Assignment 1: Basics

Language: Python

Topics and Keywords / Symbols Used

This assignment covers the topics in the table. There are certain keywords that the given topic uses which is listed below the given topic. Not all the symbols and keywords will be used, but many will.

Table

Variables	Collections	Conditional Statements	Iteration	Functions	Exceptions	Modules
=	[]	if(c):	for	def f(p):	try:	import
input(str)	{}	elif(c):	while(c)	return	except exc:	from
int(e)	pop(i?)	else:	range(o,n,Δ)			
	append(e)		in			
	len(l)					

Legend

Character	Definition
c	some condition must go here
e	some element or item must go here
o	some original value
n	some ending value
Δ	some changing factor
exc	a specific exception
f	function name
p	parameter
str	some string
i	index
l	list
?	optional value

Note on Topics

Files aren't included in any of these parts as that goes beyond the pure basics. This assignment is an assignment to apply your knowledge and understanding of the basics of Python. The next assignment will include harder or more complicated topics such as Files and Recursion. For now, as this is to build your foundation, the topics will be limited to more foundational topics and aspects of Python. The topics and keywords are generally limited to the list above. The aforementioned list isn't an extensive list of all the keywords you may or will use, but they contain the general terms you will likely require.

Explanation

This assignment has a couple parts. Read below for the descriptions of the parts and their breakdowns. The parts aren't related so you don't have to do them in any particular order, but I'd advice going down the parts. Not all parts use everything you've learned, but most of the concepts in the list above are included.

At the very bottom, there's a reference guide attached. This reference guide includes a brief writeup of what the keywords or functions do along with what they return, if they return anything.

Task

The tasks are listed below. The descriptions are on the subsequent pages.

Part 1 : Mathematical Functions
Part 2 : Shopping List
Part 3 : Phone Contacts

Online Resources

I recognize that I can't prevent you from searching up everything and using StackOverflow or similar resources. I do advise using these resources, but I also recognize that you hardly learn when you use them. If you're going to use any online resource, use a comment and say where you got it from. I'm most likely going to still ask you to code it differently and possibly code it again, but only if I suspect that you don't know what the code you copied does.

Make an attempt to code it all on your own. Review the notes from our sessions as I believe they cover the important aspects for these projects.

Questions

If you have any questions, feel free to contact me. I'll be checking my email and my socials over the break, so you should be able to contact me with anything we have, except by text messages as I'll be back on my HK number.

Submission

When you're done with everything, or if you just want me to check through any part, send me the compressed version of all the files (or the parts) you have. Tell me which parts you've done so I know what I'm looking at. I'll run the code and see how you did everything. I'll send you feedback regarding your implementation, naming conventions, or potential problems that may have been missed.

Good Luck!

Now that that's all out of the way, good luck with everything! If you can solve all of this by the time we get back from the break, you're going to be perfectly fine going into CS110. If you're struggling, let me know and honestly ask me questions. I'm happy to answer any questions you may have.

Part 1: Mathematical Functions

In this part, you will write a number of functions progressively getting more difficult. These functions will be laid out in the following table with the name of the function, what parameters are used, what the return type and values are, and what exception may need to be caught. The only two builtin functions or variables you may use are *math.sqrt* and *math.pi*.

Function name	Parameters	Return Type	Return Value / Description	Exception
add	(a,b)	int	sum of a and b	N/A
subtract	(a,b)	int	difference between a and b	N/A
multiply	(a,b)	int	product of a and b	N/A
divide	(a,b)	int	quotient of a over b	ZeroDivisionError
pow	(a,b)	int	power of a^b	N/A
pythagoras	(a,b)	int	the value of c $c = \sqrt{a^2 + b^2}$	N/A
dist	(x1, y1, x2, y2)	int	distance between coordinates $\sqrt{(x1 - x2)^2 + (y1 - y2)^2}$	N/A
quad	(a,b,c)	[list]	possible values as a list for values of a, b, and c $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$	ZeroDivisionError
degtorad	(deg)	int	the radian equivalent to a degrees value $rad = \frac{deg * \pi}{180}$	N/A
isprime	(a)	boolean	True if number is prime, False otherwise	N/A
sum	(n)	int	sum of the values from 0 to n $(\sum_{i=0}^{n} i)$	N/A
fact	(n)	int	factorial of n $(n!)$	N/A
sumofsquares	(n)	int	sum of the squared values from 0 to n $(\sum_{i=0}^{n} i^2)$	N/A

Part 2: Shopping List

In this part, you are going to write a small program to serve as a shopping list. It will have an interactive component to it. You should have at least one variable of the list being used as the shopping list. Shopping list should start as an empty shopping list. Functions should be used to manipulate the list. You can implement as many helper functions, but the functions below must be present and functional.

Function Name	Parameters	Return Type	Return Value / Description	Potential Problems
contains	(e)	boolean	True if found in list, False otherwise	N/A
add	(e)	boolean	Add element to list, return True if was added, False if it couldn't be added	N/A
remove	(e)	string	Item removed	Item Not In List
find	(e)	int	Index of item if found, -1 if not found	N/A
clear	N/a	N/a	Clears or resets the list	N/a
printall	N/a	N/a	Print out values in list as a visual list Example: Shopping List: - Apples - Tomato Sauce	List is empty
isover*	N/a	boolean	End interactive aspect	N/a
getinput*	N/a	str	Get the input from the user	N/a
validate*	str	int	Used to receive action from user as to what to do. If `ValueError`, Must find a way to validate what the user inputs and keep asking until they give a valid response	`ValueError`

Note: * denotes function is used for interactive aspect.

Part 3: Phone Contacts

In this part, you will create an empty dictionary that will store the contacts of a person. In essence, you are writing a system similar to how your phone or email keeps track of your contacts.

The dictionary named `contacts` will store key:value pairs of:
`contactName:[phoneNumber, address, email]`.

Func. Name	Params	Return Type	Description	Exception
contains	(name)	boolean	Checks if name is in dictionary, if found, return True, if not, return False	KeyError
add	(name, phoneNumber, address, email)	boolean	Adds to dictionary if not already in dictionary. return True if successful, False is record already found	N/a
remove	(name)	boolean	Removes record from dictionary, returns True if removed, False if not removable	KeyError
get	(name)	[list]	Returns records of the name if found, returns empty list if no record found	KeyError
printall	N/a	N/a	Print out all records in the dictionary with format of : Name: {name}, Number: {phoneNumber}, Address: {address}, Email: {email}	N/a
sortkeys	(contacts)	[list]	Sort the keys of the contacts in alphabetical order	N/a
reordercontacts	(sortedkeys)	{dictionary w/ sorted keys}	Using the list of sorted keys, reorganize the contacts and return a new ordered contacts list	N/a

Extra: Make it interactive using the interactive components / functions from Part 2!

Reference Guide

This page serves as a reference guide as to what each keyword or built in function does.

Keyword/Function	Purpose	Return / Crashed Exception
input	Takes in a string parameter to print to the user when requesting a value	Returns the value input by the user as a string
int	Takes in an element, attempts to cast to an integer	int value of element passed in, crashes with `ValueError` if cannot cast element to an int
pop	Removes item from list based on index. Removes last if no parameter given.	element removed
append	Takes in an element and adds it into the list	Nothing is returned (void)
len	return length of list taken in	returns length of list
if	executes code if specific condition is met	N/a
elif	executes code if `if` is not met, but given condition is met	N/a
else	executes code if `if` and `elif` clauses are not met, no condition is used	N/a
for	uses a variable provided to iterate through a range or collection	N/a
while	takes a condition and keeps looping until condition is met	N/a, can cause infinite loop
range	takes an original starting value, ending value, and some changing factor. Allows for iteration through a series of numbers	a sequence of iterable numbers
in	used for the `for` loop to iterate through range or collection	N/a
def	defines a function with any number of parameters	N/a
return	allows a value to exit a function and be used elsewhere	N/a
try	attempts to execute code	N/a
except	if try fails, will go to except clause with specific exception that broke the code	N/a
import	imports a module	N/a
from	used together with imports for shorter use `from math import pi` allows for the use of `pi` instead of `math.pi`	N/a