Functions I

Warm-up

Last class, we looked at sequences. Test your knowledge by answering the following questions:

Questions (10 min) **Start time:**

1. Evaluate each expressions in order and write down what it evaluates to. If an error occurs, write what type of error. Place an asterisk (*) next to any output for which you are unsure.

Python code	Output
`seq1 = "one two"`	
`type(seq1)`	<class 'str'>
`len(seq1)`	7
`seq1[3] = ','`	Runtime Error (str is immutable)
`seq1.replace(' ',',')`	
`print(seq1)`	`'one two'`
`seq2 = ["one", "two"]`	
`type(seq2)`	<class 'list'>
`seq2[1] = 1`	
`print(seq2)`	['one', 1]
`seq2.append(1)`	
`print(seq2)`	['one', 1, 1]
`seq2.count(1)`	2
`seq3 = 'abcdefg'`	
`seq3[:]`	'abcdefg'
`seq3[::-1]`	'gfedcba'
`seq3[:-3]`	'abcd'
`seq3[:-3:-1]`	'gf'
`seq3[-3:]`	'efg'
`seq3[-3:]`	'edcba'

2. What are similarities between `lists` and `strings`?

Answers may vary; they are both sequences, but are declared differently, strings with quotes while lists with brackets and commas between elements. Lists are mutable while strings are immutable.

Meta Activity: Group vs Team

Throughout the course, you will need to examine and process information, ask and answer questions, construct your own understanding, and develop new problem-solving skills.

Questions (8 min)

Start time:

3. What are some advantages to working in groups/teams?

You get to know other people and make new friends. Different people have different backgrounds and skills. The responsibility is shared.

4. What are some disadvantages to working in groups/teams?

Some group members may decide not to contribute. One or two students may be absent. People may not always get along with each other.

5. Based on the images above, what is the difference between a group and a team? Come up with a precise answer.

A group is students who just sit by each other and turn in the same assignment. A team actually works together toward a common goal, drawing on each other's strengths.

6. How can working as a team help you accomplish the tasks described above? Give at least two specific examples.

Working as a team makes it easier to examine and process information, because different people have different perspectives. We can also develop new problem-solving skills by observing how each other approaches the problems.

Model 1 Beyond built-in functions

In addition to built-in functions Python has module functions. For example, you can use Python's `math` module to perform more complex mathematical operations like square root. Also, you can generate a sequence of ***pseudorandom*** numbers using the Python `random` module.

Questions (15 min) Start time:

7. Write the corresponding output for each statement assuming that they are executed in order. If an error occurs, write what type of error. Place an asterisk (*) next to any output for which you are unsure.

Python code	Output
`abs(-2) ** 4`	16
`math.pow(2, 4)`	NameError
`import math`	
`math.pow(2, 4)`	16.0
`sqrt(4)`	NameError
`math.sqrt(4)`	2.0

8. Identify four examples of:

a) mathematical operator +, *, /, **

b) mathematical function abs, round, math.pow, math.sqrt

9. The code above has two errors. Explain the reason for each:

a) 1st NameError need to import math

b) 2nd NameError need "math." before function

10. Identify two differences between using a Python built-in function (e.g., `abs`) and a function from the `math` module.

Need to "`import math`" first, and all function names start with "`math.`" before the function.

11. What is the name of the module that must be imported before generating a random number? What are the names of the functions defined in this module to generate a single ***pseudo-random*** number?

random module; randint() and randrange() functions

12. Write the corresponding output for each statement assuming that they are executed in order. If an error occurs, write what type of error. Place an asterisk (*) next to any output for which you are unsure.

Python code	Output
`import randint`	ImportError
`import random`	
`randint(1, 10)`	NameError
`random.randint(1, 10)`	integer in range [1..10]
`random.randrange(1, 10)`	integer in range [1..9]
`random.choice('abcd')`	a randomly selected character from the specified sequence
`random.choice([1,2,3,4])`	a randomly selected number from the specified sequence

13. In addition to using Python's built-in functions (e.g., `print`, `abs`) and functions defined in other modules (e.g., `math.sqrt`), you can write your own functions. Trace the program below, indicate its output and list each line of code in order of their execution separated by colon:

Python code	Output

`1 def print_lyrics() -> None:`
`2     print("I'm a lumberjack, and I'm okay.")`
`3     print("I sleep all night and I work all day.")`
`4`
`5 def main() -> None:`
`6     print_lyrics()`
`7     print_lyrics()`
`8`
`9 main()`
	I'm a lumberjack, and I'm okay.
	I sleep all night and I work all day.
	I'm a lumberjack, and I'm okay.
	I sleep all night and I work all day.

9,5,6,1,2,3,7,1,2,3

a) What is the Python keyword for defining a function? def

b) On what line is the `print_lyrics` function... defined? 1 called? 6 & 7

c) On what line is the `main` function... defined? 5 called? 9

Model 2 Functional Decomposition and Composition

Questions (20 min) **Start time:**

Your task is to write a program to print three types of guitars on the screen using ASCII characters: a classical guitar, a guitar with a retro head, and a long guitar with a neck twice as long as the classical guitar. We have provided you with all the different print statements needed to draw the three types of guitars:

```
1  #classical guitar
2  print("        ___")
3  print("      o|* *|o")
4  print("      o|* *|o")
5  print("      o|* *|o")
6  print("       \===/")
7  print("        |||")
8  print("        |||")
9  print("     ___|||___")
10 print("    /   |||   \ ")
11 print("   /    |||    \ ")
12 print("  |     |||     |")
13 print("   \   (|||)   /")
14 print("    |   |||   |")
15 print("   /    |||    \ ")
16 print("  /     |||     \ ")
17 print(" /      |||      \ ")
18 print(" |     [===]     |")
19 print("  \             /")
20 print("   '.         .'")
21 print("     '-------'")
22 #retro head
23 print("       ._-_.")
24 print("      +|\G/|+")
25 print("      +|\./|+")
26 print("      +|\./|+")
27 print("       \===/")
28 #long neck
29 print("       \===/")
30 print("        |||")
31 print("        |||")
32 print("        |||")
33 print("        |||")
34 print("     ___|||___")
```

14. Your task is to split these statements between different functions and then combine them by calling them in the correct order. For code that is identical to the provided code indicate the line number or range, do NOT waste time copying it.

```python
def ________________________________________________________:
  # indicate which lines of code go here



def ________________________________________________________:
  # indicate which lines of code go here



def ________________________________________________________:
  # indicate which lines of code go here



def ________________________________________________________:
  # indicate which lines of code go here



def ________________________________________________________:
  # indicate which lines of code go here



def ________________________________________________________:
  # indicate which lines of code go here



def ________________________________________________________:
  # indicate which lines of code go here



def ________________________________________________________:
  # indicate which lines of code go here



def main() -> None:
  # indicate which lines of code go here



main()
```

```python
def print_head() -> None:
  # lines 1-6

def print_retro_head() -> None:
  # lines 23-27

def print_neck():
  # lines 7-8

def print_body():
  # lines 9-21

def print_classic_guitar() -> None:
  print_head()
  print_neck()
  print_body()

def print_retro_guitar() -> None:
  print_retro_head()
  print_neck()
  print_body()

def print_long_guitar() -> None:
  print_head()
  print_neck()
  print_neck()
  print_body()

def main() -> None:
  print_classic_guitar()
  print_retro_guitar()
  print_long_guitar()

main()
```