# Spring 2025 – Exam 1 Solutions

**CIS 1100**

*2025-10-13*

---

## Problem 1: Types

| Code | Type |
|---|---|
| `28 % 7` | int |
| `not not not True` | bool |
| `"August " + 29` | error |
| `"August " + "29"` | str |
| `chr(68)` | str |
| `[3, 4, 5, 2, 1, 2][3]` | int |
| `{3, 4, 5}` | set |
| `range(3, 6)` | range |
| `{3 : 4, 5 : 6}` | dict |

## Problem 2: Values

1. 8
2. 3
3. 10 + 20
4. True
5. False
6. 4
7. Congrats, buddy!!!

## Problem 3: Debugging

1. Line 7, wrong order for calling `add`
2. Line 8, wrong boolean expression
3. Line 11, wrong index for first letter
4. Line 12, incorrect condition
5. Line 16, no value returned.

*Full corrected solution:*

```
1  def fancy_words_only(words: list[str]) -> list[str]:
2    output = []
3    for word in words:
4      # no repeat letters
5      letters = set()
6      for letter in word:
7        letters.add(letter)
8      no_repeats = len(letters) == len(word)
9
10     # capitalized
11     first_letter = word[0]
12     is_capitalized = "A" <= first_letter <= "Z"
13
14     if no_repeats and is_capitalized:
15       output.append(word)
16   return output
```

## Problem 4: FITB

1. `filename`
2. `"ACTOR"`
3. `[1]`
4. `[]`
5. `actor_lines[current_actor]`
6. `actor_lines`

```python
def parse_actor_lines(filename: str) -> dict[str, list[str]]:
  # open the file and read all lines
  file_handler = open(filename, "r")
  lines = file_handler.readlines()
  file_handler.close()

  actor_lines = {}
  current_actor = ""
  lines = lines[2:] # skip title and setting
  for line in lines:
    line = line.strip()
    if "ACTOR" in line: # detect actor name
      # read <name> from "ACTOR <name>"
      current_actor = line.split()[1]

      if current_actor not in actor_lines:
        actor_lines[current_actor] = []

    else:
      actor_lines[current_actor].append(line)

  return actor_lines
```

# Problem 5: Price Tracker

## 5.1 `lowest_in_k_days`

```python
def lowest_in_k_days(threshold: int, history: list[int]) -> int:
    for idx, price in enumerate(history):
        if price <= threshold:
            return idx + 1
    return len(history)
```

## 5.2 `is_rare_price`

```python
def is_rare_price(price: int, history: list[int]) -> bool:
    days_below = 0
    for old_price in history:
        if price < old_price:
            days_below += 1
    return (days_below / len(history)) < 0.5
```

## 5.3 `should_buy_now`

```python
def should_buy_now(price: int, history: list[int]) -> bool:
    been_a_while = lowest_in_k_days(price, history) >= 3
    rare_price = is_rare(price, history)
    return been_a_while and rare_price
```