Algorithms and Abstraction

1. As you learned in lecture, computer science boils down to two main ideas: algorithms and abstraction.
    a. Identify/Define the difference between algorithms and abstraction, and provide an example for each.
    b. What are the 5 key components of a good algorithm?

    Answer:
    a. Algorithms: algorithms are a set of instructions to solve a given task or problem.
        i. An example of an algorithm is a food recipe or a “how-to” manual

    Abstraction: technique used to make complex systems manageable by reducing how much detail is used to represent/interact with the system
        ii. An example of abstraction is your internet browser. Instead of needing to know the exact destination of the information you want to see on the internet, you can search websites by their url or by keywords relating to it.
    b. Key components:
    1) **Input:** Should have a specified input needed in the beginning
    2) **Output:** Should have a specified output produced at the end
    3) **Correctness:** Should produce the correct output for all given inputs
    4) **Efficiency:** Should NOT take too long to finish
    5) **Clarify:** Others should be able to understand/modify it

2. Trace the following algorithm with the input of 5 (n=5).
    Algorithm:

```
Step 1: Input a number n
Step 2: Set variable x equal to 1
Step 3: Set variable y equal to 1
Step 5: Set variable count equal to 0
Step 6: If count is less than n do the following, otherwise skip to Step 7
        Step 6a: Print the string “Value: “ + str(x)
        Step 6b: If count is equal to 5:
                Step 6b i: Print the string “Oops”
        Step 6c: Set variable temp equal to the sum of variables x and y
        Step 6d: Set variable x equal to variable y
        Step 6e: Set variable y equal to variable temp
        Step 6f: Set variable count equal to the value of count + 1
        Step 6g: Repeat step 6 with updated values
Step 7: Print the string “Result: “ + str(x)
```

    Answer the following questions:
    a. How many times do we run step 6 only? (be sure to ONLY count step 6 itself, not the sub steps of 6) In other words, how many times is the comparison in step 6 evaluated?

b. How many times do we print “Oops”?
c. How many lines are printed and what does each line read?
d. Describe in words what this algorithm does.

Answer:

a. Step 6 runs **6 times**. (0<5, 1<5, 2<5, 3<5, 4<5, 5<5) Note that we still count the final comparison of 5<5 because that statement must be evaluated before we know it is False.
b. “Oops” never prints because the if statement count==5 is never True for input n = 5 (when count = 5, step 6 evaluates to False and the algorithm will skip to step 7)
c. There are 6 print statements.

```
Value: 1
Value: 1
Value: 2
Value: 3
Value: 5
Result: 8
```

d. This algorithm prints all Fibonacci numbers up to and including the nth Fibonacci number. Alternatively, the algorithm takes two variables, printing the value of the first variable, and continuously updates the values of the two variables so that the first variable is updated to be the value of the second variable and the second variable is updated to be the value of the sum of the two variables. It does this n times and then prints the final value of the first variable.
(If you don’t know what the Fibonacci sequence is, you can read about it here https://en.wikipedia.org/wiki/Fibonacci_number)

Programming Basics

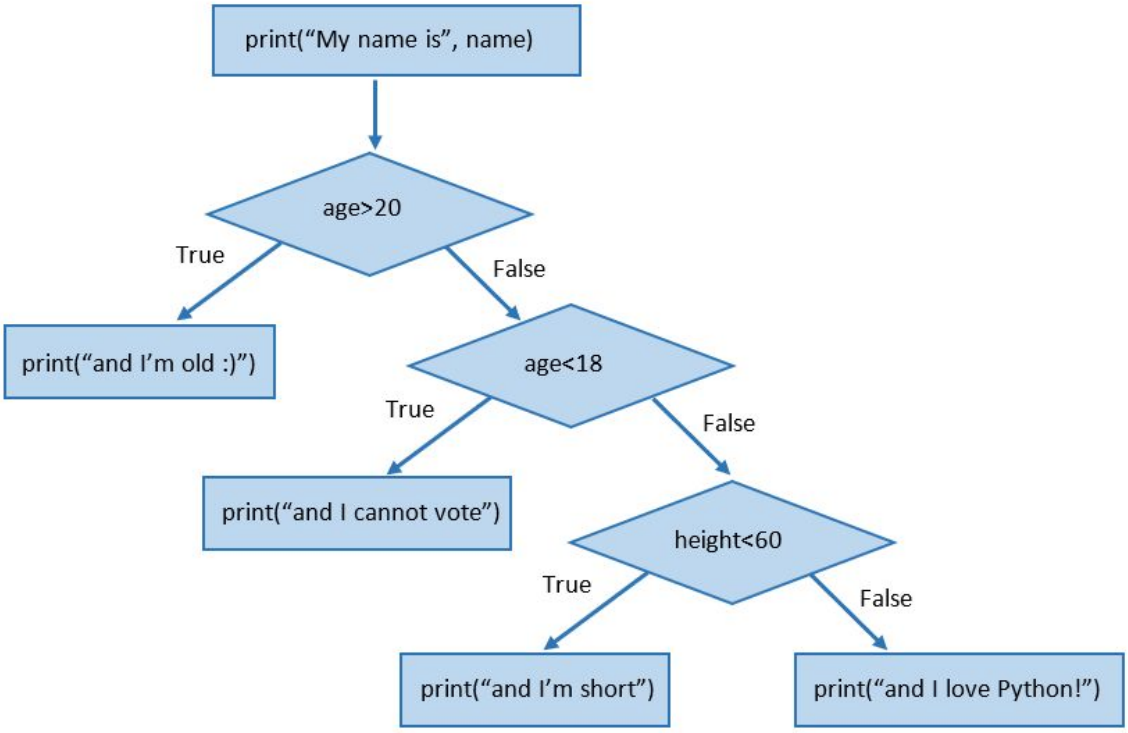
3. For each of the following Python expressions, identify the data type of the value it will evaluate to.
   a. 3
   b. 3.0
   c. “3”
   d. 3 < 3
   e. “3” + “3”
   f. 3.0 * 3.0
   g. 3 / 3
   h. 3 // 3

Answer:

a. Integer (int)
b. Floating point number (float)
c. String (str)

d. Boolean (bool)
e. String (str)
f. Floating point number (float)
g. Floating point number (float)
h. Integer (int)

4. Answer the following questions about variables you could use to describe yourself.
   a. Suppose you are creating a few variables to describe yourself or another person. Name the data type that would **best** fit each variable description:
      i. Name
      ii. Age
      iii. Height (inches)
      iv. Ability to vote
      v. Address
      vi. Zip code
      vii. Weight
      viii. School you attend
      ix. Ability to speak more than one language
   b. Consider the following flow chart (which uses some of these variables), and convert it into equivalent Python code.



Answer:

a. Variable types:

i. Str
ii. Int (float could work but is less fitting)
iii. Float
iv. Bool (look below)
v. String
vi. Int or String
vii. Int or Float
viii. String
ix. Boolean

b. Equivalent code:

```
print("My name is ", name)
if age>20:
      print("and I'm old")
elif age<18:
      print("and I cannot vote")
elif height<60:
      print("and I'm short")
else:
      print("and I love Python!")
```

5. Explain the following error messages and provide one possible fix:

a.

```
Traceback (most recent call last):
  File "<tmp 1>", line 1, in <module>
    print("hello"+1)
TypeError: can only concatenate str (not "int") to str
```

b.

```
    print("name"",name)
                  ^
SyntaxError: EOL while scanning string literal
```

c.

```
def f():
    print("hello 15-110")
        return 15-110
```

```
  File "<tmp 1>", line 3
    return 15-110
    ^
IndentationError: unexpected indent
```

Answer:

a. Type error: You cannot add a string to an integer. You can change 1 to the string of "1" to add them together, assuming the string you would like to print is "hello1".

b. Syntax error: The quotation marks do not match properly and Python cannot interpret them. Fix this by removing one of the quotes that are next to each other, so the statement should read print("name", name).
c. Indentation Error: The return statement should not be indented. Fix this by deleting the indent and realigning the return statement with the print statement.

6. Evaluate the print statements and state what the value of y and x are at the end of the function.

```
x = 15+1

def f():
      y = 12
      print(y * 2)
      print(y)
      y = "apple"
      print("bananas, 15-110, and",y)
      print(x+3)

f()
```

Answer:

Statements printed:

24
12
"bananas, 15-110, and apple"
19

The value of y at the end of the code is "apple". The value of x is 16.

Data Representation

7. Answer the following questions regarding number systems:
   a. What is a number system?
   b. How many values can a specific digit have in a decimal number? In a binary number?
   c. Let's say that we are representing length in imperial units. You are given inches, feet, and yards. How could you represent 10,000 inches in this system (show your work)? (hint: 12 inches=foot, 3 feet=yard 1760 feet=mile)

Answer:

a. A way of representing numbers using symbols
b. Decimal: 10 (0,1,2,3,4,5,6,7,8,9)
   Binary: 2 (0,1)
c. 10000 [inches] // 36 [inches] = 277 [yards]
   10000 [inches] % 36 [inches] = 28 [inches]
   28 [inches] // 12 [inches] = 2 [feet]

28 [inches] % 12 [inches] = 4 [inches]

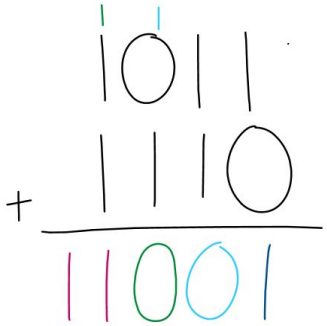
Yards	Feet	Inches
277	2	4

277 yards and 2 feet and 4 inches is equivalent to 10,000 inches
(277yards + 2 feet + 4 inches) == (10000 inches)

8. Answer the following questions about binary addition
    a. Add binary numbers 1011 and 1110 and be sure to show your work.
    b. How many bits did you use to represent the sum? Could the result be represented in fewer bits?

Answer:

a. The resulting sum is 11001



Check your work: 1011 -> 11
1110 -> 14
11+14=25
25 -> 11001 (same answer!)

b. The result is represented in 5 bits. No, it could not be represented in fewer bits, the largest number that can be represented in 4 bits is 15.

9. Complete the following conversions and show appropriate work:
    a. Convert 10111011 from binary to decimal
    b. Convert 11111111 from binary to decimal
    c. Convert 01101110 from binary to decimal
    d. Convert 57 from decimal to 8-bit binary
    e. Convert 63 from decimal to 8-bit binary
    f. Convert 88 from decimal to 8-bit binary

Answer:

a. 187 (1+2+8+16+32+128=187)
b. 255 (1+2+4+8+16+32+64+128=255 OR note that the next bit would be 256, so we know making all 8 bits 1 will be 255)
c. 110 (2+4+8+32+64=110)
d. 00111001 (57=32+16+8+1)

e. 00111111 (63=32+16+8+4+2+1)
f. 01011000 (88=64+16+8)

10. The following questions relate to interpreting binary numbers as abstracted types
    a. Answer the following questions relating to abstracting colors:
        i. How many bytes are needed to represent an RGB value?
        ii. What are the highest and lowest values for any color?
        iii. Convert the color purple to an RGB value. Provide the values in decimal and binary.
        iv. Convert 11010101 11110000 00110011 to a color (You can use this link to determine what color is appears to be: https://www.w3schools.com/colors/colors_rgb.asp)
    b. Answer the following questions relating to abstracting text (ASCII table: http://www.asciitable.com/):
        i. Convert 00111010 00101001 to ASCII characters.
        ii. Convert 'CMU cmu' into 8-bit binary (neglect the quotation marks).

Answers:
a. Colors:
    i. 3 bytes (24 bits, 3 8-bit values)
    ii. 255 is the highest value, 0 is the lowest
    iii. Purple is (R=255,G=0,B=255) or 11111111 00000000 11111111
    iv. (213, 240, 51)->yellowish
b. Text:
    i. :) (00111010=58, 00100101=41)
    ii. 01000011 01001101 01010101 00100000 01100011 01101101 01110101
    'C'=67 'M'=77 'U'=85 ' '=32 (space!)
    'c'=99 'm'=109 'u'=117

Functions

11. Consider the following function:

```
def helloThere(name):
    print("Hello there,", name, "!")
```

    a. What will you see in the shell if you call `print(helloThere("Stella"))` ?
    b. Does this function have side effect(s)? If so, state the side effect(s).
    c. Does this function have a returned value? If so, state the returned valued.
    d. What do we call the statement `helloThere("Stella")`? What do we call the string `"Stella"` in that statement?

Answer:

a. Shell output:
```
Hello there, Stella !
None
```
b. Yes, the print statement in the function is the side effect
c. Yes, the returned value is None. (all functions return None by default if another value is not specified)
d. `helloThere("Stella")` -> function call
`"Stella"` -> input or argument are both acceptable (parameter is incorrect, the variable *name* is the parameter)

12. Consider the following algorithm for a function:
Algorithm:
    1. Input a number `n`
    2. If `n` is greater than or equal to 0 do the following, otherwise skip to step 3:
        a. If `n` is divisible by 10 then print out the value of `n` followed by `" is cool number`", otherwise skip to step 2b
        b. If step 2a did not occur, check if `n` is divisible by 4, and if so print out the value of `n` followed by `" is a strange number`", otherwise skip to step 2c
        c. Skip to xstep 4
    3. Print the value of `n` followed by `" is a negative number!!`"
    4. Print the string `"ALL DONE"`

    Identify the input(s), returned value(s), and side effects of this algorithm. Then write the corresponding Python function called `divisionCheck`.

    Answer:
    The input to the function is `n`. The returned value will be `None`, since we don't return anything in the function. The side effects are the possible print statements.

```
def divisionCheck(n):
    if (n >= 0):
        if ((n % 10) == 0):
            print(n, "is a cool number")
        elif ((n%4) == 0):
            print(n, "is a strange number")
    else:
        print(n, "is a negative number!!")
    print("ALL DONE")
```

13. The function `fruitCalculator` has parameters `percentApples`, `totalFruit`, and `farm`. `fruitCalculator` calculates the number of apples harvested by the farm,

prints the number of apples and the number of other fruit, and returns a string announcing the harvest of apples. For example, `fruitCalculator(0.75, 400, "Carnegie Farms")` would return `"Carnegie Farms harvested 300 apples this year!"`, and the following would be printed in the console.

```
300 apples were harvested!
100 other fruits were also harvested!
```

Write the function `fruitCalculator` according to the description above.

Answer:

```
def fruitCalculator(percentApples, totalFruit, farm):
    numApples = percentApples * totalFruit
    numOther = totalFruit - numApples
    print(numApples, "apples were harvested!")
    print(numOther, "other fruits were also harvested!")
    return farm + " harvested " + str(numApples) + " apples
    this year!"
```

14. Suppose we had the following segment of code:

```
course = "15110"
grade = 95.0
def randomFunction(x, y):
    sum = x + y
    difference = x - y
    print(course)
    return (sum + difference) / 2
```

Give the scopes, either local or global, of the following variables: course, grade, sum, x, y, and difference.

Answer:

Global variables: course, grade
Local variables: x, y, sum, difference

How Python Works

15. Answer the following questions about how the interpreter converts Python understandable for the computer:
    1) What is the process in which the interpreter breaks down a Python file into key words?
    2) What is bytecode? In which process are sequenced tokens converted to bytecode?
    3) What is the process by which the interpreter finds the correct sequence of tokens?

Answer:

1) Tokenizing
2) Bytecode is machine code that all computers understand. Translating.
3) Parsing

16. You have the following segment of bytecode:

```
LOAD_CONST 0
LOAD_CONST 1
BINARY_ADD
STORE_NAME 0
LOAD_CONST 1
LOAD_CONST 2
BINARY_MULTIPLY
STORE_NAME 1
LOAD_NAME 0
LOAD_NAME 1
BINARY_SUBTRACT
STORE_NAME 2
```

Variables Table		
ID	Name	Value
0	x	
1	y	
2	z	

Constants Table	
ID	Value
0	5
1	6
2	7

Translate the bytecode into python using the provided tables

Answer:

```
x = 5 + 6
```

y = 6 * 7
z = x - y

Variables Table		
ID	Name	Value
0	x	11
1	y	42
2	z	-31

17. The following function takes in two numbers a and b, prints them as “a, b” and then returns their sum (for example, given a=3 and b=7 the function will print “3, 7” and it will return 10). Find the errors and classify them (into one of the three broad types of errors).

```
def printValuesReturnSum(a, b)
      print(a, b)
      return "a" +  b
```

Answer:
Line 1 - Syntax Error (missing colon)
Line 2 - Logical error (this will print “a b” without the comma)
Line 3 - Runtime error (can’t concatenate string and int)

18. The following lines of code contain multiple errors.

```
numerator = 0                #line 1
denominator = 6              #line 2
x = denominator/Numerator    #line 3
print("x', x)               #line 4
```

a. Find the errors and categorize them into the three general types of errors discussed in class.
b. If you run these lines of code in Pyzo, it will produce an error message. State which error in the code will cause the error message and explain your answer. In other words, which error is detected first and why? (You should be able to answer this without actually running the code)

Answer:
a. Line 3 - Runtime error: name error, capital “N” used in the variable numerator (remember that case matters when assigning variables) (even though this might

have been caused by a simple typo, remember that it is a runtime error not a syntax error!)
Line 3 - Runtime error: division by zero
Line 3 - Logical error: numerator and denominator are mixed up
Line 4 - Syntax error: EOL due to miss-matched parentheses

b. The syntax error on line 4 causes the error. Even though there are runtime errors occurring in an earlier line, syntax errors are always recognized first. We know this because syntax errors are caused by issues with tokenizing or parsing, but runtime errors occur later when bytecode is being executed.