15-110 Review Session - Quiz 1

-The 110 TAs

Algorithms and Abstraction

Overview:

- Computer Science is the study of **computation** and **computational devices**
  - Computation = Algorithms & Abstraction
- **Algorithms:** procedures that specifies how to do a task or solve a problem; used to standardize processes so that they could be communicated to others! (like recipes, gps directions, instructions in a lego set)
- **Abstraction:** a technique used to make complex systems manageable by changing the amount of detail used to represent or interact with the system; identifying most important features of a system and generalizing unessential ones (like

Plain-Language Algorithm Practice:

Task: A student is beginning is first day of class at CMU, they live in Mudge and need directions to the school of music practice rooms. Give an algorithm/set of directions for them to get there.

Plain-Language Algorithm Practice:

Task: A student is beginning is first day of class at CMU, they live in Mudge and need directions to the school of music practice rooms. Give an algorithm/set of directions for them to get there.

1. Walk south on Morewood Ave
2. Cross Forbes intersection
3. Walk past the Tennis Courts
4. Enter CFA
5. Take the elevator to Floor M

Adding more intermediate steps lowers abstraction, taking out steps increases abstraction

Adding more steps might start to include unessential/obvious steps (like first walk with your left foot, then right) You need to identify the more important steps to leave in the algorithm/instructions.

Variables, Data types and Error Messages

Variables

- stores data
  - *variable = expression*
- case sensitive
  - ex. foo != Foo
- able to change value of a variable
- define variable before use
  - Python is sequential

Data Types

- Int
- Float
  - + - / // * **
- String
  - “ ” or ‘ ’
- Bool
  - < > <= >= == !=

Error Types

- Runtime
  - ex. TypeError, ZeroDivisionError
- Syntax
  - ex. IndentationError
- Logical

Practice

Trace variable values:

x = 5

y = x

x = y + 3

y = x - 2

z = 1 == 2

print(x, y, z)

Predict the data type:

- 5 * 2.0
- 8 / 4
- “abc” == “ABC”
- 10 // 2
- “6” + “10”

Determine and fix the error:

- print(“Hello’)
- print(“This is” + 15110)
- 42 = a
- Return “Hello world”
- addNum = (6 * 7)

Practice Solutions

Trace variable values:

x = 5

y = x

x = y + 3

y = x - 2

z = 1 == 2

print(x, y, z)

8 6 False

Predict the data type:

5 * 2.0 float

8 / 4 float

“abc” == “ABC” bool

10 // 2 int

“6” + “10” string

Determine and fix the error:

- print(“Hello’)
  - syntax
  - “Hello” or ‘Hello’
- print(“This is” + 15110)
  - runtime
  - “This is” + “15110”
- 42 = a
  - syntax
  - a = 42
- Return “Hello world”
  - syntax
  - return “Hello world”
- addNum = (6 * 7)
  - logical
  - addNum = 6 + 7

Binary Numbers

Decimal

1 * 10^4	5 * 10^3	1 * 10^2	1 * 10^1	0 * 10^0
1	5	1	1	0

1 * 10^4 + 5 * 10^3 + 1 * 10^2 + 1 * 10^1 + 0 * 10^0 = 15110

Binary

1 * 2^4	0 * 2^3	1 * 2^2	1 * 2^1	1 * 2^0
1	0	1	1	1

1 * 2^4 + 50* 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 10^0 =
= 16 + 4 + 2 + 1
= 23

Binary Conversion Practice

Convert binary to decimal and decimal to binary

1. 10110100
2. 01101100
3. 135
4. 220

Binary Conversion Practice

Convert binary to decimal and decimal to binary

1. 10110100 -> 180
2. 01101100 -> 108
3. 135 -> 10000111
4. 220 -> 11011100

Quick Vocab

Bit
- a single 1 or 0

Byte
- 8 bits

Unsigned Integer
- Represents only positive numbers

Signed Integer
- Represents a range of negative and positive numbers

Integer Overflow
- When an integer gets so positive or negative that it wraps back to either to max or min value

RGB

RGB values make different colors using red, green, and blue components.

- Uses a single byte to represent each color

https://www.w3schools.com/colors/colors_rgb.asp

ASCII

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

ASCII Practice

Convert strings to binary (a whole byte) and binary to strings

1. “Ah!”
2. 00110001 00110101 00110001 00110001 00110000

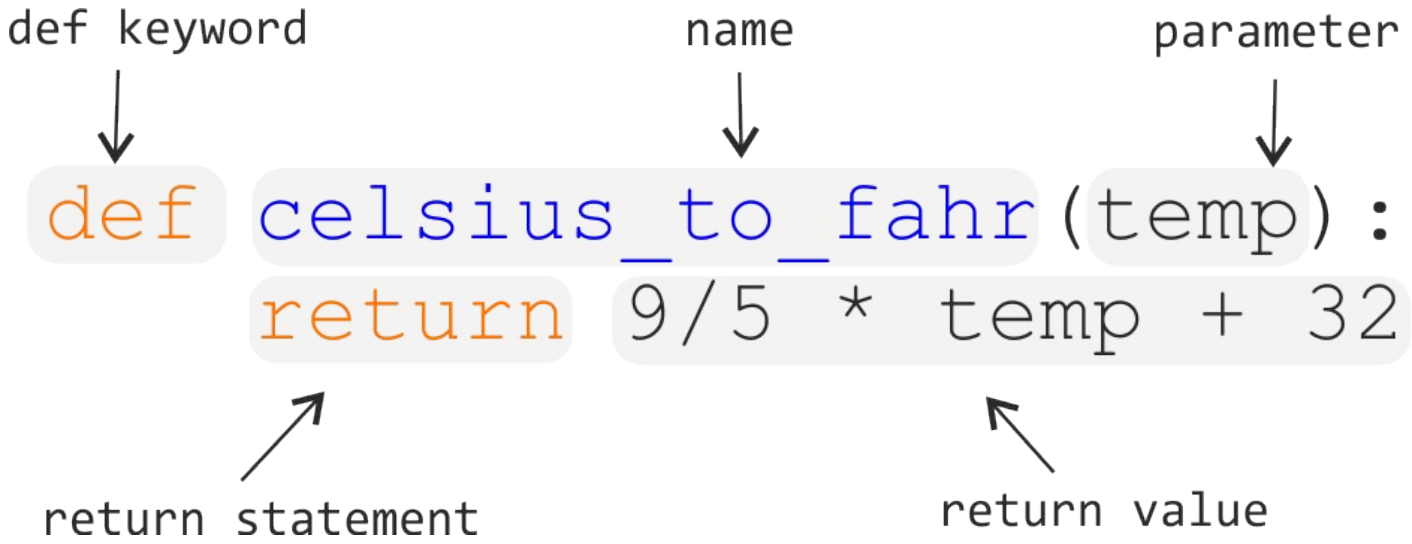
ASCII Practice

Convert strings to binary (a whole byte) and binary to strings

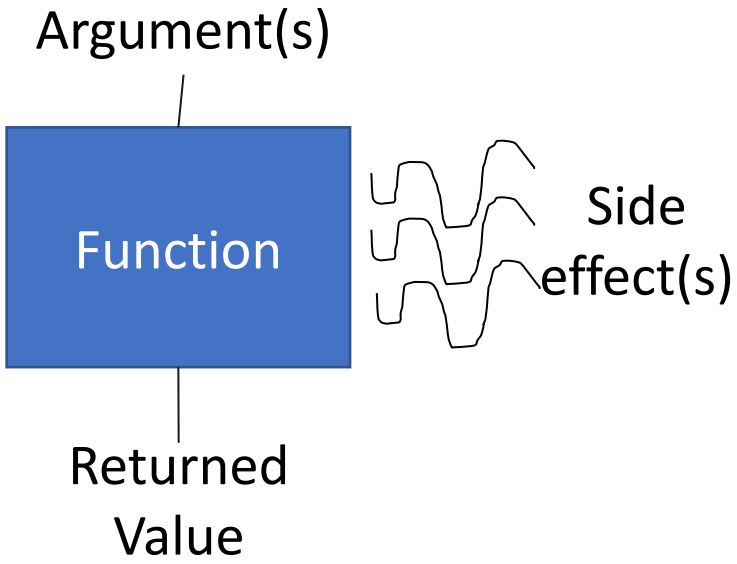
1. “Ah!” -> 01000001 1101000 00100001
2. 00110001 00110101 00110001 00110001 00110000 -> 15110

Functions

Anatomy of a Function



Function Calls & Libraries




```
import random, math

random.random()
random.randint(a,b)

math.ceil(a)
math.round(a)
math.floor(a)
```

Scope

Local Variables - function parameters, variables initialized inside a function

Global Variables - Anything initialized outside a function

Practice Problem

The 110 Professors decide that the TAs should take their recitation to dinner, using some new CS grant money. They have a great meal, but then the check comes. Oh no! They have to calculate a tip! And then it turns out the TA's were only given $50 for the whole recitation! The 110 students band together and help the TAs out in figuring out how much everyone owes. They create a function:

```
def studentCost(bill, tipPercent, numStudents)
```

Which takes in the bill cost, the tip percentage, and the number of students in the recitation section and returns the cost to each student.

NOTE: If the original starting bill is over 200 dollars, we want to print a message saying "You bought too much food!" and still return the student cost.

Ex: studentCost(175, 0.20, 15) = 10.66

Solution

```
def studentCost(bill, tipPercent, numStudents):
    if bill > 200:
        print("You bought too much food!")
    total = bill + bill*tipPercent
    total = total - 50
    return total/numStudents
```

Call Stacks

Overview:

- **Call Stacks** are how Python keeps track of nested function calls, functions waiting to be given a value;
- In this class call stacks grow down, basically listing (1, 2, 3, ...) the functions in the order they appear, with the first function as 1.

Practice:

Write the call stack for the given code.

What is the value for each function in the call stack?

How much money remains? (value of `wallet`)

```
def calculateRate(x):
    return x / 100

def calculateTip(cost, x):
    tipRate = calculateRate(x)
    return cost * tipRate

def payForMeal(cash, cost):
    cost = cost + calculateTip(cost,
    cash = cash - cost
    return cash

wallet = 20.00
wallet = payForMeal(wallet, 8.00)
```

Practice:

Call Stack:

wallet = payForMeal(wallet, 8.00)	= cash = 20 - 9.6 = 10.4
cost = cost + calculateTip(cost, 20)	= 8.00 + (8.00 * 0.2) = 9.6
tipRate = calculateRate(x)	= calculateRate(20) =>20/100 = 0.2

wallet = 10.4

```
def calculateRate(x):
    return x / 100

def calculateTip(cost, x):
    tipRate = calculateRate(x)
    return cost * tipRate

def payForMeal(cash, cost):
    cost = cost + calculateTip(cost,
    cash = cash - cost
    return cash

wallet = 20.00
wallet = payForMeal(wallet, 8.00)
```

Good Luck!