Intro to Programming (No Prior Experience)

Topics Covered on Final

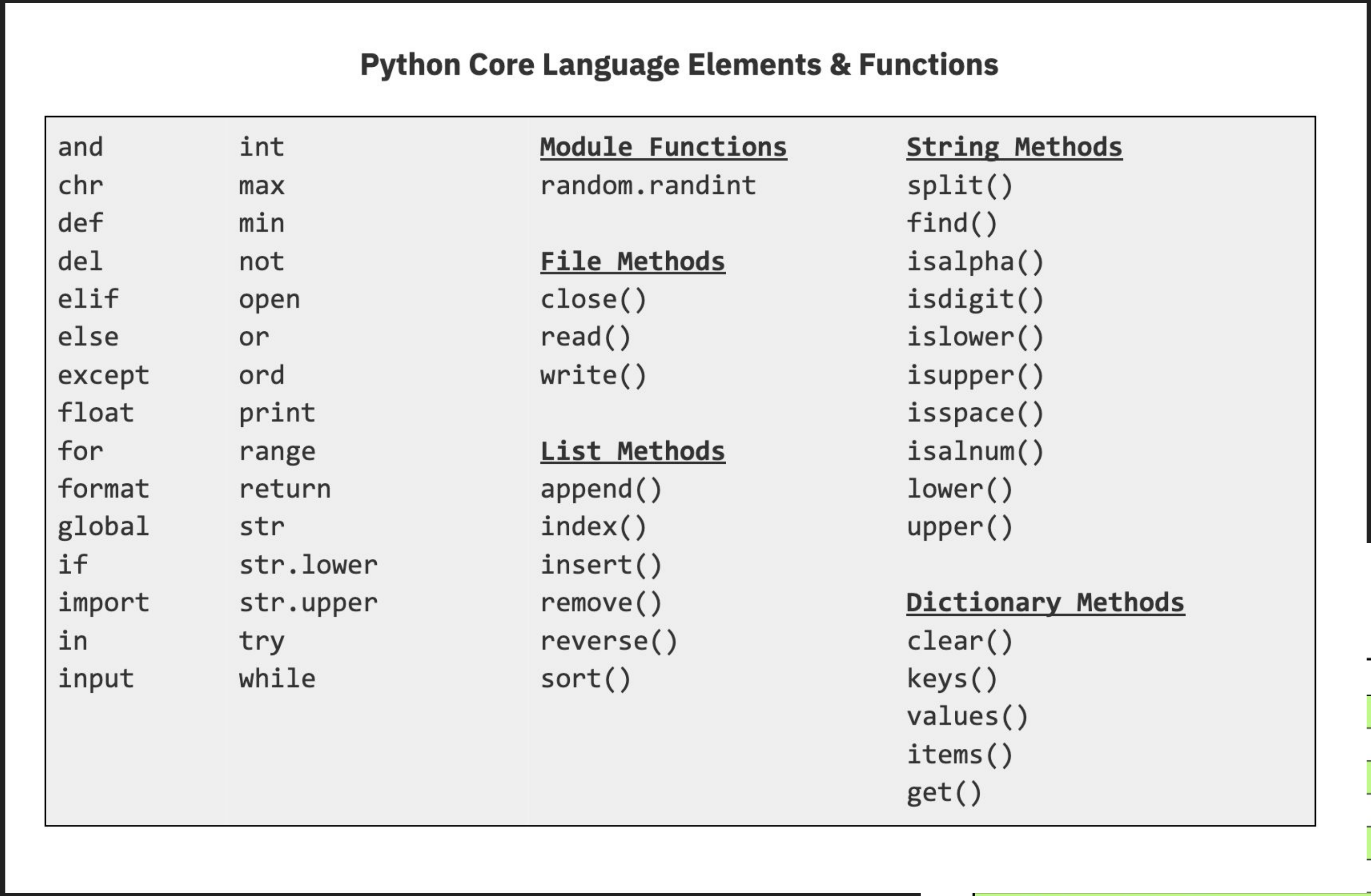
Emily Zhao
CSCI-UA-0002

Exam Info

- In-person, in classroom
- Time limit: 1 hour 15 minutes
- Paper exam
  - Will be scanned, writing in pen recommended
  - Scratch paper will be provided!

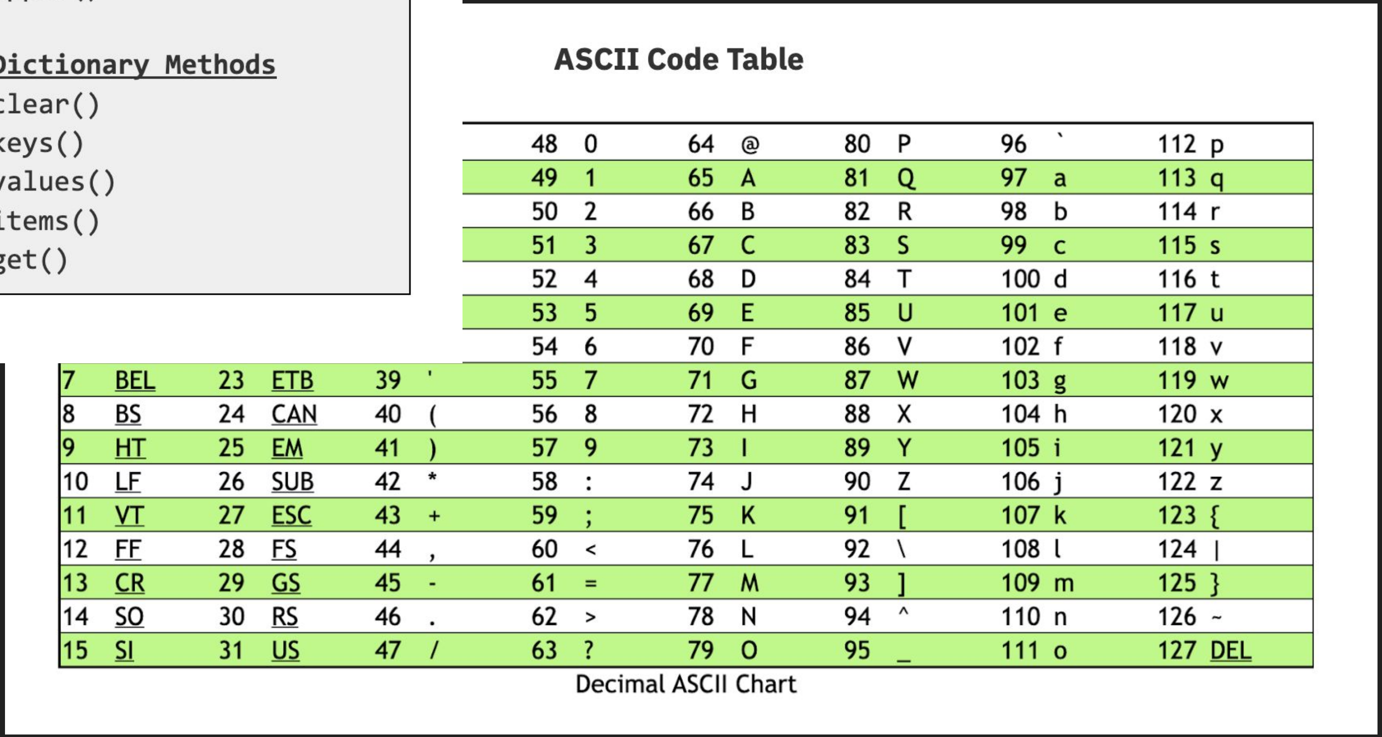
Question Types

- Short fill-in-the-blank
  - What’s the output?
  - What’s the missing code?
- Multiple Choice
- **Long programming questions**
  - These are worth the most points so I recommend starting with these!
  - I will also give partial credit for pseudo code that has good logic!



Python Core Language Elements & Functions

and	int	**Module Functions**	**String Methods**
chr	max	random.randint	split()
def	min		find()
del	not	**File Methods**	isalpha()
elif	open	close()	isdigit()
else	or	read()	islower()
except	ord	write()	isupper()
float	print		isspace()
for	range	**List Methods**	isalnum()
format	return	append()	lower()
global	str	index()	upper()
if	str.lower	insert()	
import	str.upper	remove()	**Dictionary Methods**
in	try	reverse()	clear()
input	while	sort()	keys()
			values()
			items()
			get()

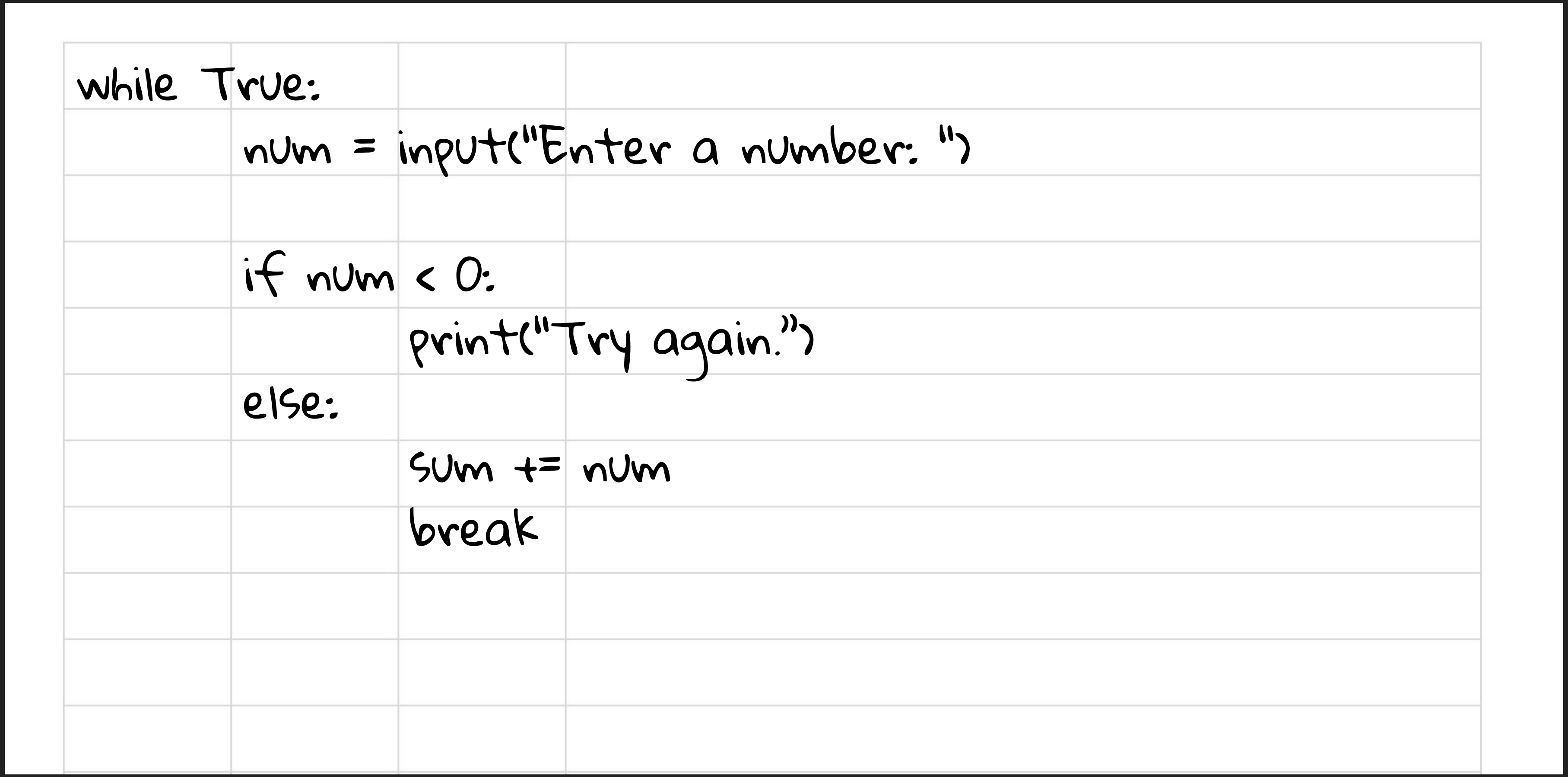


ASCII Code Table

						48	0	64	@	80	P	96	`	112	p
						49	1	65	A	81	Q	97	a	113	q
						50	2	66	B	82	R	98	b	114	r
						51	3	67	C	83	S	99	c	115	s
						52	4	68	D	84	T	100	d	116	t
						53	5	69	E	85	U	101	e	117	u
						54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	|
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

Decimal ASCII Chart

Long programming paper



```
while True:
    num = input("Enter a number: ")

    if num < 0:
        print("Try again.")
    else:
        sum += num
        break
```

^ ^ ^ ^ Lines are guides for indentation

Modules 1 – 11 will be covered on the final

- Module #1 (Variables, Statements, etc.)
- Module #2 (Types, Operators, Debugging)
- Module #3 (Boolean Logic, Using Modules)
- Module #4 (While Loops)
- Module #5 (For Loops, Nested Loops)
- Module #6 (Functions)
- Module #7 (Strings, Sequences, Slicing)
- Module #8 (Lists)
- Module #9 (Exceptions, Input/Output)
- Module #10 (Dictionaries)
- Module #11 (Object Oriented Programming)

Math Expressions

- Math operators (+, -, /, //, *)
- Writing math expressions
- Evaluating math expressions
- Storing & printing the results of math expressions
- Difference between the two division operators (/ and //)
- Order of operations in math expressions
- The exponent operator (**)
- The modulo operator (%)

First-half Material

Second-half Material

Data Types

- What is a data type?
- Strings
- Numeric data types
  - Integers (int)
  - Floating point numbers (float)
- Mixed type expressions
- Data type conversion
  - Using the float() and int() function to convert strings into numbers
  - User input & data types (converting strings to floats / ints for calculation purposes)
- The Boolean data type
- Boolean variables

Output with the print() function

- General use of the print function and its default behavior
  - Unlimited arguments
  - Spaces inserted between arguments
  - Line break after each call to the function
- Customizing line endings (end=”)
- Customizing argument separators (sep=”)
- Escape characters (\n, \t, etc.)

Basic String Manipulation

- Combining two strings (concatenation) – "+" operator
- Multiplying a string (repetition) – "*" operator
- Formatting numbers using the format() function
  - Formatting Strings – width, left align, right align, center align
  - Formatting Integers – width, left align, right align, center align
  - Formatting Floats – width, left align, right align, center align, # of decimal places, "," separator
- Case manipulation using str.lower() and str.upper()
- Calculating string length using the len() function

Selection Statements

- The structure of an IF statement (IF keyword, condition, colon, indentation)
- Writing a condition for an IF statement
- Boolean operators (<, >, ==, !=, >=, <=)
- Comparing numeric values using Boolean expressions
- Comparing string values using Boolean expressions
- Using the IF-ELSE statement
- Nesting decision structures (IF statements inside other IF statements)
- The IF-ELIF-ELSE statement
- Logical operators (and, or, not)

Condition Controlled Loops

- The structure of a "while" loop
- Mechanics & how they work
- Setting up conditions for a while loop
- Infinite loops and how to work with them
- Sentinels (defining a value that the user enters that causes the loop to end)
- Input validation loops (asking the user to continually enter a value until that value matches some condition)
- Setting up and using accumulator variables
- Self referential assignment statements (i.e. counter = counter + 1)
- Augmented assignment operators (i.e. counter += 1)

The Range Function

- mechanics and how the function works
- creating simple ranges (i.e. range(5))
- creating ranges with defined start and end points (i.e. range(3,10))
- creating ranges with a step value (i.e. range(5,50,5))
- creating ranges that count backwards (i.e. range(50,5,-5))
- user controlled ranges (i.e. range(1, somevariable))

Functions

- mechanics and how functions work
- function definitions
- arguments
- return values
- calling a function
- local variables (variables that are defined inside a function and can only be accessed inside that function)
- passing arguments to your own functions
- passing multiple arguments to your own functions
- global variables (variables created outside a function that can be accessed by any part of your program)
- making changes to global variables inside a function using the ‘global’ keyword
- writing a value returning function (i.e. using the ‘return’ keyword to send a result from your function to the part of your program that called that function)
- returning multiple values from a function
- Input, Processing & Output notation

Miscellaneous Concepts

- Generating random numbers
- Errors & error types
- Debugging strategies
- Pseudocoding

Modules

- Creating a module
- Defining functions in a module
- Calling functions in a module

Exceptions

- Preventing exceptions using selection statements (i.e. using an “if” statement to prevent an error from occurring)
- Using the try / except / else suite to test problematic code for an error and “catch” it before it has a chance to crash your program.

Lists

- Simple Variables vs. Lists (simple variables can only hold one piece of data, but lists can hold multiple values) – you can think of a list like a “book” and a variable like a “sheet of paper”
- Defining lists in Python (i.e. mylist = [1,2,3])
- Concatenating lists with the “+” operator
- Repeating lists with the “*” operator
- Referencing list items using index notation (i.e. mylist[0])
- Iterating through a list using a “while” loop
- Iterating through a list using a “for” loop
- Using the len() function to determine the # of items in a list
- Updating the value of an item in a list using bracket notation
- Creating empty lists

Lists (cont’d)

- Finding an item in a list using the “in” operator
- Adding items to a list using the append method
- Sorting items in a list using the sort method
- Reversing items in a list using the reverse method
- Finding the position of an item in a list using the index method
- Inserting an item in a list at a specific index using the insert method
- Finding the largest and smallest values in a list using the min and max methods
- Totaling the values of all elements in a list using an accumulator variable
- Removing an item from a list using the remove method
- Storing lists in files
- Reading lists from files using the readlines method

Strings Manipulation

- Iterating through all characters in a string using a for loop
- Indexing a specific character in a string using bracket notation
- Iterating through all characters in a string using a while loop
- String immutability (you can’t change a string using bracket notation like you would change a list element)
- Testing a string for substrings using the “in” operator
- Detecting character types in a string using the built-in string testing methods (isdigit, isalpha, isalnum, islower, isupper, isspace)
- Splitting a string into a list using the “split” method

File Input & Output

- Opening a file for writing
- Opening a file for reading
- Writing data to a file
- Delimiters (separating data in a file)
- Reading data from a file using the read() method
- Reading data from a file into a list
- Processing data stored in a file

Dictionaries

- What is a Dictionary?
- Basic usage of Dictionaries
- Differences between Lists and Dictionaries
- Knowing when to use which data type (lists for numerically indexed data, dictionaries for string-indexed data)

Object Oriented Programming

- What is a class?
- Setting up instance variables
- Accessing instance variables through 'dot syntax'
- Constructor functions
- Mutability of instance variables
- Instance methods
- The 'self' keyword