Week 2: Agenda

- Quiz #1: Academic Honesty Policy – Grades will be released soon
  - Common *misconceptions*: outside tutor, help from other 112 students
- This week: Quiz #2 Data and Expressions, Conditionals
- Data and Expressions, Conditionals: Review
- Loops

Data and Expressions: Types of problems

- Math Problems
  - Examples: `isRightTriangle`
- Extracting Digits
  - Examples: `isSmallFair`, `isSmallPal (Quiz 1, Pitt S22)`

`isRightTriangle(x1,y1,x2,y2,x3,y3)`

- Check if the triangle defined by points (x1,y1) (x2,y2) (x3,y3) is a right triangle. Return `True` if the triangle is right-angled, `False` otherwise

How to approach a programing problem

1. Read the Problem
   - Understand the requirements and constraints.
2. Brainstorm Test Cases
   - Generate examples to clarify expected results and edge cases.
3. Devise an Algorithm
   - Outline a clear step-by-step plan.
4. Validate the Algorithm
   - Check logic against your test cases; refine if needed.
5. Implement the Code
   - Translate the algorithm into a working program (and run tests)

`isRightTriangle(x1,y1,x2,y2,x3,y3)`

- Check if the triangle defined by points (x1,y1) (x2,y2) (x3,y3) is a right triangle. Return `True` if the triangle is right-angled, `False` otherwise

`isSmallFair(n)` (Pittsburgh F21)

- We will say that a value **n** is *"fair"* if it is an integer and it has the same number of even digits as odd digits (ignoring leading 0's). A *"small fair"* number is a fair number with exactly 4 digits.
- For example, 1083, 1081, and -1092 are each small fair numbers because each have two odds and two evens.
- With this in mind, and without using strings or loops, write the function `isSmallFair(n)` that takes a value **n**, that may or may not be an integer, and returns **True** if **n** is a small fair number, and **False** otherwise. **Do not crash if n is not an integer!**

Data and Expressions: Types of problems

- Math Problems
  - Examples: `isRightTriangle`
- Extracting Digits
  - Examples: `isSmallFair`, `isSmallPal (Quiz 1, Pitt S22)`
- Other topics: E.g., use of modulo

(b) (3 points) CT2

```
def f(x,y):
    if x % y == 3:
        y = x // 2
        print("A1", x, y)
    elif x % 2 == 0:
        y = x * 2
        print("A2", x, y)
    if y == 8:
        x = 4 * y + 2
        print("A3", x, y)
    return (x+y)//10 % 10

def ct2():
    print("L1", f(4, 7))

ct2()
```

`clockHour(currentHour, difference)`

- It's 10:00 (am/pm doesn't matter). Where will the hour hand be in 7 hours? In practice the hour will be 17, but we are restricted to show 5 since the hour hand only goes up to 12!
- Create a function called `clockHour(currentHour, difference)` that calculates the hour hand's position on a 12-hour clock after a certain number of hours have passed. The function takes two parameters: currentHour, an integer in the range 1–12 indicating the current hour hand position, and `difference`, an integer representing how many hours have passed since the current hour. The function should return the new hour hand position, ensuring that the result stays within 1–12. For example, if currentHour = 11 and difference = 2, the hour hand moves from 11 to 12, and then to 1, so the function returns 1.
- Example:
  - clockHour(1, 3) == 4
  - clockHour(8, 11) == 7
  - clockHour(4, 13) == 5
  - clockHour(4, 24) == 4
  - clockHour(12, 1) == 1
  - clockHour(11, 2) == 1
  - clockHour(6, 25) == 7
  - clockHour(12, 12) == 12
  - clockHour(3, 24) == 3
  - clockHour(1, 24) == 1

Data and Expressions: Types of problems

- Math Problems
  - Examples: `isRightTriangle`
- Extracting Digits
  - Examples: `isSmallFair`, `isSmallPal (Quiz 1, Pitt S22)`
- Other topics: E.g., use of modulo
- Mixed

`timeInterval(t1,t2)` Quiz, Qatar F21

Write the function `timeInterval(t1, t2)` which, given two non-negative integers `t1`, `t2`, that encode two 24-hour times in the format `hhmm`, returns the time interval, in minutes, between those two times. If `t2 < t1`, you should assume that `t2` refers to a next day time. You can assume that `0 < hh < 24` is the hour, and `0 < mm < 60` are the minutes. If `hh > 0`, then mm is always a two-digit number. If `hh == 0`, then mm can be either a one or a two-digit number, depending on its value.

- 1503 is 15 hours, 3 minutes, or 3:03pm.
- 849 is 8 hours, 49 minutes, or 8:49am.
- 0 is 0 hours, 0 minutes, or 12:00am midnight.
- 59 is 0 hours, 59 minutes, or 12:59am.
- 101 is 1 hour, 1 minute, or 1:01am.

For example...

- `timeInterval(1400, 1545)` returns the time interval between 14 o'clock and 15:45 (same day) which is 105 minutes.
- `timeInterval(2359, 31)` returns the time interval between 23:59 and 00:31 (next day), which is 32 minutes.
- `timeInterval(31, 2359)` returns the time interval between 00:31 and 23:59 (same day), which is 1408 minutes.
- `timeInterval(1200, 0)` returns the time interval between noon and midnight (next day), which is 720 minutes.
- Hint: There are 1440 minutes in a day.

Loops

Code that repeats itself

Guess the number

- Goal: Pick a secret number and have the user guess it until correct (or until they enter the “master key”).
- Steps:
  - Choose a secret number (random).
  - Prompt the user repeatedly for guesses.
  - End the program if they guess correctly or if they enter 42 (the “master key”).

Optional: Count and display the number of incorrect attempts.

`sumDigits(n)` with `n < 99999`

Write a function sumDigits(n) that computes the **sum of the digits** of an integer n, assuming n has **at most 5 digits**.

Example: sumDigits(123) should return 1+2+3=6

You cannot use strings

`sumDigits(n)` with arbitrary length

Write a function sumDigits(n) that computes the **sum of the digits** of an integer n

Example:

- sumDigits(123) should return 1+2+3=6
- sumDigits(111111111111) should return 12

You cannot use strings

What you need to know

- Difference between for-loop and while-loop
  - Use cases
- range: different forms
  - `range(end)`
  - `range(start,end)`
  - `range(start, end, step)`
  - `Examples:`
    - `range(1, 9, 2)`
    - `range(1, 9, 3)`
    - `range(5, 1, -2)`
    - `range(1, 9, -1)`

What you need to know

- `break` and `continue`

```
def f(x):
    print(x)
    return 42

def ct(x):
    counter = 0
    target = x
    for i in range(5, -1, -2):
        if counter == target:
            print("meet", counter)
            target += 1
        else:
            print("miss", target)
        counter += 2
    return f(counter)

ct(2)  # starts here
```

7ish numbers

- Non-negative number, and the sum of its digits a is multiple of 7
  - Examples:
    - 61: 6 + 1 = 7
    - 86: 8 + 6 = 14
    - 489: 4 + 8 + 9 = 21
- First 16 7ish numbers

0
7
16
25
34
43
52
59
61
68
70
77
86
95
106
115

`nth...` problems

- Example: `nthCircularPrime, nth7ish, ...`
- Recap: `nth7ish`
  - Part 1: Write the function `is7ish(n)`, which takes a non-negative integer **`n`** and returns **`True`** if **`n`** is a 7ish number and **`False`** otherwise.
  - Part 2: Write the function `nth7ish(n)`which takes a non-negative integer **`n`** and returns the nth *7ish* number.
    - `nth7ish(0)` should return 0, the first 7ish number.
    - `nth7ish(1)` returns 7.

0
7
16
25
34
43
52
59
61
68
70
77
86
95
106
115

What’s the nth 7ish?

Position	Number
0	0
1	7
2	16
3	25
4	34
5	43
6	52
7	59
8	61
9	68
10	70
11	77
12	86
13	95
14	106
15	115