lab1

September 25, 2023

1 ECS529U Algorithms and Data Structures

2 Lab sheet 1 (deadline: week 2)

This first lab gets you to work with Python, the programming language that we will be usingto write algorithms in this module. Python is similar to Java in syntax, although much simpler. In this lab you will write and run your first Python programs.

Look at `lab1-prep.ipynb` for examples on how to use Python commands to solve the questions.

Marks (max 5): Questions 1-2: 1 mark | Questions 3-6: 1 mark each

2.1 Question 1

We will be coding Python using the Jupyter IDE. You are asked to do the following:

0. watch the videos or/and read the slides of Week 0
1. if you skipped step 0, go back to step 0
2. start Jupyter Notebook (on Jupyter Hub, your machine, or ITL) or Google Collab
3. open `lab1-prep.ipynb` in Jupyter, read, understand and run each of its different functions

2.2 Question 2

Write a Python function:

```
def check_equal(x, y, z)
```

which checks the values of the inputs x, y, z to see if any are equal and prints whichever one of the following applies:

1. "All three inputs are equal"
2. "Only x and y are equal"
3. "Only y and z are equal"
4. "Only x and z are equal"
5. "They are all different"

The function **also returns** the corresponding number in each case (e.g. if all equal return 1, if only x and y are equal return 2, and so on). For example, running `check_equal(42,1,42)` in Jupyter we would get:

```
Only x and z are equal
4
```

Note : you will need to use the if-elif-else commands, and print.

[ ]:
```
def check_equal(x,y,z):
    if x == y and x == z:
        return("All three inputs are equal")
    elif x == y:
        return("Only x and y are equal")
    elif y == z:
        return("Only y and z are equal")
    elif x == z:
        return("Only x and z are equal")
    else:
        return("They are all different")
# below are some tests. NOTE: passing all the tests does NOT necassarily
# mean that your code is correct (there could be other tests that fail)
#
tests = [(42,1,42),(1,1,-5),(5,4,-1),(5,5,5),(0,0,1),(-9,9,9),(9,8,90)]
for x,y,z in tests:
    print(check_equal(x,y,z))
```

```
Only x and z are equal
Only x and y are equal
They are all different
All three inputs are equal
Only x and y are equal
Only y and z are equal
They are all different
```

2.3 Question 3

Write a Python function:

```
def check_equal2()
```

that is a modified version of `check_equal` which sets the three variables x, y, z to values typed in by the user when the code is run, and prints and returns as `check_equal`.

Note : this can be done using the built-in `input` function. You can assume that the inputs are integers.

[ ]:
```
def check_equal2():
    x = int(input("Enter the value of x: "))
    y = int(input("Enter the value of y: "))
    z = int(input("Enter the value of z: "))

    print(f"Values: {x} {y} {z}" )
    if x == y == z:
        return("All three inputs are equal")
    elif x == y:
        return("Only x and y are equal")
    elif y == z:
        return("Only y and z are equal")
    elif x == z:
        return("Only x and z are equal")
    else:
        return("They are all different")


print(check_equal2())
```

```
Values: 1 4 1
Only x and z are equal
```

2.4 Question 4

Write a Python function:

`def check_equal_sums(x, y, z)`

that is a modified version of `check_equal` which takes three arrays as its arguments and tests whether the sums of the contents of each of the arrays are equal.

For example, if the function takes [1,2,3], [10,20,30,40] and [65,32,3] as its three arguments, in that order, it should print “Only y and z have equal sums” and return 3.

Note : you can use a helper function that takes an array of integers and returns the result of adding them together. For example, with argument [65,32,3] the helper function would return 100.

You should **not use** the built-in `sum` function.

[ ]:
```
def calculate_sum(arr):
    total = 0
    for num in arr:
        total += num
    return total

def check_equal_sums(x, y, z):
    sum_x = calculate_sum(x)
    sum_y = calculate_sum(y)
    sum_z = calculate_sum(z)

    if sum_x == sum_y == sum_z:
        return("All three arrays have equal sums")
    elif sum_x == sum_y:
        return("Only x and y have equal sums")
    elif sum_y == sum_z:
        return("Only y and z have equal sums")
    elif sum_x == sum_z:
        return("Only x and z have equal sums")
    else:
        return("None of the arrays have equal sums")

# below are some tests. NOTE: passing all the tests does NOT necassarily
# mean that your code is correct (there could be other tests that fail)
#
tests =␣
 ↪[([1,2,3],[10,20,30,40],[65,32,3]),([10,20,30,40],[10,20,30,40],[65,32,3]),([1],[1,2],[1,2,
        ␣
 ↪([],[10,20,30,40],[]),([10,20,30,40],[1,-2],[65,32,3]),([-1,-1,-1,2],[1,-2,0],[65,32,3])]
for x,y,z in tests:
    print(check_equal_sums(x,y,z))
```

```
Only y and z have equal sums
All three arrays have equal sums
None of the arrays have equal sums
Only x and z have equal sums
Only x and z have equal sums
Only x and y have equal sums
```

2.5 Question 5

Write a Python function:

`def compare_lengths(x, y, z)`

which takes as arguments three arrays and checks their lengths and returns them as a triple in order of length.

For example, if the function takes [1,2,3], [10,20,30,40] and [65,32,7] as input, it should return either `([1,2,3], [65,32,7], [10,20,30,40])` or `([65,32,7], [1,2,3], [10,20,30,40])`.

Note: you can use the built-in `len` function which returns the length of an array, and also use Python tuples.

```
[ ]: def compare_lengths(x, y, z):
         return tuple(sorted([x,y,z], key=lambda x:len(x)))

     # below are some tests. NOTE: passing all the tests does NOT necassarily
     # mean that your code is correct (there could be other tests that fail)
     #
     tests =␣
      ↪[([1,2,3],[10,20,30,40],[65,32,3]),([0,10,20,30,40],[10,20,30,40],[65,32,3]),([1],[1,2],[1,
             ␣
      ↪([],[10,20,30,40],[]),([10,20,30,40],[1,-2],[65,32,3]),([-1,-1,-1,2],[],[65,32,3])]
     for x,y,z in tests:
         print(compare_lengths(x,y,z))
```

```
([1, 2, 3], [65, 32, 3], [10, 20, 30, 40])
([65, 32, 3], [10, 20, 30, 40], [0, 10, 20, 30, 40])
([1], [1, 2], [1, 2, 3])
```

```
([], [], [10, 20, 30, 40])
([1, -2], [65, 32, 3], [10, 20, 30, 40])
([], [65, 32, 3], [-1, -1, -1, 2])
```

2.6 Question 6

In Python a variable can be set to refer to a function, which also means a function can be passed as an argument to another function. Write a Python function:

```
def check_equal_fun(f, x, y, z)
```

that is a modified version of `check_equal` which takes four arguments, one being a function and the rest being three values. The function should print whichever one of the following applies:

1. "Applying f to all three values gives the same result"
2. "Only applying f to x and y gives the same result"
3. "Only applying f to y and z gives the same result"
4. "Only applying f to x and z gives the same result"
5. "Applying f to x,y,z gives all different results"

and return the corresponding number. Observe that `check_equal_fun` should work like `check_equal_sums` if the helper function in Question 4 is given as first argument.

Another example is the following. Let `fst` be the function that returns the first element of an array (if there is one):

```
def fst(A):
    return A[0]
```

Then, e.g. `check_equal_fun(fst,[1,2,3],[2,1,3],[3,2,1])` should print "Applying f to x,y,z gives all different results" and return 5.

[ ]:
```
def check_equal_fun(f, x, y, z):
    result_x = f(x)
    result_y = f(y)
    result_z = f(z)

    if result_x == result_y == result_z:
        print("Applying f to all three values gives the same result")
        return 1
    elif result_x == result_y:
        print("Only applying f to x and y gives the same result")
        return 2
    elif result_y == result_z:
        print("Only applying f to y and z gives the same result")
        return 3
    elif result_x == result_z:
        print("Only applying f to x and z gives the same result")
        return 4
    else:
        print("Applying f to x, y, z gives all different results")
```

```python
        return 5

# Example usage with the `fst` function
def fst(A):
    try:
        return A[0]
    except:
        return None

result = check_equal_fun(fst,[1,2,3],[2,1,3],[3,2,1])
print(result)

result = check_equal_fun(fst,[],[2,1,3],[2,5242,1])
print(result)

result = check_equal_fun(fst,[2,31241,1242,241],[2,1,3],[2,5242,1])
print(result)
```

```
Applying f to x, y, z gives all different results
5
Only applying f to y and z gives the same result
3
Applying f to all three values gives the same result
1
```