Tin Sinh học
Bioinformatics

Bài thực hành BioPython 1

TS. Nguyễn Hồng Quang
Khoa Kỹ thuật máy tính
Trường Công nghệ thông tin và Truyền thông
Đại học Bách Khoa Hà Nội

Nội dung

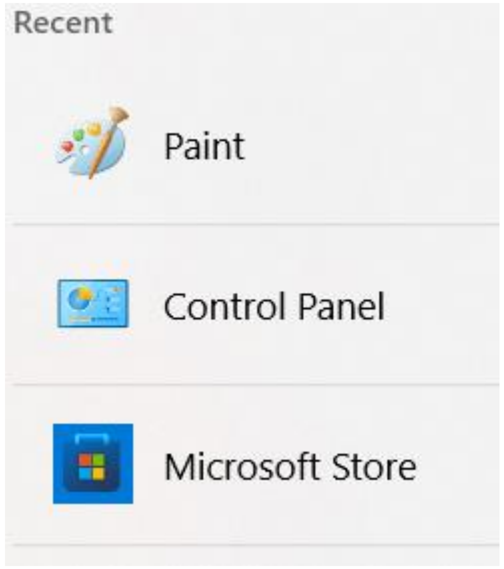
1. Cài đặt môi trường Window Subsystem Linux và Ubuntu

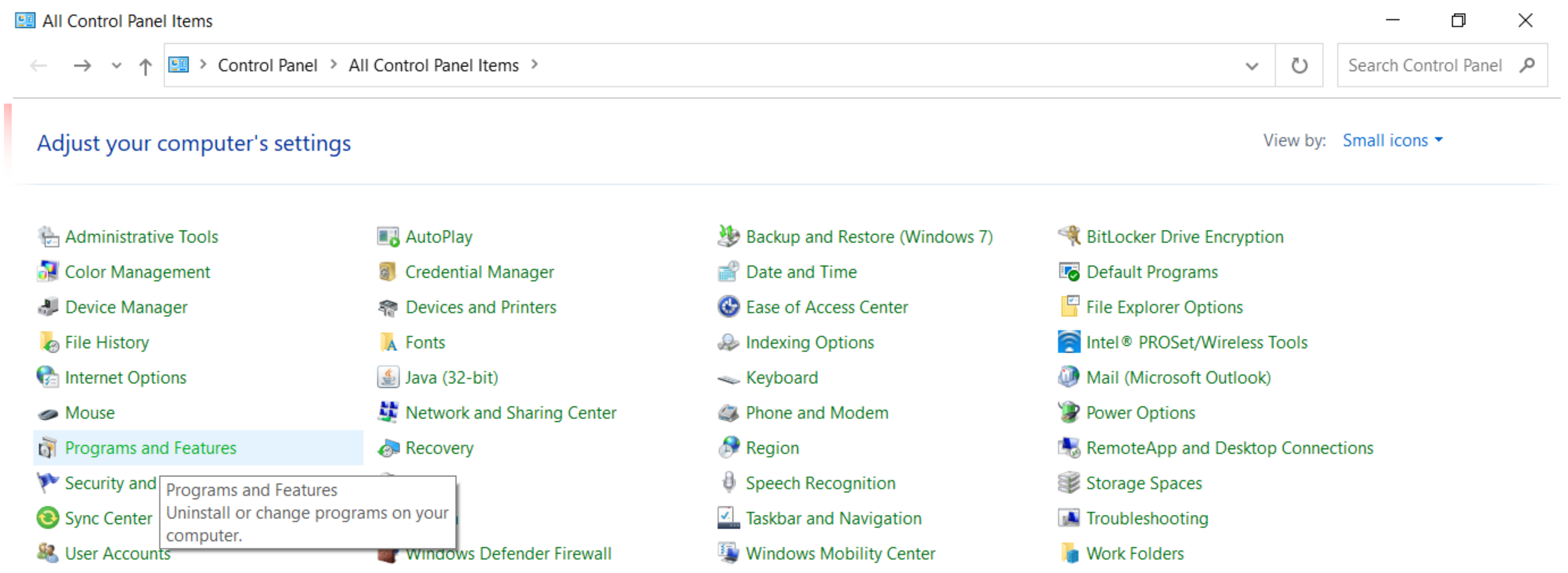
Báo cáo bài thực hành

- Thực hiện các thao tác trong bài thực hành
- Ghi lại kết quả vào báo cáo

I. Cài đặt Windows Subsystem Linux

- Vào Control Panel > Program and Features > Turn Windows feature on or off





All Control Panel Items

Control Panel > All Control Panel Items >

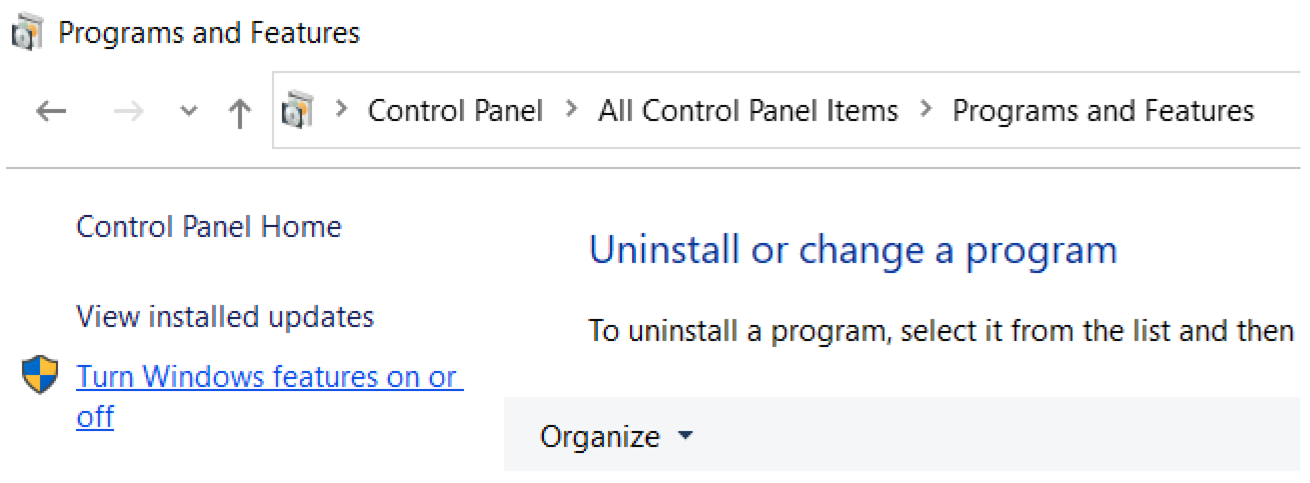
Search Control Panel

Adjust your computer's settings

View by: Small icons

Administrative Tools	AutoPlay	Backup and Restore (Windows 7)	BitLocker Drive Encryption
Color Management	Credential Manager	Date and Time	Default Programs
Device Manager	Devices and Printers	Ease of Access Center	File Explorer Options
File History	Fonts	Indexing Options	Intel® PROSet/Wireless Tools
Internet Options	Java (32-bit)	Keyboard	Mail (Microsoft Outlook)
Mouse	Network and Sharing Center	Phone and Modem	Power Options
Programs and Features	Recovery	Region	RemoteApp and Desktop Connections
Security and		Speech Recognition	Storage Spaces
Sync Center		Taskbar and Navigation	Troubleshooting
User Accounts	Windows Defender Firewall	Windows Mobility Center	Work Folders

Programs and Features
Uninstall or change programs on your computer.



Programs and Features

Control Panel > All Control Panel Items > Programs and Features

Control Panel Home

View installed updates

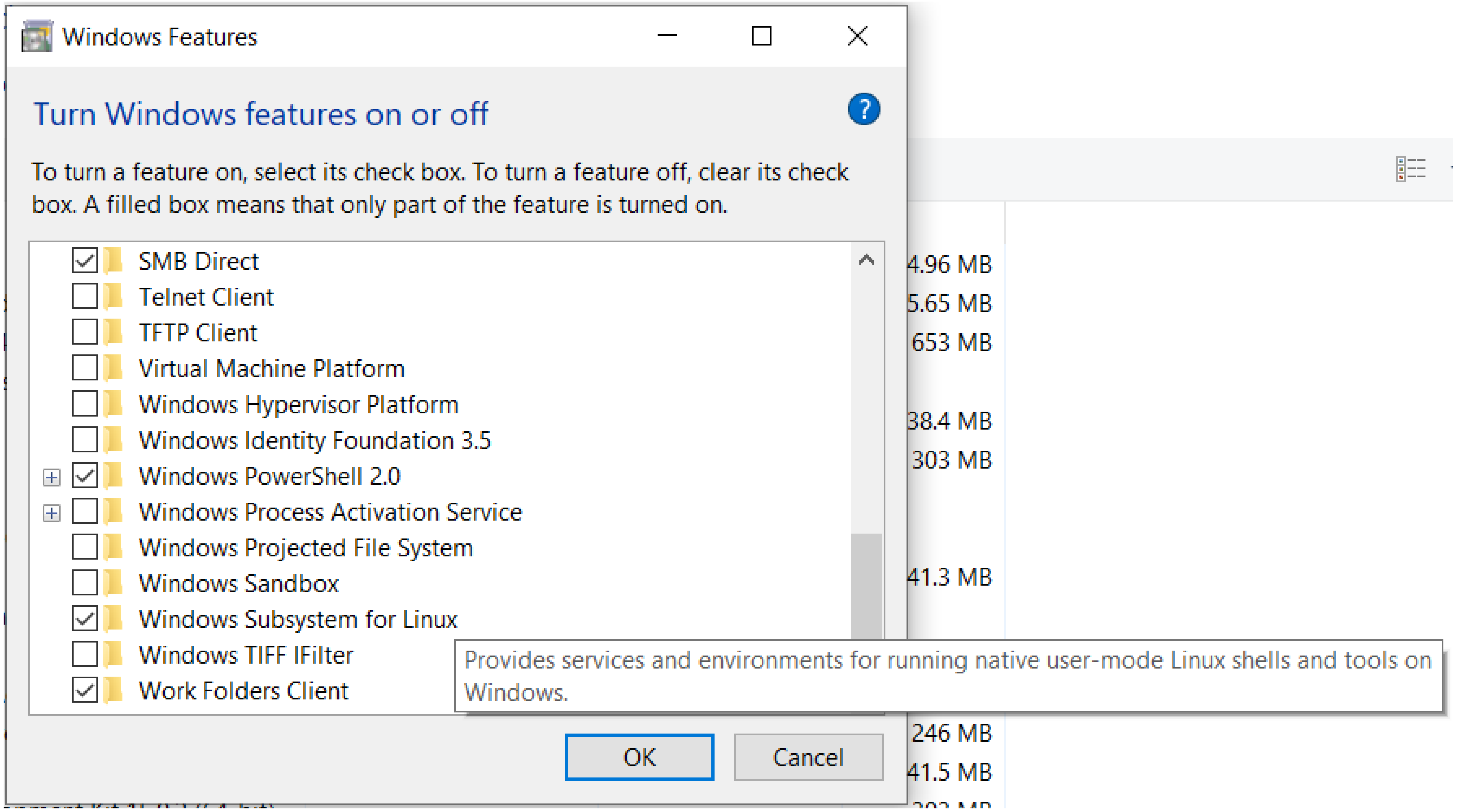
Turn Windows features on or off

Uninstall or change a program

To uninstall a program, select it from the list and then

Organize

Cài đặt Windows Subsystem Linux



Windows Features

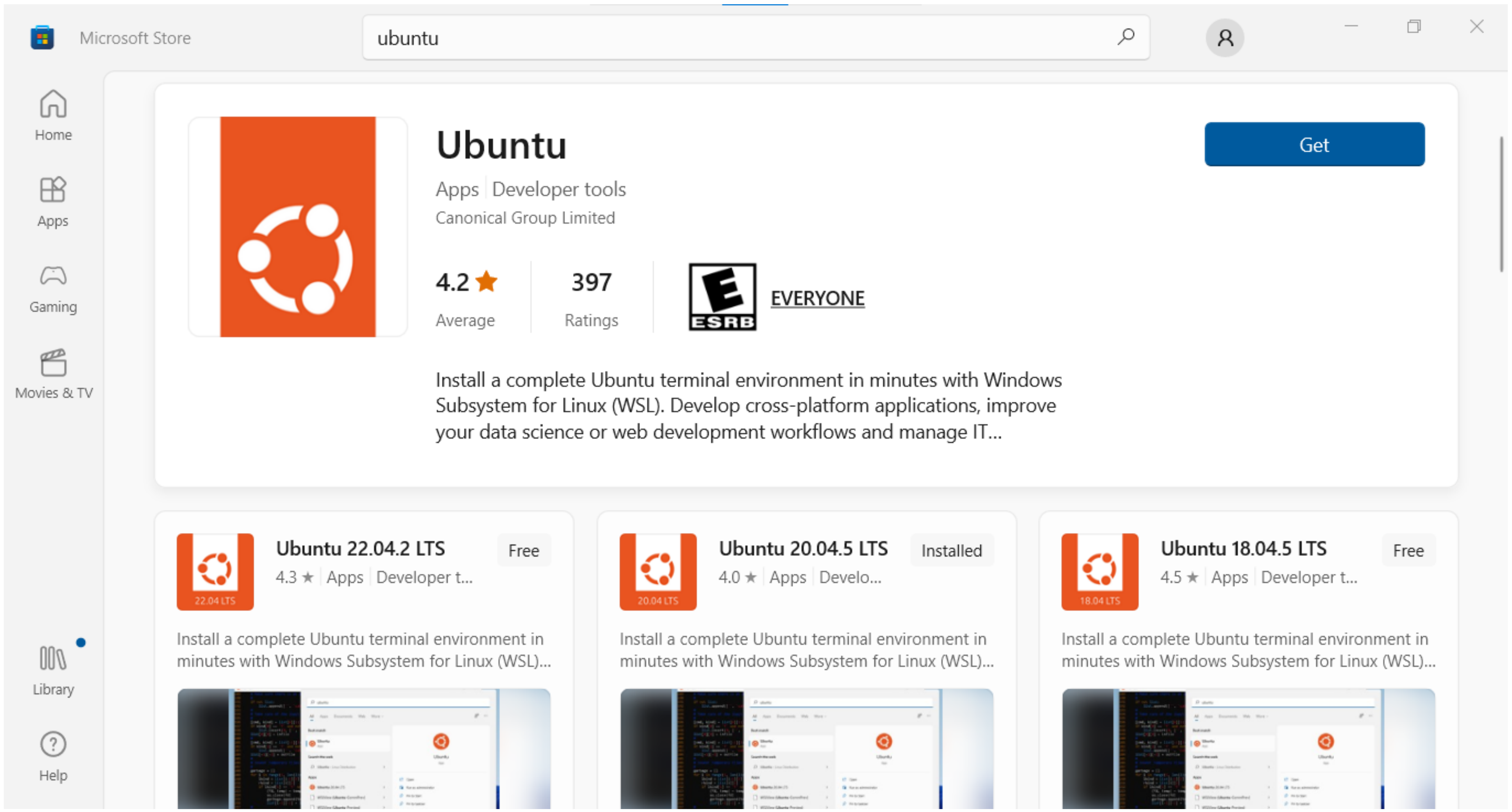
Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- [✓] SMB Direct
- [ ] Telnet Client
- [ ] TFTP Client
- [ ] Virtual Machine Platform
- [ ] Windows Hypervisor Platform
- [ ] Windows Identity Foundation 3.5
- [✓] Windows PowerShell 2.0
- [ ] Windows Process Activation Service
- [ ] Windows Projected File System
- [ ] Windows Sandbox
- [✓] Windows Subsystem for Linux
- [ ] Windows TIFF IFilter
- [✓] Work Folders Client

Provides services and environments for running native user-mode Linux shells and tools on Windows.

Cài đặt Ubuntu 22.04 LTS Vào Microsoft Store



Ubuntu
Apps | Developer tools
Canonical Group Limited

4.2 ★ Average | 397 Ratings

Install a complete Ubuntu terminal environment in minutes with Windows Subsystem for Linux (WSL). Develop cross-platform applications, improve your data science or web development workflows and manage IT...

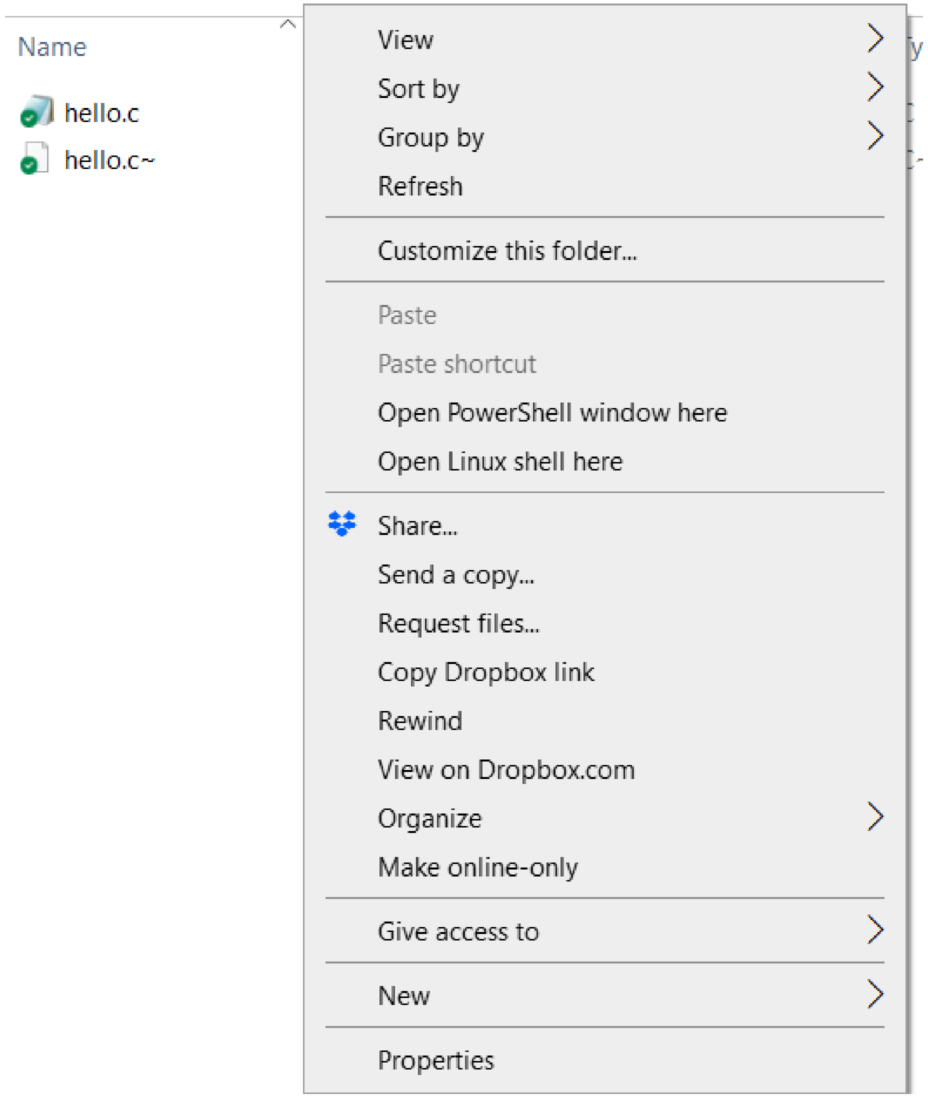
Ubuntu 22.04.2 LTS — Free
4.3 ★ | Apps | Developer t...
Install a complete Ubuntu terminal environment in minutes with Windows Subsystem for Linux (WSL)...

Ubuntu 20.04.5 LTS — Installed
4.0 ★ | Apps | Develo...
Install a complete Ubuntu terminal environment in minutes with Windows Subsystem for Linux (WSL)...

Ubuntu 18.04.5 LTS — Free
4.5 ★ | Apps | Developer t...
Install a complete Ubuntu terminal environment in minutes with Windows Subsystem for Linux (WSL)...

Để vào môi trường WSL Linux

- Vào thư mục hiện tại
- Giữ nút Shift và ấn chuột phải
- Chọn "Open Linux shell here"



2. Quick Start

What can you do with Biopython?

```
import Bio
from Bio.Seq import Seq
from Bio import SeqIO
from Bio.SeqUtils import GC
print("biopython version: ", Bio.__version__)
my_seq = Seq("AGTACACTGGT")
print(my_seq)
print(my_seq.complement())
print("Reverse complement: ",
my_seq.reverse_complement())
```

Đọc các định dạng file phổ biến: FASTA, GENBANK

```
# FASTA parsing example
for seq_record in SeqIO.parse("ls_orchid.fasta", "fasta"):
    print("Id của chuỗi: ", seq_record.id)
    print(repr(seq_record.seq))
    print(seq_record.seq, " => ", len(seq_record.seq), " nucleotides")

    print("\n", seq_record)
    break
```

Báo cáo: Mô tả chi tiết định dạng FastA

GenBank file format

```
for seq_record in SeqIO.parse("ls_orchid.gbk", "genbank"):
    print(seq_record.id)
    print(repr(seq_record.seq))
    print(seq_record.seq, " => ", len(seq_record.seq))

    print("\n### Thông tin bản ghi theo định dạng GENBANK")
    print(seq_record)
    break
```

Báo cáo: Mô tả chi tiết định dạng GenBank

3. Các thao tác xử lý chuỗi trình tự

3.1 Sequences act like strings

```
my_seq = Seq("GATCG")
for index, letter in enumerate(my_seq):
    print("%i %s" % (index, letter))
    #print(index, letter)
    #print(my_seq[index])

print("Len sequence: ", len(my_seq))
print(my_seq[0])
print(my_seq[2])
print(my_seq[-1])
```

Non-overlapping count

```
print("AAAA".count("AA"))

my_seq = Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC")
print("Len my_seq: ", len(my_seq))
print("Tổng số G: ", my_seq.count("G"))
print("Tổng số C: ", my_seq.count("C"))
gc = 100 * float(my_seq.count("G") + my_seq.count("C")) / len(my_seq)
print("Tỷ lệ GC: ", gc)

my_seq = Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC")
print("Tỷ lệ GC: ", GC(my_seq))
```

3.2. Slicing a sequence

```
my_seq = Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC")
my_seq[4:12]

# get the first, second and third codon positions of this DNA sequence:
# với 3 ORF (Open Reading Frame)
print(my_seq[0::3])
print(my_seq[1::3])
print(my_seq[2::3])

# Reverse the string
print(my_seq[::-1])
```

3.3. Turning Seq objects into strings

```
print(str(my_seq))

# Convert to FASTA format
fasta_format_string = ">Name\n%s\n" %
my_seq
print("\nFASTA:\n", fasta_format_string)
```

3.4. Concatenating or adding sequences

```
dna_seq_1 = Seq("ACGT")
dna_seq_2 = Seq("CGTATG")
dna_seq = dna_seq_1 + dna_seq_2
print("dna_seq: ", dna_seq)

list_of_seqs = [Seq("ACGT"), Seq("AACC"), Seq("GGTT")]
concatenated = Seq("")
for s in list_of_seqs:
    concatenated += s
print("concatenated: ", concatenated)
```

join method

```
contigs = [Seq("ATG"), Seq("ATCCCG"),
Seq("TTGCA")]
spacer = Seq("N"*10)
new_seq = spacer.join(contigs)
print("new_seq: ", new_seq)
```

Changing case

```
dna_seq = Seq("acgtACGT")
dna_seq_upper = dna_seq.upper()
dna_seq_lower = dna_seq.lower()
print("dna_seq_upper: ", dna_seq_upper)
print("dna_seq_lower: ", dna_seq_lower)
```

# Nucleotide sequences and (reverse) complements

```
my_seq = Seq("ACGGTA")
print("\n Complement: ",
my_seq.complement())
print("Reverse complement: ",
my_seq.reverse_complement())
```

3.7. Quá trình phiên mã : Transcription

- Consider the following (made up) stretch of double stranded DNA which encodes a short peptide:
- DNA coding strand (aka Crick strand, strand +1)
- `5’ ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG 3’`

  `|||||||||||||||||||||||||||||||||||||||`
- `3’ TACCGGTAACATTACCCGGCGACTTTCCCACGGGCTATC 5’`
- DNA template strand (aka Watson strand, strand −1)
- `|`
- `Transcription`
- `↓`
- `5’ AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG 3’`
- Single stranded messenger RNA

- The actual biological transcription process works from the template strand, doing a reverse complement (TCAG → CUGA) to give the mRNA. However, in Biopython and bioinformatics in general, we typically work directly with the coding strand because this means we can get the mRNA sequence just by switching T → U.
- These should match the figure above - remember by convention nucleotide sequences are normally read from the 5′ to 3′ direction, while in the figure the template strand is shown reversed.

```
coding_dna = Seq("ATGGCC")
print("coding_dna:    ", coding_dna)
template_dna = coding_dna.reverse_complement()
print("template_dna:  ", template_dna)

messenger_rna = coding_dna.transcribe()
print("messenger_rna: ", messenger_rna)

# The Seq object also includes a back-transcription method for
# going from the mRNA to the coding strand of the DNA.
# Again, this is a simple U → T substitution:
print("back_transcribe: ", messenger_rna.back_transcribe())
```

3.8. Bảng dịch mã Translation Table

```
from Bio.Data import CodonTable
standard_table = CodonTable.unambiguous_dna_by_name["Standard"]
#standard_table = CodonTable.unambiguous_dna_by_id[1]

# Vertebrate Mitochondrial: ty thể của động vật có xương sống
mito_table = CodonTable.unambiguous_dna_by_name["Vertebrate
Mitochondrial"]
mito_table = CodonTable.unambiguous_dna_by_id[2]

print(standard_table)
#print(mito_table)

print("Stop codons: ", standard_table.stop_codons)
print("Start codons: ", standard_table.start_codons)
```

3.9. Quá trình dịch mã: Translation

```
messenger_rna =
Seq("AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG")
protein_seq = messenger_rna.translate()
print("protein_seq: ", protein_seq)
print(messenger_rna.translate(to_stop=True))

# You can also translate directly from the coding strand DNA sequence:
coding_dna =
Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG")
protein_seq = coding_dna.translate(to_stop=True)
print("protein_seq: ", protein_seq)
```

3.10. Comparing Seq objects

```
# Comparing Seq objects
seq1 = Seq("ACGT")
print("ACGT" == seq1)
print(seq1 == "ACGT")
```

3.11. MutableSeq objects

```
from Bio.Seq import Seq
my_seq =
Seq("GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA")
my_seq[5] = "G"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-50e848a55f5c> in <module>
----> 1 my_seq[5] = "G"

TypeError: 'Seq' object does not support item assignment
```

```
from Bio.Seq import MutableSeq
mutable_seq =
MutableSeq("GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA")
mutable_seq[5] = "C‶
mutable_seq
mutable_seq.remove("T")
mutable_seq
mutable_seq.reverse()
mutable_seq
# get back to a read-only Seq object
new_seq = mutable_seq.toseq()
new_seq
```

3.12. UnknownSeq objects

- The UnknownSeq object is a subclass of the basic Seq object and its purpose is to represent a sequence where we know the length, but not the actual letters making it up.
- You could of course use a normal Seq object in this situation, but it wastes rather a lot of memory to hold a string of a million “N” characters when you could just store a single letter “N” and the desired length as an integer.

```
from Bio.Seq import UnknownSeq
unk = UnknownSeq(20)
unk
print(unk)
len(unk)
```

```
????????????????????
20
```

- For DNA or RNA sequences, unknown nucleotides are commonly denoted by the letter “N”, while for proteins “X” is commonly used for unknown amino acids.
- When creating an ‘UnknownSeq‘, you can specify the character to be used instead of “?” to represent unknown letters.

```
from Bio.Seq import UnknownSeq
unk_dna = UnknownSeq(20, character="N")
unk_dna
print(unk_dna)
```

```
NNNNNNNNNNNNNNNNNNNN
```

Tài liệu tham khảo

- Biopython Tutorial and Cookbook
- Link: http://biopython.org/DIST/docs/tutorial/Tutorial.html