

## Lecture 12: Algorithms and efficiency

Morten Rieger Hannemose, Vedrana Andersen Dahl

Fall 2023

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$

## Course overview

- ▶ Data types `int` and `float` and computation
- ▶ Functions
- ▶ Flow control with conditions and loops
- ▶ Data types `str`, `list`, methods, indexing and traversing
- ▶ Data types `dict`, `tuple`
- ▶ Reading and writing files
- ▶ Object-oriented programming
- ▶ Numpy, matplotlib

Last two weeks of the course (this week and next week)

- ▶ Algorithms and efficiency  
(writing efficient and readable code)
- ▶ Summary and discussion of the exam  
(wrapping up, revisiting midterm exam, and extras)

## Code quality

- ▶ Software quality: reliability, efficiency, security, maintainability.
- ▶ In this course, the focus is on:
  - ▶ Correctness (the only thing we test)
  - ▶ Efficiency
  - ▶ Readability
  - ▶ Style

## Code size

- ▶ In this course: 10-20 lines of code.
- ▶ My largest project: a few thousand lines of code.
- ▶ Video game: A few million lines of code

## Code efficiency and style

### Examples on

- ▶ Counting things
- ▶ Searching for things
- ▶ (a bit on) Sorting and merging things

### Focus on

- ▶ Avoiding unnecessary computation
- ▶ Carefully choosing variables
- ▶ (a bit on) Commenting
- ▶ Common pitfalls

## Counting and searching

What is the intended result?

- ▶ Is there a ...
- ▶ Where is the ...
- ▶ How many ...
- ▶ What are ...

What is the occurrence I'm searching for?

- ▶ ... number 3? ... letter 'F'?
- ▶ ... number larger than 3? ... capital letter?
- ▶ ... number larger than both its predecessor and successor?
- ▶ ... an item best according to some measure?
- ▶ ... number with an odd index which is larger than 3?

Remember from the mid-term test exam

- ▶ First alarm: When did the alarm occur? (Index of the first occurrence of a number either ...)
- ▶ Typical successor: What is typically following a letter? (What is ...)
- ▶ Dice fairness. What appears most frequently and how many times?

## Code used for coding examples

### Simplifying code

```
1 text = 'Something'
2 too_long = len(text)>10
3 if too_long: # instead too_long==True
4     print('The text is too long')
5
6 def should_pay_half_price(age):
7     # instead if-sentence
8     return (age < 18) or (age > 65)
9
10 age = 75
11 full_price = 100
12 # either full price or half price
13 # instead of if-sentence
14 price = 0.5 * full_price + 0.5 * full_price
15     * (18 <= age <= 65)
16
```

### Avoid unnecessary computation

```
1 text = 'This is a very long text which is
2     slow to compute the length of.'
3 len_text = len(text)
4 for p in [10, 50, 90]:
5     print(f'{p}% is {p / 100 * len_text}')
6
```

## Code used for coding examples

### Searching and counting, lists

```
1 items = [5, 6, 8, 2, 4, 5, 7, 8, 4, 6, 4, 3, 5, 6, 7, 3, 2,
2         4, 5, 6, 7, 8, 9]
3
4 # Use built-in list methods
5 print(3 in items)
6 print(items.index(3))
7 print(items.count(3))
8
9 # Is there an occurrence?
10 found_it = False
11 for item in items:
12     if item > 6:
13         found_it = True
14         break
15
16 # Where is the first occurrence?
17 index = -1 # a dedicated value
18 for i in range(len(items)):
19     item = items[i]
20     if item > 6:
21         index = i
22         break
23
24 # How many occurrences?
25 counter = 0
26 for item in items:
27     if item > 6:
28         counter += 1
```

### Searching and counting, lists

```
1 # How many occurrences?
2 counter = 0
3 for item in items:
4     if item > 6:
5         counter += 1
6 print(counter)
7
8 # Looking for the index of somehow best item, with smallest
9   abs(item - 5)
10 # max and min are special cases of this
11 best_distance = abs(items[0] - 5)
12 best_distance = 1000
13 for item in items:
14     this_distance = abs(item - 5)
15     if this_distance < best_distance:
16         best_distance = this_distance
17
18 # Larger than both neighbors
19 for i in range(1, len(items) - 1):
20     if items[i] > items[i - 1] and items[i] > items[i + 1]:
21         print(items[i])
22
23 # Odd index and larger than 6
24 for i in range(len(items)):
25     if i % 2 == 1 and items[i] > 6:
26         print(items[i])
```

## Code used for coding examples

### Searching and counting, numpy and lists

```
1 import numpy as np
2
3 numpy_items = np.array(items)
4 print(3 in numpy_items)
5 # print(numpy_items.index(3)) # This will not work
6 print(numpy_items == 3)
7 print((numpy_items == 3).any())
8
9 # print(numpy_items.count(3)) # This will not work
10 print((numpy_items == 3).sum())
11
12
13 print(np.where(numpy_items == 3))
14 print(numpy_items[:,2])
15
16 peak = (numpy_items[1:-1] > numpy_items[2:]) & (
17         numpy_items[1:-1] > numpy_items[:-2])
18 print(peak)
19
20 print(numpy_items.max())
21 print(numpy_items.argmax())
22
```

### Sorting and merging

```
1 items = [5, 6, 8, 2, 4, 5, 7, 8, 4, 6, 4, 3, 5, 6,
2           7, 3, 2, 4, 5, 6, 7, 8, 9]
3 print(sorted(items))
4 print(np.sort(numpy_items))
5 print(np.unique(numpy_items))
6
7 items = [4, 6, 3, 8, 5]
8 other_items = [5, 8, 11, 13, 9]
9 for i in other_items:
10     if i not in items:
11         items.append(i)
12 print(items)
```