Links

https://docs.github.com/en/github/managing-files-in-a-repository/adding-a-file-to-a-repository-using-the-command-line

Chapter 1: Getting Started

* Print: print()

Running Python programs from a terminal

* Change directory: cd Desktop\python\_work
* View directory: dir
* Run program: python hello\_world.py

Chapter 2: Variables and Simple Data Types

Variables

* Variables: var = ?

Strings

* Strings: a series of characters, anything in quotes
  + string[x] is the character in position x of string.
* Whitespace: any nonprinting character, such as spaces, tabs, and end-of-line symbols
* Methods: method()
  + Good for storing data, for users who don’t capitalize properly: var.title(), var.upper(), var.lower()
  + Stripping whitespace, which is good for inputs: var.strip(), var.lstrip(), var.rstrip()
* Format: f””
* F-strings: put {var} in quote
* End-of-line symbols
  + Tab: \t, new line: \n

Numbers

* Integers, floats (decimal can be anywhere), underscore doesn’t matter, multiple assignments, constant (make var name in all CAPS)

Comments

* #insert comment here

Chapter 3: Introducing Lists

* Lists: var = []
* var[1] gives you the second element

Changing, adding, and removing elements

* Replace element: var[] = ?, list.append(‘element’), list.insert(2, ‘element’), if you know the position: del list[0], popping one item from the top of the stack/end of the list: list.pop(0), if you know the value: list.remove(‘element’)

Organizing a list

* Alphabetize: list.sort(), reverse alphabetize:; list.sort(reverse=True), temporarily alphabetize: sorted(list), temporarily reverse alphabetize: sorted(list, reverse=True), reverse order of list: list.reverse(), finding the length: len(list)
* Sorted is a function.

Chapter 4: Working With Lists

Looping through an entire list

* for value in list:
* The for loop takes a collection of items and executes a block of code once for each item in the collection.

Making numerical lists

* range(), min(), max(), sum()

Working with part of a list

* A slice is a specific group of items in a list: list[0:3]
* You can put only one number: list[:3]

Tuples

* A tuple is an immutable list: tuple = ()
* If you do not put brackets, it will be a tuple.

Chapter 5: If Statements

A simple example

* If and else  
  if var == ‘element’: do this  
  else: do this

Conditional tests

* Equality: ==, inequality: !=, <, <=, >, >=
* Checking multiple conditions: and, or
* Checking whether a value is in a list: ‘element’ in list
* Checking whether a value is not in a list: ‘element’ not in list
* A Boolean expression is either true or false.

If statements

* if conditional\_test: do something
* if conditional\_test: do something  
  else: do something else
* if conditional\_test\_: do something  
  elif conditional\_test: do something else  
  else: do another thing

Using if statements with lists

* If the conditional\_test is the name of a list, then Python will check to see if the list has at least one element.

Using multiple lists

Chapter 6: Dictionaries

A simple dictionary

* A dictionary is a collection of key-value pairs. A key’s value can be a number, string, another dictionary, or any object that can be created in Python.
* dictionary = {‘key1’: ‘string’, ‘key2’: number}
* dictionary[‘key’], dictionary.get(‘key’, ‘return this if key doesn’t exist’)

Looping through a dictionary

* The method items() returns a list of key-value pairs.
  + for key, value in dictionary.items():
* The method keys() returns a list of keys
  + for key in dictionary.keys():
* The method values() returns a list of values
  + for value in dictionary.values():
* A set is a collection in which each item must be unique.
  + set = {‘element1’, ‘element2’, ‘element1’}

Nesting: storing things within other things

* Storing a dictionary in a list, a list in a dictionary, and a dictionary in a dictionary

Chapter 7: User Input and While Loops

How the input() function works

* input() displays a prompt and receives what’s inputted as a string. The output of input() is what was inputted.
  + Adding to a string: string += more\_string
* int() converts the argument into an integer.

The modulo operator

* The module operator (%) divides one number by another number and returns the remainder.

Introducing while loops

* The while loop runs as long as a condition is true.

Using a flag

* A flag is a variable that determines whether or not the entire program is active. A flag acts as a signal to the program.
* If flag == True, program continues to run. If flag == False, program ends.

Using break to exit a loop

* The break statement directs the flow of your program.
  + if var == ‘string’: break

Using continue in a loop

* The continue statement returns you to the beginning of the loop based on the result of the conditional test.
* CTRL+C or just close the terminal window if you’re stuck in an infinite loop.

Using a while loop with lists and dictionaries

* You can go through a list as long as it has elements in it.
  + while list:
* You can use a while loop to remove every instance of an element in a list.
  + while ‘element’ in list: list.remove(‘element’)

Chapter 8: Functions

* A function is a block of code that is designed to do one specific job.
* Modules are separate files to store functions.

Defining a function

* def function():
* Docstrings are enclosed in triple quotes, which Python looks for when it generates documentation for the functions in your programs.
  + “““docstring”””
* A function call tells Python to execute the code in the function.
* To call a function, write function().

Arguments and parameters

* A parameter is a piece of information the function needs to do its job.
  + def function(parameter):
* An argument is a piece of information that’s passed from a function call to a function.
  + function(‘argument’)

Positional arguments

* def function(parameter1, parameter2):
* function(‘argument1’, ‘argument2’)

Keyword arguments

* function(parameter1 = ‘argument1’, parameter2 = ‘argument2’)

Default values

* def function(parameter1, parameter2=‘argument2’):
  + Default values can be overridden if you use new arguments when you call the function.

Return values

* The value the function returns is called a return value.
* The return statement takes a value from inside a function and sends it back to the line that called the function.

Making an argument optional

Returning a dictionary

* None can be thought of as a placeholder value. In conditional tests, None evaluates to False.

Passing a list to a function

* function(list1, list2)

Preventing a function from modifying a list

* You can send a copy of a list to a function, so that the function doesn’t modify the original list.
  + function(list[:])