Challenges 2

December 1, 2025

1 Tasks 1 - Function examples

1. Define a function `is_odd`, which returns True or False depending on whether a number is odd (i.e., divisible by 2 with a remainder)

```
def is_odd(a: int) -> bool:
    pass

assert is_odd(3) is True
```

[3]:
```
def is_odd(a: int) -> bool:
    return a % 2 == 1
    if a % 2 == 1:
        return True
    return False


assert is_odd(3) is True
```

2. Use the function is_odd to print the output of computing it in an fstring (e.g., "Is the number 2 odd? The answer is False")

[4]:
```
a = 2
print(f"Is the number {a} odd? The answer is {is_odd(a)}")
```

```
Is the number 2 odd? The answer is False
```

3. Implement `is_even`, but use your previous `is_odd` function to implement the function in a single line

[5]:
```
def is_even(a: int) -> bool:
    return not is_odd(a)

assert is_even(3) is False
```

2 Tasks 2 - Converting Loops

1. Translate the while-loop into a for-loop (Hint: https://docs.python.org/3/library/functions.html#funcrange)

```
def every_other_number_including_n(n: int) -> None:
    i = 0
    while i <= n:
        print(i)
        i += 2

# Testcase, should print 0, 2, 4, 6
every_other_number_including_n(6)
```

[8]:
```
def every_other_number_including_n(n: int) -> None:
    for i in range(0, n + 1, 2):
        print(i)
every_other_number_including_n(6)
```

```
0
2
4
6
```

2. Translate the for-loop into a while-loop (If unsure, call `range?` to see documentation in Jupyter)

```
def func2():
    for i in range(100, 0, -2):
        print(f"{i} squared equals {i ** 2}")
```

[11]:
```
def func2():
    i = 100
    while i > 0:
        print(f"{i} squared equals {i ** 2}")
        i -= 2
func2()
```

```
100 squared equals 10000
98 squared equals 9604
96 squared equals 9216
94 squared equals 8836
92 squared equals 8464
90 squared equals 8100
88 squared equals 7744
86 squared equals 7396
84 squared equals 7056
82 squared equals 6724
80 squared equals 6400
78 squared equals 6084
76 squared equals 5776
74 squared equals 5476
72 squared equals 5184
70 squared equals 4900
```

```
68 squared equals 4624
66 squared equals 4356
64 squared equals 4096
62 squared equals 3844
60 squared equals 3600
58 squared equals 3364
56 squared equals 3136
54 squared equals 2916
52 squared equals 2704
50 squared equals 2500
48 squared equals 2304
46 squared equals 2116
44 squared equals 1936
42 squared equals 1764
40 squared equals 1600
38 squared equals 1444
36 squared equals 1296
34 squared equals 1156
32 squared equals 1024
30 squared equals 900
28 squared equals 784
26 squared equals 676
24 squared equals 576
22 squared equals 484
20 squared equals 400
18 squared equals 324
16 squared equals 256
14 squared equals 196
12 squared equals 144
10 squared equals 100
8 squared equals 64
6 squared equals 36
4 squared equals 16
2 squared equals 4
```

3 Tasks 3 - Loops and Functions

You must not use any modules that require importing for these tasks! 1. Write a function called ascii_to_string(some_list), which takes a list of integers, converts each to the corresponding ASCII character (https://docs.python.org/3/library/functions.html#chr) and returns the result as a string Hint:

```
"".join(['a', 'b', 'c']) == 'abc'
```

[12]:
```
def ascii_to_string(my_list: list) -> str:
    my_chr_list = []
    for i in my_list:
        my_chr_list.append(chr(i))
```

```
    return "".join(my_chr_list)

assert ascii_to_string([72, 101, 108, 108, 111]) == 'Hello'
```

2. . Write a fibonacci function that returns the n-th fibonacci number. You must not use recursion for this task. (Reminder: Fibonacci Sequence is 1, 1, 2, 3, 5, 8, 13, 21, …)

- Hint 1: fib(n+2) = fib(n+1) + fib(n)
- Hint 2: consider using a list, which should start with [1, 1] and can be appended to

```
assert fib(10) == 55
```

[19]:
```
def fib(n: int) -> int:
    fib_list = [1, 1]
    for _ in range(n - 2):
        fib_list.append(fib_list[-2] + fib_list[-1])
    print(fib_list)
    return fib_list[n-1]

assert fib(10) == 55
```

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

4 Task 4 - Set Intersection

Write a function called `sets_intersection(a: list, b: list, c: list, d: list) -> set`, which takes the union of `a` and `b` and computes the intersection to the union of `c` and `d`. The function must be a single line only!

[20]:
```
def sets_intersection(a: list, b: list, c: list, d: list) -> set:
    return (set(a) | set(b)) & (set(c) | set(d))
    pass

assert sets_intersection([1, 2], [3, 4], [4, 1], [6, 7]) == {1, 4}
```

5 Task 5 - Find in Dict

Write a function `find_by_value` which iterates over a dict's items and returns the *first* key for which the value is a substring match - Hint 1: recall the `in` operator can be used to check if a string is contained in another - Hint 2: some_dict.items() returns `[(key1, value1), (key2, value2), ... ]` (technically an iterator, but you can just assume a list-like structure)

[23]:
```
def find_by_value(haystack: dict, needle: str) -> str | None:
    for key, value in haystack.items():
        if needle in value:
            return key
    return None
```

```
test_dict = {}
test_dict['ben'] = 'stock@cispa.de'
test_dict['simon'] = 'simon@simon.de'
assert(find_by_value(test_dict, 'i') == 'ben')
```

[ ]: