (operator precedence)
- Comparisons (e.g. !=, <,>) with strings
- Interpret code with nested conditionals with comparison operators (e.g. !=, <,>=) and numbers
- Able to find the common elements between 2 lists
- Able to understand and use a nested for loop
- Operator precedence for evaluating expressions
- Able to concatenate lists
- Accumulation pattern for strings and lists

Week 5 Examples

```
file = open("myfile.txt")
line = file.readline()
for line in file:
  print(line)
lines = file.readlines()
my_list = my_string.split()
alist[2][0]  # Indexing in a list of lists
print(
  mystring[0], mystring[-1], mystring[:],
  mystring[3:5], mystring[:3], mystring[3:],
  mystring[3:-1]
)
"a"*3
["a"]*3
alist[2][0]
list1+list2
list1 = list1 + [elem]
"a" < "b"
"aa" < "ab"
```

Week 6 Concepts

- Maximum/Minimum values among a list

Week 6 Examples

```
list_a = [10, 30, 20, 30]
max_index = 0
max_val = list_a[0]
for i in range(len(list_a)):
  if list_a[i] > max_val:
    max_val = list_a[i]
    max_index = i
print(max_index, max_val)
```

Week 7 Concepts

- Bits as the main unit of information