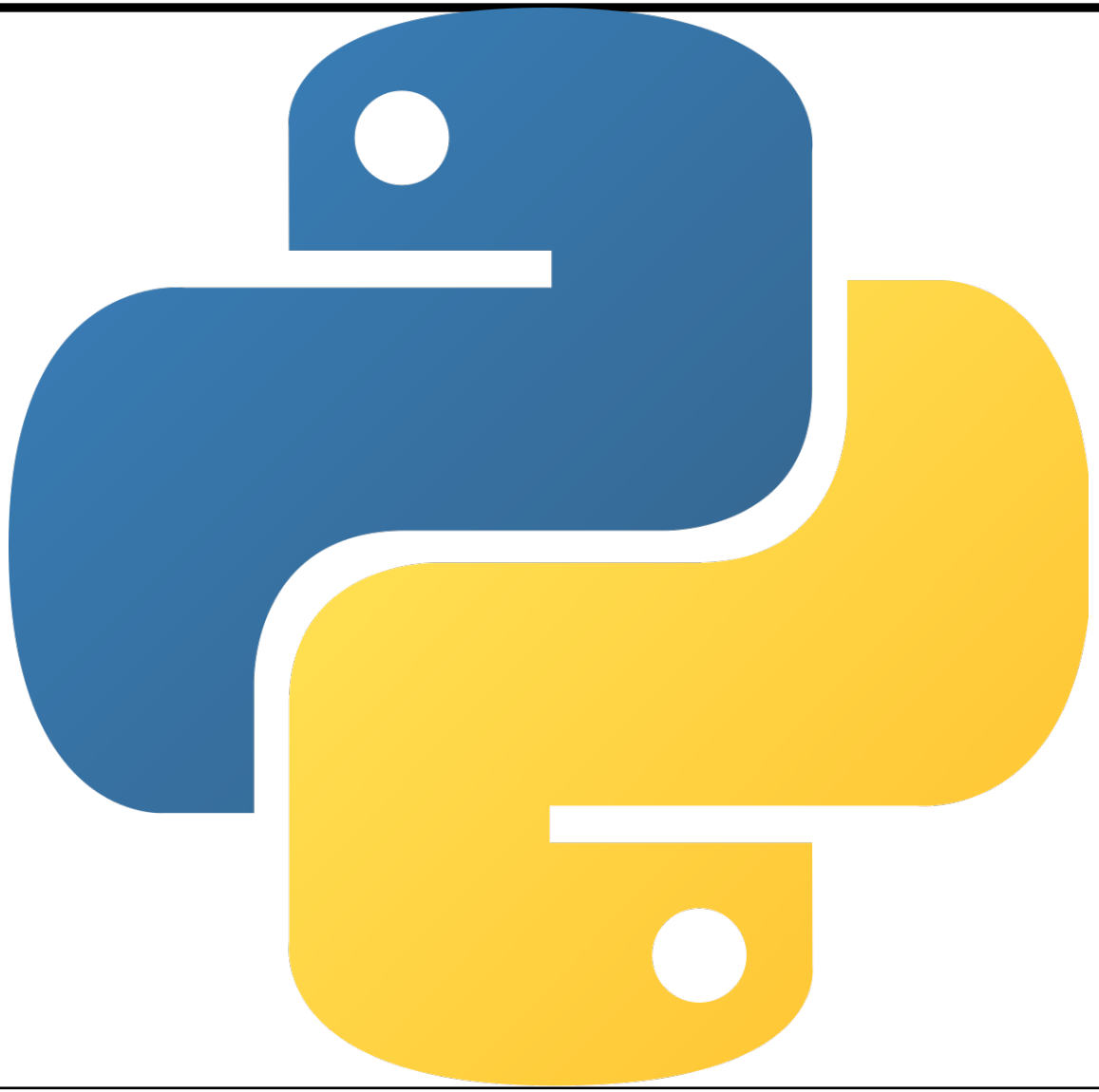
PYTHON PROGRAMMERING



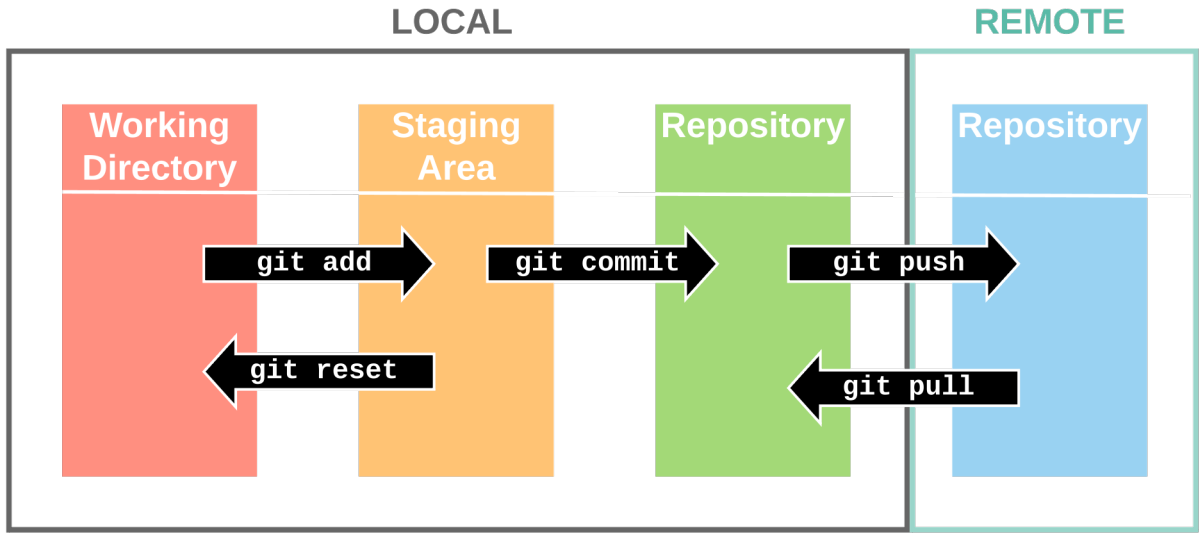
Föreläsning 2

DAGENS AGENDA

- GitHub Desktop
- Data strukturer
- Komplexitet i kod
- Slicing sequences
- Sets
- Functions
- Main
- God programmeringssed: moduler, paket, importer, DRY coding, kommentera kod

FÖRRA FÖRELÄSNING

- Git och GitHub



DATA STRUCTURES

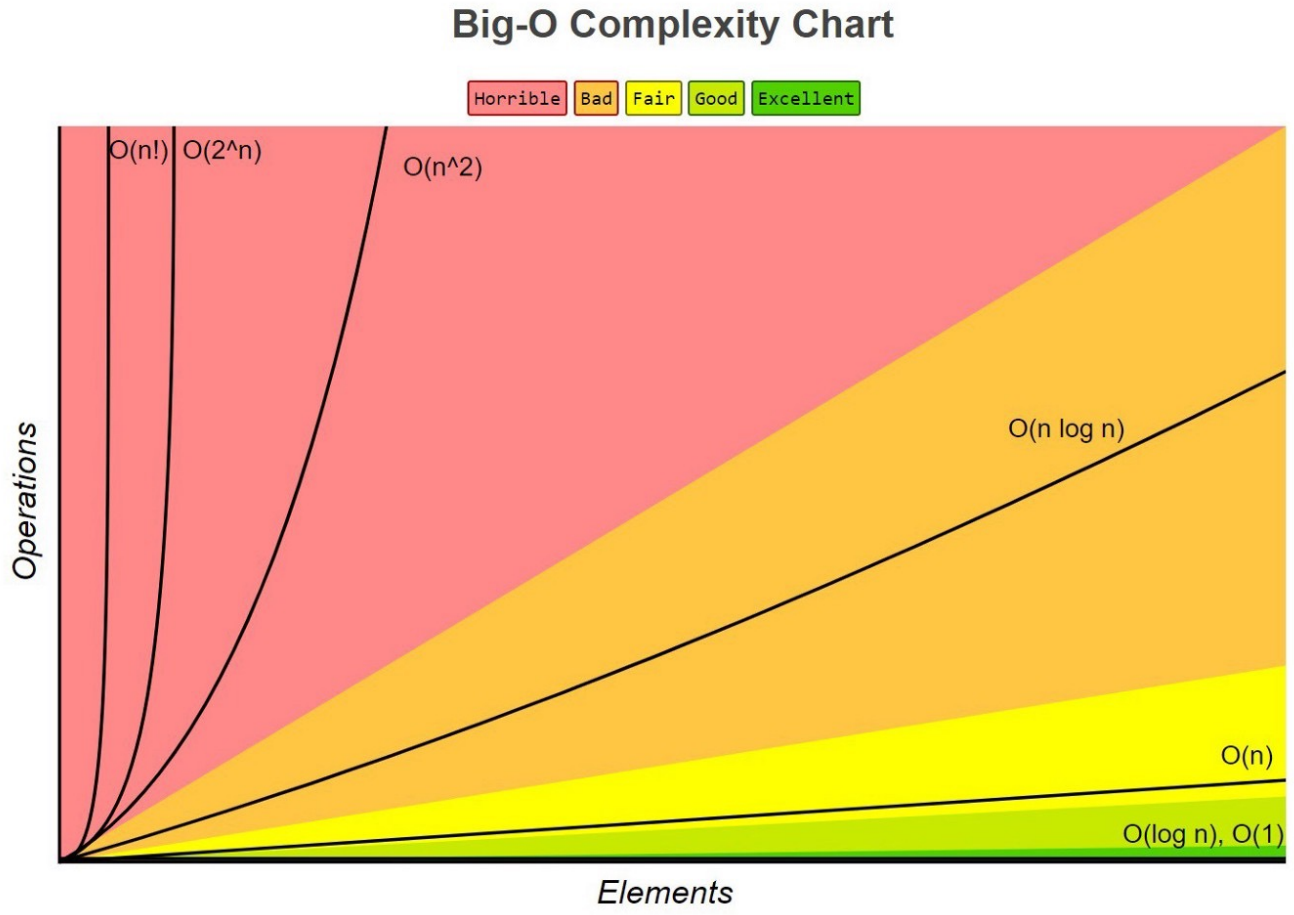
- In computers science data structures are used to organize, manage and store data in ways that enables efficient access and modification
- A data structures is a collection of data values(elements) that uses different algorithms to solve similar problems. Because of this some data structures are more suitable to certain task than others. Using the right data structures for the correct task makes your code more performant and easier to understand
- In python some basic data structures are called sequences. Each element of a sequence is assigned an index. Python has built-in functions that works for all sequences for example getting length, indexing, slicing, adding etc. List, Tuple and Set are sequences

COMPLEXITY ANALYSIS

- When talking about how efficient data structures/algorithms are, we talk about Time Complexity and Space complexity. Both are described using Big O notation
- Time Complexity signifies the amount of time an algorithm takes to run to complete in terms of the amount of input. "Time“ is the number of elementary operations performed by the algorithm
- Space Complexity is a measure of the amount of memory an algorithm needs. Consists of two parts a fixed part which is the space required to store the data that is not dependent on the problem. A variable part that is dependent on the size of the problem

BIG O CHEAT SHEET

- https://www.bigocheatsheet.com/
- https://www.geeksforgeeks.org/complexity-cheat-sheet-for-python-operations/



SLICING SEQUENCES

```
string = "0123"
skip_first = string[1:]
print(skip_first) # prints 1, 2, 3
sequence = [0, 1, 2, 3]
skip_first = sequence[-3:]
print(skip_first) # prints [1, 2, 3]
```

- Slicing is a very powerful part of python that can be applied to sequences and strings
- Slicing can be used to retrieve a slice(subset) of a sequence by using the [start:end] notation where start and end are indexes.
- A slice contains the value at start and all the values between start and end for example [1:3] would contain the value in index 1 and 2
- Start can be omitted if it is the start will be the first index of the sequence (0)
- End can be omitted if it is, end will be the last index of the sequence
- The indexes used when slicing can be negative in which case, they start from the end instead. For example, the index –1 is the last item, -2 is the second last item etc.

SETS

- A set is a collection of distinct(unique), unordered and unindexed elements.
- Common use is membership testing, removing duplicates from a sequence
- Set does not support indexing, slicing, or other sequence-like behavior
- Set can be used in for loops and supports the *in* keyword
- To add an element to a set use *add()*
- To remove an element use *remove()*
- Sets are denoted with curly brackets
- https://www.w3schools.com/python/python_sets.asp

```
fruits = {"apple", "banana", "cherry"}
fruits.add("orange")
print("banana" in fruits) #prints True
fruits.remove("apple")
```

FUNCTIONS

- A function is a block of code which only runs when it is called.
- You have already been using functions for example print() and len() which are built-in functions
- In python functions are defined using the *def* keyword have parenthesis and ends with a colon the functions code block must be indented
- You can pass data, known as parameters (or arguments), into a function. There is no limit to how many arguments can be passed but it is recommended to use as few as possible
- Functions name follow the same naming conventions as variables. Their name should describe what they do

```
def say_hello(name):
    print("Hello {}!".format(name))

say_hello("World") # prints Hello World
say_hello("Foo") # prints Hello Foo
```

MORE ON FUNCTIONS

- Function can return data as a result often called an out parameter. To return data use the *return* keyword
- When using multiple parameters the order of the parameters are important. You can also use keyword arguments to make it clearer when using multiple arguments the order of these does not matter
- You can also use a default parameter value which will be used if the parameters is not passed. To set a default parameter use a = they must be the last parameters defined
- Function are one of the most powerful tools in a programmer's arsenal they provide better modularity and a high degree of code reusing. Reducing bugs, making code easier to maintain and reason about
- Functions should do only one thing

```
def combine_name(fname, sname, mname = ""):
    return "{} {} {}".format(fname, mname, sname)

full_name = combine_name("Tim", "Nielsen", "Daldorph")
print(full_name) # prints Tim Daldorph Nielsen
full_name = combine_name(sname="Nielsen", fname="Tim")
print(full_name) # prints Tim Nielsen
```

SCOPE AND BLOCKS

- The concept of scope rules names(variables, classes, constants, functions) are looked up in you code. It determines the visibility of a name within the code.
- In python you have two scopes Global and Local scope. Names defined in the global scope is visible in all of you code. Names defined in a local scope is only visible within that local scope
- One of the easiest way to write better code is making sure that names exist in only the scope in which it is used. Therefore we should avoid defining names in the global scope
- Anything defined directly in the python file is in the global scope. So most of what we have defined thus far has been in the global scope
- Name defined in functions and classes are in a local scope

```
# Global variable
foo = "foo"
if foo == "foo":
    bar = "bar" #Global variable

# Global function
def global_function(world): # parameters are local
    hello = "hello" # local variable

global_function("world")

print(foo) # prints foo
print(bar) # prints bar

print(hello) # NameError: name 'hello' not defined
print(world) # NameError: name 'world' not defined
```

PYTHON MAIN

- Many programming languages have a special main functions that acts as the entry point when running the program. On the other hand the python interpreter executes scripts starting at the top of the file.
- But having a defined starting point for the program makes it easier to understand how a program works
- A common code pattern has been defined to faciltate this. Where you check the value of the special __name__ variable which changes based on how the script was executed.
- Using the *if __name__ == "__main__"* idiom means that the main function is only called when the script is excuted directly. So if your script was imported the main function would not run.
- You should be using main functions for all the python scripts you write

```python
def main():
    print("Hello main!") # will be printed

if __name__ == "__main__":
    main() # call the main function
```