Data 6 Summer 2023

Control and Iteration

Week 5 Tutoring Worksheet

August 2023

1. Comparisons and Boolean Expressions

Boolean comparisons allow Python to compare values and check if certain conditions are met. Comparisons and other boolean expressions form the foundation of **control**, which dictates what and in what order code is run.

Evaluate each of the following expressions just as Python would in a Jupyter notebook. If evaluating the expression causes an error, write "Error".

a. `3 > 7`

   `False`

b. `'kevin' == 'Kevin'`

   `False`

c. `'cute' in 'Acute'`

   `True`

d. `'a' in make_array('a', 'b', 'c', 'd')`

   `True`

e. `True and False`

   `False`

f. `True or 10 / 0`

   `True` (since the first value of the `or` is `True`, Python doesn't need to evaluate the second expression)

g. `False and 10 / 0`

   `False` (since the first value of the `and` is `False`, Python doesn't need to evaluate the second expression)

h. `True and 10 / 0`

   `ERROR` (Python must evaluate both sides of the `and`, so the division by 0 causes an error)

i. `False == 0.0`

   `True` (`float(False)` is 0.0)

j. `5 > True`

   `True` (`float(True)` is 1)

k. `‘cat’ > ‘dog’`

`False` (Earlier letters in the alphabet are defined to be ‘smaller’)

l. `5 > ‘4’`

`ERROR` (You can’t compare a string and int)

2. If This Then That

If-statements are crucial to **controlling the flow of execution** of code, allowing Python to **‘make decisions’ about which code to execute** based on the value of a boolean expression. An if-statement has exactly **one `if` clause**, **zero or more `elif` (else if) clauses**, and **zero or one `else` clauses**.
James is trying to book a flight to his next far-flung vacation destination. He is making his decision based off of three rules:

1. If the destination is more than (or exactly) 2000 miles away and the price is less than $500, then book that flight.
2. If the flight costs more than $500, book the flight if the destination has a ‘J’ (uppercase) in its name. (James has always wanted to visit Juneau, Alaska, and San Jose, Costa Rica)
3. If the destination is one of Tokyo, Auckland, Buenos Aires, or Copenhagen, then book the flight regardless of its cost. (These destinations are on James’ must-see list)

Complete the `book_flight` function below to help James decide if he should book a particular flight based on its destination (`dest`), cost (`cost`), and distance from Berkeley (`mi`). To ‘book’ the flight, print “Booking flight to [Destination]”. If the flight is not booked, then print “Bad deal”. *Hint:* It might be helpful to refer back to Q1(d).

```
def book_flight(dest, cost, mi):

    if dest in make_array(‘Tokyo’, ‘Auckland’, ‘Buenos Aires’, ‘Copenhagen’):
        print("Booking flight to " + dest)
    elif mi >= 2000 and cost < 500:
        print("Booking flight to " + dest)
    elif ‘J’ in dest:
        print("Booking flight to " + dest)
    else:
        print("Bad deal")
```

`book_flight(‘Las Vegas’, 300, 410)` should print “Bad deal”
`book_flight(‘Juneau’, 600, 1519)` should print “Booking flight to Juneau”
`book_flight(‘Copenhagen’, 2000, 5449)` should print “Booking flight to Copenhagen”

3. Loop-De-Loop

The ability to **repeat code** is essential to modern programming, and is a super time saver for programmers. Python has two types of loops, the **for loop** and the **while loop**. A **for loop** is used to **iterate** over a sequence of values (often an array), and the number of loops is known ahead of time. A **while loop** repeats **while a certain boolean expression is true**. This is extremely useful when you don't know ahead of time how many repetitions you will need.

a. Given the array `[10, 20, 30, 40, 50]`, use a for loop to print each number multiplied by 30. Also keep track of the sum of all of the (original) numbers. *Do not use any array arithmetic for this.*

```
my_arr = make_array(10, 20, 30, 40, 50)
total = 0

for v in my_arr:

    total += v

    print(v * 30)
```

b. Let's define 'Mike's Magic Sequence' as a sequence of numbers (starting with a given first number) generated according to the following rules:

   i. If the previous number is even, the next number is the previous number divided by 3 and rounded down
   ii. If the previous number is odd, the next number is the previous number multiplied by 3, minus one (i.e. n * 3 - 1)
   iii. If the previous number is less than or equal to 1, then the sequence is ended

   Complete the code below to generate 'Mike's Magic Sequence' for the number 19.

```
n  = 19

while (n > 1):

    print(n)

    if (n % 2 == 0):

        n = n // 3

    else:

        n = n * 3 - 1
```