POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Introduction to Python and its applications in bioinformatics Part one

Basics of Bioinformatics

Lecture 3

Adam Kuzdraliński

2026/2027

Sources: Biopython: applications

What should the project produce?

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

The program downloads or reads a defined dataset.

It checks the contents and calculates selected properties.

It saves the results in a table and presents them clearly.

The conclusion answers the question and considers the limits of the data.

Sources: Wilson et al.: good computing practices | Biopython: applications

Different libraries have different roles

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Biopython reads sequences and performs biological sequence operations.

Requests downloads data over HTTP.

pandas handles tables, and Matplotlib creates charts.

Standard Python modules help manage files and record analysis details.

Sources: Biopython: applications | Requests: requests and responses | pandas: reading tables | Matplotlib: histogram

The analysis environment

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Install the package as biopython, but import Bio in the code.

The libraries must be available in the environment running the program.

A separate project environment reduces version conflicts.

Record the versions of the libraries you use.

```
# Run in the terminal, not inside a script:
python -m pip install biopython requests pandas matplotlib

# Python code:
import Bio
print(Bio.__version__)
```

Sources: Python: virtual environments | Biopython: installation and version

Input files and results

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Keep downloaded data unchanged.

Save results in a separate directory.

A relative path depends on the current working directory.

The README describes the data sources and how to run the program.

```
project/
    data/        # original input files
    results/      # output tables and charts
    analysis.py  # analysis script
    README.md
    requirements.txt
```

Sources: Python: pathlib | Wilson et al.: good computing practices

Paths with pathlib

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Path represents the location of a file or directory.

Before reading, check that the specified file exists.

Create the results directory before saving files.

resolve() produces an absolute path and resolves symbolic links.

```
from pathlib import Path
Path("data").mkdir(exist_ok=True)
Path("results").mkdir(exist_ok=True)
file = Path("data") / "proteins.fasta"
print(Path.cwd())
print(file.resolve())
if not file.is_file():
    raise FileNotFoundError(file)
```

Sources: Python: pathlib

File formats and parsers

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

A parser reads text according to the rules of a particular format.

FASTA contains headers and sequences, while GenBank also includes detailed annotations.

TSV is a table, and JSON can contain nested objects.

A filename extension does not guarantee the file contents.

Sources: Biopython: SeqIO | pandas: reading tables | Python: JSON

FASTA: one record, several lines

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

The > sign begins the header for the next sequence.

A sequence can be split across several lines.

The example contains two fictional DNA fragments, each 12 nucleotides long.

The parser joins lines belonging to the same sequence.

```
>dna_1 fictional DNA fragment
ATGCGT
ACGTAA
>dna_2 fictional DNA fragment
ATGCNNACGTAA
```

Sources: Biopython: SeqIO

Seq and SeqRecord

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Seq stores a sequence and provides operations on it.

SeqRecord combines a sequence with an identifier and description.

Annotations are available only if the source contains them.

A Seq object does not automatically determine whether the data are DNA or protein.

Record attribute	Meaning
record.seq	Sequence
record.id	Identifier
record.description	Header description
record.features	Annotated sequence regions

Sources: Biopython: Seq objects | Biopython: SeqRecord and annotations

Reading multiple sequences

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

SeqIO.parse reads successive records in the specified format.

A loop lets us process them one at a time.

len(record.seq) gives the sequence length.

The length unit depends on whether the data are DNA, RNA or protein.

```
from Bio import SeqIO
from pathlib import Path

for record in SeqIO.parse(Path("data/dna.fasta"), "fasta"):
    print(record.id, len(record.seq), record.seq)

# dna_1 12 ATGCGTACGTAA
# dna_2 12 ATGCNNACGTAA
```

Sources: Biopython: SeqIO

Reading a single record

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

SeqIO.read requires exactly one record.

Zero records or more than one record causes an error.

This is useful when we expect a specific reference sequence.

After reading, check the identifier and length.

```
from Bio import SeqIO

record = SeqIO.read("data/mtDNA.gb", "genbank")
print(record.id)
print(len(record.seq))
# NC_012920.1
# 16569
```

Sources: Biopython: SeqIO | NCBI: NC_012920.1

Length and ambiguous symbols

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

In DNA, N represents an unspecified nucleotide.

It still occupies one position in the sequence.

Decide how to handle these positions before analysis.

Removing them changes both length and position numbering.

```
s = "ATGCNNACGTAA".upper()
if not s:
    raise ValueError("Empty sequence")
n = s.count("N")
print(len(s), n, round(100 * n / len(s), 1))
# 12 2 16.7
```

Sources: Biopython: Seq objects

GC content in DNA

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

GC content is the proportion of guanine and cytosine in a DNA sequence.

Without ambiguous symbols, calculate (G + C) / length.

gc_fraction returns a fraction, so multiply by 100 for a percentage.

Specify how ambiguous symbols are handled.

```
from Bio.SeqUtils import gc_fraction

s = "ACGTN"
print(100 * gc_fraction(s, ambiguous="remove"))  # 50.0
print(100 * gc_fraction(s, ambiguous="ignore"))  # 40.0
```

Sources: Biopython: gc_fraction

Biological positions and Python indices

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

The inclusive range 4-8 contains five nucleotides.

Python indices start at zero, and a slice excludes its end index.

The inclusive range 4-8 therefore becomes seq[3:8].

Check the source coordinate convention for every range you use.

```
from Bio.Seq import Seq
seq = Seq("ATGCGTACGTAA")
start, end = 4, 8
fragment = seq[start - 1:end]
print(fragment, len(fragment))
# CGTAC 5
```

Sources: Biopython: Seq objects | Biopython: SeqRecord and annotations

The reverse complement

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

DNA bases pair as A with T and C with G.

The two strands run in opposite directions.

To write the other strand from 5’ to 3’, complement the bases and reverse their order.

reverse_complement performs both operations.

```
from Bio.Seq import Seq
seq = Seq("ATGC")
print(seq.complement())           # TACG
print(seq.reverse_complement())   # GCAT
assert seq.reverse_complement().reverse_complement() == seq
```

Sources: Biopython: Seq objects

Transcription and translation in code

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Biological transcription produces RNA using DNA as a template.

The transcribe method replaces T with U in the supplied sequence.

For an intron-free coding DNA strand, this gives the corresponding RNA sequence.

translate reads nucleotide triplets using the selected genetic code table.

```
from Bio.Seq import Seq
cds = Seq("ATGTTCTAA")  # fictional coding region
rna = cds.transcribe()
print(rna)                          # AUGUUCUAA
print(rna.translate(table=1))        # MF*
print(rna.translate(table=1, to_stop=True))  # MF
```

Sources: Biopython: Seq objects | NCBI: genetic code tables

Reading frames and stop codons

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

The starting position changes how a sequence is divided into triplets.

translate does not automatically find the start of a gene.

to_stop ends the output at the first stop codon.

cds=True checks selected conditions for a complete coding sequence.

```
from Bio.Seq import Seq
cds = Seq("ATGTTCTAA")
print(cds.translate(table=1, cds=True))  # MF

# CDS is a protein-coding sequence
```

Sources: Biopython: Bio.Seq

The genetic code table matters

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

The standard genetic code does not cover every case.

Some codons have different meanings in vertebrate mitochondria.

NCBI table 1 is the standard code, and table 2 is the vertebrate mitochondrial code.

Choose the table using annotations and biological context.

```
from Bio.Seq import Seq
codon = Seq("TGA")
print(codon.translate(table=1))  # *  (STOP)
print(codon.translate(table=2))  # W  (tryptophan)
```

Sources: NCBI: genetic code tables | Biopython: Seq objects

GenBank: sequences with annotations

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Record NC_012920.1 describes reference human mtDNA.

SeqIO reads the sequence, organism information and annotated features.

A CDS feature describes a protein-coding region.

Not every item in features is a gene or a CDS.

```
from Bio import SeqIO
record = SeqIO.read("data/mtDNA.gb", "genbank")
print(record.annotations.get("organism"))
for feature in record.features:
    if feature.type == "CDS":
        print(feature.location)
# COX1: [5903:7445](+) using Python coordinates
```

Sources: Biopython: SeqRecord and annotations | NCBI: NC_012920.1

Extracting a region using annotations

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

feature.extract selects the region specified by an annotation.

It handles strand orientation and joins the parts of a compound feature.

qualifiers holds details such as the gene name and genetic code table.

Also check whether the CDS is complete and whether exceptions apply.

```
cds_feature = next(
    f for f in record.features
    if f.type == "CDS" and "COX1" in f.qualifiers.get("gene", [])
)
cds_seq = cds_feature.extract(record.seq)
table = int(cds_feature.qualifiers["transl_table"][0])
print(len(cds_seq), table)  # 1542 2
```

Sources: Biopython: SeqRecord and annotations | NCBI: NC_012920.1

FASTA does not preserve every annotation

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Converting GenBank to FASTA preserves the sequence and header.

Standard FASTA does not preserve gene location annotations.

Keep the source file if this information may be needed.

Choose the output format according to the information needed next.

```
from Bio import SeqIO

n = SeqIO.write([record], "results/mtDNA.fasta", "fasta")
print(n)  # 1
check = SeqIO.read("results/mtDNA.fasta", "fasta")
assert check.seq == record.seq
```

Sources: Biopython: SeqIO | Biopython: SeqRecord and annotations

Large and compressed files

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

An iterator lets us process records one at a time.

Creating a list of all records increases memory use.

A .gz file can be read without manually unpacking it.

In this example, the sequence parser requires a text stream.

```
import gzip
from pathlib import Path
from Bio import SeqIO
# First download proteins.fasta using slide 34.
with gzip.open("data/proteins.fasta.gz", "wb") as file:
    file.write(Path("data/proteins.fasta").read_bytes())
with gzip.open("data/proteins.fasta.gz", "rt") as file:
    count = sum(1 for _ in SeqIO.parse(file, "fasta"))
print(count)  # 3
```

Sources: Python: gzip | Biopython: SeqIO

Three records for checking the program

P42212 is GFP from the jellyfish *Aequorea victoria*.

P69905 and P68871 describe the human haemoglobin alpha and beta subunits.

We will use these records to check data reading and merging.

UniProt ID	Record description
P42212	Green fluorescent protein GFP
P69905	Human haemoglobin alpha subunit
P68871	Human haemoglobin beta subunit

Sources: UniProt: GFP P42212 | UniProt: alpha P69905 | UniProt: beta P68871

Downloading the GFP sequence over HTTP

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

The URL identifies a specific UniProt record in FASTA format.

Check the HTTP status before saving.

Save the file in the input data directory.

Reading and calculations can then be repeated without downloading again.

```
from pathlib import Path
import requests

Path("data").mkdir(exist_ok=True)
url = "https://rest.uniprot.org/uniprotkb/P42212.fasta"
r = requests.get(url, timeout=30)
r.raise_for_status()
Path("data/gfp.fasta").write_bytes(r.content)
```

Sources: Requests: requests and responses | UniProt: API examples

Checking a downloaded record

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Read the saved file with SeqIO.read.

The FASTA header identifier follows the source format.

In a UniProt header, the accession is between vertical bars.

For this GFP record, we expect a length of 238 amino acids.

```
from Bio import SeqIO

gfp = SeqIO.read("data/gfp.fasta", "fasta")
accession = gfp.id.split("|")[1]  # UniProt header
print(accession, len(gfp.seq))
assert accession == "P42212"
assert len(gfp.seq) == 238
```

Sources: Biopython: SeqIO | UniProt: GFP P42212

Searching for a selected set of records

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

An OR query selects three specified UniProt identifiers.

We select the accession, organism and sequence length.

The size parameter limits the number of records per page.

The same search condition can return TSV or FASTA.

```
query = " OR ".join(
    f"accession:{x}" for x in ["P42212", "P69905", "P68871"]
)
url = "https://rest.uniprot.org/uniprotkb/search"
params = {"query": query, "format": "tsv", "size": 2,
          "fields": "accession,organism_name,length"}
```

Sources: UniProt: API examples | UniProt: result fields

Downloading further pages in Python

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

We read the next page address from r.links.

We follow the next link until there are no more pages.

We keep column names only from the first TSV page.

We do not add the initial query parameters to the complete next-page URL.

```
lines = []
while url:
    r = requests.get(url, params=params, timeout=30)
    r.raise_for_status()
    page = r.text.splitlines()
    lines.extend(page if not lines else page[1:])
    url = r.links.get("next", {}).get("url")
    params = None
Path("data/proteins.tsv").write_text(
    "\n".join(lines) + "\n", encoding="utf-8")
```

Sources: UniProt: API examples | Requests: Link headers

Connection errors and data errors

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

A timeout or server error may require another attempt after a pause.

An invalid query needs a correction, rather than repeated requests.

We also check the format, columns and number of records.

An empty file can look like a valid result.

Sources: Requests: requests and responses | UniProt: API examples

Downloading from NCBI with Bio.Entrez

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Bio.Entrez provides functions that use NCBI E-utilities.

EFetch downloads a record specified by its identifier.

We provide our own contact email, as required by NCBI.

This example saves the complete GenBank record NC_012920.1.

```
import os
from Bio import Entrez
from pathlib import Path
Entrez.email = os.environ["NCBI_EMAIL"]
with Entrez.efetch(db="nuccore", id="NC_012920.1",
                   rettype="gb", retmode="text") as handle:
    text = handle.read()
Path("data/mtDNA.gb").write_text(text, encoding="utf-8")
```

Sources: Biopython: NCBI Entrez | NCBI: NC_012920.1

Download details and data versions

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

We save the source, query and download date alongside the data.

A versioned identifier specifies a sequence version, if the database supports this.

We retain the database release information and a file checksum.

Repeating a query against an updated database may give a different result.

```
import hashlib, json
from datetime import datetime, timezone
description = {"database": "UniProtKB", "query": query,
        "downloaded_at": datetime.now(timezone.utc).isoformat(),
        "sha256": hashlib.sha256(
            Path("data/proteins.tsv").read_bytes()).hexdigest()}
Path("data/download.json").write_text(
    json.dumps(description, indent=2), encoding="utf-8")
```

Sources: Python: JSON | Wilson et al.: good computing practices

Reading a TSV table with pandas

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

A DataFrame is a table with named columns and labelled rows.

TSV separates columns with a tab, specified by sep="\t".

After reading, we check column names, data types and the first few rows.

Here, each row represents one UniProtKB protein record.

```
import pandas as pd
meta = pd.read_csv("data/proteins.tsv", sep="\t")
print(meta.head())
print(meta.dtypes)
meta = meta.rename(columns={"Entry": "accession",
    "Organism": "organism", "Length": "database_length"})
```

Sources: pandas: reading tables | UniProt: result fields

Checking columns and values

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

We check required columns and look for missing identifiers.

Sequence length must be a positive integer.

A missing value means missing information, not a length of zero.

Rows that fail these checks need investigation.

```
required = {"accession", "organism", "database_length"}
if not required.issubset(meta.columns):
    raise ValueError("Required columns are missing")
if meta["accession"].isna().any():
    raise ValueError("Missing identifier")
meta["database_length"] = pd.to_numeric(
    meta["database_length"], errors="raise")
d = meta["database_length"]
if not (d.notna() & (d > 0) & (d % 1 == 0)).all():
    raise ValueError("Invalid length")
```

Sources: pandas: missing data | pandas: reading tables

Repeated IDs and repeated sequences

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Two rows with the same ID may indicate a download or merge error.

Different IDs may represent an identical sequence.

These are two different types of duplication.

Removing duplicates requires a clear reason.

```
duplicates = meta["accession"].duplicated(keep=False)
print(meta.loc[duplicates])
if duplicates.any():
    raise ValueError("Duplicate identifiers in the table")
```

Sources: pandas: joining tables | Biopython: SeqIO

Sequences for the same query

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

We reuse the query that selects our three example records.

We download FASTA through the stream endpoint.

We expect all three records for this query.

For large datasets, we choose a download method supported by the service documentation.

```
url_fasta = "https://rest.uniprot.org/uniprotkb/stream"
r = requests.get(url_fasta,
    params={"query": query, "format": "fasta"}, timeout=30)
r.raise_for_status()
Path("data/proteins.fasta").write_bytes(r.content)
```

Sources: UniProt: API examples | Requests: requests and responses

Building a table from sequences

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

For each record, we retain its accession and calculated length.

Each loop iteration creates one row of the future table.

Protein sequence length is the number of amino acids.

We do not identify a record by its position in a list.

```
rows = []
for record in SeqIO.parse("data/proteins.fasta", "fasta"):
    accession = record.id.split("|")[1]
    rows.append({"accession": accession,
                 "sequence_length": len(record.seq)})
metrics = pd.DataFrame(rows)
print(metrics)
```

Sources: Biopython: SeqIO | pandas: reading tables

Checking that the data are complete

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

We expect exactly three specified identifiers.

We check the descriptions and calculated values separately.

Three rows alone do not rule out duplicates or an incorrect record.

Checking the set of IDs is stronger than checking the record count alone.

```
expected = {"P42212", "P69905", "P68871"}
for table in [meta, metrics]:
    if set(table["accession"]) != expected:
        raise ValueError("Unexpected set of records")
    if len(table) != 3:
        raise ValueError("Unexpected row count")
```

Sources: UniProt: API examples | Biopython: SeqIO

Joining tables by identifier

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

We join the data using the accession column.

validate="one_to_one" requires a unique key in both tables.

An outer join also keeps records without a match.

The _merge column shows where each row comes from.

```
result = meta.merge(metrics, on="accession", how="outer",
                    validate="one_to_one", indicator=True)
print(result[["accession", "_merge"]])
if not result["_merge"].eq("both").all():
    raise ValueError("Some records have no match")
```

Sources: pandas: joining tables

Comparing sequence lengths

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

We compare the database length with the length calculated from FASTA.

Agreement checks one specific part of downloading and reading the data.

A difference may reflect another protein form or data version.

We keep rows with differences so that we can investigate them.

```
matching = result["sequence_length"].eq(result["database_length"])
if not matching.all():
    print(result.loc[~matching])
    raise ValueError("Sequence lengths do not match")
result.to_csv("results/proteins_checked.tsv",
              sep="\t", index=False)
```

Sources: Biopython: SeqIO | pandas: reading tables

Sequence length distribution on a histogram

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

A histogram counts sequences within length intervals.

The horizontal axis needs a unit, here the number of amino acids.

The number of intervals affects the appearance of the plot.

Three records are enough only to test the code.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(7, 4))
ax.hist(result["sequence_length"], bins=5, edgecolor="white")
ax.set_xlabel("Protein sequence length [amino acids]")
ax.set_ylabel("Number of records")
fig.tight_layout()
fig.savefig("results/lengths.png", dpi=150)
plt.close(fig)
```

Sources: Matplotlib: histogram

Group summaries and the median

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Grouping calculates a separate summary for each organism.

The median is the middle value, or the mean of the two middle values for an even count.

We also report the number of observations and the range.

Group size and record selection affect our interpretation.

```
summary = result.groupby("organism")["sequence_length"].agg(
    count="count", median="median", minimum="min", maximum="max"
)
print(summary)
summary.to_csv("results/summary.tsv", sep="\t")
```

Sources: pandas: grouping

What does a difference between groups mean?

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

The result describes records selected using a particular rule.

Databases are not random samples of all molecules found in nature.

Reviewed status, fragments and isoform selection affect the dataset.

Finding a difference does not explain its cause.

Sources: UniProt: API examples | UniProt: result fields | Wilson et al.: good computing practices

Protein properties calculated from sequence

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

ProtParam calculates selected properties from an amino acid sequence.

For example, we can estimate the molecular mass of the amino acid chain.

The sequence must use symbols supported by the method.

The result does not automatically account for protein processing or all chemical modifications.

```
from Bio.SeqUtils.ProtParam import ProteinAnalysis
s = str(gfp.seq).upper()
if set(s) - set("ACDEFGHIKLMNPQRSTVWY"):
    raise ValueError("Non-standard amino acid symbol")
mass = ProteinAnalysis(s).molecular_weight()
print(round(mass, 1), "Da")
```

Sources: Biopython: ProtParam | UniProt: GFP P42212

Searching for a short pattern

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

A pattern is a specified arrangement of symbols in a sequence.

The usual count method counts non-overlapping occurrences.

Overlapping occurrences require a different counting rule.

```
s = "AAAA"
print(s.count("AA"))  # 2
positions = [i for i in range(len(s) - 1)
             if s[i:i + 2] == "AA"]
print(positions)         # [0, 1, 2]
```

Sources: Biopython: Seq objects | Biopython: Bio.Seq

FASTQ: a sequence with quality scores

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

FASTQ stores a read sequence and a quality score for each position.

A read is a sequence fragment obtained through sequencing.

A Phred score of Q=30 corresponds to an estimated error probability of 0.001.

This example creates a fictional ACGT read with Q=30 at every position.

```
from pathlib import Path
from Bio import SeqIO
Path("data").mkdir(exist_ok=True)
# Fictional example:
Path("data/reads.fastq").write_text("@test\nACGT\n+\n????\n")
for read in SeqIO.parse("data/reads.fastq", "fastq"):
    qualities = read.letter_annotations["phred_quality"]
    assert len(read.seq) == len(qualities)
    print(read.id, len(read.seq), min(qualities))  # test 4 30
```

Sources: Biopython: SeqIO | Biopython: SeqRecord and annotations

Small test examples

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Before a large analysis, we test data with known results.

Checks include lengths, fragment boundaries and matching identifiers.

Empty or invalid inputs should also produce predictable results.

```
from Bio.Seq import Seq
s = Seq("ATGCGTACGTAA")
assert len(s[3:8]) == 5
assert s.reverse_complement().reverse_complement() == s
assert str(Seq("ATGTTCTAA").translate(cds=True)) == "MF"
```

Sources: Biopython: Bio.Seq | Python: assert statement

Organising the program into stages

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

We separate downloading, reading, checking and calculations.

Each function performs a clearly defined task.

We keep analysis parameters in one place.

The program should run from the beginning without manual editing.

```
def length_table(path):
    rows = []
    for record in SeqIO.parse(path, "fasta"):
        rows.append({"accession": record.id.split("|")[1],
                     "sequence_length": len(record.seq)})
    return pd.DataFrame(rows)

metrics = length_table("data/proteins.fasta")
```

Sources: Biopython: SeqIO | Wilson et al.: good computing practices

Project: comparing protein lengths

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

The question concerns length distributions in clearly selected UniProtKB datasets.

We use the same selection rules for both organisms.

We check complete downloads, IDs, fragments and isoform handling.

We report group sizes, a plot and a conclusion limited to these data.

Sources: UniProt: API examples | UniProt: result fields | pandas: grouping | Matplotlib: histogram

Other uses of Python in bioinformatics

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

GenBank files support analysis of annotated sequence regions.

Tables of measurements also require sample metadata and clear units.

Specialised modules can read three-dimensional structure data.

We choose a project scope that matches the data and methods we understand.

Sources: Biopython: applications | Biopython: SeqRecord and annotations | pandas: reading tables

Preparing for part two

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

We can read sequences and tables while retaining their identifiers.

We can check coordinates, DNA orientation and basic translation assumptions.

We connect downloading with checks and saving results.

Part two will develop sequence comparison and the interpretation of tool results.

Sources: Biopython: SeqIO | Biopython: sequence alignment | Wilson et al.: good computing practices

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Thank you for your attention

Basics of Bioinformatics

Lecture 3

Python in bioinformatics - part one