

BIOINFORMATICS

SESSION 2 PRACTICE

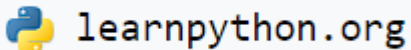
2023-09-11

Working with the molecules of life in the computer

Contents

1. How to use jupyter notebook
2. Basic python programming; strings, loops, dictionary
 - DNA sequence as string
 - inferring products of DNA replication
 - inferring RNA products of trascription
 - inferring protein products of translation
3. Exercise & Assignment

Python tutorial



<https://www.learnpython.org/>

Hello, World!

Python is a very simple language, and has a very straightforward syntax. It encourages programmers to program without boilerplate (prepared) code. The simplest directive in Python is the "print" directive - it simply prints out a line (and also includes a newline, unlike in C).

There are two major Python versions, Python 2 and Python 3. Python 2 and 3 are quite different. This tutorial uses Python 3, because it more semantically correct and supports newer features.

For example, one difference between Python 2 and 3 is the `print` statement. In Python 2, the "print" statement is not a function, and therefore it is invoked without parentheses. However, in Python 3, it is a function, and must be invoked with parentheses.

To print a string in Python 3, just write:

```
script.py | IPython Shell
1 print("This line will be printed.")
In [1]: |
```

Run

Powered by DataCamp

Chapters

- [Hello, World!](#)
- [Variables and Types](#)
- [Lists](#)
- [Basic Operators](#)
- [String Formatting](#)
- [Basic String](#)
- [Operations](#)
- [Conditions](#)
- [Loops](#)
- [Functions](#)
- [Classes and Objects](#)
- [Dictionaries](#)
- [Modules and Packages](#)
- [Numpy Arrays](#)
- [Pandas Basics](#)
- [Generators](#)
- [List Comprehensions](#)
- [Lambda functions](#)
- [Multiple Function Arguments](#)
- [Regular Expressions](#)
- [Exception Handling](#)
- [Sets](#)
- [Serialization](#)
- [Partial functions](#)
- [Code Introspection](#)
- [Closures](#)
- [Decorators](#)
- [Map, Filter, Reduce](#)
- [Contributing Tutorials](#)

Python tutorial

BAE/KJOON >
ONLINE JUDGE

<https://www.acmicpc.net/>

BAE/KJOON >
ONLINE JUDGE

문제 > 문제집 > 대회 > 채점 현황 > 경향 > 기사판 > 그룹 > 더 보기 > Q

문제	문제	출처	ICPC
> 전체 문제	> 푼 사람이 한 명인 문제	> ICPC	> Regionals
> 문제 출처	> 아무도 못 푼 문제	> Olympiad	> World Finals
> 단계별로 풀어보기	> 최근 제출된 문제	> 한국정보올림피아드	> Korea Regional
> 알고리즘 분류	> 최근 풀린 문제	> 한국정보올림피아드시도지역본선	> Africa and the Middle East Regionals
> 추가된 문제	> 편람	> 전국 대학생 프로그래밍 대회 동아리 연합	> Europe Regionals
> 문제 순위		> 대학교 대회	> Latin America Regionals
		> 카카오 코드 페스티벌	> North America Regionals
		> Coder's High	> South Pacific Regionals

단계	제목
1	입출력과 사칙연산
2	조건문
3	반복문
4	1차원 배열
5	문자열
6	심화 1
7	2차원 배열
8	일반 수학 1
9	약수, 배수와 소수
10	기하: 직사각형과 삼각형
11	시간 복잡도
12	브루트 포스
13	정렬
14	집합과 맵
15	약수, 배수와 소수 2
16	스택, 큐, 덱
17	=== 짝치선 ===

단계	문제 번호	제목
1	2557	Hello World
Hello World를 화면에 출력하는 문제 (예제 출력과 똑같이)		
2	1000	A+B
두 수를 입력받고 합을 출력하는 문제		
3	1001	A-B
두 수를 입력받고 뺄셈 한 결과를 출력하는 문제		
4	10998	A*B
곱셈 문제		
5	1008	A/B
나눗셈 문제. 이 문제에는 "스페셜 저지" 표시가 붙어 있는데,		
6	10869	사칙연산
모든 연산 문제		
7	10926	???
입출력을 응용하는 문제??		
8	18108	1998년생인 내가 태국에서는
식물 직접 세워서 계산하는 문제		
9	10430	나이차
네 개의 계산식을 계산하는 문제. 이 문제를 쓴 다음에는 직접		
10	2588	곱셈
변 간에 들어갈 수는?		
11	11382	꼬아 정민
더 큰 수를 더하는 문제		
12	10171	고양이
\ ' 등의 문자에 주의하여 고양이를 출력하는 문제		
13	10172	개
\ ' 등의 문자에 주의하여 개를 출력하는 문제		

Python tutorial

2557만 **제출** [맞힌 사람](#) [순코딩](#) [재채점 결과](#) [재점 현황](#) [내 제출](#) [난이도 기여](#) [질문 게시판](#)

Hello World 프로그

☆

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	1012941	408848	287838	39.483%

[문제](#)

Hello World!를 출력하십시오.

[입력](#)

없음

[출력](#)

Hello World!를 출력하십시오.

[예제 입력 1](#) 복사

```
< >
```

[예제 출력 1](#) 복사

```
Hello World!
```

[알고리즘 분류](#)

- 구현

[메모](#)

[메모 작성하기](#)

2557만 **제출** [맞힌 사람](#) [순코딩](#) [재채점 결과](#) [재점 현황](#) [내 제출](#) [난이도 기여](#) [질문 게시판](#)

Hello World

언어 Python 3 언어 설정

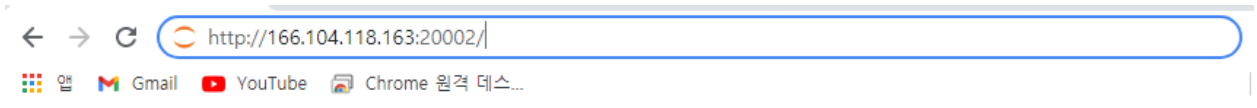
소스 코드 공개 공개 비공개 맞았을 때만 공개

소스 코드

```
1
```

제출

Jupyter notebook 접속



Password:

Log in



2023bio

Jupyter notebook 접속



jupyter

Files Running Clusters

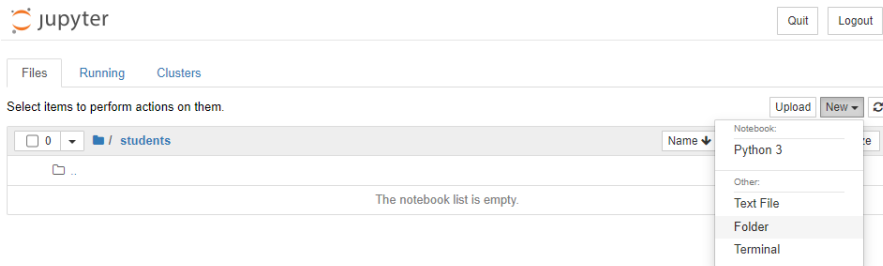
Select items to perform actions on them.

Upload New

	Name	Last Modified	File size
<input type="checkbox"/>	/		
<input type="checkbox"/>	materials	몇 초 전	
<input type="checkbox"/>	programs	2년 전	
<input type="checkbox"/>	R	2년 전	
<input type="checkbox"/>	students	14분 전	
<input type="checkbox"/>	trashcan	13분 전	
<input type="checkbox"/>	tutor	1분 전	
<input type="checkbox"/>	run_jupyter.sh	13일 전	64 B



Make a new directory:
학번_영어이름
Ex) 2023123456_hyunwookim



jupyter

Files Running Clusters

Select items to perform actions on them.

Upload New

	Name	Last Modified	File size
<input type="checkbox"/>	/ students		
<input type="checkbox"/>	..		

The notebook list is empty.

- Notebook:
 - Python 3
- Other:
 - Text File
 - Folder
 - Terminal

Make Session2 directory



Files

Running

Clusters

Select items to perform actions on them.



0



/ tutor

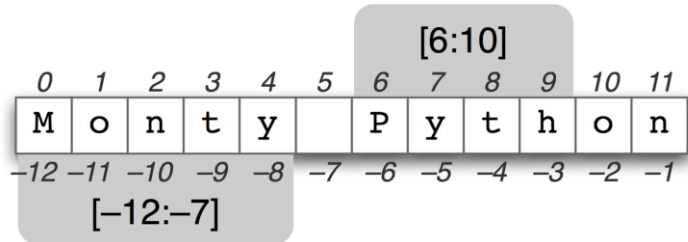


Session2

Basic Python - string

python implements 0-based indexing

```
$ python_string.ipynb
```



```
>>> rna='AGCTT'  
>>> print(rna[2:4])  
>>> print(rna[0:1])  
>>> print(rna[3:])  
>>> print(rna[:3])  
>>> print(rna[:-2])  
>>> print(rna[::-1])
```

[OUTPUT]
CT
A
TT
AGC
AGC
TTCGA

Basic Python - string

```
>>> dna = 'GCAATGG'
>>> print(dna)                "GCAATGG"
>>> rev = dna[::-1]
>>> print(rev)                "GGTAACG"
```

Basic Python - string

Code continues from the previous one

```
>>> rev = dna[::-1]
>>> comp = rev.maketrans('ACGT', 'TGCA')
>>> rev_comp = rev.translate(comp)
>>> print(rev_comp)
"CCATTGC"
```

DNA replication I (ssDNA)

replication.ipynb

```
import string  
  
dna = 'GCAATGG'  
rev = dna[::-1]  
comp = rev.maketrans('ACGT', 'TGCA')  
rev_comp = rev.translate(comp)  
print( rev_comp )
```

CCATTGC

DNA replication I (dsDNA) – small practice

```
replication2.ipynb
```

print the dsDNA sequence representation including the 5'- and -3' notation as below

```
5'-GCAATGG-3'  
3'-CGTTACC-5'
```

```
import string  
dna = 'GCAATGG'  
comp = dna.maketrans('ACGT','TGCA')  
rev_comp = dna.translate(comp)  
print ( "5#'-" + dna + "-3#'")  
print ( "3#'-" + rev_comp + "-5#'")
```

```
5'-GCAATGG-3'  
3'-CGTTACC-5'
```

Inferring RNA products of transcription

```
transcription.ipynb
```

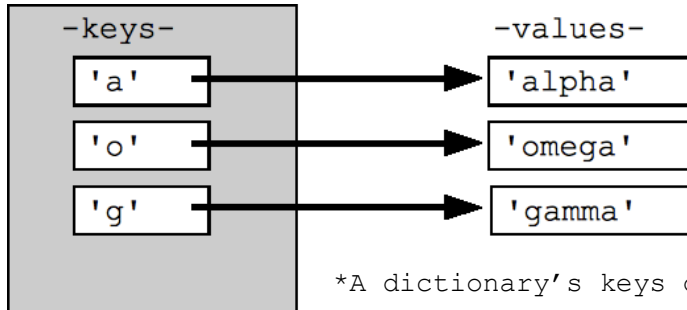
```
>>> dna = 'GCAATGG'  
>>> print("The DNA sequence is " + dna )  
>>> rna = dna.replace('T', 'U')  
>>> print("and the RNA sequence is " + rna )
```

```
dna = 'GCAATGG'  
print ( "The DNA sequence is " + dna )  
rna = dna.replace('T', 'U')  
print ( "and the RNA sequence is " + rna )
```

```
The DNA sequence is GCAATGG  
and the RNA sequence is GCAAUGG
```

Basic Python - dictionary

dictionary.ipynb



*A dictionary's keys do not allow duplicates

```
code = {'UUU':'F', 'UUC':'F', 'UUA':'L'}  
code  
print ( code['UUU'], code['UUC'] )
```

F F

```
code = {'UUU' : 'F', 'UUC' : 'F', 'UUU' : 'A', 'UUA' : 'L'}  
print(code['UUU'])
```

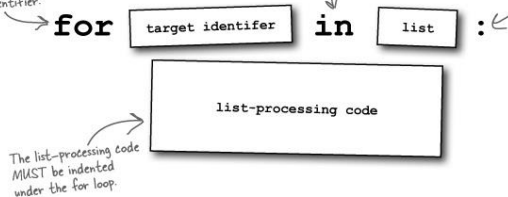
A

Basic Python - for loop

The keyword "for" indicates the start of the loop and comes before the target identifier.

The keyword "in" separates the target identifier from your list.

A colon ":" follows your list name and indicates the start of your list-processing code.



```
for i in range(0, 5, 1):  
    print ( i )  
print ( 'range(0,5,1)' )  
print ( list(range(0, 5, 1)) )  
print ( 'range(0,5,3)' )  
print ( list(range(0, 5, 3)) )
```

```
0  
1  
2  
3  
4  
range(0,5,1)  
[0, 1, 2, 3, 4]  
range(0,5,3)  
[0, 3]
```


Code1.1 translation.py

translation.ipynb

```
code = {
    'UUU': 'F', 'UUC': 'F', 'UUA': 'L', 'UUG': 'L', 'CUU': 'L', 'CUC': 'L', 'CUA': 'L', 'CUG': 'L',
    'AUU': 'I', 'AUC': 'I', 'AUA': 'I', 'AUG': 'M', 'GUU': 'V', 'GUC': 'V', 'GUA': 'V', 'GUG': 'V',
    'UCU': 'S', 'UCC': 'S', 'UCA': 'S', 'UCG': 'S', 'CCU': 'P', 'CCC': 'P', 'CCA': 'P', 'CCG': 'P',
    'ACU': 'T', 'ACC': 'T', 'ACA': 'T', 'ACG': 'T', 'GCU': 'A', 'GCC': 'A', 'GCA': 'A', 'GCG': 'A',
    'UAU': 'Y', 'UAC': 'Y', 'UAA': '*', 'UAG': '*', 'CAU': 'H', 'CAC': 'H', 'CAA': 'Q', 'CAG': 'Q',
    'AAU': 'N', 'AAC': 'N', 'AAA': 'K', 'AAG': 'K', 'GAU': 'D', 'GAC': 'D', 'GAA': 'E', 'GAG': 'E',
    'UGU': 'C', 'UGC': 'C', 'UGA': '*', 'UGG': 'W', 'CGU': 'R', 'CGC': 'R', 'CGA': 'R', 'CGG': 'R',
    'AGU': 'S', 'AGC': 'S', 'AGA': 'R', 'AGG': 'R', 'GGU': 'G', 'GGC': 'G', 'GGA': 'G', 'GGG': 'G'
}

dnaseq = 'GAACTGGGT'
print ( dnaseq )
rnaseq = dnaseq.replace('T', 'U')
print ( rnaseq )

for i in range(0, len(rnaseq), 3):
    codon = rnaseq[i:i + 3]
    amino_acid = code[codon]
    print ( '', amino_acid, end=' ' )
```

```
GAACTGGGT
GAACUGGGU
E L G
```

What we learned today

python keywords, variables, strings, numerics, loops, dictionary

```
code = {
    'UUU': 'F', 'UUC': 'F', 'UUA': 'L', 'UUG': 'L', 'CUU': 'L', 'CUC': 'L', 'CUA': 'L', 'CUG': 'L',
    'AUU': 'I', 'AUC': 'I', 'AUA': 'I', 'AUG': 'M', 'GUU': 'V', 'GUC': 'V', 'GUA': 'V', 'GUG': 'V',
    'UCU': 'S', 'UCC': 'S', 'UCA': 'S', 'UCG': 'S', 'CCU': 'P', 'CCC': 'P', 'CCA': 'P', 'CCG': 'P',
    'ACU': 'T', 'ACC': 'T', 'ACA': 'T', 'ACG': 'T', 'GCU': 'A', 'GCC': 'A', 'GCA': 'A', 'GCG': 'A',
    'UAU': 'Y', 'UAC': 'Y', 'UAA': '*', 'UAG': '*', 'CAU': 'H', 'CAC': 'H', 'CAA': 'Q', 'CAG': 'Q',
    'AAU': 'N', 'AAC': 'N', 'AAA': 'K', 'AAG': 'K', 'GAU': 'D', 'GAC': 'D', 'GAA': 'E', 'GAG': 'E',
    'UGU': 'C', 'UGC': 'C', 'UGA': '*', 'UGG': 'W', 'CGU': 'R', 'CGC': 'R', 'CGA': 'R', 'CGG': 'R',
    'AGU': 'S', 'AGC': 'S', 'AGA': 'R', 'AGG': 'R', 'GGU': 'G', 'GGC': 'G', 'GGA': 'G', 'GGG': 'G'
}
```

```
dnaseq = 'GAACTGGGT'
print ( dnaseq )
rnaseq = dnaseq.replace('T', 'U')
print ( rnaseq )

for i in range(0, len(rnaseq), 3):
    codon = rnaseq[i:i + 3]
    amino_acid = code[codon]
    print ( '', amino_acid, end=' ' )
```

```
GAACTGGGT
GAACUGGGU
E L G
```

```
import string
```

```
dna = 'GCAATGG'
rev = dna[::-1]
comp = rev.maketrans('ACGT', 'TGCA')
rev_comp = rev.translate(comp)
print( rev_comp )
```

CCATTGC

```
>>> rna='AGCTT'
>>> print(rna[2:4])
>>> print(rna[0:1])
>>> print(rna[3:])
>>> print(rna[:3])
>>> print(rna[:-2])
>>> print(rna[::-1])
```

Exercise

- **Modify code 1.1(translation.ipynb) so the reverse complementary strand of the original strand sequence (GAACTGGGT) is translated.** As with the original strand, we only consider the first reading frame of the nucleotide sequence, i.e. the first codon of the complementary strand is ACC

Exercise script and result

```
dnaseq = 'GAACTGGGT'  
rev = dnaseq[::-1]  
comp = rev.maketrans('ATGC', 'UACG')  
rnaseq = rev.translate(comp)  
print(rnaseq)  
  
for i in range(0, len(rnaseq), 3):  
    codon = rnaseq[i:i+3]  
    amino_acid = code[codon]  
    print('', amino_acid, end = ' ')
```

```
ACCCAGUUC  
T Q F
```

Assignment

- Using the code translation.ipynb and *dna_seq* below, generate and print **1) RNA (*rna_seq*) sequence** which is reverse complementary to the underlined DNA sequence segment, and **2) its translated amino acid sequence (*aminoacid_seq*)**. The variable “*dna_seq*” must be present in your python script.

```
dna_seq = "ATGCCTTGCAAATACGTCACGACAGTGAAAAA"
```

*hint 해당 sequence 부분을 indexing (ex, dna_seq[x:x])
을 사용해 추출 ->Reverse complementary

```
rna_seq = "UGUCGUAUUUGCAAG"
```

```
aminoacid_seq = ??
```

Translation

- 과제 제출 : 09/17 Sunday 23:59
- 해당 코드 캡처를 한 뒤 워드에 첨부. 기입 하고 코드에 대한 설명 간략히 작성 워드 파일명은 n주차_학번_이름 형식으로 제출(e.g. 2주차_2023123456_김현우)