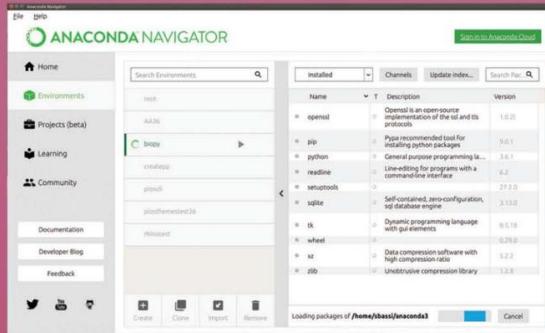


Chapman & Hall/CRC  
Mathematical and Computational Biology Series

# PYTHON FOR BIOINFORMATICS

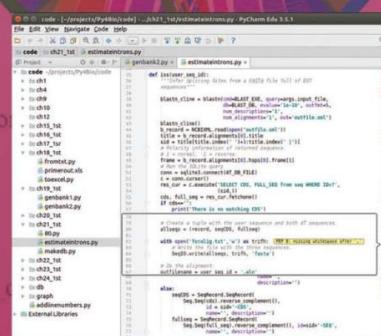
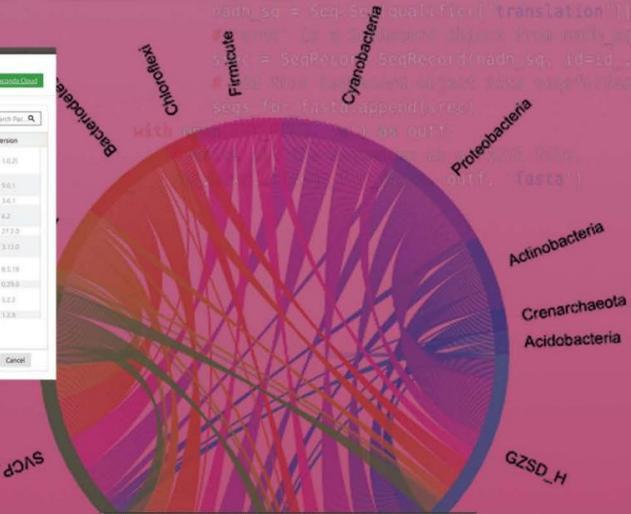
## SECOND EDITION



```
def sortmarkers(crms, end):
    """ Sort markers into chromosomes """
    i = 0
    crms_o = [] for r in range(len(end))]
    crms_fo = [] for r in range(len(end))]
    for r in range(len(end)):
        for mark in crms:
            if mark in crms[i]:
                # Add the marker start position to each chromosome
                crms_fo.append(mark[1])
                crms_o[i].append(mark[0]) # Save the marker positions.
        i += 1
    for order in crms_fo:
        # Sort markers in order and set terms to fill crms_o
        with all the marker information
        for pos in order:
            for mark in crms[i]:
                if pos == mark[1]:
                    crms_o[i].append(mark)
        i += 1
    return crms_o
```

 **biopython**

**SEBASTIAN BASSI**



 **CRC Press**  
Taylor & Francis Group

A CHAPMAN & HALL BOOK



# PYTHON UCHUN BIOINFORMATIKA

IKKINCHI NASHRI

# CHAPMAN & HALL/CRC

## Matematik va hisoblash biologiyasi turkumi

### Maqsadlar va

**qamrov:** Ushbu seriya yangi ishlanmalarni qo'lga kiritish va matematik va hisoblash biologiyasi va tibbiyotining butun spektrida ma'lum bo'lgan narsalarni umumlashtirishga qaratilgan. U ko'plab darsliklar, ma'lumotnomalar va qo'llanmalarni nashr etish orqali matematik, statistik va hisoblash usullarini biologiyaga integratsiyalashuvini rag'batlantirishga intiladi. Seriyaga kiritilgan sarlavhalar talabalar, tadqiqotchilar va matematika, statistik va hisoblash fanlari, fundamental biologiya va bioinjeneriya sohasidagi mutaxassislar, shuningdek, ushbu soha bilan shug'ullanadigan fanlararo tadqiqotchilarga murojaat qilish uchun mo'ljallangan. Aniq misollar va ilovalar, dasturlash texnikasi va misollarini kiritish juda rag'batlantiriladi.

### Seriya muharrirlari

NF Britton

*Bath universiteti matematika fanlari bo'limi*

Xixon Lin

*Biostatistika kafedrasи*

*Garvard universiteti*

Nikola Mulder

*Keyptaun universiteti*

*Janubiy Afrika*

Mariya Viktoriya Shnayder

*Yevropa bioinformatika instituti*

Mona Singx

*Kompyuter fanlari kafedrasи*

*Prinston universiteti*

Anna Tramontano

*Fizika kafedrasи*

*Rim La Sapienza universiteti*

Seriya uchun takliflar yuqoridaqgi seriya muharrirlaridan biriga yoki bevosita quyidagi manzilga yuborilishi kerak:

**CRC Press, Teylor va Frensis guruhi**

3 Park maydoni, Milton bog'i

Abingdon, Oksfordshir OX14 4RN

## Chop etilgan sarlavhalar



**Tizim biologiyasiga kirish:**

**Biologik konturlarni loyihalash tamoyillari**  
Uri Alon

**Glycome Informatics: usullari va ilovalari**

Kiyoko F. Aoki-Kinoshita

**Hisoblash tizimlari biologiyasi**

**Saraton**

Emmanuel Barillot, Lorens Kalzone,

Filipp Hupe, Jan-Filip Vert va

Andrey Zinovьев

**Bioinformatika uchun Python, Ikkinci nashr**

Sebastyan Bassi

**Miqdoriy biologiya: molekulyardan**

**Uyali tizimlar**

Sebastyan Bassi

**Tibbiy informatikaning usullari:**

**Sog'liqni saqlash asoslari**

**Perl, Python va Ruby tillarida dasturlash**

Jyul J. Berman

**Xromatin: tuzilishi, dinamikasi,**

**Reglament**

Ralf Blossey

**Hisoblash biologiyasi: statistik**

**Mekanika nuqtai nazari**

Ralf Blossi

**Biologiyada o'yin nazariy modellari**

Mark Broom va Yan Rychtář

Chimera Forbes J. Burkowski yordamida

**strukturaviy bioinformatika uchun hisoblash va vizualizatsiya usullari**

**Strukturaviy bioinformatika: Algoritmik yondashuv**

Forbes J. Burkowski

**Fazoviy ekologiya**

Stiven Kurrell, Kris Kozner va

Shigui Ruan

**Hujayra mexanikasi: bitta mashtabdan**

**Ko'p miqyosli modellashtirishga asoslangan modellar**

Arno Shovier, Luidji Preziosi va Klod

Verdier

**Bayes filogenetikasi: usullar, algoritmlar va**

**ilovalar** Ming-Xui Chen, Lin Kuo va Pol O. Lyuis

**QTL xaritalash uchun statistik usullar**

Zexua Chen

**Jismoniy onkologiyaga kirish:**

**Qanday mexanik matematik**

**Modellashtirish saraton kasalligini davolashni yaxshilashi mumkin**

**Natijalar**

Vittorio Cristini, Eugene J. Koay va

Zhihui Vang

**Oddiy rejim tahlili: nazariya va**

**Biologik va kimyoga qo'llanilishi**

**Tizimlar**

Qiang Cui va Ivett Bahar

**Tizim biologiyasida kinetik modellashtirish**

Oleg Demin va Igor Goryanin

**DNK mikroarraylari uchun ma'lumotlarni tahlil qilish vositalari**

Sorin Dragichi

**Statistika va ma'lumotlarni tahlil qilish**

**R va Bioconductor yordamida mikromassivlar,**

**Ikkinci nashr**

Sorin Dragichi ў

**Hisoblash nevrologiyasi:**

**Kompleks yondashuv**

Jianfeng Feng

**SeqAn C++ kutubxonasi yordamida biologik**

**ketma-ketlikni tahlil qilish**

Andreas Gogol-Döring va Knut Reynert

**Gen ifodasini o'rGANISHDAN foydalanish**

**Affymetrix mikromassivlari**

Hinrich Gohlmann va Villem Talloen

**Yashirin Markov modellari bo'yicha qo'llanma**

**Bioinformatika bo'yicha**

Martin Goleri

**Meta-tahlil va genetika va genomikada**

**ma'lumotlarni birlashtirish** Rudy Guerra va Darlen

R. Goldstein

**Differensial tenglamalar va matematik biologiya,**

**ikkinci nashr** DS Jones, MJ Plank va BD Sleeman

**Proteomikada bilimlarni kashf qilish**

Igor Jurisica va Dennis Wigle

**Proteinlarga kirish: tuzilishi,**

**Funktsiya va harakat**

Amit Kessel va Nir Ben-Tal

## Chop etilgan sarlavhalar (davomi)

<b>RNK-seq ma'lumotlarini tahlil qilish: amaliy Yondashuv</b> Eija Korpelainen, Jarno Tuimala, Panu Somervuo, Mikael Huss va Garri Vong	<b>Bio-ontologiyalarga kirish</b> Piter N. Robinson va Sebastyan Bauer
<b>Matematik onkologiyaga kirish</b> Yang Kuang, Jon D. Nagy va Steffen E. Eikenberry	<b>Biologik tizimlar dinamikasi</b> Maykl Kichkina
<b>Biologik hisoblash</b> Ehud Lamm va Ron Unger	<b>Genom izohi</b> Jung Soh, Pol MK Gordon va Kristof V. Sensen
<b>Biologik uchun qo'llaniladigan optimal nazorat Modellar</b> Suzanna Lenhart va Jon T. Workman	<b>Niche Modeling: dan bashoratlar</b> <b>Statistik taqsimotlar</b> Devid Stokvell
<b>Bioinformatika va giyohvand moddalarini kashf qilishda klasterlash</b> Jon D. Makkuish va Norah E. MakKuist	<b>Keyingi avlod uchun algoritmlar</b> <b>Ketma-ketlik</b> Wing-Kin Sung
<b>Ekologiya va epidemiologiyada fazoviy-vaqt namunalari: nazariya, modellar va simulyatsiya</b> Horst Malchow, Sergey V. Petrovskiy va Ezio Venturino	<b>Bioinformatikada algoritmlar: amaliy Kirish</b> Wing-Kin Sung
<b>Tizimlar uchun stokastik dinamika</b> <b>Biologiya</b> Kristian Mazza va Mishel Benaim	<b>Bioinformatikaga kirish</b> Anna Tramontano
<b>Molekulyar biologiya uchun statistik modellashtirish va mashinani o'rganish</b> Alan M. Moses	<b>Eng ko'p qidirilayotgan o'nta yechim</b> <b>Protein bioinformatikasi</b> Anna Tramontano
<b>Muhandislik genetik sxemalari</b> Chris J. Myers	<b>Kombinatoriy naqshlarni moslashtirish</b> <b>Hisoblash biologiyasida algoritmlar</b> Perl va R dan foydalanish Gabriel Valiente
<b>Bioinformatikada naqsh kashfiyoti:</b> <b>Nazariya va algoritmlar</b> Laxmi Parida	<b>Bioologik ma'lumotlaringizni boshqarish</b> <b>Python</b> Allegra Via, Kristian Roter va Anna Tramontano
<b>Biologik aniq echiladigan modellar</b> <b>Bosqin</b> Sergey V. Petrovskiy va Bai-Lian Li	<b>Saraton tizimlari biologiyasi</b> Edvin Vang
<b>Hisoblash gidrodinamika</b> <b>Kapsulalar va biologik hujayralar</b> C. Pozrikidis	<b>Tizimlar uchun stokastik modellashtirish</b> <b>Biologiya, ikkinchi nashr</b> Darren J. Uilkinson
<b>Kapsulalar va biologik hujayralarni modellashtirish va simulyatsiya qilish</b> C. Pozrikidis	<b>Bioinformatika uchun katta ma'lumotlarni tahlil qilish va Biomedikal kashfiyotlar</b> Shui Qing Ye
<b>Saratonni modellashtirish va simulyatsiya qilish</b> Luidji Preziosi	<b>Bioinformatika: amaliy yondashuv</b> Shui Qing Ye
	<b>Hisoblash faniga kirish</b> <b>Proteomika</b> Golan Yona

# PYTHON UCHUN BIOINFORMATIKA

IKKINCHI NASHRI

SEBASTIAN BASSI



CRC Press  
Taylor & Francis Group  
Boca Raton London New York

---

CRC Press is an imprint of the  
Taylor & Francis Group, an **informa** business  
A CHAPMAN & HALL BOOK

MATLAB® The MathWorks, Inc. kompaniyasining savdo belgisi bo'ylib, ruxsat bilan foydalaniladi. MathWorks ushbu kitobdag'i matn yoki mashqlarning to'jig'ligiga kafolat bermaydi. Ushbu kitobda MATLAB® dasturiy ta'minoti yoki tegishli mahsulotlardan foydalanish yoki muhokama qilish MathWorks tomonidan ma'lum bir pedagogik yondashuv yoki MATLAB® dasturiy ta'minotidan alohida foydalanishni ma'qullah yoki homiylik qilish hisoblanmaydi.

CRC matbuot

Taylor va Frencis guruhı

6000 Broken Sound Parkway NW, Suite 300

Boca Raton, FL 33487-2742

© 2018 Taylor & Francis Group, LLC tomonidan

CRC Press - bu Informa biznesi bo'lgan Taylor & Francis Group kompaniyasining izidir

AQSh hukumatining asl ishlariqa da'vo yo'q

Kisolatasiz qog'ozga bosilgan

Versiya sanasi: 20170626

Xalqaro standart kitob raqami-13: 978-1-1380-3526-3 (Hardback)

Ushbu kitobda haqiqiy va yuqori baholangan manbalardan olingan ma'lumotlar mavjud. Ishonchli ma'lumotlar va ma'lumotlarni nashr qilish uchun oqilona harakatlar qilingan, ammo muallif va noshir barcha materiallarning haqiqiyligi yoki ulardan foydalanish oqibatlari uchun javobgarlikni o'z zimmasiga olmaydi. Mualliflar va noshirlar ushbu nashrda ko'satilgan barcha materiallarning mualliflik huquqi egalarini aniqlashga harakat qilishdi va agar ushbu shaklda nashr etish uchun ruxsat olinmagan bo'lsa, mualliflik huquqi egalaridan uzr so'raydilar. Agar mualliflik huquqi bilan bog'liq materiallar tan olinmagan bo'lsa, iltimos, yozing va bizga xabar bering, shunda biz har qanday keyingi nashrda tuzatishimiz mumkin.

AQSh Mualliflik huquqi qonunida ruxsat etilgan hollar bundan mustasno, ushbu kitobning biron bir qismi hozirda ma'lum bo'lgan yoki bundan keyin ixtiro qilingan har qanday elektron, mexanik yoki boshqa vositalar bilan, jumladan, fotokopiya, mikrofilmga olish va yozib olish orqali qayta nashr etilishi, ko'paytirilishi, uzatilishi yoki har qanday shaklda ishlatalishi mumkin emas. yoki har qanday axborotni saqlash yoki qidirish tizimida nashriyotlarning yozma ruxsatsiz.

Ushbu asardan nusxa olish yoki materialdan elektron foydalanishga ruxsat olish uchun [www.copyright.com](http://www.copyright.com) saytiga kiring (<http://www.copyright.com/>) yoki Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400 bilan bog'laning. CCC notijorat tashkilot bo'lib, turli foydalanuvchilar uchun litsenziyalar va ro'yxatdan o'tishni ta'minlaydi. CCC tomonidan nusxa ko'chirish litsenziyasi berilgan tashkilotlar uchun alohida to'lov tizimi yo'lga qo'yilgan.

**Savdo belgisiga oid eslatma:** Mahsulot yoki korporativ nomlar tovar belgilari yoki ro'yixatdan o'ytgan tovar belgilari bo'ylishi mumkin va ular huquqbazarlik maqsadisiz faqat identifikasiya va tushuntirish uchun ishlataliladi.

---

#### Kongress kutubxonasi nashriyot ma'lumotlarini kataloglash

---

Ismilar: Bassi, Sebastyan, muallif.

Sarlavha: Bioinformatika uchun Python / Sebastyan Bassi.

Tavsiy: Ikkinci nashr. | Boca Raton: CRC Press, 2017. | Seriya: Chapman & Hall/CRC matematik va hisoblash biologiyasi | Bibliografik manbalar va indeksni o'z ichiga oladi.

Identifikatorlar: LCCN 2017014460 | ISBN 9781138035263 (pbk. : alk. qog'oz) | ISBN 9781138094376 (qattiq muqovali : alk. qog'oz) | ISBN 9781315268743 (elektron kitob) | ISBN 9781351976961 (elektron kitob) | ISBN 9781351976954 (elektron kitob) | ISBN 9781351976947 (elektron kitob)  
Mavzular: LCSH: Bioinformatika. | Python (Kompyuter dasturlari tili)  
Tasnifi: LCC QH324.2. B387 2017 | DDC 570.285--dc23 LC yozuvi <https://lccn.loc.gov/2017014460> manzilida mavjud

---

**Taylor & Frencis veb-saytiga tashrif buyuring:**

<http://www.taylorandfrancis.com>

va CRC Press veb-sayti <http://www.crcpress.com>

---

# Tarkib

---

Raqamlar ro'yxati	xvii
Jadvallar ro'yxati	xxi
Birinchi nashrga so'zboshi	xxiii
Ikkinci nashrga so'zboshi	xxv
Minnatdorchilik	xxix
<b>I bo'lim Dasturlash</b>	
<b>1 -bob Kirish</b>	<b>3</b>
<b>1.1 BU KITOBNI KIM O'QISHI KERAK</b>	<b>3</b>
<b>1.1.1 O'quvchi nimani bilishi kerak</b>	<b>4</b>
<b>1.2 USHBU KITOBdan FOYDALANISH</b>	<b>4</b>
<b>1.2.1 Matbaa qoidalari 1.2.2 Python</b>	<b>4</b>
<b>versiyalari 1.2.3 Kod uslubi 1.2.4 Bu</b>	<b>5</b>
<b>kitobdan hammasini o'qimay turib, undan</b>	<b>5</b>
<b>maksimal foyda oling</b>	<b>6</b>
<b>1.2.5 Ushbu kitob bilan bog'liq onlayn manbalar</b>	<b>7</b>
<b>1.3 NEGA DASTURLASHNI O'RGANISH KERAK?</b>	<b>7</b>
<b>1.4 ASOSIY DASTURLASH TUSHUNCHALARI</b>	<b>8</b>
<b>1.4.1 Dastur nima?</b>	<b>8</b>
<b>1.5 NEGA PYTHON?</b>	<b>10</b>
<b>1.5.1 Pythonning asosiy xususiyatlari</b>	<b>10</b>
<b>1.5.2 Pythonni boshqa tillar bilan solishtirish</b>	<b>11</b>
<b>1.5.3 U qanday ishlataladi?</b>	<b>14</b>
<b>1.5.4 Pythondan kim foydalanadi?</b>	<b>15</b>
<b>1.5.5 Python lazzatlari 1.5.6</b>	<b>15</b>
<b>Maxsus Python tarqatishlari</b>	<b>16</b>
<b>1.6 QO'SHIMCHA RESURSLAR</b>	<b>17</b>

## viii Tarkib

<b>2- bob Python bilan birinchi qadamlar</b>	<b>19</b>
<b>2.1 PYTHON O'R NATISH</b>	<b>20</b>
<b>2.1.1 Pythonni undan foydalanish orqali</b>	<b>20</b>
<b>2.1.2 o'rganining Python-ni mahalliy sifatida o'rnatning</b>	<b>20</b>
<b>2.1.3 Python Online-dan foydalanish</b>	<b>21</b>
<b>2.1.4 Python-ni sinovdan o'tkazish</b>	<b>22</b>
<b>2.1.5 Birinchi foydalanish</b>	<b>22</b>
<b>2.2 INTERFAOL REJIM</b>	<b>23</b>
<b>2.2.1 Chaqaloq</b>	<b>23</b>
<b>qadamlari 2.2.2 Asosiy kiritish va chiqish</b>	<b>23</b>
<b>2.2.3 Interaktiv rejim haqida batafsil</b>	<b>24</b>
<b>2.2.4 Matematik amallar 2.2.5 Python</b>	<b>26</b>
<b>qobig'idan chiqish</b>	<b>27</b>
<b>2.3 BATCH REJIM</b>	<b>27</b>
<b>2.3.1 Sharhlar</b>	<b>29</b>
<b>2.3.2 Chiziq</b>	<b>30</b>
<b>2.4 MUHARRIRINI TANLASH</b>	<b>32</b>
<b>2.4.1 Ajoyib matn</b>	<b>32</b>
<b>2.4.2 Atom</b>	<b>33</b>
<b>2.4.3 PyCharm 2.4.4</b>	<b>34</b>
<b>Spyder IDE</b>	<b>35</b>
<b>2.4.5 Tahrirlovchilar haqida yakuniy so'zlar</b>	<b>36</b>
<b>2.5 BOSHQA ASOBOTLAR</b>	<b>36</b>
<b>2.6 QO'SHIMCHA RESURSLAR</b>	<b>37</b>
<b>2.7 O'Z-O'ZINI BAHOLASH</b>	<b>37</b>
<b>3- bob Asosiy dasturlash: Ma'lumotlar turlari</b>	<b>39</b>
<b>3.1 STRINGS</b>	<b>40</b>
<b>3.1.1 Satrlar Unicode belgilar ketma-ketligidir 3.1.2 Stringni manipulyatsiya qilish 3.1.3 Satrlar bilan bog'liq usullar</b>	<b>41</b>
<b>42</b>	<b>42</b>
<b>3.2 RO'YXATLAR</b>	<b>44</b>
<b>3.2.1 Ro'yxat elementlari ro'yxatiga bir nechta takrorlanadigan elementlar bilan 3.2.3 kirish. Ro'yxatni tushunish 3.2.4 Ro'yxatlarni o'zgartirish</b>	<b>45</b>
<b>45</b>	<b>46</b>
<b>47</b>	<b>47</b>

<b>3.2.5 Ro'yxatni nusxalash</b>	<b>49</b>
<b>3.3</b>	<b>49</b>
<b>3.3.1 Kortejlar o'zgarmas ro'yxatlardir</b>	<b>49</b>
<b>3.4 KETTLIKLARNING UMUMIY XUSUSIYATLARI</b>	<b>51</b>
<b>3.5 LIG'ATLAR</b>	<b>54</b>
<b>3.5.1 Xaritalash: Har bir qiymatni nom bilan chaqirish 3.5.2</b>	<b>54</b>
<b>Lug'atlar bilan ishlash</b>	<b>56</b>
<b>3.6 TOPLAR</b>	<b>59</b>
<b>3.6.1 Ob'ektlarning tartibsiz to'plami 3.6.2</b>	<b>59</b>
<b>O'rnatish operatsiyalari 3.6.3 Boshqa ma'lumotlar</b>	<b>60</b>
<b>turlari bilan umumiyl operatsiyalar</b>	<b>62</b>
<b>3.6.4 O'zgarmas to'plam: Frozenset</b>	<b>63</b>
<b>3.7 OB'YEKTLARNI NOM BERISH</b>	<b>63</b>
<b>3.8 O'ZGARGANGA QIYMAT BERISH O'ZGARTIRISH O'ZGARTIRISH O'ZGARTIRISHGA QIYMAT BERISH, OB'YEKTGA NOMI BOG'LASH.</b>	<b>64</b>
<b>3.9 QO'SHIMCHA RESURSLAR</b>	<b>67</b>
<b>3.10 O'Z-O'ZI BAHOLASH</b>	<b>68</b>
<b>4 -bob Dasturlash: Oqimni boshqarish</b>	<b>69</b>
<b>4.1 AGAR-BOSHQA</b>	<b>69</b>
<b>4.1.1 O'tish bayonoti</b>	<b>74</b>
<b>4.2 FOR LOOP</b>	<b>75</b>
<b>4.3 WHILE LOOP</b>	<b>77</b>
<b>4.4 BUZISH: LOOPNI UZISH</b>	<b>78</b>
<b>4.5 UNNI QO'YISH</b>	<b>80</b>
<b>4.5.1 Proteinning sof zaryadini hisoblash 4.5.2 Past degeneratsiya zonasini qidirish</b>	<b>80</b>
<b>4.6 QO'SHIMCHA RESURSLAR</b>	<b>83</b>
<b>4.7 O'Z-O'ZI BAHOLASH</b>	<b>83</b>
<b>5 -bob Fayllar bilan ishlash</b>	<b>85</b>
<b>5.1 FAYLLARNI O'QISH</b>	<b>86</b>
<b>5.1.1 Fayllar bilan ishlash misoli</b>	<b>87</b>
<b>5.2 FAYLLARNI YOZISH</b>	<b>89</b>
<b>5.2.1 Fayllarni o'qish va yozishga misollar</b>	<b>90</b>
<b>5.3 CSV FAYLLARI</b>	<b>90</b>

x **Tarkib**

5.4 PICKLE: VARI MAZMUNINI SAQLASH VA QAYTA QILISH QOLIB	94
5.5 JSON FAYLLARI	96
5.6 FAYLLARNI ISHLATISH: OS, OS.PATH, SHUTIL VA PATH.PY MODULI	98
<b>5.6.1 path.py moduli</b>	<b>100</b>
<b>Bir nechta DNK ketma-ketliklarini bitta FASTA fayliga birlashtirish</b>	<b>102</b>
5.7 QO'SHIMCHA RESURSLAR	102
5.8 O'Z-O'ZINI BAHOLASH	103
<b>6 -bob Kodni modullashtirish</b>	<b>105</b>
6.1 KODLARNI MODULARLASHTIRISHGA KIRISH	105
6.2 FUNKSIYALAR	106
<b>6.2.1 Python kodini modulli qilishning standart usuli</b>	<b>106</b>
<b>Funktsiya parametrлari parametrлari</b>	<b>110</b>
<b>6.2.3 Generatorlar</b>	<b>113</b>
6.3 MODULLAR VA PAKETLAR	114
<b>6.3.1 Modullardan</b>	<b>115</b>
<b>foydalinish 6.3.2</b>	<b>116</b>
<b>6.3.3 Paketlar Uchinchi tomon modullarini o'rnatish</b>	<b>117</b>
<b>6.3.4 Virtualenv: Izolyatsiya qilingan Python muhitlari</b>	<b>119</b>
<b>6.3.5 Conda: Anaconda virtual muhit</b>	<b>121</b>
<b>6.3.6 Modullarni yaratish</b>	<b>124</b>
<b>6.3.7 Sinov modullari</b>	<b>125</b>
6.4 QO'SHIMCHA RESURSLAR	127
6.5 O'Z-O'ZINI BAHOLASH	128
<b>7 -bob Xatolarni hal qilish</b>	<b>129</b>
7.1 XATOLARNI QO'LLANISHGA KIRISH	129
<b>7.1.1 Urinish va istisno</b>	<b>131</b>
<b>7.1.2 Istisno turlari</b>	<b>134</b>
<b>Istisnolarni ishga tushirish</b>	<b>135</b>
7.2 SOZLANGAN ISTISOZLARNI YARATISH	136
7.3 QO'SHIMCHA RESURSLAR	137
7.4 O'Z-O'ZI BAHOLASH	138
<b>8- bob Ob'ektga yo'naltirilgan dasturlashga kirish (OOP)</b>	<b>139</b>
8.1 OBYEKT PARADIGMASI VA PYTHON	139

8.2 JARGONNI O'rganish	140
8.3 SINFLAR TUZISH	142
8.4 MEROS	145
8.5 MAXSUS USULLAR	149
8.5.1 O'rnatilgan ma'lumotlar turidan foydalanib yangi ma'lumotlar turini yaratish	154
8.6 KODIMIZNI XUSUSIY QILISH	154
8.7 QO'SHIMCHA RESURSLAR	155
8.8 O'Z-O'ZINI BAHOLASH	156
<b><u>9 -bob Biopythonga kirish</u></b>	<b><u>157</u></b>
9.1 BIOPITON NIMA?	158
9.1.1 Loyihani tashkil etish	158
9.2 BIOPYTHONNI O'R NATISH	159
9.3 BIOPITON KOMPONENTLARI	162
9.3.1 Alifbo	162
9.3.2 Seq	163
9.3.3 MutableSeq 9.3.4	165
SeqRecord 9.3.5 Align	166
9.3.6 AlignIO	167
	169
9.3.7 ClustalW	171
9.3.8 SeqIO	173
9.3.9 AlignIO	176
9.3.10 PORTLASH	177
9.3.11 Biologik bog'liq ma'lumotlar	187
9.3.12 Entrez	190
9.3.13 PDB	194
9.3.14 PROZIT	196
9.3.15 Cheklash	197
9.3.16 SeqUtils	200
9.3.17 Sequencing	202
9.3.18 SwissProt	205
9.4 XULOSA	207
9.5 QO'SHIMCHA RESURSLAR	207
9.6 O'Z-O'ZINI BAHOLASH	209

## xii Tarkib

**II bo'lim Ilg'or mavzular**

<b>10- bob Veb-ilovalar</b>	<b>213</b>
10.1 WEBDA PYTHONGA KIRISH	213
PYTHONDA 10.2 CGI	214
<b>10.2.1 Veb-serverni CGI uchun sozlash 10.2.2</b>	<b>215</b>
<b>Proteinning aniq zaryadini hisoblash uchun</b>	<b>215</b>
<b>10.2.3 Veb-dasturimiz bilan serverni sinovdan o'tkazish (CGI versiyasi)</b>	<b>219</b>
10.3 WSGI	221
<b>10.3.1 Shisha: WSGI uchun Python veb-ramkasi 10.3.2</b>	<b>222</b>
<b>Shishani o'rnatish 10.3.3 Minimal shisha ilovasi</b>	<b>223</b>
<b>10.3.4 Shisha komponentlari</b>	<b>224</b>
<b>10.3.5 Proteinning aniq zaryadini hisoblash uchun veb-dastur (shisha versiyasi)</b>	<b>229</b>
<b>10.3.6 Apache da WSGI dasturini o'rnatish</b>	<b>232</b>
10.4 PYTONGA ASOSLANGAN DINAMIKA QILISH UCHUN ALTERNATIV VARITALAR	232
WEB SAYTLAR	232
10.5 SCKRIPT XAVFSIZLIGI HAQIDA BA'ZI SO'ZLAR	232
10.6 PYTHON DASTURLARINI QAYERDA HOZLASH MUMKIN	234
10.7 QO'SHIMCHA RESURSLAR	235
10.8 O'Z-O'ZINI BAHOLASH	236
<b>11 -bob XML</b>	<b>237</b>
11.1 XML TILIGA KIRISH	237
11.2 XML HUJJATINING TUZILISHI	241
11.3 XML HUJJATDAGI MA'LUMOTLARGA KIRISH USULLARI	246
<b>11.3.1 SAX: cElementTree Iterparse</b>	<b>246</b>
11.4 XULOSA	251
11.5 QO'SHIMCHA RESURSLAR	252
11.6 O'Z-O'ZINI BAHOLASH	252
<b>12- bob Python va ma'lumotlar bazalari</b>	<b>255</b>
12.1 MA'LUMOT BAZALARIGA KIRISH	256
<b>12.1.1 Ma'lumotlar bazasini boshqarish:</b>	<b>257</b>
<b>RDBMS 12.1.2 Relyatsion ma'lumotlar bazasi komponentlari</b>	<b>258</b>

12.1.3 Ma'lumotlar bazasi ma'lumotlar turlari	260
12.2 MA'LUMOTLAR BAZAGA ULANISH	261
12.3 MYSQL MA'LUMOTLAR BAZASINI YARATISH	262
<b>12.3.1 Jadvallarni yaratish</b>	<b>263</b>
<b>12.3.2 Jadvalni yuklash</b>	<b>264</b>
12.4 REJAJLASH	266
<b>12.4.1 PythonU: Ma'lumotlar bazasi namunasi</b>	<b>266</b>
12.5 TANLASH: MA'LUMOTLAR BAZASINI SO'ROV QILISH	269
<b>12.5.1 So'rovni yaratish 12.5.2</b>	<b>271</b>
<b>Ma'lumotlar bazasini yangilash</b>	<b>273</b>
<b>12.5.3 Ma'lumotlar bazasidan yozuvni o'chirish</b>	<b>273</b>
12.6 PYTHONDAN MA'LUMOTLAR BAZAGA KIRISH	274
<b>12.6.1 PyMySQL moduli 12.6.2</b>	<b>274</b>
<b>Ulanishni o'rnatish 12.6.3 Python'dan so'rovni bajarish</b>	<b>274</b>
12.7 SQLITE	276
12.8 NOSQL MA'LUMOT BAZALARI: MONGODB	278
<b>12.8.1 MongoDB dan PyMongo bilan foydalanish</b>	<b>278</b>
12.9 QO'SHIMCHA RESURSLAR	282
12.10 O'Z-O'ZINI BAHOLASH	284
<b>13- bob Oddiy iboralar</b>	<b>285</b>
13.1 Muntazam iboralarga kirish (REGEX)	285
<b>13.1.1 REGEX sintaksisi</b>	<b>286</b>
13.2 RE MODULI	287
<b>13.2.1 Shaklni tuzish 13.2.2 REGEX misollari 13.2.3 Shaklni almashtirish</b>	<b>290</b>
	292
	294
13.3 BIOINFORMATIKADA REGEX	294
<b>13.3.1 Ketma-ketlikni tozalash</b>	<b>296</b>
13.4 QO'SHIMCHA RESURSLAR	297
13.5 O'Z-O'ZINI BAHOLASH	298
<b>14- bob Pythonda grafika</b>	<b>299</b>
14.1 BOKEHGA KIRISH	299
14.2 BOKEH O'R NATISH	299
14.3 BOKEH FOYDALANISH	301

## xiv Tarkib

14.3.1 Oddiy XY syujeti	303
14.3.2 Ikki ma'lumotlar seriyasining syujeti	304
14.3.3 Tarqalish sxemasi	306
14.3.4 Issiqlik xaritasi	308
14.3.5 Akkord diagrammasi	309
<b>III bo'lim Python retseptlari sharhlangan manba kodi bilan</b>	
<b>15- bob To'plamdag'i ketma-ketlikni manipulyatsiya qilish</b>	<b>315</b>
15.1 MUAMMOLARNING TA'RIFI	315
15.2 BIRINCHI MAMUL: RANDOM SE BILAN TEZ FAYL YARATING QUUENCES	315
<b>15.2.1 Sharhlangan manba kodi</b>	<b>315</b>
15.3 Ikkinci muammo: FASTA faylidan bo'sh EMAS ketma-ketliklarni filtrlang	316
<b>15.3.1 Sharhlangan manba kodi</b>	<b>317</b>
15.4 UCHINCHI MAMUL: FASTA FAYLNING HAR YOZISHINI O'zgartirish <b>15.4.1 Sharhlangan manba kodi</b>	<b>319</b>
<b>16- bob Vektorli ifloslanishni filtrlash uchun veb-ilova</b>	<b>321</b>
16.1 MUAMMOLARNING TA'RIFI	321
<b>16.1.1 Sharhlangan manba kodi</b>	<b>322</b>
16.2 QO'SHIMCHA RESURSLAR	326
<b>17- bob Primer3 yordamida PCR primerlarini qidirish</b>	<b>329</b>
17.1 MUAMMOLARNING TA'RIFI	329
17.2 O'ZG'ARCHI UZUNLIK HUDDATIGA BO'LGAN PRIMER DIZAYNI <b>17.2.1 Sharhlangan manba kodi</b>	<b>330</b>
17.3 O'ZG'ARCHI UZUNLIK MINTAQINDAGI PRIMER DIZAYNI, BIOPITON BILAN	331
17.4 QO'SHIMCHA RESURSLAR	332
<b>18- bob Primerlar to'plamidan erish haroratini hisoblash</b>	<b>335</b>
18.1 MUAMMOLARNING TA'RIFI	335
<b>18.1.1 Sharhlangan manba kodi</b>	<b>336</b>
18.2 QO'SHIMCHA RESURSLAR	336
<b>19- bob GenBank faylidan maxsus maydonlarni filtrlash</b>	<b>339</b>
19.1 TANLANGAN PROTEINLARNI EKSTRAKTALASH	339

<b>19.1.1 Sharhlangan manba kodi</b>	<b>339</b>
19.2 TANLANGAN PRONING YUQORI REGIONINI OLISH TEINS	340
<b>19.2.1 Sharhlangan manba kodi</b>	<b>340</b>
19.3 QO'SHIMCHA RESURSLAR	341
<b>20 -bob Qo'shilish joylari haqida xulosa chiqarish</b>	<b>343</b>
20.1 MUAMMOLARNING TA'RIFI	343
<b>20.1.1 Sharhlangan manba kodi bilan birlashma saytlarini xulosa qilish</b>	<b>345</b>
<b>20.1.2 Tahmini Intron dasturining namunaviy ishlashi</b>	<b>347</b>
<b>21- bob Bir nechta tekislash uchun veb-server</b>	<b>349</b>
21.1 MUAMMOLARNING TA'RIFI	349
<b>21.1.1 Veb-interfeys: Front-End. HTML kodi</b>	<b>349</b>
<b>21.1.2 Veb-interfeys: Server tomonidagi skript. Sharhlangan manba kodi</b> <b>351</b>	<b>351</b>
21.2 QO'SHIMCHA RESURSLAR	353
<b>22- bob Ma'lumotlar bazasida saqlangan ma'lumotlardan foydalangan holda marker pozitsiyalarini chizish</b> <b>355</b>	<b>355</b>
22.1 MUAMMOLARNING TA'RIFI	355
<b>22.1.1 Ma'lumotlar bo'yicha dastlabki ish</b> <b>22.1.2</b>	<b>355</b>
<b>MongoDB versiyasi sharhlangan manba kodi bilan</b>	<b>358</b>

## IV bo'lim Ilovalar

Ilova A hamkorlikda ishlab chiqish: GitHub [365](#) bilan versiyani boshqarish

A.1 VERSIYALARNI BOSHQARISHGA KIRISH	366
A.2 VERSIYA KODINGIZNI	367
A.3 KODINGIZNI BERING	375
A.4 BOSHQA LOYIHALARGA HISSA QO'SHING	381
A.5 XULOSA	382
A.6 USULLAR	384
A.7 QO'SHIMCHA RESURSLAR	384

Ilova B PythonAnywhere-da shisha ilovasini o'rnatning

B.1 PyTHONAN QAYERDA	385
<b>B.1.1 PythonAnywhere nima</b> <b>B.1.2</b>	<b>385</b>
<b>PythonAnywhere-da veb-ilovani o'rnatish</b>	<b>385</b>

xvi      **Tarkib**

---

Ilova C Ilmiy Python Cheat Sheet	393
C.1 SOZ PYTHON	394
C.2 VIRTUALENV	400
C.3 CONDA	402
C.4 IPYTHON	403
C.5 NUMPY	405
C.6 MATPLOTLIB	410
C.7 SCIPY	412
C.8 PANDAS	413
Indeks	417

---

---

# Raqamlar ro'yxati

---

2.1 Anaconda macOS-da o'rnatiladi.	21
2.2 Anaconda Python interaktiv terminali.	23
2.3 PyCharm Edu xush kelibsiz ekranı.	35
3.1 Chorraha.	60
3.2 Birlashma.	61
3.3 Farq.	61
3.4 Simmetrik farq.	62
3.5 1-holat.	65
3.6 2-holat.	66
5.1 Excel formatlangan elektron jadval sampledata.xlsx.	93
8.1 IUPAC nuklein kislotasi belgilari jadvali.	147
9.1 BLAST natijasining anatomiysi.	181
10.1 Bizning birinchi CGI.	216
10.2 CGI veb-server o'rniiga mahalliy diskdan foydalанилди.	217
10.3 greeting.html: Juda oddiy shakl.	217
10.4 greeting.html ni qayta ishlovchi CGI dasturining chiqishi.	218
10.5 Protcharge.html formasi topshirishga tayyor.	220
10.6 Net to'lov CGI natijasi.	222
10.7 Brauzerda ko'rinish turganidek, Shishada yaratilgan Salom Dunyo dasturi.	224
10.8 Proteinning sof zaryadini hisoblash uchun veb-ilova uchun shakl.	229
11.1 XML ko'rish dasturining skrinshoti.	244
11.2 Codebeautify, vebga asoslangan XML ko'rish dasturi.	245
12.1 PhpMyAdmin skrinshoti.	258
12.2 phpMyAdmin yordamida yangi ma'lumotlar bazasini yaratish.	262
12.3 phpMyAdmin yordamida yangi jadval yaratish.	264

12.4 Talabalar jadvalining ko'rinishi.	266
12.5 Qasddan noto'g'ri "Baholar" jadvali.	267
12.6 Yaxshiroq "Baholar" jadvali.	267
12.7 Kurslar jadvali: Qidiruv jadvali.	268
12.8 O'zgartirilgan "Baholar" jadvali.	268
12.9 SQLite menejerining skrinshoti.	277
12.10 MongoDB bulut provayderidan ko'rish.	281
14.1 Bokeh bilan aylana.	302
14.2 Bokeh bilan to'rtta doira.	303
14.3 Bokeh bilan oddiy syujet.	305
14.4 Bokeh bilan ikkita ma'lumot seriyali syujeti.	306
14.5 Tarqalgan chizma grafikasi.	308
14.6 Mikroarray tajribasidan issiqlik xaritasi.	310
14.7 Akkord diagrammasi.	312
16.1 Ketma-ket filtrlash uchun HTML formasi.	327
16.2 Ketma-ket filtrlash uchun HTML formasi.	328
21.1 Muscle Web interfeysi.	350
22.1 Demo ma'lumotlar to'plamidan (NODBDEMO) foydalangan holda 22.2 Listing mahsuloti.	356
A.1 Git qo'shish/commit jarayoni.	369
A.2 Mahalliy ombor bilan ishlash.	370
A.3 Yagona foydalanuvchi sifatida mahalliy va masofaviy ombor bilan ishlash. 379 A.4 Ochiq kodli loyihalarga hissa qo'shish.	383
B.1 "Konsollar" yorlig'i.	386
B.2 "Veb" yorlig'i.	386
B.3 Domen turini yangilash opsiyasi.	387
B.4 Veb ramka ekranini tanlang, Shishani tanlang.	388
B.5 Python va Bottle versiyasini tanlang.	389
B.6 Veb-ilovaning yo'lini kiritish uchun shakl.	390
B.7 Namuna veb-ilovasi foydalanishga tayyor.	390
B.8 "Fayl" yorlig'i.	391
B.9 PythonAnywhere-da yangi katalog yaratish uchun forma.	391
B.10 Hisob qaydnomangizga fayllarni ko'rish va yuklash.	391

**B.11 PythonAnywhere-da joylashtirilgan Shisha yordamida oqsil zaryadini hisoblash uchun dasturning asosiy qismi.**

**392**



**Taylor & Francis**  
Taylor & Francis Group  
<http://taylorandfrancis.com>

---

# Jadvallar ro'yxati

---

<b>2.1 Arifmetik uslubdagi operatorlar</b>	<b>26</b>
<b>3.1 Umumiy ro'yxat operatsiyalari</b>	<b>48</b>
<b>3.2 Lug'atlar bilan bog'liq usullar</b>	<b>58</b>
<b>9.1 Ketma-ketlik va hizalama formatlari 9.2</b>	<b>175</b>
<b>Blast dasturlari</b>	<b>178</b>
<b>9.3 eUtils</b>	<b>191</b>
<b>10.1 Veb-ishlab chiqish uchun ramkalar</b>	<b>233</b>
<b>12.1 Python universiteti talabalari 12.2</b>	<b>259</b>
<b>Asosiy kalitli jadval 12.3 MySQL</b>	<b>260</b>
<b>maýlumotlar turlari</b>	<b>261</b>
<b>13.1 REGEX maxsus ketma-ketliklari</b>	<b>287</b>
<b>A.1 Resurslar</b>	<b>367</b>



**Taylor & Francis**  
Taylor & Francis Group  
<http://taylorandfrancis.com>

# Birinchi nashrga so'zboshi

---

Ushbu kitob qishloq xo'jaligi biotexnologiyasi kompaniyasida bir necha yil davomida to'plangan tajriba natijasidir. Genomik ma'lumotlar bazasi kuratori sifatida men bioinformatika bo'yicha keng ko'lamli ehtiyojlarga ega bo'lgan xodimlarga yordam berdim. Ulardan ba'zilari qo'lda bajarayotgan protsedurani avtomatlashtirishni xohlashdi, boshqalari esa biologik muammolar bilan kelib, bioinformatika yechimlari bor-yo'qligini so'rashdi. Aksariyat hollarda umumiy narsa bor edi: dasturlash bilimi muammoga yechim topish uchun zarur edi. Ushbu kitobning asosiy maqsadi biologik muammolarini hal qilmoqchi bo'lgan olimlarga dasturlash asoslarini tushunishga yordam berishdir. Shu maqsadda men har qanday dasturlash bilan bog'liq tushunchalarни qabul qilishdan qochishga harakat qildim. Ushbu vazifa uchun tanlangan til Python.

Python - o'rganish oson bo'lgan kompyuter tili bo'lib, olimlar orasida qiziqish uyg'otmoqda. Buning sababi, uni ishlatalish oson, lekin ko'pchilik dasturlash maqsadlariga erishish uchun etarlicha kuchli. Python yordamida o'quvchi haqiqiy dasturlashni juda tez boshlashi mumkin. Fan va muhandislikda hisoblash, Bioinformatikada brifinglar va PLOS Hisoblash biologiyasi kabi jurnallarda Python haqida kirish maqolalari chop etilgan. Olimlar Python-dan molekulyar vizualizatsiya, genomik annotatsiya, ma'lumotlarni manipulyatsiya qilish va boshqa son-sanoqsiz ilovalar uchun foydalanmoqda.

Hayotiy fanlarning alohida misolda Pythonning rivojlanishi juda muhim edi; eng yaxshi ko'rsatkich Biopython to'plamidir. Shu sababli, II bo'lim Biopythonga bag'ishlangan. Qanday bo'lmasin, men Biopython dunyodagi har bir biologiya muammosining echimi deb da'vo qilmayman. Ba'zan oddiygina tayyorlangan yechim mavjud muammoga yaxshiroq mos kelishi mumkin. BioNEB va CoreBio kabi boshqa paketlar mavjud, ularni o'quvchi sinab ko'rishi mumkin.

Kitob birinchi bo'limgan ("Dasturlash") o'quvchiga dasturlash tamoyillarini o'rgatishdan boshlanadi. Men boshidanor amaliyatga alohida e'tibor qarataman, chunki men dasturlashni bajarish orqali eng yaxshi o'rganiladigan narsa deb hisoblayman. Shuning uchun kitob bo'ylab tarqalgan kod bo'laklari mavjud. O'quvchi ular bilan tajriba o'tkazishi va ularni ichkilashtirishga harakat qilishi kutilmoqda. Boshqa tillar bilan ba'zi zaxira taqqoslashlar ham mavjud; ular joriy mavzuni yoritgandagina kiritiladi. Menimcha, ko'pchilik tilni taqqoslash yangi tilni o'rgatishda foydadan ko'ra ko'proq zarar keltiradi. Ular ko'pchilik o'quvchilar uchun tushunarsiz va ahamiyatsiz bo'lgan ma'lumotlarni taqdim etadilar.

O'quvchining qiziqishini saqlab qolish uchun ko'pchilik misollar qandaydir tarzda biologiya bilan bog'liq. Shunga qaramay, o'quvchi ushbu sohada aniq bilimga ega bo'lmasa ham, bu misollarga amal qilish mumkin.

Ushbu kitobning amaliy mohiyatini mustahkamlash, shuningdek, ma'lumotnomha sifatida foydalanish

material, IV bo'lim "Izohlangan manba kodi bilan Python retseptlari" deb nomlanadi. Ushbu dasturlardan avvalgidek foydalanish mumkin, ammo boshqa loyihibar uchun asos sifatida foydalanish mo'ljallangan. O'quvchilar ba'zi misollar juda oddiy ekanligini tushunishlari mumkin; ular o'z ishlarini juda ko'p qo'ng'iroqlar va hushtaklarsiz bajaradilar. Bu qasddan. Buning asosiy sababi, o'quvchini keraksiz xususiyatlar bilan chalg'itmasdan, ilovaning ma'lum bir jihatini ko'rsatish, shuningdek, murakkab dasturlar bilan o'quvchini xafa qilmaslikdir. Asosiy ma'lumotlar o'rganilgandan so'ng, har doim xususiyatlar va sozlashlarni qo'shish uchun vaqt bo'ladi.

III boylim ("Kengaytirilgan mavzular") sarlavhasi qo'yrqinchli ko'yrinishi mumkin, ammo bu holda ilg'yor degani har doim ham qiyin degani emas. Oxir-oqibat, hamma ushbu bo'limdagi bo'limlardan foydalanadi [ayniqsa, relyatsion ma'lumotlar bazasini boshqarish tizimi - RDBMS - va XML]. Bioinformatika ishining muhim qismi ma'lumotlar bazalarini yaratish va so'rashdir, shuning uchun men MySQL kabi RDBMSni bilish bioinformatika mahoratining tegishli qismi deb hisoblayman. Turli manbalardan olingen ma'lumotlarni integratsiyalash bioinformatikada eng ko'p bajariladigan vazifalardan biridir. Bu vazifa uchun tanlangan vosita XML hisoblanadi. Ushbu standart ilovalar o'rtasida ma'lumotlar almashinuvni uchun keng qo'llaniladigan platformaga aylanmoqda. Pythonda bir nechta XML parserlari mavjud va biz ularning aksariyatini ushbu kitobda tushuntirami.

Ilova B, "Tanlangan maqolalar" Python bo'yicha kirish darajasidagi hujjatlarni taqdim etadi. Mavzular bir-biriga o'xshash bo'lsa-da, bu bir xil mavzu bo'yicha bir nechta nuqtai nazarni ko'rsatish uchun qilingan.

Bu kitob foydali bo'ladigan yagona tadqiqotchilar emas. Bundan tashqari, universitet darsligi sifatida foydalanish uchun tuzilgan. Talabalar undan dasturlash darslarida, ayniqsa bioinformatikaning yangi yo'nalishlarida foydalanishlari mumkin.

---

# Ikkinchisiga so'zboshi

## Nashr

---

Bioinformatika uchun Python ning birinchi nashri 2008 yilda yozilgan va 2009 yilda nashr etilgan. Hatto sakkiz yil o'tgan bo'lsa ham, bu kitobdagi saboqlar hali ham qimmatlidir. Bu shunday tez sur'atlar bilan rivojlanayotgan sohada juda yutuq. Foydali bo'lishiga qaramay, kitob o'zining yoshini ko'rsatmoqda va ikkinchi nashrdan katta foya keltiradi.

Python-ning asosiy versiyasi 3.6, ammo Python 2.7 hali ham ishlab chiqarish tizimlarida qo'llaniladi. Ushbu versiyalar o'rtasida nomuvofiqliklar mavjud bo'lganligi sababli, Python 3 kitobidagi barcha kodlarni mos qilish uchun ko'p harakat qilindi.

So'nggi sakkiz yil ichida nafaqat dasturiy ta'minot o'zgardi, balki umuman ochiq kodli dasturiy ta'minotga va ayniqsa Pythonga nisbatan korporativ munosabat va qo'llab-quvvatlash keskin o'zgardi. Bundan tashqari, hamkorlikda ishlab chiqish va bulutli hisoblash kabi e'tibordan chetda qoldirib bo'lmaydigan yangi hisoblash paradigmalari mavjud.

Asl kitobda **14-bob** "Hamkorlikda ishlab chiqish: Versiyani boshqarish" deb nomlangan va hozirda foydalanilayotgan taqsimlangan ishlab chiqish ish oqimiga amal qiluvchi, ammo bugungi kunda ko'ypchilik ishlab chiquvchilar tomonidan qo'llanilmaydigan Bazaar dasturiga asoslangan. Hozirgacha eng ko'p dasturiy ta'minot ishlab chiqish GitHub'da Git bilan amalga oshiriladi. Ushbu bob joriy amaliyotlarga e'tibor qaratish uchun qayta yozildi.

Veb-ishlab chiqish sezilarli darajada o'zgargan yana bir sohadir. Bu veb-ishlab chiqish kitobi bo'lmasa-da, "Veb-i-lovalar" bo'limi endi CGI/WSGI va o'rta dasturga asoslangan ilovalar o'rniqa uzoq davom etadigan jarayonlar va ramkalardan joriy foydalanishni aks ettiradi. Ushbu bobda ramkalar yon eslatma sifatida muhokama qilindi, ammo endi bob ramka (shisha) atrofida asoslanadi va eski usulni tarixiy izoh sifatida qoldiring.

Ma'lumotlar bazalarida NoSQL birinchi nashrda o'q nuqtasi bo'lgani uchun juda ko'p tortishuvlarga erishdi, endi MongoDB-dan foydalangan holda o'z bo'limiga ega va Python retsepti ushbu NoSQL ma'lumotlar bazasidan foydalanish uchun o'zgartirildi.

Grafik kutubxonalar 2009 yildan beri yaxshilandi va Python uchun ajoyib sifatli raqobatbardosh grafik kutubxonalar mavjud. Bokehga bag'ishlangan to'liq bob mavjud, bepul interaktiv vizualizatsiya kutubxonasi.

Ushbu kitobda aks ettirilgan yana bir o'zgarish - Anaconda va Jupyter noutbuklaridan foydalanish (barcha kodlar Microsoft Azure<sup>1</sup> tomonidan taqdim etilgan bulut daftaridagi ).

---

<sup>1</sup><https://notebooks.azure.com/py4bio/libraries/py3.us> ga qarang

Manba kodiga kelsak, <https://github.com/Serulab/Py4Bio> manzilida GitHub ombori mavjud. bu erda siz ushbu kitobda ishlataladigan barcha kod va namuna fayllarini yuklab olishingiz mumkin.

Har bir bobda tuzatishlar mavjud. Ba'zida haqiqiy xatolar bo'lgan, ammo tuzatishlarning aksariyati Python 3-ni yangilash bilan bog'liq va hozirgi yaxshi amaliyotlarga muvofiq edi. Tuzatishlarga kelsak, men ushbu kitobga tuzatishlar kerak bo'lishi mumkin deb umid qilaman, shuning uchun men o'quvchilar yangilanishlarni olishlari mumkin bo'lgan veb-sahifa yaratdim. Iltimos, <http://py3.us> saytiga qarang va past hajmli pochta ro'yxatiga obuna bo'ling.

Dasturiy ta'minot evolyutsiyasi va paradigmalarning o'zgarishi bilan bir qatorda, men rivojlanish tajribasiga ega bo'ldim va pedagogik masalalar bo'yicha qarashlarimni o'zgartirdim. Bu yillar davomida men xalqaro konsorsiumda genom sekvensiyasi loyihasida va NYSE listing kompaniyasida (Globant) katta dasturiy ta'minot ishlab chiqaruvchisi sifatida ishladim. So'nggi 5 yil ichida men Salesforce va National Geographic kabi bir qancha taniqli mijozlarda ishladim. Men hozirda PLOS (Public Library of Science) da ishlayapman.

MATLAB so'roviga ko'rya, men ularning aloqa majlumotlarini kiritaman: MATLAB ® The MathWorks, Inc kompaniyasining ro'yixatdan oytgan savdo belgisidir. Mahsulot haqida majlumot olish uchun quyidagi manzilga murojaat qiling: The MathWorks, Inc. 3 Apple Hill Drive Natick, MA, 01760-2098 AQSh Tel: 508- 647-7000 Faks: 508-647-7001 E-mail: [info@mathworks.com](mailto:info@mathworks.com) Veb-sayt: [www.mathworks.com](http://www.mathworks.com)

Muqovada ishlataladigan Biopython logotipiga kelsak, u foydalanish litsenziyasi (bu barcha Biopython fayllarini, shu jumladan uning logotipini ham qamrab oladi): Biopython hozirda "Biopython litsenziya shartnomasi" doirasida chiqarilgan (quyida to'liq keltirilgan). Agar alohida fayl sarlavhalarida boshqacha ko'rsatilmagan bo'lsa, barcha Biopython fayllari "Biopython litsenziya shartnomasi" ostida.

Ba'zi fayllar siz tanlagan "Biopython Litsenziya shartnomasi" yoki "BSD 3-band litsenziyasi" (ikkalasi ham quyida to'liq keltirilgan) bo'yicha ikki tomonlama litsenziyalangan. Bu keyinchalik ushbu ikki tomonlama litsenziyalash yondashuvi ostida barcha Biopython-ni taklif qilish niyatida.

## Biopython litsenziya shartnomasi

Ushbu dasturiy ta'minotni va uning hujjatlarini o'zgartirishlarsiz yoki o'zgartirishlarsiz, har qanday maqsadda va to'lovsiz ishlatalish, nusxalash, o'zgartirish va tarqatish uchun ruxsat beriladi, agar mualliflik huquqi to'g'risidagi har qanday eslatma barcha nusxalarda ko'rsatilsa va mualliflik huquqi to'g'risidagi bildirishnomalar ham, ushbu ruxsatnomaga haqida eslatma ham mavjud bo'lsa. qo'llab-quvvatlovchi hujjatlarda ko'rsatilishi va hissa qo'shuvchilar yoki mualliflik huquqi egalarining ismlaridan oldindan ruxsatisiz dasturiy ta'minotni tarqatish bilan bog'liq reklama yoki reklamada foydalanilmasligi kerak.

Ushbu yumshoq sharoitning hissa qo'shuvchilar va mualliflik huquqi egalari, shu jumladan savdogarlik va fitness bo'yicha barcha kafolatlarni, shu jumladan savdogarlar va fitnaviy kafolatlar, hech qachon biron bir maxsus, bilvosita yoki kompensial shikastlanishlar uchun javobgar bo'lmaydi yoki HAR QANDAY ZARARLAR NATIJASI

**USHBU DASTURNI FOYDALANISH YOKI ISHLAB CHIQISH BILAN YOKI BILAN YOKI BO'LGAN SHARTNOMA HARAKAT, EHBARATLIK YOKI BOSHQA ZARARLI HARAKATLarda FOYDALANISH, MA'LUMOT YOKI FOYDALANISHNI YO'qotilishidan.**

### **BSD 3-band litsenziyasi**

**Mualliflik huquqi (c) 1999-2017, The Biopython Contributors Barcha huquqlar himoyalangan.**

**Qayta taqsimlash va manba va ikkilik shakllarda o'zgartirish yoki o'zgartirishsiz foydalanishga quyidagi shartlar bajarilgan taqdirda ruxsat beriladi:**

**Manba kodini qayta tarqatishda yuqoridagi mualliflik huquqiga oid eslatma, ushbu shartlar rojyxati va quyidagi rad javobi boylishi kerak. Ikkilik shakldagi qayta tarqatishlar yuqoridagi mualliflik huquqi eslatmasini, ushbu shartlar ro'yxatini va tarqatish bilan birga taqdim etilgan hujjalarni va yoki boshqa materiallardagi quyidagi rad etishni takrorlashi kerak. Mualliflik huquqi egasining nomi va uning hissa qo'shuvchilarining ismlaridan oldindan yozma ruxsatsiz ushbu dasturiy ta'minotdan olingan mahsulotlarni ma'qullash yoki targ'ib qilish uchun foydalanish mumkin emas. USHBU DASTURIY HUQUQ ENGLIKLARI VA HISSA QILISHLARI TARAFINDAN "XAMDA" VA HAR QANDAY OG'ROQ YOKI KO'RSATILGAN KAFOLATLAR TOMONIDAN TAQDIM ETILGAN. HECH HOLDA NUSHORA HUQUQ SOG'LIGI YOKI HISSA QO'SHISHLARI HAR QANDAY TO'G'OVVOZ, INDI REKST, TASOSODIY, MAXSUS, NURUN YOKI NATIBAT TO'G'ONLAR UCHUN (JUMLADAN, BIR QO'YIB BO'LGAN, CHEK BO'LGAN BO'LGANLAR; , YOKI FOYDA; YOKI TADBIRKORLIKNING UZISHI) SHARTNOMADA BO'LGAN VA HAR QANDAY JAVOBGARLIK NAZARIYASI BO'YICHA, ISHLAB CHIQARISH BO'YICHA ETILGAN BO'LGAN BO'LGAN BO'LGAN BO'LGAN BO'LGAN QAT'iy MA'BARAT YOKI SHU BO'LGAN SHU BO'LGAN SHART BUNDAY ZARAR BERISH MUMKINLIGI.**



**Taylor & Francis**  
Taylor & Francis Group  
<http://taylorandfrancis.com>

---

# Minnatdorchilik

---

Bu kitob kabi loyihani faqat bir kishi bajara olmaydi. Shu sababli, mening minnatdorchiligidagi loyiq bo'lgan odamlarning uzoq ro'yxati bor. Oddiy o'quvchi ismlarga ahamiyat bermasligiga va kimnidir tark etish xavfiga qaramay, men quyidagi odamlarni tan olishni istardim: xotinim Virjiniya Gonsales (Vikki) va o'g'lim Maksimo Bassi. bir yildan ko'proq vaqt davomida virtual yo'qligim bilan kurashish uchun. Viki ham qo'lyozmalarni tayyorlash jarayonida menga behisob yordam berdi. Ota-onam va professorlar menga muhim saboq berishdi. Mening oilam (Oskar, Grasiela va Ramiro) menga Gyugo va Lukas Bejar bilan birga ingliz tilini tahrirlashda yordam berishdi. Viki, Griselda va Eugenio ham yozuvchilar va ishlab chiquvchilar uchun zarur bo'lgan rivojlanish mavhum qatlagini taqdim etdi.

Mahalliy Python hamjamiyatidagi odamlarga minnatdorchilik bildirmoqchiman (<http://www.python.org.ar>): Fakundo Batista, Lusio Torre, Gabriel Genellina, Jon Lenton, Alejandro J. Kura, Manuel Kaufmann, Gabriel Patinyo, Alejandro Vayl, Marselo Fernandes, Ariel Rossanigo, Mariano Dragi va Buanzo. Men yana Python-ni ushbu ajoyib jamoa uchun tanlagan bo'lardim. Men Biopython-dagi odamlarga ham rahmat aytaman: Jeffri Chang, Bred Chapman, Piter Kok, Mishel de Xun va Iddo Fridberg. Piter Kokga Biopython bobidagi sharhlari uchun alohida minnatdorchilik bildiriladi. Shuningdek, LATEX2e masalalarida menga yordam bergen Shashi Kumar va Pablo Di Napoliga va menga birinchi daqiqalardanoq ishongan Sunil Nairga minnatdorchilik **bildiraman**. Shuningdek, Globantda menga ishongan Guido Barosio, Jozefina Chausovskiy, Lukas Kampos, Pablo Brenner va Gibert Englebien kabi odamlar. Pedro Mourelle, Kris DeBlois, Rodrigo Obi-Van Iloro, Karlos Del Rio va Alejandro Valle kabi Globant hamkasblari.



**Taylor & Francis**  
Taylor & Francis Group  
<http://taylorandfrancis.com>

---



# Dasturlash



**Taylor & Francis**  
Taylor & Francis Group  
<http://taylorandfrancis.com>

# Kirish

---

## MAZMUNI

1.1 Ushbu kitobni kim o'qishi kerak . . . . .	3
1.1.1 O'quvchi nimani bilishi kerak . . . . .	4
1.2 Ushbu kitobdan foydalanish . . . . .	4
1.2.1 Tipografik konvensiyalar . . . . .	4
1.2.2 Python versiyalari . . . . .	5
1.2.3 Kod uslubi . . . . .	5
1.2.4 Barchasini o'qimay turib, ushbu kitobdan maksimal darajada foydalaning 1.2.5 Ushbu kitobga tegishli onlayn manbalar . . . . .	6
1.3 Nima uchun dasturlashni o'rganish kerak? . . . . .	7
1.4 Dasturlashning asosiy tushunchalari . . . . .	8
1.4.1 Dastur nima? . . . . .	8
Python? . . . . .	10
xususiyatlari. . . . .	10
solishtirish. . . . .	11
mumkinligi . . . . .	12
Tezlik . . . . .	13
ishlatiladi? . . . . .	14
foydalanadi? . . . . .	15
lazzatlari. . . . .	15
tarqatishlari . . . . .	16
manbalar . . . . .	17
1.5 Nima uchun	
1.5.1 Pythonning asosiy	
xususiyatlari. . . . .	
solishtirish. . . . .	
mumkinligi . . . . .	
Tezlik . . . . .	
ishlatiladi? . . . . .	
foydalanadi? . . . . .	
lazzatlari. . . . .	
tarqatishlari . . . . .	
manbalar . . . . .	
1.5.2 Pythonni boshqa tillar bilan	
1.5.3 U qanday	
1.5.4 Pythondan kim	
1.5.5 Python	
1.5.6 Maxsus Python	
1.6 Qo'shimcha	

Buni qilishning eng samarali usuli - buni qilish.

Amelia Erhart

## 1.1 BU KITOBNI KIM O'QISHI KERAK

Ushbu kitob dasturlashni o'rganmoqchi bo'lgan hayot fanlari tadqiqotchisi uchun.

U ilgari kompyuter dasturlash bilan shug'ullangan bo'lishi mumkin, ammo bu kitobni tushunish uchun kerak emas (garchi bu albatta yordam beradi).

Ushbu kitob bir nechta alohida, ammo bir-biriga bog'liq bo'lgan auditoriya, talabalar, bitiruvchilar, postdoclar va xodimlar uchun foydali bo'lishi uchun mo'ljallangan, chunki ularning barchasi dasturlashni bilishdan foyda olishlari mumkin.

## 4 Bioinformatika uchun Python

Talabalarni karerasining dastlabki bosqichlarida dasturlash bilan tanishtirish ularning ijodkorligi va mantiqiy fikrlash qobiliyatini oshirishga yordam beradi va ikkala ko'nikma ham tadqiqotda qo'llanilishi mumkin. Talabalar uchun o'quv jarayonini engillashtirish maqsadida barcha fanlar minimal shartlar bilan kiritiladi. Har bir bobning oxirida savollar ham mavjud. Ulardan qancha o'rganganingizni o'z-o'zini baholash uchun foydalanish mumkin. Javoblar o'qituvchilar uchun alohida qo'llanmada mavjud.

Haqiqiy dasturlash ehtiyojlariaga ega bo'lgan bitiruvchilar va xodimlar uni topishlari kerak bir nechta real misollar va ko'plab ma'lumotnomalar juda qimmatli.

### 1.1.1 O'quvchi nimani bilishi kerak

Ushbu kitob Bioinformatika uchun Python deb nomlanganligi sababli , u quyidagi taxminlarni hisobga olgan holda yozilgan:

- Hech qanday dasturlash bilimi mavjud emas, lekin matn muharriridan foydalanish va operatsion tizimingizda (OT) asosiy vazifalarni bajarish uchun o'quvchi minimal kompyuter bilimiga ega bo'lishi kerak. Python ko'p platformali bo'lgani uchun ushbu kitobdagagi ko'rsatmalarning aksariyati eng keng tarqalgan operatsion tizimlarga (Windows, macOS va Linux) taalluqlidir; faqat ma'lum bir OS uchun amal qiladigan buyruq yoki protsedura mavjud bo'lganda, u aniq qayd etiladi.
- O'quvchi bioinformatika vositalari bilan ishlayotgan bo'lishi kerak (yoki hech bo'lmaganda ishlashni rejalashtirmoqda). NCBI BLAST yordamida ketma-ketlikni aniqlash, oqsillarni tekislash, primerlarni qidirish yoki filogenetik daraxtni baholash kabi kichik hajmdagi qo'lda ishlangan ishlar ham misollarga amal qilish foydali bo'ladi. O'quvchi bioinformatika bilan qanchalik yaxshi tanish bo'lsa, u ushbu kitobda o'rganilgan tushunchalarni qanchalik yaxshi qo'llay oladi.

---

## 1.2 USHBU KITOBDAN FOYDALANISH

### 1.2.1 Tipografik konvensiyalar

Men kitob davomida bir xilda foydalanishga harakat qilgan ba'zi tipografik konvensiyalar mavjud. Ular o'qishga yordam berishi kerak va foydalanuvchi tomonidan yaratilgan nomlarni (yoki o'zgaruvchilarni) til kalit so'zlaridan ajratish uchun tanlangan. Bu yangi kompyuter tilini o'rganishda qo'l keladi.

**Qalin:** Python va uchinchi tomon modullari tomonidan taqdim etilgan ob'ektlar. Shu bilan birga, round foydalanuvchi tomonidan belgilangan nom emas, balki tilning bir qismi ekanligi aniq bo'lishi kerak. Matn qismlarini ajratib ko'rsatish uchun qalin harf ham qo'llaniladi. Bir jasur foydalanishni boshqasi bilan aralashtirib yuborishning iloji yo'q.

**Mono intervalli shrift:** Foydalanuvchi e'lon qilgan o'zgaruvchilar, kodlar va fayl nomlari. Masalan: ketma-ketlik = 'MRVLLVALALLALAASATS'.

**Kursiv:** Buyruqlarda u turli qiymatlarni qabul qila oladigan o'zgaruvchini belgilash uchun ishlataladi. Masalan, len(iterable) da "iterable" turli qiymatlarni qabul qilishi mumkin. ichida ishlataligan

matn, u yangi so'z yoki tushunchani belgilaydi. Masalan, "Bunday asosiy ma'lumotlar tuzilmasi bu lug'atdir."

\$ (dollar belgisi) bilan boshlanadigan satrlarning mazmuni operatsion tizim konsolida (shuningdek, Windowsda buyruq satri yoki macOS da terminal deb ataladi) terilishi uchun mo'ljallangan. ýy : Chiziqni ajratish. Ba'zi satrlar chop etilgan

sahifadagi mavjud bo'sh joydan uzunroqdir, shuning uchun bu belgi sahifaning keyingi qatorida joylashgan narsa kompyuter ekranidagi bir xil satrni ifodalashini bildirish uchun kiritilgan. Ichki kod, ishlatiladigan belgi

<=.

### 1.2.2 Python versiyalari

Hozirgi vaqtida Python-ning joriy versiyasi 3.6.1. 2.7.12 versiyasi mavjud, chunki u 2.7 filialidan foydalangan holda ishlab chiqarishda hali ham ko'p sonli ilovalar mavjud. 3.x va 2.x versiyalari mos kelmaydigan nuqtada bir oz farq qiladi. Python 3 ko'p jihatdan Python 2 ga qaraganda samaraliroq.

Instagram kabi yirik veb-saytlar protsessor va xotira sarfini 30% gacha tejash uchun Python 2.7 dan Python 3.6 ga oytdi. Ushbu kitob Python 3.6 dan foydalanadi.

Python 2.7 dan foydalanish kerak boylgan yagona stsenariy, eski kodni saqlashdan tashqari, Python 3 uchun maxsus kutubxona mavjud boylimaganida. almashtiriladigan kutubxona. Misol uchun, siz MySQL ma'lumotlar bazasiga ularishni xohlaysiz va sizga MySQLdb dan foydalanish aytildi, chunki bu paket Python 3 bilan mos kelmaydi; Python 2.7 dan foydalanish o'rniiga mysqlclient yoki mysql-connector-python dan foydalaning, ikkalasi ham Python 3 bilan ishlaydi.

### 1.2.3 Kod uslubi

Ushbu kitobda keltirilgan Python manba kodi ro'yxatlar sifatida taqdim etilgan. Ushbu ro'yxatlarning har bir qatori raqamlangan. Bu raqamlar terish uchun mo'ljallanmagan; ular matndagi har bir satrga murojaat qilish uchun ishlatiladi. Kodni kitobdan nusxalashingiz shart emas, chunki uni <https://github.com/Serulab/Py4Bio>.

Kod bir necha usulda formatlanishi mumkin va Python inter uchun amal qiladi preter. Ushbu kod sintaktik jihatdan to'g'ri:

```
def GetAverage(X):
    avG=sum(X)/len(X)
    """
    O'rtachani hisoblang
    avGni qaytarish
```

Shuningdek, bu:

---

<sup>1</sup>Python 2.7.x ning amal qilish muddati 2020 yilda tugaydi. Python 2.8 bo'lmaydi. Qo'shimcha ma'lumot olish uchun qarang: <https://www.python.org/dev/peps/pep-0373/>.

## 6 Bioinformatika uchun Python

```
def get_average (elementlar):
    """ O'rtachani hisoblang
    """
    o'rtacha = summa (elementlar) / len
    (elementlar) o'rtacha daromad
```

Avvalgi kod namunasi Python uchun eng ko'p qabul qilingan kodlash uslublariga amal qiladi.<sup>2</sup> Kitob davomida siz asosan ikkinchi namuna sifatida formatlangan kodni topasiz. Kitobdagি ba'zi kodlar quyidagi uchun qabul qilingan kodlash uslublariga amal qilmaydi sababli:

- Kodning ma'lum qismini ko'rsatishning eng didaktik usuli uslublar qo'llanmasiga zid bo'lgan holatlar mavjud. O'sha bir necha hollarda men ravshanlik foydasiga uslublar qo'llanmasidan chetga chiqishni tanlayman.
- Chop etilgan kitobning hajmi cheklanganligi sababli, ayrim nomlar qisqartirildi va kodlash uslublaridan boshqa kichik o'zgarishlar kiritildi.
- Xuddi shu kodni yozishning bir nechta usullari mavjudligini ko'rsatish. Kodlash uslubi ko'rsatma bo'lib, amal qilish til darajasida amalga oshirilmaydi, shuning uchun ba'zi dasturchilar unga to'liq amal qilmaydi. Siz "yomon" kodni o'qiy olishingiz kerak, chunki ertami-kechmi siz boshqa odamlarning kodini o'qishingiz kerak bo'ladi.

#### 1.2.4 Hammasini o'qimay turib, ushbu kitobdan maksimal foyda oling

- Agar dasturlashni o'rganmoqchi bo'lsangiz, **1-bobdan 8 -bobgacha** bo'lgan birinchi bo'limni o'qing . Agar REGEX bilan shug'ullanishingiz shart bo'lmasa, Muntazam ifodalar (REGEX) bo'limini (**13-bo'lim**) o'tkazib yuborishingiz mumkin.
- Agar siz Python-ni bilsangiz va shunchaki Biopython haqida bilmoqchi bo'lsangiz, avval **9-bobni** (158-betdan 209-betgacha) o'qing. Bu Biopython modullari va funktsiyalari haqida. Keyin III bo'limda joylashgan dasturlarga amal qilishga harakat qiling (315-betdan 363-betgacha).
- Mustaqil o'qilishi mumkin bo'lgan uchta qo'shimcha mavjud. Ilova A (Hamkorlikda ishlab chiqish: GitHub bilan versiyani boshqarish) "Git va GitHub bilan versiyalarni boshqarishga tezkor kirish" nomli maqolani qayta ishlab chiqaradi. Qo'shimcha B Python Anywhere yordamida veb-iovani qanday o'rnatishni ko'rsatadi. Ilova C - bo'limni o'qimasdan tez javob berish kerak bo'lganda hiyla-nayrang sifatida foydalanish mumkin bo'lgan ma'lumotnoma.

---

<sup>2</sup>Python uslubi bo'yicha rasmiy qo'llanma <https://www.python.org/dev/peps/pep-0008> manzilida joylashgan , va o'qish uchun qulayroq uslublar qo'llanmasi [http://docs.python-guide.org/en/latest/write/style\\_manzilida\\_joylashgan](http://docs.python-guide.org/en/latest/write/style_manzilida_joylashgan).

### 1.2.5 Ushbu kitob bilan bog'liq onlayn manbalar

Kitob veb-sayti <http://py3.us> manzilida joylashgan. Ushbu saytda siz xatoliklarni, Python haqida yangilanib turish uchun pochta ro'yxatini va manba kodlari omborlariga havolalarni topasiz. Manba kodiga kelsak, ushbu kitobning rasmiy manba kodi ombori GitHub (<https://github.com/Serulab/Py4Bio>) da joylashgan. Ushbu saytdan siz onlayn tarzda tekshirishingiz yoki ushbu kitobda ishlataligan barcha kodlarni yuklab olishingiz mumkin. Barcha skriptlarni yuklab olish uchun "Klonlash yoki yuklab olish" yashil tugmasiga o'ting va uni bosing. Agar sizda Git kompyuteringizda o'rnatilgan bo'lsa (va undan qanday foydalanishni bilsangiz<sup>3</sup>), ushbu reklama libosi yordamida omborni klonlang: [git@github.com:Serulab/Py4Bio.git](mailto:git@github.com:Serulab/Py4Bio.git). Yana bir variant - "ZIP-ni yuklab olish" tugmasini bosish. Mashinangizda omborga ega bo'lganiningizdan so'ng, kod papkasiga o'ting, u erda papkalar to'plami mavjud, har birida bobga tegishli skriptlar mavjud. Kitobdagi har bir skriptning nomi bor va bu fayl nomiga mos keladi. Noutbuklar deb nomlangan yana bir papka mavjud bo'lib, u mahalliy sifatida ishga tushirilishi mumkin bo'lgan Jupyter daftarlарini o'z ichiga oladi. Jupyter noutbukini qanday ishlatish haqida ko'proq ma'lumot olish uchun <http://jupyter.org>

Yana bir onlayn resurs Microsoft Azure Notebook veb-saytida (<https://notebooks.azure.com/py4bio/libraries/py3.us>) mavjud Jupyter noutbuklaridir . Kitob omboridagi bir xil daftarlardan ushbu saytda onlayn foydalanish mumkin.

### 1.3 NEGA DASTURLASHNI ORGANISH KERAK?

Tadqiqotchi o'z kompyuterida bajaradigan ko'pgina vazifalar takrorlanadi: Veb-sahifadan ma'lumotlarni to'plash, fayllarni bir formatdan boshqa formatga o'tkazish, yuzlab BLAST natijalarini bajarish yoki sharhlash, primer dizayni, cheklash fermentlarini qidirish va hokazo. ko'p hollarda, bu kompyuter yordamida, biz tomonidan kamroq kuch sarflagan holda va charchoq yoki chalg'itishdan kelib chiqadigan xatolarga yo'l qo'ymasdan bajarilishi mumkin bo'lgan vazifalar ekanligi ayon bo'ladi.

Dasturni yaratish yoki yaratmaslikni baholashda muhim e'tibor muammoni aniqlash va shakllantirish, uni kod bilan amalga oshirish va keyin uni disk raskadrovska qilishda (xatolarni tuzatishda) yo'qolgan vaqtadir. Muammoni aniqlash va baholashni vaqtini behuda sarflash deb hisoblash noto'g'ri. Umuman olganda, jarayonning aynan shu nuqtasida biz duch kelayotgan muammoni yaxshilab tushunamiz. Muammoni shakllantirishga urinish paytida biz dastlabki taxminlarimizning ko'pchiligi noto'g'ri bo'lganini tushunamiz. Bu, shuningdek, rejalshtirish jarayonini qachon qayta boshlash zarurligini aniqlashga yordam beradi. Bu sodir bo'lganda, biz loyihaning o'rtasida bo'lgandan ko'ra, rejalshtirish bosqichida sodir bo'lgan yaxshiroqdir. Bunday hollarda dasturni rejalshtirish tejalgan vaqtini anglatadi. E'tiborga olish kerak bo'lgan yana bir afzallik shundaki, dasturni yaratish uchun bir marta sarflangan vaqt biz uni har safar ishga tushirganimizda vazifalarni bajarish tezligi bilan qoplanadi.

<sup>3</sup> A ilovasida GitHub-dan qanday foydalanish bo'yicha qo'llanma mavjud

## 8      Bioinformatika uchun Python

U nafaqat biz qo'lda bajaradigan protseduralarni avtomatlashtirishi mumkin, balki boshqa hollarda amalga oshirish mumkin bo'limgan narsalarni ham qila oladi.

Ba'zan ma'lum bir vazifani dastur tomonidan bajarish mumkinligi juda aniq emas. Bu kabi kitobni o'qish (misollarni o'z ichiga olgan holda) sizni aniqlashga yordam beradi dasturiy ta'minot yordamida qaysi vazifalarni avtomatlashtirish mumkin va qaysi biri yaxshiroq bajariladi qo'lda.

## 1.4 ASOSIY DASTURLASH TUSHUNCHALARI

---

Python-ni o'rnatishdan oldin, ba'zi dasturlash asoslarini ko'rib chiqaylik. Agar sizda avvalroq dasturlash tajribangiz boylsa, ushbu boylimni oytakazib yuborib, toyg'yridan-toyg'yri "Python-ni o'rnatish" **2-bobiga** oytishingiz mumkin. Ushbu bolimda ko'satmalar, ma'lumotlar turlari, o'zgaruvchilar **va ushbu kitob** davomida qo'llaniladigan boshqa tegishli atamalar kabi asosiy tushunchalar keltirilgan.

### 1.4.1 Dastur nima?

Kompyuterlar faqat siz ularga nima deyotganingizni biladi. Ularga biror narsa qilishni aytishning yo'lli dastur orqali. Dastur - bu kompyuterga biror narsa qilishni buyurish uchun mo'ljallangan tartiblangan ko'satmalar to'plami. "Buyurtma qilingan" so'zi mavjud, chunki nima qilish kerakligini e'lon qilish uchun etarli emas, lekin yo'nalihsarning haqiqiy tartibi ham ko'satilishi kerak.<sup>4</sup>

Dastur ko'pincha retsept sifatida tavsiflanadi. Odatiy retsept ingredientlar ro'yxatidan iborat bo'lib, undan keyin idishni tayyorlash bo'yicha bosqichma-bosqich ko'satmalar mavjud.

Ushbu o'xshashlik bir nechta dasturlash veb-saytlari va o'quv qo'llanmalarida "retsept" va "pazandachilik kitobi" so'zleri bilan aks ettirilan. Laboratoriya protokoli yana bir foydali o'xshashlikdir. Protokol "eksperimentlarni loyihalash va amalga oshirishda oldindan belgilangan yozma protsessual usul" deb ta'riflanadi.

Bu erda deyarli har kuni bir nechta molekulyar laboratoriyalarda bajariladigan odatiy protokol:

**Listing 1.1: Lambda DNK hazm qilish protokoli**

---

Lambda DNKsining hazm bo'lishini cheklash

#### Materiallar

5,0 mkl	Lambda DNK (0,1 g/L) 10x
2,5 mkl	bufer
16,5 mkl H2O	
1,0 mkl	EcoRI

<sup>4</sup>Deklarativ tillar mavjud bo'lib, ular dastur qanday bajarilishini tasvirlashdan ko'ra, nimaga erishishi kerakligini bildiradi. Ko'pgina kompyuter tillari (Python kiritilgan) deklarativ o'rninga imperativdir.

Jarayon

Reagentlarni 37°C da 1 soat davomida inkubatsiya qiling.

2,5 mcL yuklovchi bo'yоq qо'shing va yana 15 daqiqа davomida inkubatsiya qiling.

20 mcL hazm qilish aralashmasini minigelga soling

Protokolning kamida ikkita komponenti mavjud: materiallar yoki ingredientlar va protseduralar. Jarayon inkubatsiya, qо'shish, aralashtirish, yuklash va boshqalar kabi aniq tartibni ta'minlaydi. Xuddi shu narsa kompyuter dasturiga ham tegishli. Dasturchi kompyuterga aniq buyruq beradi: chop etish, o'qish, yozish, qо'shish, ko'paytirish, dumaloq qilish va boshqalar.

Protokol protseduralari dastur ko'rsatmalari bilan bog'liq bo'lса-da, materiallar ma'lumotlарdir . Protokollarda materialarga protseduralar qо'llaniladi: 2,5 μL buferni 5 μL Lambda DNK va 16,5 μL H2O bilan aralashtirib, minigelga 20 μL yuklang. Dasturda ma'lumotlarga ko'rsatmalar qо'llaniladi: "Salom" matn qatorini chop eting, ikkita butun son qо'shing, suzuvchi raqamni aylantiring.

Protokol sifatida biz turli tillarda (masalan, ingliz, ispan yoki frantsuz) yozishimiz mumkin, kompyuterni dasturlash uchun turli tillar mavjud. Fanda ingliz tili de-fakto tildir. Tarixiy, tijorat va amaliy sabablarga ko'ra, kompyuter fanida bunday ekvivalent yo'q. Bir nechta tillar mavjud, ularning har biri o'zining kuchli va zaif tomonlariga ega. Tez orada mantiqiy bo'ladijan sabablarga ko'ra, Python ushbu kitob uchun tanlangan kompyuter tili edi.

Keling, oddiy Python dasturini ko'rib chiqaylik:

**Listing 1.2: sample.py: Python dasturi namunasи**

```
1 seq_1 = 'Salom,' 2
seq_2 = 'siz!' jaketraha =
ket_1 + ketma-ket_2 4 ketma-
ketlik_size = liniya (jami) 5 ta
chop etish (seq_size)
```

Eslatma: Har bir satr boshidagi raqamlar faqat ma'lumot uchun mo'ljallangan, ular terish uchun mo'ljallanmagan.

Ushbu kichik dasturni "Salom satriga nom bering, seq\_1 deb o'qishingiz mumkin. Tarmoqqa o'zingiz nom bering! 2-qism sifatida. seq\_1 va seq\_2 nomli satrlarni qо'shing va natijani jami deb chaqiring. Total deb nomlangan satr uzunligini oling va bu qiymatni seq\_size deb nomlang. seq\_size qiymatini chop eting." Ushbu dastur 11 raqamini chop etadi.

Ko'rsatilganidek, ma'lumotlarning har xil turlari mavjud (ko'pincha "ma'lumotlar turlari" yoki faqat "turlar" deb ataladi). Raqamlar (butun sonlar yoki float), matn qatori va boshqa ma'lumotlar turlari 3- bobda yoritilgan. Print(seq\_size) da ko'rsatma chop etiladi, seq\_size esa ma'lumotlar nomidir. Ma'lumotlar ko'pincha o'zgaruvchilar sifatida taqdim etiladi. O'zgaruvchi - bu dasturni bajarish jarayonida o'zgarishi mumkin bo'lgan qiymatni bildiruvchi nom. O'zgaruvchilar bilan dasturchi "2.9-raund" o'rniiga "round n" kabi umumiy buyruqni ifodalashi mumkin. Shu tarzda u sobit bo'limgan (shuning uchun o'zgaruvchan) qiymatni hisobga olishi mumkin. Qachon

## 10 Bioinformatika uchun Python

dastur bajarilgan bo'lsa, "n" ma'lum bir qiymatga ega bo'lishi kerak, chunki "n" ni yaxlitlashning imkon yo'q. Buni o'yzgaruvchiga qiymat berish yoki qiymatga nom bogylash orqali amalga oshirish mumkin.<sup>5</sup> "O'yzgaruvchiga qiymat berish" va "nomni qiymatga bogylash" o'rtasidagi farq **3-bobda** (dan boshlab) batafsil tushuntirilgan. 64-bet). Har holda, u quyidagicha ifodalanadi:

```
o'zgaruvchi_nomi = qiymat
```

E'tibor bering, bu matematikada ko'rindigan tenglik emas. Tenglikda atamalar almashtirilishi mumkin, lekin dasturlashda o'ngdagi atama (qiymat) chapdagি atama nomini oladi (o'zgaruvchi\_nomi). Masalan,

```
seq_1 = "Salom"
```

Ushbu topshiriqdan so'ng seq\_1 o'zgaruvchisidan foydalanish mumkin, masalan:

```
chop etish (qator_1)
```

Bu "seq\_1 deb nomlangan qiymatni chop etish" deb tarjima qilingan. Bu buyruq "Salom"ni qaytaradi, chunki bu o'zgaruvchining qiymati.

## 1.5 NEGA PYTHON?

---

Keling, Pythonning ba'zi xususiyatlarini ko'rib chiqaylik.

### 1.5.1 Pythonning asosiy xususiyatlari

- O'qilishi: O'qish mumkinligi haqida gapirganda, biz kodni tushunishga qiziqqan har qanday boshqa shaxs kabi original dasturchiga murojaat qilamiz. Kimdir kodni yozib, bir oydan keyin unga qaytib, tushunish qiyin bo'lishi odatiy hol emas. Ba'zan Python "odam o'qiy oladigan til" deb ataladi.
- O'rnatilgan funksiyalar: Python "batareyalar" bilan birga keladi. Uning boy va ko'p qirrali standart kutubxonasi mavjud bo'lib, u foydalanuvchi alohida paketlarni yuklab olmasdan darhol foydalanish mumkin. Python yordamida siz bir nechta satrlar bilan XML va JSON fayllarini o'yqish va yozishingiz, elektron pochta xabarlarini tahlil qilishingiz va yaratishingiz, zip arxividan fayllarni chiqarib olishingiz, URL-manzilni xuddi fayl kabi ochishingiz va boshqa tillarda mavjud boylgan boshqa imkoniyatlarni yaratishingiz mumkin. uchinchi
- Keng spektrdagи faoliyat uchun uchinchi tomon modullarining mavjudligi. Ma'lumotlarni vizuallashtirish<sup>6</sup> va chizma, PDF yaratish, bioinformatika tahilili, 7 tasvir

<sup>5</sup> Pythonda oxirgi shakl ishlatalidi.

<sup>6</sup> Matplotlib (<http://matplotlib.org/>) va bokeh <http://bokeh.pydata.org/en/latest/> eng ko'p qo'llaniladi.

7 O'z bioinformatika dasturlarini yaratish uchun Biopython kutubxonasi (<http://biopython.org/>).

qayta ishlash, 8 mashinani o'rganish, 9 o'yinni ishlab chiqish, mashhur ma'lumotlar bazalari bilan interfeys, 10 va amaliy dasturiy ta'minot Python funksionalligini kengaytirish uchun o'rnatilishi mumkin bo'lgan modullarning bir nechta misolidir.

- Yuqori darajadagi o'rnatilgan ma'lumotlar tuzilmalari: Lug'atlar, to'plamlar, ro'yxatlar, kortejlar va boshqalar. Bular real dunyo ma'lumotlarini modellashtirish uchun juda foydali. NumPy va SciPy kabi uchinchi tomon modullari tuzilmalarni kd-daraxtlar, n-o'lchovli massivlar, matritsa operatsiyalari, vaqt seriyalari, tasvir ob'ektlari va boshqalarga kengaytirishi mumkin.
- Multiparadigma: Python "klassik" protsessual til yoki "zamonaviy" ob'ektga yo'naltirilgan dasturlash (OOP) tili sifatida ishlatalishi mumkin. Aksariyat dasturchilar kodni protsessual tarzda yozishni boshlaydilar va kerak bo'lganda, ular OOP ga yangilashadi. Python dasturchilarni oddiy skript yozishni xohlaganlarida OOP kodini yozishga majburlamaydi.
- Kengaytirish imkoniyati: Agar o'rnatilgan usullar va mavjud uchinchi tomon modullari sizning ehtiyojlarингiz uchun etarli bo'lmasa, Python tilini hatto boshqa dastur tillarida ham osongina kengaytirishingiz mumkin. Ko'pincha Python-da yozilgan, lekin C yoki FORTRAN-da tartib talab qiladigan protsessorga ega bo'lgan ba'zi ilovalar mavjud. Python, shuningdek, R yoki MATLAB<sup>11</sup> kabi maxsus yuqori darajadagi tillarga ulanish orqali kengaytirilishi mumkin .
- Ochiq manba: Python liberal ochiq kodli litsenziyaga ega, bu esa uni hatto tijorat maqsadlarida foydalanish uchun ham erkin foydalanish va tarqatish imkonini beradi.
- O'zaro platforma: Python tilida yaratilgan dastur Python tarjimoniga ega bo'lgan har qanday kompyuter ostida ishlashi mumkin. Shunday qilib, Windows 10 ostida yaratilgan dastur Linux yoki OSX da o'zgartirilmagan holda ishlashi mumkin. Python tarjimonlari ko'pgina kompyuterlar va operatsion tizimlar va hatto Raspberry Pi kabi o'rnatilgan kompyuterlari bo'lgan ba'zi qurilmalar uchun mavjud.
- Rivojlanayotgan hamjamiyat: Python bugungi kunda olimlar va tadqiqotchilar uchun foydalaniladigan dasturlash tilidir.<sup>12</sup> Bu sizning loyihalarингiz va yordam so'rab murojaat qilishingiz mumkin bo'lgan odamlar uchun ko'proq kutubxonalarga aylanadi.

### 1.5.2 Pythonni boshqa tillar bilan solishtirish

Siz nima uchun Java, PHP yoki C++ kabi taniqli tillarni emas, balki Python tilini ishlatishingiz kerakligi haqida savol tug'dirayotgandirsiz. Bu yaxshi savol. Dasturlash tili

**8Scikit-image qog'ozi:** <http://peerj.com/articles/453>

<sup>9</sup> scikit-learn veb-sayti: <http://scikit-learn.org/stable/> <sup>10</sup><https://wiki.python.org/moin/DatabaseProgramming> <sup>11</sup>MATLAB® The

MathWorks, Inc kompaniyasining ro'yixatdan oytgan savdo belgisidir. Mahsulot haqida maylumot olish uchun quyidagi manzilga murojaat qiling: The MathWorks, Inc. 3 Apple Hill Drive Natick, MA, 01760-2098 AQSh. Tel: 508-647-7001. Elektron pochta: [info@mathworks.com](mailto:info@mathworks.com). Veb-sayt: [www.mathworks.com](http://www.mathworks.com).

**12**<http://www.nature.com/news/programming-pick-up-python-1.16833>

## 12 Bioinformatika uchun Python

**vosita sifatida qaralishi mumkin va ish uchun eng yaxshi vositani tanlash juda ko'p narsalarni qiladi tuyg'u.**

O'qish qobiliyati

Professional bo'limgan dasturchilar o'rganish egri chizig'ini kodning oyoq qobiliyati kabi qadrlashadi (ikkala jihat bir-biri bilan chambarchas bog'liq).

Python-dagi oddiy "salom dunyo" dasturi quyidagicha ko'rindan:

```
chop ("Salom dunyo!")
```

Uni Java'dagi ekvivalent kod bilan solishtiring:

```
umumiyl sinf Salom {
```

```
    public static void main(String[] args)
        { System.out.printf("Salom dunyo!");
    }
}
```

Keling, C tilida kod namunasini ko'rib chiqaylik. Quyidagi dastur faylni o'qiydi (input.txt) va uning mazmunini boshqa faylga (output.txt) nusxalaydi:

```
#include <stdio.h>
int main(int argc, char **argv) { FILE *in,
    *out; int c; in = fopen("input.txt", "r");
    out = fopen("output.txt", "w"); while ((c
    = fgetc(in)) != EOF) { fputc(c, out);
}
} fclose(out);
fclose(in);}
```

Python-da xuddi shu dastur qisqaroq va o'qish osonroq:

```
input_file sifatida open("input.txt") bilan:
    open("output.txt") bilan output_file sifatida:
        output_file.writelines(in)
```

Keling, bir qator raqamlarning o'rtacha qiymatini hisoblaydigan Perl dasturini ko'rib chiqaylik:

```

sub avg(@_) { $sum
    += $_ foreach @_; $sum / @_ ni
    qaytaring, agar @_ == 0 bo'lmasa; qaytish 0;

}

print avg((1..5))."\n";

```

Pythondagi ekvivalent dastur:

```

def avg(ma'lumotlar):
    agar len(ma'lumotlar)==0:
        0 qaytarining
    boshqa:
        summani qaytarish (ma'lumotlar)/len (ma'lumotlar)
chop etish (oýrtacha([1,2,3,4,5]))

```

Ushbu Python dasturining maqsadini faqat ingliz tilini bilish orqali deyarli to'liq tushunish mumkin.

Python juda o'qilishi mumkin bo'lgan til bo'lishi uchun yaratilgan.<sup>13</sup> Ingliz kalit so'zlaridan foydalanish, kod bloklari va uning ichki mantig'ini cheklash uchun bo'shlardan foydalanish bu maqsadga yordam beradi. Pythonda o'qish qiyin bo'lgan kodni yozish mumkin, lekin kodni chalkashtirib yuborish uchun ataylab harakat qilish kerak.<sup>14</sup>

## Tezlik

Dasturlash tilini tanlashda e'tiborga olish kerak bo'lgan yana bir mezon - bu bajarilish tezligi. Kompyuter dasturlashning dastlabki kunlarida kompyuterlar shunchalik sekin ediki, tilni amalga oshirish tufayli ba'zi farqlar juda muhim edi. Dasturning tarjima qilingan tilda bajarilishi uchun bir hafta vaqt ketishi mumkin, kompilyatsiya qilingan tildagi bir xil kod esa bir kunda bajarilishi mumkin. Tarjima qilingan va kompilyatsiya qilingan tillar o'rtaсидagi ishlash farqi hali ham bir xil nisbatga ega, ammo u kamroq ahamiyatga ega. Buning sababi shundaki, ishga tushirish uchun bir hafta kerak bo'lgan dastur endi o'n soniyadan kamroq vaqtini oladi, kompilyatsiya qilingani esa taxminan bir soniya davom etadi. Farq muhim bo'lib ko'rinsa ham (kamida bitta kattalik tartibi), agar biz uni ishlab chiqish uchun zarur bo'lgan vaqtini hisobga olsak, u unchalik ahamiyatlama emas.

Bu bajarilish tezligini hisobga olish kerak emas degani emas. Ba'zi yuqori samarali hisoblash operatsiyalarida 10X tezlik farqi hal qiluvchi bo'lishi mumkin.

Ba'zan optimallashtirilgan kodni yozish orqali ko'plab yaxshilanishlarga erishish mumkin. Agar kod tezlikni optimallashtirishni hisobga olgan holda yozilgan bo'lsa, natijalarga erishish mumkin

---

<sup>13</sup>Boshqa tillar "faqat yozish" deb hisoblanadi, chunki bir marta yozilgani uchun uni tushunish juda qiyin bu.

<sup>14</sup>Oddiy bosma 'Salom Dunyo' dasturi, agar siz juda moyil bo'lsangiz, chop etish ".join([chr((L>=65 va L<=122) va (((L>=97) va kabi yozilishi mumkin. (L-96) yoki (L-64))-1)+13)%26+((L>=97) va 97 yoki 65)) yoki L) uchun L) [ ord(C) uchun C 'Uryyb Jbeyq!']]]) (py3.us/1).

## 14 Bioinformatika uchun Python

kompilyatsiya qilingan tilda olinishi mumkin bo'lganlarga o'xshash. Agar dasturchi Python tomonidan olingen tezlikdan qoniqmasa, boshqa tilda (masalan, C yoki Fortran) yozilgan tashqi kutubxonaga ularish mumkin. Shunday qilib, biz ikkala dunyoning eng yaxshisini olishimiz mumkin: kompilyatsiya qilingan til tezligi bilan Python dasturlashning qulayligi.

### 1.5.3 U qanday ishlataladi?

Python keng ko'lamli dasturlarga ega. Mobil telefonlardan veb-serverlarga, eng xilma-xil sohalarda minglab Python ilovalari mavjud. Vikipediya robotlarini quvvatlaydigan, Industrial Light & Magic-da yangi avlod maxsus effektlarini yaratishga yordam beruvchi Python kodi mavjud<sup>15</sup>, D-link modemlari va marshrutizatorlariga o'rnatilgan<sup>16</sup> va bu OpenOffice to'plamining skript tilidir<sup>17</sup>.

Ba'zi tillar bir joyda kuchli (masalan, veb-illovalar uchun PHP, ish stoli dasturlari uchun Java), ammo Python-ni osongina yozish mumkin emas.

Bitta kod bazasi bilan Python ish stoli ilovalari bir nechta platformalarda o'ziga xos ko'rinish va hissiyot bilan ishlaydi. Ushbu toifadagi taniqli misollar orasida BitTorrent p2p mijoz/server, Calibre, elektron kitoblar menejeri, Sage Math (matematik ematik dasturiy ta'minot tizimi), Dropbox mijozи va boshqalar kiradi.

Veb-illovalarni yaratish tili sifatida Python-ni Reddit, NationalGeographic, Instagram va NASA kabi yuqori trafikli saytlarda topish mumkin. Pythonda Django, Web2Py, Pyramid, Flask va Bottle kabi veb-saytlar (webframeworks deb ataladi) yaratish uchun maxsus dasturlar mavjud.

Tizim ma'muriyatidan ma'lumotlar tahliligacha, Python shu maqsadda keng ko'lamli vositalarni taqdim etadi:

- Umumiy operatsion tizim xizmatlari (OS, io, vaqt, la'natlar)
- Fayl va katalogga kirish (os.path, glob, tempfayl, shutil)
- Ma'lumotlarni siqish va arxivlash (zipfile, gzip, bz2)
- Jarayonlararo aloqa va tarmoq (subprocess, socket, ssl)
- Internet (elektron pochta, mimetools, rfc822, cgi, urllib)
- String xizmatlari (string, re, kodeklar, unicodedata)

Python ilmiy hamjamiyat uchun standart kompyuter tili sifatida tobora kuchayib bormoqda. SciPy<sup>18</sup> va Anaconda kabi ilmiy foydalanuvchilarga yo'naltirilgan bir nechta kutubxonalar mavjud . <sup>19</sup> Ikkala taqsimot ham chiziqli algebra uchun modullarni birlashtiradi,

---

<sup>15</sup><https://www.python.org/about/success/>  
<sup>16</sup><https://www.python.org/about/success/dlink/> <sup>17</sup><http://wiki.services.openoffice.org/wiki/Python> <sup>18</sup><https://www.scipy.org> <sup>19</sup><https://www.continuum.io/anaconda-overview>

signallarni qayta ishlash, optimallashtirish, statistika, genetik algoritmlar, interpolyatsiya, ODE hal qiluvchi, maxsus funktsiyalar va boshqalar.

Python pyMPI va 2D/3D ilmiy ma'lumotlarni chizish bilan parallel dasturlashni qo'llab-quvvatlaydi.

Python muhandislik, elektron kabi keng va xilma-xil sohalarda qo'llanilishi ma'lum ics, astronomiya, biologiya, paleomagnitizm, geografiya va boshqalar.

#### 1.5.4 Pythondan kim foydalanadi?

Python kichik va noma'lum do'konlardan tortib Google, National Geographic, Disney, NASA, NYSE va boshqa ko'plab sohalardagi yirik o'yinchilargacha bo'lgan bir nechta kompaniyalar tomonidan qo'llaniladi.

Bu Java, C++ va Go tillari orasida Googlening to'rtta "rasmiy tillari" dan biridir.

Ularda Python-da yaratilgan veb-saytlar, mustaqil dasturlar va hattoki hosting bor.<sup>20</sup> Google Python-ga jiddiy yondashayotganining tasdig'i sifatida 2005-yil dekabr oyida ular Python-ni yaratuvchisi Guido van Rossumni yollashdi. Bu Google-ning asosiy tili bo'lmasisligi mumkin, ammo bu ularning kuchli tarafdori ekanligini ko'rsatadi.

Hatto ochiq kodli dasturlarni qo'llab-quvvatlashi bilan mashhur bo'lImagen Microsoft kompaniyasi ham o'zining ".Net" platformasini (IronPython) Bepul, ochiq manbada ishlatalish hamda Visual Studio uchun Python Tools pluginini ishlab chiqdi, bu <sup>21</sup> uchun Python versiyasini ishlab chpligin Visual Studio ni Python IDE ga aylantiradi.

Ko'pgina taniqli Linux distributorlari o'zlarining asosiy vositalarida allaqachon Python-dan foydalanadilar. Ubuntu Linux "hamjamiyat Pythonda ishlashga hissa qo'shishni afzal ko'radi". Python Linux-ga shu qadar qattiq integratsiyalashganki, ba'zi distributivlar Python-ning ishchi nusxasisiz ishlamaydi.

#### 1.5.5 Python lazzatlari

Garchi bu kitobda men Pythonni dasturlash tili sifatida ko'rsatsam ham, Python aslida til ta'rifidir. Biz ko'pincha ishlataladigan narsa - bu CPython, ya'ni C tilida amalga oshirilgan Python tilining ta'rifi.

Ushbu dastur eng ko'p qo'llaniladiganligi sababli, biz Pythonni CPython ilovasiga chaqiramiz.

Eng dolzarb Python ilovalari quyidagilardir: CPython, PyPy,<sup>22</sup> Stackless,<sup>23</sup> Jython<sup>24</sup> va IronPython.<sup>25</sup> Ushbu kitob Python standart versiyasiga (CPython) qaratilgan, ammo turli versiyalar haqida bilishga arziydi.

- CPython: eng ko'p ishlataladigan Python versiyasi, shuning uchun CPython va Python atamalari bir-birining o'rnida ishlataladi. U asosan C tilida yaratilgan (ba'zi modullar yaratilgan

<sup>20</sup><https://cloud.google.com/appengine/>

<sup>21</sup><https://www.visualstudio.com/vs/python/>

<sup>22</sup><http://codespeak.net/pypy/dist/pypy/doc/home.html>

<sup>23</sup><http://www.stackless.com> <sup>24</sup><http://www.jython.org/>

Project 25<http://ironpython.net>

## 16 Bioinformatika uchun Python

Python da) va Python rasmiy veb-saytida (<http://www.python.org>) mavjud bo'lgan versiya .

- PyPy: Python-da yaratilgan Python versiyasi. Bu dasturchilarga tilni moslashuvchan tarzda sinab ko'rishga imkon berish uchun ishlab chiqilgan (C tilini bilmasdan Python kodini o'zgartirish uchun). Bu asosan eksperimental platforma.
- Stackless: yana bir eksperimental Python ilovasi. Ushbu dasturning maqsadi PyPy kabi moslashuvchanlikka e'tibor qaratmaydi; Buning o'rniغا, u "standart" Python versiyasida mavjud bo'lмаган ilg'or xususiyatlarni taqdim etadi. Bu Python rivojlanish tarixida qabul qilingan ba'zi dizayn qarorlarini engish uchun amalga oshiriladi. Stackless maxsus ishlab chiqilgan Python dasturiga CPython hamkasblariга qaraganda yaxshiroq miqyoslash imkonini beradi. Ushbu dastur EVE Online ommaviy ko'p o'yinchi onlayn o'yinida, Civilization IV, Second Life va Twistedda qo'llaniladi .
- Jython: Java tilida yozilgan Python versiyasi. U JVM (Java Virtual Machine) da ishlaydi. Jython-ning ilovalaridan biri Jython kutubxonalarini o'zlarining Java tizimiga qo'shishdir, bu foydalanuvchilarga ilovaga funksionallik qo'shish imkonini beradi. Juda mashhur 3D dasturlash muhiti (Alice<sup>26</sup>) foydalanuvchilarga o'z skriptlarini dasturlash imkonini berish uchun Jython-dan foydalanadi.
- IronPython: Microsoft tomonidan ".Net" va ".Mono" platformalarida ishlashga moslashtirilgan Python versiyasi. .Net bu "bir marta yozish, hamma joyda ishlaydi" bo'yicha Java bilan raqobatlashadigan texnologiya.

### 1.5.6 Maxsus Python taqsimotlari

Python ilovalaridan tashqari, ma'lum maqsadlar uchun paketlangan original CPython-ning ba'zi maxsus moslashuvlari mavjud. Ular Python to'plamlari yoki taqsimotlari deb ataladi. Ularning aksariyati muharrirlar, vizualizatsiya modullari va Jupyter noutbuki kabi uchinchi tomon dasturlarini stolga olib keladi. Bu jonli kod, tenglamalar, vizualizatsiya va tushuntirish matnnini o'z ichiga olgan hujjatlarni yaratish va almashish imkonini beruvchi veb-ilova. Bu erda eng foydali tarqatishlar ro'yxati<sup>27</sup>:

precom-ni taqdim etadi [28 Komp'yutorda Python dasturlash uchun korporatsiyalarga qo'shilish](#)  
 kafolatlangan Python distributivi korporatsiyalarga ochiq kodli mahsulotlarni qo'llab-quvvatlash siyosati talablariga rioya qilishni osonlashtiradi. Texnik nuqtai nazardan, u oldindan o'rnatilgan eng ko'p ishlatiladigan tashqi modullar bilan barcha zamonaviy Python versiyalarini taqdim etadi. Shuningdek, u o'zining paketlarni boshqarish va tashqi modullar omboriga ega (PyPM<sup>29</sup>)

26Alice <http://www.alice.org> saytida bepul mavjud .

27Python ilovalari va tarqatishlarining to'liq ro'yxati uchun <https://www.python.org/download/alternatives>

28<http://www.activestate.com/activepython>

29<https://code.activestate.com/pypm/>

Enthought Canopy: NumPy,<sup>30</sup> SciPy, Python, y<sup>2</sup>Da b<sup>3</sup> Dasturlash tizimi o'yin shingizda hisoblanadi. Adapterlari va boshqalar kabi 450 ta asosiy ilmiy tahliliy va Python to'plamlari. Jupyter noutbuklarini qo'llab-quvvatlaydigan kod muharriri ham o'z ichiga oladi. Unda sizga yangilanishlar haqida xabar beradigan, bir marta bosish bilan o'rnatiladigan va paket versiyalarini orqaga qaytarishga yordam beradigan grafik paket menejeri kabi ba'zi qo'shimchalar mavjud. Hamma narsa uchta asosiy operatsion tizim uchun bir marta bosish bilan o'rnatuvchi sifatida mavjud. Ushbu to'plam ilmiy foydalanuvchilar uchun mos bo'lib, u NumPy va SciPy-ni yaratgan odamlar tomonidan yaratilgan. Bepul akademik litsenziyalar va turli xil pullik tijorat korxonalari litsenziyalari kabi turli xil litsenziyalar mavjud.

- WinPython: <sup>31</sup> U o'zini Windows 7/8/10 va ilmiy va ta'lrim maqsadlarida foydalanish uchun Python dasturlash tilining bepul ochiq manbali portativ tarqatilishi sifatida belgilaydi. Shuningdek, olimlar, ma'lumotshunoslar va ta'lrim uchun mos paketlarni o'z ichiga oladi (NumPy, SciPy, Sympy, Matplotlib, Pandas, pyqtgraph va boshqalar). Noto'g'ri muharrir sifatida Spyder (Scientific PYthon Development EnviRonment) dan foydalanadi va foydalanuvchi WinPython katalogini ko'chirishi va barcha sozlamalar saqlanib qolishi ma'nosida ko'chma hisoblanadi. Siz izolyatsiyalangan va o'z-o'zidan mos keladigan WinPython o'rnatishlarining bir nechta nusxa.
- Anakonda: <sup>32</sup> Ilmiy hisoblash uchun Python va R tarqatish. Ma'lumotlarni tayyorlash, ma'lumotlarni tahlil qilish, ma'lumotlarni vizuallashtirish, mashinani o'rganish va interaktiv ma'lumotlar faniga oid 720 dan ortiq paketlarni o'z ichiga oladi. U maqsad va foydalanuvchi turini Enthought Canopy bilan baham ko'radi. Shuningdek, standart kod muharriri sifatida Spyder bilan birga keladi. Uni boshqa Python tarqatishdan ajratib turadigan bir nechta mahsulotlar mavjud, masalan, Repository, Accelerate, So Ushbu xizmatlarning aksariyati faqat qimmat obunalar uchun mavjud. Agar siz ushbu xizmatlardan foydalanmasangiz, siz hali ham ajoyib Python tarqatilishini olasiz. Continuum, Anaconda ortida turgan kompaniya Project Jupyterning institutsional hamkorি hisoblanadi, ya'ni ular brauzerda Python kodini ishga tushirish uchun veb-ilova bo'lgan Jupyter Notebook-ni ishlab chiqishni qo'llab-quvvatlaydi.

Siz qaysi birini ishlatishti (yoki oddiy "oddiy vanil" dan foydalanishni) qiziqtirgan bo'lisingiz mumkin. Python). Bu savolga yagona va to'g'ri javob yo'q, chunki bu sizning ehtiyojlaringiz, ish odatlaringiz, byudjetingiz va shaxsiy imtiyozlaringizga bog'liq bo'ladi. Shaxsan men serverlarda standart Python va dasturiy ta'minotni ishlab chiqishda foydalanadigan kompyuterlarda Anaconda dan foydalanishga moyilman.

## 1.6 QO'SHIMCHA RESURSLAR

- Interaktiv noutbuklar: kodni almashish. Interaktiv noutbuklar: kodni almashish. Bepul IPython noutbuki ma'lumotlar tahlilini yozib olishni osonlashtiradi

<sup>30</sup><https://www.enthought.com/products/canopy/> <sup>31</sup><http://winpython.github.io/>  
<sup>32</sup><https://www.continuum.io/anaconda-overview>

## 18 Bioinformatika uchun Python

turish va ko'paytirish. Helen Shen. Tabiat 515, 151–152 (06-noyabr 2014 yil)  
doi:10.1038/515151a <https://goo.gl/HfBJ12>

- **Badiiy film uchun**  
Python: <http://dgovil.com/blog/2016/11/30/python-for-feature-film/>
- **Muqobil Python ilovalari:** <https://www.python.org/download/alternatives/>
- **IPython:** interaktiv hisoblash muhiti. <http://ipython.org/>
- **bpython:** Unix-ga o'xshash ishlash uchun Python tarjimoniga ajoyib interfeys tizimlar:  
<https://www.bpython-interpreter.org>
- **Python tarixi, Guido van Rossum blogi:**  
<http://python-history.blogspot.com>

# Python bilan birinchi qadamlar

---

## MAZMUNI

2.1	Python o'rnatish .....	<a href="#">20</a>	2.1.1 Python tilini undan foydalanish orqali o'rganing .....	<a href="#">20</a>
			Python-ni mahalliy o'rnatish .....	<a href="#">20</a>
			Anaconda-ni o'rnatish .....	<a href="#">20</a>
			Python Online dan foydalanish .....	<a href="#">20</a>
			2.1.4 Python dasturini sinash. ....	<a href="#">21</a>
			2.1.5 Birinchi foydalanish. ....	<a href="#">22</a>
2.2	Interaktiv rejim. ....	<a href="#">23</a>		
	2.2.1 Chaqaloq qadamlari. ....	<a href="#">23</a>		
	2.2.2 Asosiy kirish va chiqish .....	<a href="#">23</a>		
	Chiqish: Chop etish .....	<a href="#">23</a>	Kirish:	
	Kirish .....	<a href="#">24</a>	2.2.3 Interaktiv	
	rejim haqida ko'proq ma'lumot. ....	<a href="#">24</a>	2.2.4 Matematik	
	amallar. ....	<a href="#">26</a>		
	-bo'lim. ....	<a href="#">27</a>	2.2.5 Python	
	qobig'idan chiqish. ....	<a href="#">27</a>	2.3 To'plam	
	rejimi. ....	<a href="#">27</a>	2.3.1	
	Sharhlar .....	<a href="#">29</a>	2.3.2	
	Chekish. ....	<a href="#">30</a>	2.4 Muharrir	
	tanlash .....	<a href="#">32</a>	2.4.1 Sublime	
	matn. ....	<a href="#">32</a>	2.4.2	
	Atom. ....	<a href="#">33</a>	2.4.3	
	PyCharm. ....	<a href="#">34</a>	2.4.4 Spyder	
	IDE. ....	<a href="#">35</a>	2.4.5 Tahrirlov	<a href="#">36</a>
	yakuniy so'zlar. ....	<a href="#">35</a>	2.5 Haqidagi	
	asboblar. ....	<a href="#">36</a>	2.6 Qo'shimcha	
	manbalar .....	<a href="#">37</a>	2.7 O'z-o'zini	
	baholash. ....	<a href="#">37</a>		

Ming kilometrlik sayohat bir qadamdan boshlanadi.

Lao Tzu

## 2.1 PYTHON O'RNATISH

---

### 2.1.1 Python tilini undan foydalanish orqali o'rganing

Ushbu bo'lim o'z dasturlaringizni ishga tushirish uchun Python-ni qanday o'rnatishni ko'rsatadi. Amalga oshirish orqali o'rganish - o'rganishning eng samarali usuli. Bu shunchaki kitobni (hatto bu kitobni) o'qishdan ko'ra yaxshiroqdir. Siz "Python interaktiv rejimi" ni shu ma'noda juda foydali deb topasiz, chunki u sizning savollaringizga kitob yoki qidiruv tizimidan tezroq javob berishi mumkin. Bonus sifatida Python interaktiv rejimidan olgan javoblaringiz aniq.

Shu sabablarga ko'ra men ushbu kitobni o'qishni davom ettirishdan oldin Python-ni o'rnatishni taklif qilaman.

### 2.1.2 Python-ni mahalliy sifatida o'rnatning

Python macOS va ko'pgina Linux distributivlarida oldindan o'rnatilgan. Windows-da Windows x86-64 veb-asoslangan o'rnatuvchini Python yuklab olish sahifasidan (<https://www.python.org/downloads/windows/>) yuklab olishingiz kerak. va keyin uni o'rnatning. Agar siz Windows dasturlarini o'rnatishga odatlangan bo'lsangiz, o'rnatish juda oddiy. O'rnatish faylini ikki marta bosing va Python o'rnatish ustasini ishga tushiring.

Standart sozlamalarni qabul qiling va Python bir necha daqiqada muammosiz o'rnatiladi.

Esda tutingki, rasmiy oýrnatuvchiga muqobil sifatida siz 16-sahifada koýrsatilgan Python distributivlaridan birini yuklab olishingiz va oýrnatishingiz mumkin. Hozir mening shaxsiy afzalligim Anaconda distributividir, lekin bu kitobga amal qilish uchun Anaconda-ni oýrnatishingiz shart emas, har qanday Python tarqatish buni amalga oshiradi.

#### Anaconda o'rnatilmoqda

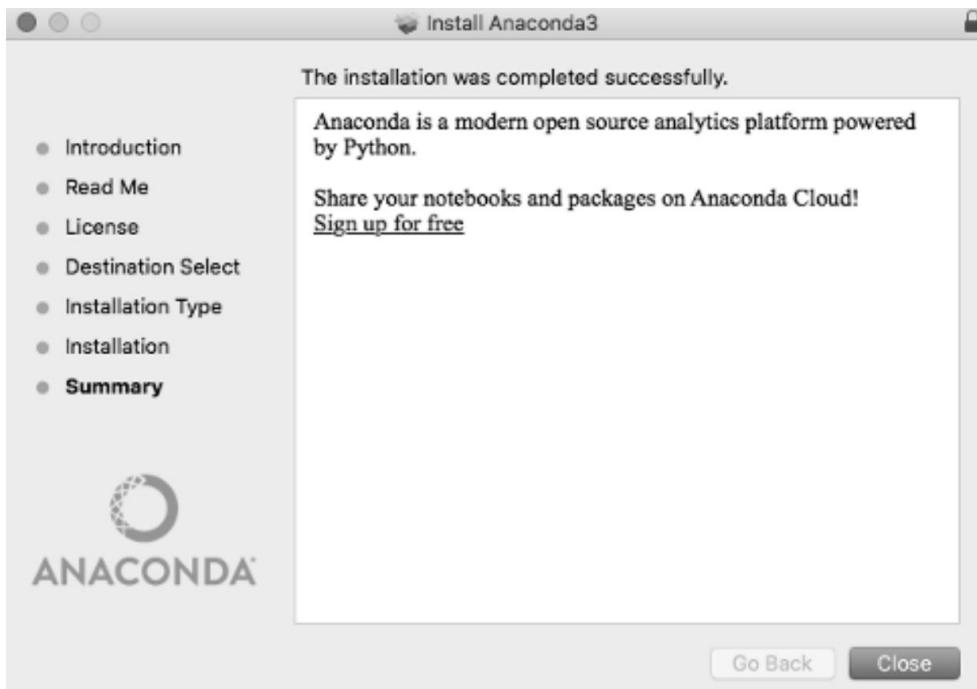
Quyidagi ko'rsatma macOS va Windows uchun keng tarqalgan (keyingi xatboshida Linuxni o'rnatishga qarang). Operatsion tizimingizga mos grafik oýrnatuvchini<sup>1</sup> yuklab oling va ustiga ikki marta bosing.<sup>2</sup> Unda “Anaconda 3 oýrnatuvchisiga xush kelibsiz” sarlavhali oýrnatuvchi koýrsatiladi. Kirish, Meni o'qing va Litsenziyada Davom etish tugmasini bosing. Agar siz uni faqat joriy foydalanuvchi uchun o'rnatmoqchi bo'lsangiz, Destination Select-da Davom etish tugmasini bosing. Keyingi qadam O'rnatish tugmasini bosishdir. O'rnatish bir necha daqiqa davom etadi va siz 2.1-rasmdagi kabi ekranni ko'rasiz . MacOS-da, o'rnatuvchini yopganingizda, o'rnatuvchini axlat qutisiga o'tkazish taklif etiladi, bunda faqat o'rnatuvchi o'chiriladi.

Anaconda-ni Linux-ga o'rnatish uchun Linux versiyasini <https://www.continuum.io/downloads#linux> manzilidan yuklab **oling** , Anaconda3-4.3.1-Linux-x86\_64.sh ga o'xshash nomga ega faylni olasiz. Terminalda ishga tushiring

```
$ bash Anaconda3-4.3.1-Linux-x86_64.sh
```

<sup>1</sup><https://www.continuum.io/downloads>

<sup>2</sup>Agar Mac dan foydalansangiz, macOS versiyasi 10.12.3 (Sierra) yoki undan keyingi versiya bo'lishi kerak.



## 2.1-rasm macOS-da Anaconda o'rnatilishi.

va ko'rsatmalarga rioya qiling. Agar siz oxirgi savolga "ha" deb javob bersangiz, sizning standart Pythoningiz Anaconda Python bo'ladi. E'tibor bering, ushbu o'zgarish yangi terminal ochganingizda kuchga kiradi.

### 2.1.3 Python Online dan foydalanish

Python-ni o'rganish yoki hatto dasturlarni ishga tushirish uchun sinab ko'rishning yana bir usuli - bu onlayn xizmatdan foydalanish. PythonAnywhere (<https://www.pythonanywhere.com>) Pythonning turli versiyalarini ishga tushirish imkonini beruvchi xizmatdir. Sizga brauzer va Internet ularishi kerak bo'ladi. PythonAnywhere Python skriptlarini onlayn ishlatalish uchun bepul xizmatni taqdim etadi. Ular cheksiz Python yoki Bash konsollarini, veb-ilovani joylashtirish uchun subdomen, 1 Gb xotira, ma'lumotlar bazalari va boshqalar bilan oyiga \$5 dan bir nechta rejalarga ega.<sup>3</sup> Ularning bepul "Boshlang'ich" xizmati tadqiqot va o'rganish uchun etarli.

Sinashga arziyidigan yana bir xizmat Microsoft Azure noutbuklaridir. Hozirda u oldindan ko'rish holatida (2017 yil iyun), lekin men uni bir necha oydan beri ishlatib kelmoqdaman va uni tavsiya qilish uchun etarlicha barqaror ko'rindi<sup>4</sup>. Bu siz joylashgan xizmat emas

<sup>3</sup>PythonAnywhere rejalar haqida qo'yshimcha maylumot olish uchun <https://www.pythonanywhere.com/narx/>.

<sup>4</sup>Siz ushbu platformadagi kitobdag'i kodni <https://notebooks.azure.com/py4bio/> orqali sinab ko'rishingiz mumkin. **libraries/py3.us**. Havolalar kitobning veb-sahifasida ham mavjud (<http://py3.us>)

## 22 Bioinformatika uchun Python

har qanday Python buyrug'iini yozish uchun Python konsoliga kirish, lekin u "Jupyter Notebook" dan foydalanadi, bu jonli kodli veb-sahifaga o'xshash veb-ilova. Bu dasturchi muhitini almashtirmaydi, lekin o'rganish va taqdimotlar uchun ishlatalishi mumkin. Oddiy Python dasturlarini ishga tushirishingiz mumkin bo'lgan yana bir veb-xizmat (masalan, ushbu kitobning birinchi besh bobida keltirilgan) bu Rep.it (<https://repl.it/languages/python3> ).

### 2.1.4 Python-ni sinovdan o'tkazish

Python o'rnatilgandan so'ng, uning ishlashiga ishonch hosil qilishingiz kerak. Windows-da Python belgisini ikki marta bosing. Linux va macOS foydalanuvchilari terminalni ochishi va keyin python yozishi mumkin.

Siz shunday ekranni ko'rishingiz kerak: 6

```
Python 3.6.0 |Anaconda 4.3.1 (64-bit)| (2016 yil 23 dekabr, 12:22:00)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] Linuxda Qo'shimcha
ma'lumot uchun "yordam", "mualliflik huquqi", "kreditlar" yoki "litsenziya" ni kriting.
>>>
```

Bu Python konsoli<sup>7</sup> va u interaktiv rejimda dasturlash uchun ishlataladi . Ushbu rejim keyingi bo'limda tushuntiriladi.

### 2.1.5 Birinchi foydalanish

Python-dan foydalanishning ikki yo'li mavjud: interaktiv va ommaviy rejim. Ushbu usullar bir-birini to'ldiradi va ular turli maqsadlarda qo'llaniladi. Interaktiv rejim dasturchiga har bir ko'rsatmaga darhol javob olish imkonini beradi. Ommaviy rejimda ko'rsatmalar bir yoki bir nechta fayllarda saqlanadi va keyin bajariladi. Interaktiv rejim asosan kichik testlar uchun ishlataladi, aksariyat dasturlar esa ommaviy rejimda ishlaydi.

Interaktiv rejim python-ni ishga tushirish orqali yoki Spyder, PyCharm, IDLE va boshqalar kabi ba'zi Python ed itorlarida ishga tushirilishi mumkin. Bundan tashqari, <https://repl.it/languages/python3> kabi saytlarda onlayn foydalanish mumkin yoki <https://www.pythonanywhere.com/>. Men Python-ni onlayn ishlatalishdan ko'ra o'z kompyuteringizga o'rnatishni tavsiya qilishman.

Agar Anaconda-dan foydalansangiz, Anaconda Navigator9-ni ishga tushiring va QTConsole-ni tanlang; ichida bu holda Python interaktiv rejimi 2.2-rasmdagi kabi ko'rindasi .

Agar siz Anaconda dan foydalanmasangiz, terminalingizdan yoki buyruq satridan python yozing.

<sup>5</sup> MacOS operatsion tizimida terminal llovalar/Utilitalar jildida joylashgan.

<sup>6</sup>Bu chiqish Python versiyasiga, asosiy ishiga qarab tizimdan tizimga farq qilishi mumkin tizim va kompilyatsiya paytida o'rnatilgan variantlar.

<sup>7</sup>Texnik nomi REPL, o'qish-baholash-chop etish tsiklidan olingan, ammo ko'pchilik uni Python deb ataydi. tarjimon, Python qobig'i yoki Python terminali

<sup>8</sup> Agar siz administrator yoki ildiz huquqlariga ega bo'lman mashinada bo'lsangiz va Python-ni o'rnatolmasangiz, onlayn variant yaxshi alternativ hisoblanadi.

<sup>9</sup>Anaconda Navigator belgisini toping yoki buyruq satridan ishga tushiring anakonda-navigator.

Jupyter QtConsole 4.2.1  
 Python 3.6.0 |Anaconda custom (x86\_64)| (default, Dec 23 2016, 13:19:00)  
 Type "copyright", "credits" or "license" for more information.  
 IPython 5.1.0 -- An enhanced Interactive Python.  
 ? -> Introduction and overview of IPython's features.  
%quickref -> Quick reference.  
help -> Python's own help system.  
object? -> Details about 'object', use 'object??' for extra details.

In [1]: |

Shakl 2.2 Anaconda Python interaktiv terminali.

Keling, interaktiv rejimdan foydalanib, Python asoslarini bilib olaylik.

## 2.2 INTERFAOL REJIM

---

### 2.2.1 Chaqaloq qadamlari

Quyidagi kod tarjimonga “Salom dunyo!” qatorini chop etish uchun qanday buyruq berishni ko'rsatadi

```
>>> chop ('Salom dunyo!')  
Salom Dunyo!
```

Uchta kattaroq belgilarga e'tibor bering (>>>); bu interaktiv rejimning Python so'rovi. U allaqachon mavjud, uni kiritishingiz shart emas. Bu shuni anglatadiki, Python buyruqlarni bajarishga yoki ifodalarni baholashga tayyor.

### 2.2.2 Asosiy kirish va chiqish

Chiqish: Chop etish

Python 3 dan chop etish funksiyadir. Funktsiya - bu ma'lum bir vazifani bajarishi mumkin bo'lgan qayta ishlataladigan kod. Har bir funktsiya parametrlar deb ataladigan bir yoki bir nechta qiymatlarni qabul qilishi m... Print("Salom Dunyo!") holatida funktsiya nomi chop, parametr esa "Hello World!" qatoridir. Biz 6- bobda ba'zi tafsilotlar bilan funktsiyalarni ko'rib chiqamiz .

Chop etish funksiyasi bir nechta elementlarni qabul qilishi mumkin:

```
>>> chop ('Salom', 'Dunyo!')
```

10Dasturchilar orasida “Salom dunyo” satrini bosib chiqarish orqali til qanday ishlashini ko'rsatish odati bor. Python dasturchilari bu odatdan himoyalanganmagan. Ushbu bayonotni interaktiv rejimda yozganingizda nima bo'lishini ko'ring: import \_\_hello\_\_.

## 24 Bioinformatika uchun Python

Salom Dunyo!

Odatiy bo'lib, u bo'sh joy bilan ajratilgan barcha satrlarni chop etadi, lekin siz ajratuvchini sep nomli parametr bilan o'zgartirishingiz mumkin:

```
>>> chop ('Salom', 'Dunyo!', sep=',')  
Salom Dunyo!
```

Chiqishni faylga yo'naltiring:

```
>>> print("Salom", "Dunyo!", sep=",", file=filehandle)
```

Fayllarni qanday boshqarishni 6- **bobda ko'rib chiqamiz**.

Chiqishdagi oxirni o'zgartirish uchun end parametridan foydalaning. Oxirini o'zgartirish bu holda chiqish ikkita tashuvchining qaytishini qo'shadi (**yoki** kriting):

```
>>> print("Salom", "Dunyo!", sep=";", end="\n\n")  
Salom Dunyo!
```

### Kirish: kirish

Ishlayotgan dasturga ma'lumotlarni kiritish uchun siz kiritishdan foydalanishingiz mumkin. Quyidagi buyruq foydalanuvchidan ma'lumotlar qatorini oladi va uni nom deb nomlangan o'zgaruvchiga qaytaradi. Quyidagi kodda satr kiritilgandan so'ng o'zgaruvchi kiritiladi va o'zgaruvchining mazmuni ko'rsatiladi:

```
>>> name = input("Ismingizni kriting:")  
Ismingizni kriting: Seba  
>>> nomi  
"Seba"
```

Ko'pincha siz kiritish funktsiyasidan foydalanmaysiz, chunki ma'lumotlarni kiritishning ko'proq amaliy usullari mavjud, masalan, uni fayldan, veb-sahifadan yoki boshqa dasturning chiqishidan o'qish.

#### 2.2.3 Interaktiv rejim haqida batafsil

Interaktiv rejim kalkulyator sifatida ishlatalishi mumkin:

```
>>> 1+1  
2
```

Satrlarda "+" ishlatilsa, u birlashmani qaytaradi:

```
>>> '1'+'1'
'11'
>>> "Bir qator" + "belgilar"
"Bir qator belgilar"
```

E'tibor bering, bitta (' ) va qo'sh ( " ) qo'shtirnoqlar, agar ular izchillik bilan qo'llanilsa, noaniq tarzda ishlatalishi mumkin. Ya'ni, satr ta'rifi bir turdag'i qo'shtirnoq bilan boshlangan bo'lisa, u bilan yakunlanishi kerak. bir xil turdag'i iqtibos.

Turli xil ma'lumotlar turlarini qo'shib bo'lmaydi:

```
>>> 'Javob ' + 42
Traceback (eng oxirgi qo'ng'iroq):
  Fayl "<stdin>", 1-qator, <modul> da
TypeError: int emas, str bo'lishi kerak
```

Faqat bir xil turdag'i elementlarni qo'shish mumkin. Buni satrlar yig'indisiga aylantirish uchun raqamni satrga aylantirish kerak; Bu str() funksiysi bilan amalga oshiriladi:

```
>>> 'Javob ' + str(42)
"Javob 42"
```

Xuddi shu natijani “String formatlash operatsiyalari” bilan arxivlash mumkin:

```
>>> 'Javob: {}'.format(42)
"Javob 42"
```

E'tibor bering, qarama-qarshi transformatsiya (dan o'rniga satrdan butun songa). integer to string) int() funksiysi bilan amalga oshirilishi mumkin:

```
>>> 1 + '1'
Traceback (eng oxirgi qo'ng'iroq):
  Fayl "<stdin>", 1-qator, <modul> da
TypeError: +: 'int' va 'str' >>> 1 + int('1') uchun qo'llab-quvvatlanmaydigan operand
turlari
2
```

Siz har qanday Python elementiga nom berishingiz va keyinroq ularga murojaat qilishingiz mumkin:

```
>>> raqam = 42
>>> 'Javob: {}'.format(raqam)
"Javob 42"
```

---

**113- bobda satrlarning batafsil tavsifi berilgan.**

12 Qo'shimcha ma'lumot olish uchun (<http://www.python.org/dev/peps/pep-3101>) PEP-3101 ni o'qing .

## 26 Bioinformatika uchun Python

## 2.1-JADVAL Arifmetik uslubdagi operatorlar

Belgi tavsifi	
+	Qo'shimcha
-	Ayirish
*	Ko'paytirish
/*	Bo'lim
**	Eksponentsiya
%	Modul (qolgan)

Ismalar faqat harflar, raqamlar va pastki chiziqdandan iborat bo'lishi kerak (\_), lekin ular bo'lishi mumkin emas raqamlardan boshlang. Boshqa dasturlash tillarida nomlar o'zgaruvchilar deb ataladi.

Diqqatga sazovor bo'lgan buyruq dir(), u joriy muhitda mavjud bo'lgan nomlarni aytadi. Sinab ko'ring:

```
>>> dir()
['__builtins__', '__doc__', '__loader__', '__name__', '<='__paket__',
 '__spec__', 'raqam']
```

\_\_name\_\_ shaklidagi nomlar maxsus bo'lib, ular keyinroq muhokama qilinadi, lekin siz belgilagan barcha o'zgaruvchilar nomlari borligini ko'ring.

## 2.2.4 Matematik amallar

Python qobig'ida har qanday standart matematik operatsiyani bajarish mumkin:

```
>>> 12*2
24
>>> 30/3
10.0
>>> 2**8/2+100
228.0
```

Ikkita yulduz (\*\*) "kuchga ko'tarilgan" degan ma'noni anglatadi va teskari chiziq (/) bo'llinish operatsiyasi. Demak, bu ifoda quyidagi ma'noni bildiradi Python 100/21 dan valda qo'llab-quvvatlanadigan arifmetik uslubdagi operatorlar ro'yxati keltirilgan.

E'tibor bering, operator ustunligi matematikada ishlataladigan bilan bir xil. Oson yo'il esda tutingki, ustuvorlik tartibi PEMDAS qisqartmasi bilan:

P Qavslar eng yuqori ustunlikka ega va ifodani baholash tartibini belgilash uchun ishlataladi. Shuning uchun  $2 * (3-2)$  2 ni va  $(3-1) ** (4-1)$  8 ni beradi. Qavslar iboralarni o'qishni osonlashtirish uchun ham ishlatalishi mumkin.

E Ko'rsatkichlar ikkinchi o'rinda turadi, shuning uchun  $2^{**}2+1$  8 emas, 5 ga teng.

MD ko'paytirish va bo'llinish bir xil ustunlikka ega.  $2^{*2}-1$  o'rniغا 3 tani beradi dan 2.

**AS Qo'shish va ayirish ham bir xil (oxirgi) ustuvorlik tartibiga ega.**

Va nihoyat, bir xil ustunlikka ega operatorlar chapdan o'ngga qarab baholanadi.  
Demak,  $60/6*10$  1 emas, 100 ni beradi.

## Bo'lim

Hozirgi vaqtda ikkita bo'linish operatori mavjud. Standart, / haqiqiy bo'linish deb ataladi va bo'linishning matematik natijasini qaytaradi:<sup>13</sup>

```
>>> 10/4
```

2.5

Qavatga bo'linish deb ataladigan // ham bor, u butun son qismini qaytaradi bo'linish:

```
>>> 10//4 2
```

## 2.2.5 Python qobig'idan chiqish

Python qobig'idan chiqish uchun MacOS yoki Linuxda CRTL-D dan foydalaning (ya'ni Control va D tugmalarini bir vaqtning o'zida bosing). Windows-da CTRL-Z va Enter ni bosing. Har qanday operatsion tizimda ishlaydigan yana bir muqobil variant - exit() funksiyasidan foydalansing.

```
$ python
```

Python 3.5.1 |Anaconda 2.4.1 (64-bit)| (2015 yil 7 dekabr, 11:16:01)

[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] Linuxda Qo'shimcha

ma'lumot uchun "yordam", "mualiflik huquqi", "kreditlar" yoki "litsenziya" ni kriting. >>> exit() \$

## 2.3 BATCH REJIM

---

Interfaol tarjimon juda foydali bo'lsa-da, notrivial dasturlarning aksariyati fayllarda saqlanadi. Interaktiv seansda ishlatiladigan kodga faqat seans faol bo'lganda kirish mumkin. Har safar interaktiv seans yopilganda, kiritilgan barcha kodlar o'chib ketadi. Kodning barqarorligi uchun dasturlar matnli fayllarda saqlanadi. Dastur interaktiv tarjimonda satr satr emas, balki shunday matn faylidan bajarilsa, u ommaviy ish rejimi deb ataladi. Bular odatda ".py" kengaytmali oddiy matnli fayllardir. Bu fayllarni istalgan standart matn muharriri yordamida yaratish mumkin.<sup>14</sup>

<sup>13</sup>Python 2.x da bo'linish belgisi (/) bo'linishning butun qismini qaytarish uchun ishlatiladi.

<sup>14</sup>Python dasturlash uchun har qanday matn muharriridan foydalananish mumkin, ammo umumiylar muharriri o'rniiga maxsus matn muharriridan foydalananish tavsiya etiladi. Ushbu bobning oxirida muharrir tanlashga bag'ishlangan bo'lim mavjud.

## 28 Bioinformatika uchun Python

Unix-ga o'xshash tizim ostidagi Python skriptlarining ixtiyoriy xususiyati Python tarjimoniga yo'l bo'lgan birinchi qatordir. Agar Python tarjimon /usr/bin/python (Linux'dagi odatiy joy) da joylashgan bo'lsa, birinchi qator quyidagicha bo'ladi:

```
#!/usr/bin/python
```

Bu shebang deb ataladi va bu operatsion tizimga dastur uchun tarjimon nima ekanligini bilish imkonini beruvchi Unix konvensiyasidir va bu tarjimon foydalanuvchi python tarjimonini aniq chaqirmasdan turib bajarilishi mumkin . dasturni qobiq skripti sifatida bajarishga harakat qilish uchun operatsion tizim.

Aytaylik, sizda juda oddiy dastur bor:

**Listing 2.1: hello.py: "Salom dunyo!" dastur****1 ta bosma ("Salom dunyo!")**

Raqamni boshida yozmaslikni unutmang, u faqat ma'lumot uchun mavjud. Ushbu dastur Python tarjimonining argumenti sifatida chaqirilgan taqdirdagina buyruq satridan ishlaydi:

```
$ python hello.py
Salom dunyo!
```

Ammo agar siz uni mustaqil dastur sifatida ishga tushrimoqchi bo'lsangiz, siz shunga o'xshash narsani ko'rasisiz:

```
$ ./hello.py .
hello.py: 1-qator: kutilmagan token yaqinidagi sintaksis xatosi <=
"Salom dunyo!"
./hello.py: 1-qator: 'print('Salom dunyo!')
```

Ushbu xato xabari dasturni tizim skripti sifatida (Python-ni chaqirmasdan) bajarishga urinayotganda qobiq tomonidan yuboriladi. Dasturning birinchi qatorini tahrirlash orqali uni oldini olish mumkin:

**Listing 2.2: hello2.py: Salom dunyo! shebang bilan**

```
1#!/usr/bin/python 2
chop ("Salom dunyo!")
```

Ushbu versiya bajariladigan ikkilik fayl sifatida ishlaydi:<sup>16</sup>

<sup>15</sup>Bir nechta versiya o'rnatilgan bo'lsa, ma'lum bir Python versiyasini tanlash uchun tarjimon yo'lini belgilashingiz mumkin.

<sup>16</sup>Linux va macOS da siz faylning bajariladigan ruxsatiga ega ekanligiga ishonch hosil qilishingiz kerak, ya'nini chmod a+x hello.py bilan bajarildi .

```
$ ./hello2.py
Salom Dunyo!
```

Agar siz maxsus tarjimon o'rniga birinchi mavjud Python tarjimonini chaqirmoqchi bo'lsangiz, `#!/usr/bin/env python` dan foydalaning. Tarjimonda ma'lum bir Pythonga yo'lni faqat dasturingizni ma'lum bir versiya bilan ishga tushirishni xohlaganingizda foydalaning (masalan, `/usr/bin/python2.7`).

Windowsda bu qator e'tiborga olinmaydi, chunki tarjimon fayl kengaytmasi (`.py`) bo'yicha ishga tushiriladi.

Python matn muharriri ichidan ham bajarilishi mumkin, agar muharrirda bu funksiya mavjud bo'lsa. Ko'pgina muharrirlarda siz F5 tugmasi bilan dasturni ishga tushirishingiz mumkin.

Odatda Python kodida topiladigan yana bir maxsus izoh bu "kodlash izohi"dir. Ushbu qator hujjatning qolgan qismi uchun belgilar kodlashni belgilaydi va u quyidagi shaklni oladi:

```
# -*- kodlash: KODLASH -*-
```

Bu erda KODLASH, masalan, ascii, latin1, 8859-1, UTF-8 va boshqalar bo'lishi mumkin. Shunday qilib, UTF-8 da kodlangan belgilar bilan manba kodini kodlash qatori quyidagi qatorga ega bo'ladi:

```
# -*- kodlash: UTF-8 -*-
```

Sharhni kodlashsiz, Python tahlilchisi UTF-8 (yoki ASCII) ni qabul qiladi. Python 3-versiyadan oldin).

### 2.3.1 Sharhlar

Agar siz ushbu dasturni sintaksik rang berish qobiliyatiga ega har qanday muharrirda sinab ko'rsangiz, birinchi qator (`#!/usr/bin/python`) boshqa rangga ega ekanligini ta'kidlagan bo'lishingiz mumkin. Bu "#" belgisidan foydalanish bilan bog'liq. Bu belgi Python uchun a U tarjimon tomonidan bajarilmagan satrlarni aniqlash uchun ishlataladi. Natijada, bu satrlar "sharhlar" deb ataladi. Sharhlar dasturga funksionallik qo'shmaydi, balki kodni ishlab chiquvchiga yoki boshqa o'quvchilarga yordam beradi. Keling, oldingi kodni sharh bilan qayta yozamiz:

**Listing 2.3: Salom dunyo! sharhlar bilan**

---

```
#!/usr/bin/env python #
Keyingi qatorda "Salom Dunyo!" qatori chop etiladi. chop
("Salom dunyo!")
```

Ushbu maxsus koddagi sharh juda ma'nosiz, chunki "chop etish" funktsiyasi haqida hech qanday shubha yo'q. Boshqa dasturlarda unchalik oson bo'limgan kod mavjud

### 30 Bioinformatika uchun Python

tushunish va qayerda izoh kodni o'qishni yaxshilashi mumkin. Izohlarni siz nazarda tutayotgan koddan oldin qo'yish odatiy holdir. Sharhlar asosan boshqa birovga bizning kodimizni tushunishga yordam berish uchun qilingan, lekin ular hatto yozgandan keyin kodni ko'rgan va muntazam ish maqsadini eslamaydigan o'sha dasturchi uchun ham foydali bo'lishi mumkin.

Sharhlar kodning bir qismini o'chirish uchun ham ishlatalishi mumkin (bu dasturlash jargonida "comment out" deb ataladi). Bu odatda disk raskadrovska maqsadida amalga oshiriladi. Vazifani bajarish uchun muqobil kodlarni sinab ko'rayotganda, qaysi koddan foydalaningizga ishonchingiz komil bo'limguncha, kodning faol bo'limgan qismiga ega bo'lish yaxshiroqdir. O'chirilgan narsani qayta yozishdan ko'ra, faol bo'limgan kodni izohdan chiqarish osonroq. Bu shunday keng tarqalgan vazifikasi, barcha Python muharrirlarida matnlarning butun blokini izohlash yoki izohdan chiqarish vositalari mavjud.<sup>17</sup>

Maslahat: Python-da kengaytmalar.

**Python fayllari .py kengaytmasiga ega, ammo Python bilan bog'liq boshqa kengaytmalarni ham topishingiz mumkin:**

- **py:** standart Python fayllari.
- **pyc:** "Tuzilgan" Python fayllari. Python modulini birinchi marta import qilganingizda, u bayt kodiga kompilyatsiya qilinadi, shuning uchun keyingi safar u tezroq boshlanadi. **Compileall** modulidagi **compile\_dir** funksiyasi bilan kompilyatsiya Python'dan majburiy bo'lishi mumkin. E'tibor bering, .pyc fayllari tezroq yuklanadi, lekin tezroq ishlamaydi.
- **pyo:** "Optimallashtirilgan" kod. Bu Python tarjimonini -o bayrog'i bilan ishga tushirish orqali yaratiladi. Nomga aldanmang, aksariyat kodlar -o bayrog'i **yoqilgan bo'lsa** ham bir xil tezlikda ishlaydi.
- **pyw:** Bu kengaytmali standart Python fayli bo'ylib, Windows ularni python.exe o'rniga pythonw.exe bilan bajarishga majbur qiladi. Pythonw.exe DOS konsolini ishga tushirmaydi, shuning uchun Win dows ostidagi grafik dasturlar uchun afzalroqdir.

#### 2.3.2 Chiziq

**Python haqida dasturiy ta'minot ishlab chiquvchilari uchun birinchi o'rinda turadigan narsalardan biri bu uning kodini kiritish tizimidir. Dasturchi bo'limganlar bu vaqtida manba kodining chekinishi nima ekanligini bilishlari kerak. Mana, cheklanmagan ba'zi C kodlari:**

```
if (attr == -1){while (x<5)
{ printf("Kutilmoqda...\n");kutish(1);
```

<sup>17</sup>Python standart muharririda (IDLE) bu vosita Format menyusiga ostida joylashgan.

```
x = x+1;}printf("Hammasi joyida\n");} else
{printf("Xatolik bor\n");}
```

Xuddi shu dastur qismining (yoki "kod snippeti") bo'g'inlangan versiyasi:

```
if (attr == -1) { while
    (x<5)
        { printf("Kutilmoqda...\n");
            kuting (1); x = x+1;

        printf("Hammasi yaxshi\n");} else
{ printf("Xatolik bor\n");}
```

C ni bilmasdan ham, ikkinchi dastur birinchisidan ko'ra ko'proq o'qilishi mumkin deb ayta olamiz. C yoki Java kabi dasturlash tilida ob'ekt sifatida bajariladigan kod bloklari qavslar bilan ajratiladi. Shunday qilib, tarjimon buni biladi, masalan, printf("Hammasi joyida\n"); if tuzilmasi ichida, lekin while ichida emas. Elementlar orasidagi mantiqiy munosabatlar cheksiz dasturga qaraganda aniqroq bo'ladi. Python-da quyidagi kod parchasini tekshiring, bu erda qavslar yo'q, lekin chekinish bilan belgilangan kod bloklari mavjud.

```
agar attr==1:
    x<5 esa:
        print("Kutilmoqda...")
        kuting(1)
        x = x + 1
    print("Hammasi yaxshi")
boshqa
    chop etish ("Xatolik bor")
```

Agar siz ushbu dasturni tushunmasangiz, hozir bu muhim emas. Ushbu misoldan maqsad tilning eng yorqin jihatlaridan birini ko'rsatishdir. Bu afzallik deb hisoblanadi, chunki kodning tuzilishi etarlicha aniq bo'lsa, kodlash xatolarini kiritish imkoniyati kamroq bo'ladi. Ba'zilar kodni shu tarzda saqlash zerikarli deb aytishadi, ammo bu unday emas. Ko'pgina matn muharrirlari kodni kiritish bilan avtomatik tarzda shug'ullanadi, shuning uchun dasturchiga hech qanday yuk bo'lmaydi. Majburiy chiziqqa yana bir tanqid - kodning chuqr joylashishi; ba'zi bayonotlar o'ng tomonda joylashgan. Juda ko'p darajali chekinishlar (masalan, modulli kod yozish) bilan kod yozishni oldini olish uchun dasturlash vositalari mavjud. Ushbu vositalardan to'g'ri foydalanish kerakli mahoratga ega bo'lib, u siz foydalanadigan dasturlash tiliga bog'liq emas.<sup>18</sup> Dasturchini majburlash

---

<sup>18</sup>Linus Torvalds, Linux yadrosini yaratuvchisi, "Agar sizga 3 dan ortiq darajali chekinish kerak bo'lsa, siz baribir ishlamay qoldingiz va dasturingizni tuzatishingiz kerak", dedi.

## 32 Bioinformatika uchun Python

girintidan foydalanish Python dizayn falsafasining bir jihatiga mos keladigan xususiyatdir: O'yqish mumkinligi hisoblanadi.<sup>19</sup> Oliver Fromme "Python: Indentatsiya haqidagi afsonalar" asarida yozganidek:<sup>20</sup> "Python sizni baribir ishlatgan boylgan chekinishdan foydalanishga majbur qiladi. Siz dasturning tuzilishini chalkashtirmoqchi edingiz.

## 2.4 MUHARRIRINI TANLASH

Asosan, Pythonda dasturlash uchun har qanday matn muharriridan foydalanish mumkin. Python dasturlash uchun mo'ljallangan muharrirdan foydalanish ko'p narsaga ega bo'lsa-da, bloknotda (agar siz juda moyil bo'lsangiz) yoki biron bir engil matn muharririda dasturlashingizga hech narsa to'sqinlik qilmaydi.

Muharrir tanlash arzimas masala emas; aslida bu dasturiy ta'minot ishlab chiquvchilar o'ttasida "muharrirlar urushi" ga olib keladigan darajada bahsli masala.<sup>21</sup> Bu kurashishga arziyidigan narsa bo'lmasisligi mumkin, ammo ehtiyojlaringiz uchun eng yaxshi muharrirni tanlash unumdorligingizni oshirishi mumkin.

Quyida Python dasturchilari tomonidan ishlatiladigan mashhur muharrirlarning qisqacha sharhi keltirilgan.

### 2.4.1 Ajoyib matn

Sublime Text<sup>22</sup> eng ko'p ishlatiladigan matn muharrirlaridan biri hisoblanadi. Uni bir necha soat ishlatganingizdan so'ng, nima uchun ekanligini tushunish oson. Bu nozik, tez va kuchli, xususiyatlarni bir xil dasturda osongina topib bo'lmaydi. Foydalanuvchi interfeysi (UI) minimalist, ammo asabiyashmaydi. Sintaksisni ta'kidlash uchun ishlatiladigan standart ranglar palitrasasi ko'zni quvontiradi. Unda ko'nikkaningizdan ko'ra bir nechta yoqimli funksiyalar mavjud bo'lsa, bu funksiyalarsiz boshqa muharrirga o'tishni xohlamaysiz. Ba'zi taniqli xususiyatlar:

- **Minimap:** Muharrirning o'ng tomonida butun hujatning umumiyo ko'rinishi mavjud va ishlab chiquvchi uning ustida aylanib, kodning katta qismlarini tezda tekshirishi mumkin.
  
- **Tez global qidiruv:** Katta kod bazasi bilan ishlaydigan ishlab chiquvchilar uchun keng tarqalgan muammo barcha loyiha fayllarida matn qidirishdir. Find va grep kabi buyruq qatori yordam dasturlariga murojaat qilish o'rniiga , uni tahrirlovchidan chiqmasdan tez va intuitiv tarzda bajarish imkoniyati mavjud.
  
- **Ishlash:** Ulug'vor matn barcha hisoblar bo'yicha tezdir (boshlash vaqt, kechikish nol). Katta hajmdagi fayllarni ochganda ham muharrir sizni pastga tortmaydi.
  
- **Ustun tanlash:** Bu bizga ustun ko'rinishidagi matn qismlarini tanlash imkonini beradi va ustunni tanlagandan so'ng bir nechta qo'shish nuqtalari paydo bo'ladi, shuning uchun siz bir vaqtning o'zida bir nechta pozitsiyalarga matn kiritishingiz mumkin.

19 Iltimos, qarang: <http://www.python.org/dev/peps/pep-0020> ko'rsatma haqida ko'proq ma'lumot olish uchun Python dizayni uchun printsiplar.

<sup>20</sup>[http://www.seonetix.de/~olli/Python/block\\_indentation.hawk](http://www.seonetix.de/~olli/Python/block_indentation.hawk).

<sup>21</sup>Qarang: [https://en.wikipedia.org/wiki/Editor\\_war](https://en.wikipedia.org/wiki/Editor_war).

<sup>22</sup>[Https://www.sublimetext.com](https://www.sublimetext.com) saytida mavjud .

- **Kengaytiriladigan:** pliginlardan foydalanib, muharrir funksiyalarini ehtiyojlarigizga mos ravishda kengaytirishingiz mumkin. Foydalanuvchiga Sublime-dan chiqmasdan boshqa pliginlarni qidirish, yuklab olish va o'rnatish imkonini beruvchi Package Control<sup>23</sup> deb nomlangan plugin mavjud . Plugin Python-da yozilgan bo'lib, bu muharrir nima uchun Python dasturchilari orasida mashhurligini ham tushuntiradi.
- **Truly multiplatform:** Sublime uchta asosiy platformada bir xil ko'rindi. Har bir operatsion tizim UI ko'rsatmalariga rioya qilish uchun ba'zi asosiy yorliqlar o'zgaradi.

Ko'p jihatdan, Sublime dasturiy ta'minotni ishlab chiqish uchun eng yaxshi matn muharriridir. Garchi uning ba'zilar uchun kelishuvni buzishi yoki boshqalar uchun bezovta bo'lishi mumkin bo'lgan muammo bo'lsa ham: Sublime yopiq kodli dasturiy ta'minot va undan foydalanish uchun tijorat litsenziyasi kerak. Agar siz litsenziyani to'lay olsangiz (hozirda \$70) va yopiq kodli dasturiy ta'minot bilan bog'liq muammo bo'lmasa, bu ideal ko'p maqsadli matn muharriri bo'lishi mumkin.

## 2.4.2 Atom

Atom - GibHub foydalanuvchilari tomonidan yaratilgan matn muharriri.<sup>24</sup> Ushbu muharrirning g'oyasi mahsulotni Sublime va TextMate kabi qulay, lekin Emacs va Vim kabi kengaytiriladigan va moslashuvchan qilishdir. Avvaliga Atom Sublimega o'xshaydi. Bu tasodif emas; ularning yaratuvchilari foydalanuvchilarini jalb qilish uchun Sublime UI-dan nusxa ko'chirishdi. Uldan tashqari, ko'pgina funksionallik va yorliqlar saqlanadi, shuning uchun o'tish oson bo'lishi kerak. Asosiy texnologiya kabi ba'zi farqlar mavjud. Sublime yadroси kengaytma tili sifatida Python bilan C++ da yaratilgan boylsa, Atom Chromium (Google Crome'ning ochiq manbali versiyasi) asosida yaratilgan. Tezlik farqi sezilarli. Atom sekinroq, shuning uchun uni eski yoki quvvatsiz mashinalar bilan ishlatish tavsiya etilmaydi. Chromium-dan foydalanishning afzalligi shundaki, muharrirni JavaScript, HTML va CSS yordamida osongina sozlash mumkin. Bu Atomni veb-ishlab chiquvchilar uchun ajoyib muharrir qiladi. Atom GitHub tomonidan ishlab chiqarilganligi sababli, u boshqa tahrirlovchilarda ko'rilmagan Git integratsiyasiga ega<sup>25</sup> .

- **Aqli avtoto'Idirish:** Python kabi talqin qilinadigan tillarda avtomatik to'Idirish mukammal emas, lekin siz yozish paytida Atom mos opsiyalarni taklif qilishda yaxshi ish qiladi.
- **Modulli dizayn:** U asosiy funksiyalar bilan birga keladi va keyin sozlamalar sahifasidan paketlarni o'rnatadi. Paket tanlovi (6200 dan ortiq) Sublime-ga juda o'xshaydi va ko'p ishlatiladigan paketlar allaqachon Atomga ko'chiriladi. Paketlardan tashqari, tashqi ko'rinish va his-tuyg'ularni boshqarish uchun mavzular mavjud va hozirda 2100 dan ortiq mavzular bilan siz zerikmaysiz.

---

23Paket nazoratini <https://packagecontrol.io> saytidan yuklab oling. 24<https://atom.io> 25Atomdan

chiqmasdan eng keng tarqalgan Git operatsiyalariga kirishingiz mumkin. Qo'shimcha ma'lumot uchun <http://blog.atom.io/2017/05/16/git-and-github-integration-comes-to-atom.html> ni o'qing.

## 34 Bioinformatika uchun Python

- Ochiq manba: Siz GitHub-da manba kodiga kirishingiz va dasturchilar hamjamiyatining bir qismi bo'lisingiz mumkin. Bu holda ochiq manba ham bepul degan ma'noni anglatadi. Rivojlanish GitHub tomonidan homiylik qilinadi.
- Haqiqatan ham multiplatforma: Sublime-da bo'lgani kabi, u uchta asosiyda ham bir xil ko'rinati platformalar.

Sublime litsenziyasi uchun pul to'lay olmasangiz yoki xohlamasangiz, Atom eng yaxshi alternativ hisoblanadi. Bu mukammal emas. Ko'p hollarda tezlik muammodir. Katta hujjat bilan u RAMni shunday iste'mol qiladiki, bu sizning kompyuterizingizni javob bermaguncha sekinlashtirishi mumkin. Shubhasiz, Atom sinab ko'rishga arziydi, uning sustligini bilish uchun uni uskuna va sozlamalaringiz bilan sinab ko'ring.

### 2.4.3 PyCharm

Bu Python muharriri asosan tijorat/professional sozlamalarda ishlataladi. Yangi "taylim" versiyasi mavjud.<sup>26</sup> U oyzini "Python bilan dasturlashni o'rganish uchun bepul, oson va professional vosita" sifatida eylon qiladi. Ushbu nashr bepul va kurslarni yaratish va tarqatishni qo'llab-quvvatlaydi, shuning uchun u kurslar kutubxonasi bilan birga keladi va siz boshqa foydalanuvchilar tomonidan yaratilgan materiallarga ham kirishingiz mumkin. Men undan Python tilini o'rganish uchun foydalanmadim, shuning uchun u bu maqsad uchun mos yoki yo'qligini aya olmayman, lekin ilg'or foydalanuvchi sifatida bu ta'lim jihatni mening oddiy dasturchi ish jarayonimga ta'sir qilmasligini aniqladim, aslida bu menga yordam beradi. har safar yaxshilanish uchun joy b Ushbu ta'lim nashri turli veb-ishlab chiqish texnologiyalarini, masofaviy ishlab chiqish imkoniyatlarini yoki qo'shimcha tillarni qo'llab-quvvatlamaydi.

Ushbu xususiyatlar PyCharm-ga kiritilgan:

- Intelligent Python Assistance: Aqlii kodni to'ldirish, kodni tekshirish, xatoni tezda aniqlash va tezkor tuzatish, shuningdek, avtomatlashtirilgan kodni qayta ishlash. Ushbu oxirgi nuqta professionallar tomonidan juda qadrlanadi, xatoni ta'kidlash esa yangi boshlanuvchilar uchun foydalidir. Misol uchun, agar siz modulni import qilsangiz va fayl orqali foydalanmasangiz, u kulrang rangda belgilangan. Agar siz mavjud bo'lmagan usulni chaqirsangiz, u sariq rang bilan ta'kidlangan.
- Ilmiy asboblar: Jupyter Notebook bilan integratsiyalashgan, interaktiv Python konsoliga ega va Anaconda, shuningdek, matplotlib va NumPy kabi bir nechta ilmiy paketlarni qo'llab-quvvatlaydi.
- O'rnatilgan dasturchi asboblari: qutidan tashqaridagi asboblarning katta to'plami: integratsiyalangan tuzatuvchi va test dasturi, Python profili, o'rnatilgan terminal va asosiy VCS va o'rnatilgan ma'lumotlar bazasi vositalari bilan integratsiya.
- Web Development Frameworks: Web Development Frameworks: PyCharm zamonaviy veb-ishlab chiqish uchun ajoyib ramka-maxsus yordamni taklif qiladi

---

<sup>26</sup>[Https://www.jetbrains.com/pycharm-edu](https://www.jetbrains.com/pycharm-edu) saytida mavjud



### 2.3-rasm PyCharm Edu xush kelibsiz ekranı.

Django, Flask, Google App Engine, Pyramid va web2py kabi ramkalar.

Faqat Pro nashri.

#### 2.4.4 Spyder IDE

Spyder IDE — ilmiy dasturlash uchun ochiq manbali oýzaro platformali IDE. U Anaconda distributiviga kiritilgan, lekin uni yuklab olish va mustaqil muharrir sifatida ishlatalish mumkin. Bundan tashqari, pliginlarni tanlash Sublime va Atom kabi katta bo'lmasa-da, pliginlarni qo'llab-quvvatlaydi. Qisman bu muammo emas, chunki Spyder Python IDE hisoblanadi, shuning uchun Python funktsiyalariga ega bo'lish uchun Sublime va Atom kabi pliginlar kerak emas. Agar u Python IDE bo'lsa ham, u Python bilan cheklanmaydi; u C, C++ va Fortran kabi bir nechta tillarni qo'llab-quvvatlaydi (chunki uning kelib chiqishi fan uchun rivojlanish platformasi sifatida).

U PyCharm-ning aksariyat xususiyatlarini o'z ichiga oladi (kodni to'ldirish juda yaxshi ishlaydi), shuning uchun qaysidir ma'noda yaxshi alternativ. So'zni ta'kidlaganingizda, u avtomatik ravishda ushbu so'zning barcha nusxalarini ajratib ko'rsatadi. Interaktiv konsol Python va IPython-ni qo'llab-quvvatlaydi (yaxshiroq interaktiv konsol). Mening tajribamda u qadar barqaror emas; ba'zida konsolda ishlaydigan jarayon muharrirdagi koddan uzilib qoladi va siz IDEni qayta ishga tushirishingiz kerak.

Mavjudligi: Spyder IDE paketi Anaconda va WinPython ichida keladi. Shunday qilib, agar siz ushbu Python distributivlaridan foydalansangiz, sizda allaqachon Spyder IDE mavjud. Agar siz Spyder-ni Python distributivini o'rnatmasdan o'rnatmoqchi bo'lsangiz, uni PIP yordamida o'rnatishingiz mumkin:

## 36 Bioinformatika uchun Python

`pip install spyder`

Agar siz Conda-dan foydalansangiz, uni Conda Navigator-dan yoki buyruq satridan ishga tushirishingiz mumkin:

`conda install spyder`

### 2.4.5 Tahrirlovchilar haqida yakuniy so'zlar

Hech qanday muharrir barcha sohalarda boshqalardan shubhasiz yaxshiroq. Ko'p maqsadli muharrirlar sifatida Sublime va Atom yaxshi tanlovdir. Agar siz yopiq kodli dasturiy ta'minot bilan qulay bo'lsangiz, Sublime eng yaxshi tanlovdir (agar siz litsenziya to'lovini to'lashingiz mumkin bo'lsa). Shu bilan bir qatorda, agar kuchli mashinangiz bo'lsa, Atom Sublime-ning ko'p afzalliklarini taqdim etadi.

Python-ga xos muharrirlar sifatida Spyder ham, PyCharm ham Python dasturlash uchun kerak bo'lishi mumkin bo'lgan hamma narsaga ega. Ikkala IDE ham disk raskadrovska paytida qulay bo'lgan integratsiyalangan terminal emulyatsiyasi bilan birga keladi (va bu juda ko'p vaqtini talab qilishi mumkin). Ular, shuningdek, Anaconda Python tarqatish bilan yaxshi o'yнaydi, aslida, Spyder Anaconda qismidir.

Barcha eslatib o'tilgan muharrirlar Windows bilan ishlaydi, lekin agar siz Windows dasturchisi bo'lsangiz, Python-ni yaqinda qo'llab-quvvatlay boshlagan Visual Studio dasturini ko'rib chiqishingiz mumkin.<sup>27</sup> Bu shunchalik yangiki, uning qanday ishlashi haqida ko'p fikr-mulohaza yo'q, lekin Windows ishlab chiquvchilari o'zlarini uysa qilishadi. bu mahsulot.

Men ma'lum bir muharrirni tavsiya qilmoqchi emasman, chunki men buni shaxsiy tanlov deb bilaman. Mening tavsiyam shundaki, siz qo'lingizdan kelganini sinab ko'ring va ehtiyojlarингизга eng mos keladiganini tanlang. Men Atom-dan umumiyl maqsadli fayllarni tahrirlash uchun foydalanaman (bu kitob LATEX-da LATEX-ni tahrirlashga yordam beradigan plugin bilan Atom yordamida yaratilgan ), lekin Python uchun men odatda PyCharm Edu-dan foydalanaman.

## 2.5 BOSHQA ASOBOTLAR

---

Kod muharrirlaridan tashqari, ishlab chiquvchilar o'z ishlarini bajarishda yordam beradigan boshqa vositalardan foydalanishga moyil. Garchi ular o'quv jarayonining boshida kerak bo'lmasa-da, yo'lda foydalanishingiz mumkin bo'lgan ba'zi foydali vositalarni eslatib o'tish kerak.

- Jupyter Notebook (<http://jupyter.org>): Mahalliy ravishda ishlaydigan va veb-sahifa ichida ishlashi mumkin bo'lgan Python kodini o'z ichiga olgan hujjatlarni yaratish va almashish imkonini beruvchi veb-ilova. Tenglamalardan tashqari u matn va interaktiv grafiklarni ham ko'rsatishi mumkin. Ushbu kitobdag'i barcha kodlar ham shu formatda mavjud. Qo'shimcha ma'lumot olish uchun <http://py3.us> saytidagi kitob veb-sahifasiga qarang .
- Uçurtma (<https://kite.com>): U o'zini "Dasturlar uchun aqlli kopilot" deb belgilaydi. Bu sizning kod muharriringizga integratsiyalashgan va kodni taqdim etadigan yordamchi dastur

27Qarang: <https://www.visualstudio.com/vs/python/>.

to'ldirishlar vebdagi eng ko'p ishlataladigan variantlar, hujjatlar va misollar asosida. Ularning taqdimot videosini veb-sahifasida ko'ring. Hozirda Windows va macOS uchun mavjud bo'yigan Linux yozish vaqtida "deyarli tayyор".

- QuantifiedCode (<https://QuantifiedCode.com>): Dasturiy ta'minot omboringizdan kodingizni o'qiydigan va sharhlar va kodingiz haqida juda foydali maslahatlarni ko'rsatadigan onlayn vosita. Bu bepul xizmat emas,
- Pylint (<https://www.pylint.org>): Kodingizni tahlil qiladigan va uni qanday yaxshilash bo'yicha sharhlarni chiqaradigan buyruq qatori yordam dasturi. U kodning Python uslubi bo'yicha qo'llanmasiga amal qilishini tekshiradi, xatolarni va qayta tiklanishi mumkin bo'lgan takrorlangan kodni aniqlaydi. .
- Pylama (<https://github.com/klen/pylama>): Manba kodini tahlil qilish uchun Pylint va boshqalar kabi bir nechta vositalarni o'rabi oladi. Bu foydalanuvchi uchun qulay dastur emas, lekin u kuchli.
- ptpython (<https://github.com/jonathanslenders/ptpython>): Python interaktiv tarjimoni (yoki REPL) uchun qayta joylashtirish. U Python terminaliga sintaksisni ta'kidlash, ko'p qatorli tahrirlash va avtoto'ldirishni qo'shadi.

## 2.6 QO'SHIMCHA RESURSLAR

---

- Python tarjimonidan foydalanish.  
<https://docs.python.org/3/tutorial/interpreter.html>
- IPython: Interaktiv hisoblash muhit. <http://ipython.org/>
- PyFormat: format() va % yordamida satrlardagi qiymatlarni formatlash. <https://pyformat.info>
- Vikipediya maqolasi: "Matn muharrirlarini solishtirish".  
[http://en.wikipedia.org/wiki/Comparison\\_of\\_text\\_editors](http://en.wikipedia.org/wiki/Comparison_of_text_editors)
- Python Anywhere: "Python-ni bulutda joylashtiring, ishga tushiring va kodlang!"  
<https://www.pythonanywhere.com>
- Repl.it: Kod almashish imkonini beruvchi onlayn Python tarjimoni.  
<https://repl.it/languages/python3>

## 2.7 O'Z-O'ZINI BAHOLASH

---

1. Aniqlang: Dastur, ko'rsatma va o'zgaruvchi.
2. Python va cPython o'rtaqidagi farq nima?

**38 Bioinformatika uchun Python**

- 3. Ba'zi Python ilovalarini nomlang.**
- 4. Yagona va qo'sh tirnoqli qo'shtirnoqning qanday afzalligi bor?**
- 5. Format() nima?**
- 6. RPEL nima?**
- 7. Qachon interaktiv qobiq o'rniqa paketli rejimdan foydalanasiz?**
- 8. Chekish nima? Nima uchun Pythonda bu majburiy?**
- 9. Dastlabki koddagi izoh nima?**
- 10. Ishchi manba kodini izohlash uchun asosli sabab bormi?**
- 11. "Shebang" nima?**
- 12. "Kodlash izohi" nima. va uni qachon ishlatish kerak?**

# Asosiy dasturlash: ma'lumotlar Turlari

## MAZMUNI

<b>3.1 Satrlar .....</b>	<b>40</b>	<b>3.1.1 Satrlar</b>
Unicode belgilar ketma-ketligidir .....	41	3.1.2 String
manipulyatsiyasi. ....	42	
3.1.3 Stringlar bilan bog'liq usullar .....	42	
<b>3.2 Ro'yxatlar. ....</b>	<b>44</b>	
3.2.1 Ro'yxat elementlariga kirish. ....	45	3.2.2
Ko'p takrorlanuvchi elementlardan iborat ro'yxat .....	45	3.2.3
Ro'yxatni tushunish. ....	46	3.2.4 Ro'yxatlarni o'zgartirish .....
Qo'shish .....	47	47 Olib
tashlash. ....	47	3.2.5
Ro'yxatni nusxalash. ....	49	3.3
<b>Kortejlar. ....</b>	<b>49</b>	3.3.1 Kortejlar
o'zgarmas ro'yxatlardir. ....	49	3.4 Ketma-ketliklarning umumiy xossalari. ....
51		
Indekslash. ....	51	
Dilimslash. ....	52	
Ayzolik testi. ....	53	
Ulanish .....	53	len, max va min 53 Ketma-ketlikni ro'yxatga aylantirish .....
		54
<b>3.5 Lug'atlar. ....</b>	<b>54</b>	3.5.1
<b>Xaritalash: Har bir qiymatni nom bilan chaqirish. ....</b>	<b>54</b>	3.5.2 Lug'atlar bilan
ishlash. ....	56	lug'atlar kalit va qiymatlardan
tuzilgan .....	56	Lug'at qiymatlari
so'rovi. ....	57	Elementlarni
o'chirish. ....	58	3.6
To'plamlar. ....	59	
3.6.1 Ob'ektlarning tartibsiz to'plami. ....	59	To'plam
yaratish .....	59	3.6.2
Operatsiyalarni sozlash. ....	60	
chorraha .....	60	
Ittifoq .....	60	

<b>Farqi. ....</b>	<b>61</b>
<b>Simmetrik farq ..</b>	<b>62</b> <b>3.6.3</b>
<b>Boshqa ma'lumotlar turlari bilan birqalikdagi operatsiyalar ..</b>	<b>62</b>
<b>62 Maksimal, Minimal va Uzunlik ..</b>	<b>62</b>
<b>To'plamni ro'yxatga aylantirish ..</b>	<b>62</b>
<b>62 3.6.4 O'zgarmas to'plam: Frozenset ..</b>	<b>62</b>
<b>63 3.7 Obyektlarni nomlash ..</b>	<b>63</b>
<b>63 3.8 O'zgaruvchiga qiymat berish va ob'ektga nomni bog'lash</b> <b>64 3.9 Qo'shimcha manbalar ..</b>	<b>67</b> <b>3.10 O'z-o'zini baholash.</b> ..
	<b>68</b>

Oldingi bobda aytib o'tilganidek, ba'zi ma'lumotlar tuzilmalari turli xil kompyuter tillari o'rtaida taqsimlanadi, ammo ularning ba'zilari tilga xosdir. Shuning uchun ma'lumotlar turlari qandaydir tarzda kompyuter tilini belgilaydi. Python o'ziga xos ma'lumotlar turlariga ega.

Bunday asosiy ma'lumotlar tuzilmasi ketma -ketlikdir. Ketma-ketlik ichida elementlar ketma-ket tartibga ega. Masalan, belgilarning tartiblangan ketma-ketligi bo'lgan satr. Boshqa ketma-ketliklar ro'yxatlar va kortejlardir<sup>1</sup>. Ushbu turdag'i ketma-ketliklar o'rtaida tub farqlar mavjud bo'lsa-da, ular umumiylashtirilishi mumkin. Ketma-ket elementlarning tartibi bor, indekslanishi, tiliqilashtirilishi va takrorlanishi mumkin. Agar siz ushbu atamalardan ba'zilarini tushunmasangiz, tashvishlanmang. Faqat o'qishda davom eting. Ushbu bobda biz ushbu fikrlarning barchasini ko'rib chiqamiz.

Ketma-ketliklardan **tashqari** tartibsiz ma'lumotlar turlari ham mavjud : lug'atlar va to'plamlar. **Lug'at2** kalit va qiymat o'rtaida munosabatlarni saqlaydi, to'plam esa tartibsiz qiymatlar to'plamidir. Keyingi sahfalar tartiblangan (**string**, ro'yxat va kortej) va tartibsiz turlarga (lug'at va to'plam) qaratilgan.

### 3.1 STRINGS

---

Satr - bu bitta tirnoq ('), qo'sh tirnoq ("), uchta bitta tirnoq ("") yoki uch qo'sh tirnoq ("""") bilan ajratilgan belgililar ketma-ketligi. Shunday qilib, quyidagi satrlar ekvivalentdir:

```
"Bu Python'dagi string"
"Bu Python'dagi string"
"""Bu Python'dagi string"""
"""Bu Python'dagi string"""
```

Satrni chegaralashning ko'p usullariga ega bo'lish biroz ortiqcha tuyulishi mumkin.

<sup>1</sup>Ushbu kitobda yoritilmagan ko'proq ketma-ketlik turlari mavjud. Boshqa ketma-ketlik haqida ko'proq ma'lumot olish uchun turlari **bob oxiridagi Qo'shimcha manbalarga qarang**.

<sup>2</sup>Shuningdek, xaritalash ma'lumotlari turi sifatida tasniflanadi.

Bitta ('') va ikkita (") qo'shtirnoq chegaralovchilariga ega bo'lishning afzalligi shundaki, biz ikkita qo'shtirnoq uchun ajratilgan qatorga bitta qo'shtirnoq qo'shishimiz mumkin va aksincha:

"Ikki tirnoq ichidagi bitta tirnoq ('")

"Bu erda bizda bitta tirnoq ichida "ikkita tirnoq" bor"

Esda tutish kerak bo'lган muhim narsa shundaki, agar biz satrni kotirovka turi bilan boshlasak, uni bir xil turdag'i tirnoq bilan tugatishimiz kerak. Quyidagi qator noto'g'ri:

```
>>> "Iqtibos turlarini aralashtirish qorong'u tomonga olib keladi"
```

Fayl "<stdin>", 1-qator

```
"Iqtibos turlarini aralashtirish qorong'u tomonga olib keladi"
```

**Sintaksis xatosi:** bitta tirnoqli qatorni skanerlashda EOL

Eslatma: EOL qisqartmasi chiziq oxirini bildiradi.

Uch qo'shtirnoq bilan o'rالgan satrlarga kelsak, biz ulardan ko'p qatorli satrlarni (blok qatori sifatida ham tanilgan) ko'rsatish uchun foydalanishimiz mumkin:

```
"""Salom! Men a
```

ko'p qatorli

```
string """
```

“\n” belgisi qator oxiri (EOL) belgisini bildiradi. Shuning uchun, Yuqoridagi kod bir qatorda quyidagicha yozilishi mumkin:

```
"Salom! Men\nmultiline\nman ip"
```

Siz uni ko'rishni kutganingizdek qatorni yaratish va formatlash uchun uch qo'shtirnoqdan foydalanishingiz mumkin. Hujjatlar kabi uch tirnoqli qatorlar uchun boshqa foydalanish ham mavjud.

### 3.1.1 Satrlar Unicode belgilar ketma-ketligidir

Python 3 dan boshlab, barcha qatorlar sukut bo'yicha Unicode belgilaridir. Shunday qilib, bu to'g'ri qator:

```
>>> 'Python 3-da satrlar Unicode: ўўйўй шўйўй'
```

```
"Python 3-da satrlar Unicode: ўўйўй шўйўй"
```

Agar siz Unicode haqida qiziqsangiz, bu "barcha zamonaviy dasturiy ta'minotda istalgan tildagi matn ma'lumotlarini qayta ishlash, saqlash va almashish uchun asos bo'lган" sanoat standartidir. aniq deklaratsiyalar yoki konvertatsiya qilishni talab qilmasdan. Biroq, tashqi ma'lumotni o'qish yoki yozishda ba'zi ehtiyoj choralarini ko'rishingiz kerak bo'lishi mumkin

manbalar.

---

3 [http://www.unicode.org/faq/basic\\_q.html](http://www.unicode.org/faq/basic_q.html) manzilidagi Unicode tez-tez so'raladigan savollardan olingan .

### 3.1.2 String manipulyatsiyasi

Satrlar o'zgarmasdir. Satr yaratilgandan keyin uni o'zgartirib bo'lmaydi. Agar siz satrni o'zgartirishingiz kerak bo'lsa, siz nima qilishingiz mumkin bo'lsa, olingen satrni yaratishingiz mumkin. Bu funktsiyada parametr sifatida satr yordamida amalga oshiriladi va keyin qaytarilgan qiymatni oladi. Quyidagi misolda aminokislotalar ketma-ketligini ifodalovchi qator mavjud va u signal\_peptid deb ataladi:

```
>>> signal_peptide = "MASKATLLLAFULLFATCIA"
```

Satrning kichik harfli versiyasini olish uchun low() usulidan foydalaning:

```
>>> signal_peptide.lower()  
'maskatlllaftllfaticia'
```

Kichik harflar qatorini olganiga qaramay, asl satr o'zgartirilmagan:

```
>>> signal_peptid  
"MASKATLLLAFULLFATCIA"
```

Agar biz ushbu yangi kichik harflar qatori avvalgisi bilan bir xil nomga ega bo'lishini xohlasak birinchi, biz uni tayinlashimiz kerak:

```
>>> signal_peptide = signal_peptide.lower() >>>  
signal_peptide "maskatlllaftllfaticia"
```

Aniq effekt biz satrni o'zgartirganimiz kabi. Satrlar bilan bog'liq ba'zi usullarni ko'rish vaqt keldi.

### 3.1.3 Stringlar bilan bog'liq usullar

**replace(eski,yangi[,hisoblash]):** Satrning bir qismini (eski) **boshqasiga** (yangi) almashtirishga imkon beradi. Agar ixtiyoriy argumentlar soni ishlatsilsa, faqat oldingi sananing birinchi **ro'y** bergan sonlari almashtiriladi :

```
>>> dna_seq = 'GCTAGTAATGTG'  
>>> m_rna_seq = dna_seq.replace('T','U') >>>  
m_rna_seq 'GCUAGUAAUGUG'
```

**count(sub[, start[, end]]):** pastki satrning boshlang'ich va oxirgi **pozitsiyalari** (agar mavjud bo'lsa) o'rtaida necha marta paydo bo'lishini hisoblaydi . Keling, ketma-ketlikning CG mazmunini4 hisoblash uchun qanday foydalanish mumkinligini ko'rib chiqaylik :

---

4CG tarkibi DNK ketma-ketligidagi sitozin va guanin miqdoridir. CG tarkibi bog'liq DNKnинг erish haroratiga va boshqa fizik xususiyatlarga.

```
>>> dna_seq
"GCTAGTAATGTG"
>>> c = dna_seq.count("C") >>> g
= dna_seq.count("G") >>> (c+g)/
len(dna_seq)*100 41.66666666666667
```

`find(sub[,start[,end]])`: pastki satrning boshlang'ich va oxirgi pozitsiyalari **orasidagi (mavjud bo'lsa)** o'rnnini qaytaradi . Agar satrda pastki qator topilmasa, bu usul -1 qiymatini qaytaradi:

```
>>> m_rna_seq
"GCUAGUAAUGUG"
>>> m_rna_seq.find('AUG') 7
>>> m_rna_seq.find('GGG') -1
```

`indeks(sub[,start[,end]])`: `find()` kabi ishlaydi. **Farqi shundaki, pastki qator topilmasa, indeks ValueError istisnosini keltirib chiqaradi** . Bu usul `find()` orqali tavsiya etiladi , chunki -1 qiymati haqiqiy qiymat sifatida talqin qilinishi mumkin, `index()` tomonidan qaytarilgan ValueError esa haqiqiy qiymat sifatida qabul qilinmaydi.

`split([sep [,maxsplit]])`: Satrning "so'zlarini" ajratib, ularni ro'yxatda qaytaradi. Agar ajratuvchi (`sep`) belgilanmagan bo'lsa, standart ajratuvchi oq rang bo'ladi bo'sh joy:

```
>>> 'Ushbu qatorda bo'yshliqlar bilan ajratilgan so'zlar mavjud'.split()
['Bu', 'string', 'bor', 'so'zlar', 'ajratilgan', 'by', 'bo'shliqlar']
```

Agar oq bo'shliq ma'lumotlarni ajratuvchi bo'lmasa, biz maxsus ajratuvchini belgilashimiz kerak:

```
>>> "Aleks Doe,5555-2333,nobody@example.com".split()
['Aleks', 'Doe,5555-2333,hech kim@example.com']
```

Bu holda ajratuvchi vergul (","), shuning uchun biz buni aniq ko'rsatishimiz kerak:

```
>>> "Aleks Doe,5555-2333,nobody@example.com".split(",")
['Aleks Doe', '5555-2333', 'hech kim@example.com']
```

Bioinformatika ilovasi: BLAST fayllarini tahlil qilish.

Eng ko'p ishlataladigan bioinformatika dasturlaridan biri NCBI-BLAST (bu dastur 177-betdan ko'rib chiqiladi).

## 44 Bioinformatika uchun Python

BLAST mustaqil bajariladigan fayli “yorliqlardan ajratilgan fayl” sifatida (-m 8 argumentidan foydalanim) chiqishni yaratishi mumkin. Ushbu chiqish faylini split('t') yordamida tahlil qilish mumkin.

---

Split() funksiyasining teskari funksiyasi join():

join(seq): “Yelim belgisi” sifatida qator yordamida ketma-ketlikni birlashtiradi:

```
';'.join(['Aleks Doe', '5555-2333', 'hech kim@example.com'])  
"Aleks Doe;5555-2333;hech kim@example.com"
```

Hech qanday elim belgisiz ketma-ketlikni qo'shish uchun bo'sh tirnoqlardan foydalaning (""):

```
>>> ".qo'shilish(['A','C','A','T'])  
'MUSHUK'
```

String usullarining to'qliq tavsifi uchun konsoldagi help(str) ga qarang.

### 3.2 RO'YXATLAR

---

Ro'yxatlar Python-dagi eng ko'p qirrali ob'ekt turlaridan biridir. Ro'yxat ob'ektlarning tartiblangan to'plamidir. U vergul bilan ajratilgan va kvadrat qavslar orasiga olingan elementlar bilan ifodalanadi.

Split() funktsiyasini qo'llash natijasida biz allaqachon ro'yxatni ko'rdik:

```
>>> 'Aleks Doe,5555-2333,hi@example.com'.split(',')  
['Aleks Doe', '5555-2333', 'hi@example.com']
```

Bu uch elementli ro'yxat, "Aleks Doe", "5555-2333" va "hi@example.com", ularning hammasi torlar.

Keyingi kod ro'yxatni qanday aniqlash va nomlashni ko'rsatadi:

```
>>> birinchi_royyxat = [1, 2, 3, 4, 5]
```

Bu besh elementdan iborat ro'yxat. Bunday holda, barcha elementlar bir xil turdag'i (butun son). Ro'yxat turli xil elementlarni o'z ichiga olishi mumkin:

```
>>> other_list = [1, 'ikki', 3, 4, 'oxirgi']
```

Ro'yxat hatto boshqa ro'yxatni ham o'z ichiga olishi mumkin:

```
>>> nested_list = [1, 'ikki', birinchi_ro'yxat, 4, 'oxirgi'] >>> ichki_ro'yxat  
[1, 'ikki', [1, 2, 3, 4, 5], 4, 'oxirgi']
```

Bo'sh ro'yxat bo'sh qavslar bilan belgilanadi:

```
>>> empty_list = [] >>>  
empty_list []
```

Bo'sh ro'yxat avvalgidek ishlatalmaydi, lekin ba'zida elementlarni keyinroq qo'shish uchun bo'sh ro'yxatni belgilashni xohlashimiz mumkin.

### 3.2.1 Ro'yxat elementlariga kirish

Boshqa ketma-ketlik ma'lumotlar turlari kabi, siz mushuk ro'yxat elementlarini noldan boshlanadigan indeks bilan olasiz.

```
>>> first_list = [1, 2, 3, 4, 5] >>> first_list[0]
```

1

```
>>> birinchi_roýyxat[1]
```

2

Salbiy raqamlar ro'yxatlarga o'ngdan kirish uchun ishlataladi:

```
>>> birinchi_roýyxat = [1, 2, 3, 4, 5] >>>
```

```
birinchi_roýyxat[-1]
```

```
5 >>> birinchi_roýyxat[-4]
```

2

Ro'yxatlarni olishning yana bir usuli - ro'yxat bo'lmagan ob'ektni foydalanish orqali ro'yxatga aylantirishdir o'rnatilgan funktsiyalar ro'yxati():

```
>>> aseq = "atggctaggc" >>>
```

```
roýyxat(aseq) ['a', 't', 'g', 'g',
 'c', 't', 'a', 'g', 'g', 'c']
```

### 3.2.2 Ko'p takrorlanadigan elementlar ro'yxati

Agar bir xil element bir necha marta takrorlangan ro'yxatni ishga tushirmoqchi boýlsangiz, quyidagi kabi \* operatoridan foydalanishingiz mumkin:

```
>>> namunalar = ['qizil'] * 5 >>>
```

```
namunalar ['qizil', 'qizil', 'qizil',
 'qizil', 'qizil']
```

Bu bo'sh qiymatlar bilan ro'yxatni oldindan to'ldirish uchun ishlatalishi mumkin va katta ro'yxatlar bilan ishlashda foydali bo'lishi mumkin va elementlar soni oldindan ma'lum.

Ruxsat etilgan o'lchamdagи ro'yxatni belgilash, bo'sh ro'yxat yaratish va kerak bo'lganda uni kengaytirishdan ko'ra samaraliroqdir:

```
>>> namunalar = [Yo'q] * 5 >>>
```

```
namunalar
```

```
[Yo'q, Yo'q, Yo'q, Yo'q, Yo'q]
```

### 3.2.3 Ro'yxatni tushunish

Ro'yxatni aniqlashning yana bir usuli bor. Ro'yxat boshqa ro'yxatdan tuzilishi mumkin. Matematikada bo'lgani kabi, to'plamni uning barcha elementlarini sanab o'tish (tug'ish) yoki uning a'zolari tomonidan baham ko'rildigan xususiyatlarni tavsiflash (tushunish) orqali aniqlash mumkin bo'lganidek, Pythonda ham ro'yxat ikkala usulda ham tuzilishi mumkin.

Ro'yxatga olish bilan belgilangan to'plam,

$$A = \{0, 1, 2, 3, 4, 5\}$$

Pythonda sanab o'tilgan ro'yxat,

```
>>> a = [0, 1, 2, 3, 4, 5]
```

Tushunish bilan aniqlangan to'plam,

$$B = \{3 \mid x/x \in A\}$$

Bu ga teng

$$B = \{0, 3, 6, 9, 12, 15\}$$

Pythonda tushunish bilan belgilangan ro'yxat,

```
>>> [a ichida x uchun  
3*x] [0, 3, 6, 9, 12, 15]
```

Ro'yxatni tushunish orqali aniqlash uchun har qanday Python funksiyasi yoki usulidan foydalanish mumkin. Masalan, satrlar ro'yxatidan bir xil elementlardan iborat ro'yxat tuzamiz, lekin orqa va bosh oq bo'yshliqlarsiz:

```
>>> hayvonlar = [' King Kong', ' Godzilla ', 'Gamera '] >>> [hayvonlarda  
x uchun x.strip()]  
['King Kong', 'Godzilla', 'Gamera']
```

Natijalar to'plamini toraytirish uchun shartli bayonotni (agar) qo'shishimiz mumkin:

```
>>> hayvonlar = [' King Kong', ' Godzilla ', 'Gamera '] >>> [x.strip()  
hayvonlarda x uchun, agar x ichida 'i' bo'lsa]  
['King Kong', 'Godzilla']
```

### 3.2.4 Ro'yxatlarni o'zgartirish

Satrlardan farqli o'llaroq, ro'yxatlarni qo'shish, olib tashlash yoki o'zgartirish orqali o'zgartirish mumkin<sup>5</sup> fikrlar:

#### Qo'shish

**Ro'yxatga elementlar qo'yshishning uchta usuli mavjud:** qo'yshish, kiritish va kengaytirish. `append(element):` Ro'yxat oxiriga element qo'yshadi.

```
>>> first_list.append(99) >>>
first_list [1, 2, 3, 4, 5, 99]
```

**insert(position,element):** Element elementini pozitsion pozitsiyasiga kiritadi.

```
>>> first_list.insert(2,50) >>>
first_list [1, 2, 50, 3, 4, 5, 99]
```

**kengaytimoq(ro'yxat):** Asl ro'yxatning oxiriga ro'yxat qo'shish orqali ro'yxatni kengaytiradi.

```
>>> first_list.extend([6,7,8]) >>>
first_list [1, 2, 50, 3, 4, 5, 99, 6, 7, 8]
```

Bu + belgisini ishlatalish bilan bir xil:

```
>>> [1,2,3]+[4,5] [1, 2,
3, 4, 5]
```

#### O'chirish

**Ro'yxatdagi elementlarni o'chirishning uchta usuli mavjud.**

**pop([indeks]):** Indeks pozitsiyasidagi elementni olib tashlaydi va uni qaytaradi deb atalgan nuqta. Parametrlarsiz u oxirgi elementni qaytaradi.

```
>>> first_list [1, 2,
50, 3, 4, 5, 99, 6, 7, 8] >>> first_list.pop()
8
```

```
>>> first_list.pop(2) 50
```

```
>>> birinchi_ro'yxat
[1, 2, 3, 4, 5, 99, 6, 7]
```

---

<sup>5</sup>Ro'yxatlar Python jargonida "o'zgaruvchan" deb ataladi.

## 48 Bioinformatika uchun Python

## 3.1-JADVAL Umumiy ro'yxat operatsiyalari

**Xususiyatlар Tavsif**

`l.append(x)` s ro'yixatiga x elementini qo'yshadi  
`l.count(x)` X ning s `l.index(x)` da necha marta borligini hisoblaydi s `l.remove(x)` ro'yixatdagi x manzilini qaytaradi. ro'yixatdagi x elementi s `l.reverse()` Ro'yixatni teskari s `l.sort()`

Ro'yixatni saralash s

**olib tashlash (element):** Parametrda ko'rsatilgan elementni olib tashlaydi. Ro'yixatda bir xil ob'ektning bir nechta nusxalari bo'lsa, u chapdan sanab, birinchisini olib tashlaydi. `Pop()` dan farqli o'laroq, bu funksiya hech narsani qaytarmaydi.

```
>>> first_list.remove(99) >>>
first_list [1, 2, 3, 4, 5, 6, 7]
```

Mavjud bo'limgan elementni olib tashlashga urinish xatoga olib keladi:6

```
>>> first_list [1, 2,
3, 4, 5, 6, 7] >>>
first_list.remove(10)
Traceback (eng oxirgi qo'ng'iroq):
Fayl "<stdin>", 1-qator, ?
ValueError: list.remove(x): x ro'yixatda yo'q
```

Ro'yixat elementini o'chirishning yana bir usuli del buyrug'idan foydalanishdir, buning uchun:

`birinchi_list[0]`

Bu shunga o'xshash ta'sirga ega:

`first_list.pop(0)`

farqi bilan `pop()` chiqarilgan elementni chaqirilgan joyga qaytaradi, del esa uni o'chiradi. **3.1-jadvalda** ro'yixatlarning boshqa xossalari jamlangan.

<sup>6</sup>**7-bobda** istisnolarning batafsil tavsifi mavjud.

7Ob'ekt o'chirilmaydi. Aslida nima sodir bo'ladi, ob'ekt va uning nomi o'rtaсидаги mos yozuvlar yo'qoladi. Dasturchi uchun bu harakat xuddi o'chirilgandek ta'sir qiladi (ob'ektga kirish imkoniy yo'q). Oxir-oqibat, "Python axlat yig'uvchisi" uni shaffof va avtomatik tarzda yo'q qiladi.

### 3.2.5 Ro'yxatni nusxalash

Ro'yxatni nusxalash qiyin bo'lishi mumkin. Quyidagi kodda biz a ro'yxatini b ga nusxalashga harakat qilamiz, lekin elementni olib tashlash orqali b ni o'zgartirganimda , bu element a dan ham o'chiriladi :

```
>>> a = [1, 2, 3] >>>
```

```
b = a
```

```
>>> b.pop() 3
```

```
>>> a
```

```
[1, 2]
```

Ko'rinish turlidagi matlarni ko'chirmaydi, u asl obyektga havolani ko'chiradi.<sup>8</sup>

Ro'yxatni nusxalash uchun nusxa ko'chirish modulida nusxa ko'chirish usulidan foydalanishingiz kerak:

```
>>> import nusxasi
```

```
>>> a = [1, 2, 3] >>>
```

```
b = copy.copy(a) >>>
```

```
b.pop() 3
```

```
>>> a
```

```
[1, 2, 3]
```

Nusxa ko'chirish modulidan foydalanmasdan ham xuddi shunday qilishning bir usuli bor:

```
>>> a = [1, 2, 3] >>>
```

```
b = a[:] >>> b.pop()
```

```
3
```

```
>>> a
```

```
[1, 2, 3]
```

---

## 3.3

### 3.3.1 Kortejlar o'zgarmas ro'yxatlardir

Kortej - bir marta yaratilgan, uni o'zgartirib bo'lmaydigan xususiyatga ega bo'lgan tartiblangan ob'ektlar to'plami. Shuning uchun ular "o'zgarmas ro'yxatlar" deb nomlanadi. Python obyektlarini o'zgaruvchan va o'zgarmasga bo'lish mumkin. Nomidan ko'rinish turibdiki, o'zgarmas ob'ektlar yaratilgandan keyin ularni o'zgartirib bo'lmaydi. Ro'yxatdagi kortejni osongina ajratib ko'rsatish mumkin, chunki uning elementlari kvadrat qavslar o'rniiga qavslar orasiga olingan:

```
>>> nuqta = (23, 56, 11)
```

---

<sup>8</sup>Kaput ostida nima sodir bo'layotganini batafsil ko'rib chiqish uchun 64-betga qarang.

## 50 Bioinformatika uchun Python

nuqta uchta elementli kortejdir (23, 56 va 11).

Agar kortej faqat bitta elementga ega bo'lsa, siz keyingi verguldan foydalanishingiz kerak:

```
yolg'iz_element_tuple = (5,)
```

Bu ibora atrofidagi qavslar e'tiborga olinmaganligi sababli (5) ga ega bo'lgan noaniqlikni saralash uchun amalga oshiriladi, bu 5 (beshinch raqam) degan ma'noni anglatadi. Keyingi vergul va qavslar bilan Python tarjimoni bu ifoda emas, balki kortej ekanligini aytishi mumkin.

Tupledan elementlarni qo'yishish yoki o'y chirishga ruxsat berilmagan:

```
>>> point.append(3)
```

Traceback (eng oxirgi qo'ng'iroq):

Fayl "<stdin>", 1-qator, ?

```
AttributeError: "tuple" ob'ektida "qo'shish" atributi yo'q >>> point.pop()
```

Traceback (eng oxirgi qo'ng'iroq):

Fayl "<stdin>", 1-qator, ?

```
AttributeError: "tuple" obyektida "pop" atributi yo'q
```

Muayyan tarzda, kortej cheklangan ro'yxatga o'xshaydi (biz qila olmaydigan ma'noda cheklangan uni o'zgartiring). Xo'sh, kortejlar nima uchun yaxshi? Nega shunchaki ro'yxatni ishlatmaslik kerak?

Kortej sifatida saqlanadigan ma'lumotlar turlari va ro'yxat sifatida saqlanadigan ma'lumotlar o'rtasida kontseptual farq mavjud. Ro'yxatlar bir xil turdag'i ob'ektlarning o'zgaruvchan miqdoriga ega bo'lishi kerak. Katalogdagi barcha fayllarning fayl nomlarini o'z ichiga olgan ro'yxat ro'yxatda saqlanishi mumkin. Ularning barchasi bir xil turdag'i (string) bo'lib, ro'yxatdagi elementlar soni har bir katalogga qarab o'zgaradi. Ushbu ro'yxat ichidagi elementlarning tartibi tegishli emas.

Boshqa tomondan, kortejning odatiy misoli koordinatalar tizimidir. Uch o'lchovli koordinatalar tizimida har bir nuqta uch elementli kortej ( $x, y, z$ ) bilan ifodalanadi. Har bir kortej uchun elementlar soni o'zgarmaydi (chunki har doim uchta koordinata mavjud) va har bir pozitsiya muhim, chunki har bir nuqta ma'lum bir o'qga mos keladi.

Funktsiya yoki lug'at kalitidan qaytariladigan elementlar haqida ham xuddi shunday deyishimiz mumkin.<sup>9</sup> Kortejning yana bir afzalligi shundaki, undan xavfsizroq kod yaratish uchun foydalanish mumkin - biz o'zgartirishni istamagan ma'lumotlar "yozishdan himoyalangan" qoladi. " o'zgarmas kortejda.

Tuplelar ro'yxatlarga qaraganda kamroq xotirani oladi. Katta ma'lumotlar to'plamlari bilan ishlashda bu muhim bo'lishi mumkin. Shuningdek, kortejlar bilan bog'liq operatsiyalarining tezligi ro'yxatlarga qaraganda tezroq. Muayyan holatlarda bu to'g'ri bo'lishi mumkin bo'lsa-da, ro'yxat yoki kortejni tanlashda bu faktning o'zi asosiy e'tiborga olinmasligi kerak.

---

<sup>9</sup>Funktsiyalar va lug'atlarni ko'rgandan so'ng bu mantiqiy bo'ladi.

### 3.4 KETTLIKLARNING UMUMIY XUSUSIYATLARI

---

Ketma-ketliklar umumiy xususiyatlarga ega ekan, keling, ularni birgalikda ko'rib chiqamiz. Ushbu xususiyatlarni ro'yxatlar, kortejlar va satrlarga qo'llashingiz mumkin.

#### Indekslash

Ro'yxatlarni yoritishda indekslash muhokama qilindi, ammo to'liqlik uchun bu erda ham yoritilgan. Ketma-ketlikdagi elementlar tartiblanganligi sababli, biz noldan boshlanadigan indeks orqali istalgan elementga kirishimiz mumkin:

```
>>> nuqta = (23, 56, 11) >>>
nuqta[0] 23
```

```
>>> nuqta[1] 56
```

```
>>> ketma-ketligi = 'MRVLLVALALLALAASATS'
```

```
>>> ketma-ketligi[0]
'M'
```

```
>>> ketma-ketligi[5]
"V"
```

```
>>> parametrlari = ['UniGene', 'dna', 'Mm.248907', 5] >>>
parametrlari[2]
'Mm.248907'
```

Salbiy raqamlar yordamida ketma-ketlik elementlariga o'ngdan ham kirishimiz mumkin:

```
>>> nuqta[-1] 11
```

```
>>> nuqta[-2] 56
```

```
>>> my_sequence[-2]
'T'
```

```
>>> my_sequence[-4]
'S'
```

```
>>> my_sequence[-1] 5
```

Ketma-ket ichida joylashgan elementga kirish uchun, uning o'zi boshqasining ichida ketma-ketlikda siz boshqa indeksdan foydalanishingiz kerak:

```
>>> seqdata = ('MRVLLVALALLA', 12, '5FE9EEE8EE2DC2C7') >>>
seqdata[0][5]
"V"
```

## 52 Bioinformatika uchun Python

Birinchi indeks (0) biz seqdata birinchi elementiga kirishimizni bildiradi. Ikkinci indeks (5) birinchi elementning ('MRVLLVALALLA') 6-elementiga ('V') ishora qiladi.

## Kesish

Bo'lim belgilaridan foydalanib, ketma-ketlikning bir qismini tanlashingiz mumkin . Dilimlash ikki nuqta (:) bilan ajratilgan ikkita indeksdan foydalanishdan iborat. Ushbu indekslar elementlar orasidagi mavjud bo'shliqdagi pozitsiyani ifodalarydi. "Python" qatori quyidagicha ifodalanadi:

```
+-----+
| P | y | t | h | o | n |
+-----+
0   1   2   3 4 5           6
```

```
>>> my_sequence="Python"
>>> my_sequence[0:2]
"Py"
```

Birinchi pastki indeksni o'tkazib yuborganda, indeks qiymati sukut bo'yicha birinchi pozitsiyaga (0) keladi:

```
>>> mening_tartibim[:2]
"Py"
```

Boshqa tomondan, ikkinchi pastki indeks qoldirilsa, indeks qiymati sukut bo'yicha oxirgi pozitsiya (-1):

```
>>> my_sequence = "Python"
>>> my_sequence[4:6] 'on'

>>> my_sequence[4:]
'on'
```

Pozitsiyalarni o'tkazib yuborish uchun uchinchi, ixtiyoriy indeks mavjud (qadam argumenti):

```
>>> my_sequence[1:5]
'ytho' >>
my_sequence[1:5:2] 'yh'
```

Orqaga hisoblash uchun manfiy sonli qadam ishlataladi. Shunday qilib -1 (uchinchi holatda) ketma-ketlikni o'zgartirish uchun ishlatalishi mumkin:

```
>>> my_sequence[::-1]
'nohtyP'
```

E'tibor bering, kesish har doim boshqa ketma-ketlikni qaytaradi.

## A'zolik testi

Elementning ketma-ketlikka tegishli ekanligini in kalit so'zidan foydalanib tekshirishingiz mumkin:10

```
>>> nuqta = (23, 56, 11) >>> 11
nuqtada
To'g'ri
>>> my_sequence = 'MRVLLVALALLALAASATS'
>>> my_sequencedagi "X"
Yolg'on
```

## Birlashtirish

Siz "+" belgisi yordamida bir xil sinfning ikki yoki undan ortiq ketma-ketligini birlashtirishingiz mumkin:

```
>>> nuqta = (23, 56, 11) >>> nuqta2
= (2, 6, 7) >>> nuqta + nuqta2 (23,
56, 11, 2, 6, 7) >>> dna_seq =
'ATGCTAGACGTCCCTCAGATAGCCG
'>>> tata_box = 'TATAAA'

>>> tata_box + dna_seq
"TATAAAATGCTAGACGTCCCTCAGATAGCCG"
```

Har xil turdag'i ketma-ketliklarni birlashtirib bo'lmaydi:

```
>>> nuqta + tata_box
Traceback (eng oxirgi qo'ng'iroq):
  Fayl "<stdin>", 1-qator, <modul> da
TypeError: faqat kortejni ("str" emas) kortejga birlashtirishi mumkin
```

## len, maksimal va min

len() ketma-ketlikning uzunligini (elementlar sonini) qaytaradi:

```
>>> nuqta = (23, 56, 11) >>> len
(nuqta)
3
>>> my_sequence = 'MRVLLVALALLALAASATS' >>>
len(mening_sequence) 19
```

max() va min() raqamlar ketma-ketligida qo'llaniladigan kutilganidek qaytariladi  
maksimal va minimal qiymat:

---

10E'tibor bering, in kalit so'zi 59-betda ishlatalgan

## 54 Bioinformatika uchun Python

```
>>> nuqta  
(23, 56, 11) >>>  
max(nuqta) 56
```

```
>>> min(nuqta) 11
```

Satrlarga qo'llaniladigan max() va min() ASCII kodining maksimal yoki minimal qiymatiga muvofiq belgini qaytaradi:

```
>>> my_sequence = 'MRVLLVALALLALAASATS'  
>>> max(mening_sequence)  
"V"  
>>> min (mening_ketma-ketligim)  
'A'
```

Ketma-ketlikni ro'yxatga aylantiring

Ketma-ketlikni (masalan, kortej yoki satr) ro'yxatga aylantirish uchun list() usulidan foydalaning:

```
>>> tata_box = 'TATAAA' >>>  
ro'yxati (tata_box)  
['T', 'A', 'T', 'A', 'A', 'A']
```

Ro'yxatni ishlatalish bizga satrni bilvosita o'zgartirish usullarini beradi. Ro'yxatlar, satrlardan farqli oylaroq, oyzgaruvchan boylgani uchun, biz satrni ro'yxatga oyzgartirishimiz, ushbu ro'yxatni oyzgartirishimiz va keyin uni yana satrga aylantirishimiz mumkin (str()) bilan).11 Bu jarayon samarali emas, shuning uchun iloji boricha , boshqa qatorni olish uchun string xusu

## 3.5 LIG'ATLAR

---

### 3.5.1 Xaritalash: Har bir qiymatni nom bilan chaqirish

Lug'atlar barcha dasturlash tillarida mavjud bo'limgan maxsus ma'lumotlar turidir. Lug'atning asosiy xususiyati shundaki, u o'zboshimchalik bilan indekslangan tartibsiz ma'lumotlar turlarini saqlaydi.

Ushbu misol nima uchun ushbu ma'lumotlar turi lug'at deb atalishini ko'rsatadi:

```
>>> iupac = {'A':'Ala','C':'Cys','E':'Glu'} >>> print('C  
aminokislotalarni bildiradi {0}'.format(iupac) ['C']))  
C Cys aminokislotasini anglatadi
```

1144-betda tasvirlanganidek join() usulidan foydalanib.

iupac - uchta elementdan iborat lug'at nomi. Enclos tomonidan aniqlangan ing kalit: jingalak qavslar orasidagi qiymat juftlari ({}).

Ushbu lug'at bir harfli aminokisloti kodini uch harfli kodga tarjima qilish imkonini beruvchi tarjima jadvali sifatida ishlaydi. Har bir element juft kalitdan iborat: qiymat. Kalit qiymatni olish uchun ishlataladigan indeksdir:

```
>>> iupac['E']
"Glu"
```

Har bir ob'ektdan lug'at kaliti sifatida foydalanish mumkin emas. Kalit sifatida faqat satrlar, kortejlar va raqamlar kabi o'zgarmas ob'ektlardan foydalanish mumkin. Agar kortejda har qanday o'zgaruvchan ob'ekt bo'lsa, uni kalit sifatida ishlatib bo'lmaydi.

Lug'at dict bilan ketma-ketlikdan ham yaratilishi mumkin:

```
>>> rgb = [('qizil','ff0000'), ('yashil','00ff00'), ('ko'k','0000ff')] >>> colors_d = dict(rgb) >>>
colors_d { 'qizil': 'ff0000', 'ko'k': '0000ff', 'yashil': '00ff00'}
```

dict shuningdek, kalit so'z argumentlari ro'yxatida nom = qiymat juftlarini qabul qiladi:

```
>>> rgb = dict(red='ff0000', yashil='00ff00', ko'k='0000ff') >>> rgb {'ko'k':
'0000ff', 'yashil': '00ff00', 'qizil': 'ff0000'}
```

Lug'atni ishga tushirishning yana bir usuli bo'sh lug'at yaratish va qo'shishdir kerak bo'lganda elementlar:

```
>>> rgb = {}
>>> rgb['qizil'] = 'ff0000' >>>
rgb['yashil'] = '00ff00' >>> rgb
{'yashil': '00ff00', 'qizil': 'ff0000'}
```

len(), lug'atdagi elementlar sonini qaytaradi:

```
>>> len(iupac) 3
```

Lug'atga qiymat qo'shish uchun,

```
>>> iupac['S'] = 'Ser' >>>
len(iupac) 4
```

Lug'atlar tartibsiz bo'ladi, chunki ular o'z elementlarining tartibini kuzatmaydilar. Lug'at mazmunini ko'rishni so'raganingizda, elementlarni kiritilgan tartibda olishingiz yoki olmaysiz:

```
>>> iupac = {'A':'Ala','C':'Cys','E':'Glu'} >>> iupac
```

```
{'E': 'Glu', 'C': 'Cys', 'A': 'Ala'}
```

Yangi elementni kiritishda u ma'lum bir joyga kiritilmaydi (oxirgi kabi):

```
>>> iupac['X'] = 'Xaa' >>>
iupac
{'E': 'Glu', 'X': 'Xaa', 'C': 'Cys', 'A': 'Ala'}
```

Element tartibini kuzatish uchun lug'atga ishonmang. Agar sizga buyurtma qilingan lug'at kerak bo'lsa, OrderedDict12 dan foydalaning:

```
>>> kollektsiyalardan import OrderedDict >>> d =
OrderedDict() >>> d['a'] = 'A' >>> d['b'] = 'B' >>>
d['c'] = 'C'

>>> d
OrderedDict([('a', 'A'), ('b', 'B'), ('c', 'C')])
```

OrderedDict haqida qo'shimcha ma'lumot olish uchun <https://www.python.org/dev/peps/pep-0372> ga qarang .

Ushbu bobda topilgan o'rnatilgan ma'lumotlar turlari ko'pchilik foydalanuvchilar uchun etarli, ammo yanada rivojlangan foydalanish uchun buyurtma qilingan kontent bilan ishslash uchun uchinchi tomon moduli mavjud: SortedContainers13. Katta hajmdagi ma'lumotlar bilan ishslashda tartiblangan konteynerlar uchun tezkor va xotira optimallashtirilgan yechim kerak bo'lga

### 3.5.2 Lug'atlar bilan ishslash

Ro'yxat sifatida lug'atlarning o'z usullari mavjud.

#### Lug'atlar kalit va qiymatlardan tuzilgan

Lug'atning kalitlari yoki qiymatlarini olish uchun keys() va val ues() kabi usullar mavjud:

---

<sup>12</sup>Python 3.6 da lug'atlar buyurtma qilingan, ammo bu amalga oshirish detali hisoblanadi va unga tayanmaslik kerak. Bu Python 3.7 da o'zgarishi mumkin, shuning uchun tartibni saqlashni istasangiz OrderedDict dan foydalaning. <sup>13</sup><http://www.grantjenks.com/docs/sortedcontainers/>

```
>>> iupac
{'E': 'Glu', 'X': 'Xaa', 'C': 'Cys', 'A': 'Ala'} >>> iupac.keys()
dict_keys(['E', 'X', 'C', 'A']) >>> iupac.values() dict_values(['Glu',
'Xaa', 'Cys', 'Ala'])
```

E'tibor bering, bu usullar ro'yxatni qaytarmaydi (bu ularning Python 3 dan oldingi xatti-harakatlari edi), lekin ular lug'at ko'rinishi deb nomlangan maxsus ob'ektni qaytaradi. Ushbu ob'ekt sizga joriy kalitlar yoki qiymatlarni ko'rsatadi, shuning uchun u lug'atda o'zgarsa, lug'at ko'rinishida o'zgaradi:

```
>>> iupac.values()
dict_values(['Glu', 'Xaa', 'Cys', 'Ala']) >>> iupac.keys()
dict_keys(['E', 'X', 'C', 'A']) >>> iupac_keys =
iupac.keys() >>> iupac_vals = iupac.values() >>>
iupac.pop('X')
```

"Xaa"

```
>>> iupac_keys
dict_keys(['E', 'A', 'C']) >>>
iupac_vals dict_values(['Glu',
'Cys', 'Ala'])
```

Bu sizga dasturning bir nechta qismlarida kalitlar yoki qiymatlarni kerak bo'lganda foydali bo'lishi mumkin.

Lug'at elementlariga kirishning yana bir usuli items() dan foydalanishdir. har bir kalit/qiymat juftligi uchun lug'at ko'rinishini qaytaradi:

```
>>> iupac = {'E': 'Glu', 'X': 'Xaa', 'C': 'Cys', 'A': 'Ala'} >>> iupac.items()
dict_items([('E', 'Glu'), ('A', 'Ala'), ('C', 'Cys'), ('X', 'Xaa')])
```

## Lug'at qiymatlarini so'rash

Istisnoni chaqirish xavfisiz lug'atdan qiymat so'rash uchun get(k,x) dan foydalaning. K - ajaratib olinadigan elementning kaliti, x esa k lug'at kaliti sifatida topilmagan taqdirda qaytariladigan elementdir .

```
>>> iupac = {'E': 'Glu', 'X': 'Xaa', 'C': 'Cys', 'A': 'Ala'} >>> iupac.get('A', 'Tarjima
mavjud emas')
"Ala"
>>> iupac.get('Z', 'Tarjima mavjud emas')
"Tarjima mavjud emas"
```

## 58 Bioinformatika uchun Python

## 3.2-JADVAL Lug'atlar bilan bog'liq usullar

len(d) d[k]	Tavsif
d[k] = v	d ning elementlari soni
del d[k]	k kalitga ega bo'lgan d elementi
d.clear()	d[k] ni v ga o'rnatning
d.copy()	d [k] ni d dan olib tashlang
xususiyatlari	d dan barcha elementlarni olib tashlang
Nusxa ko'chirish d	
k dk	Agar d tugmasi k bo'lsa, rost, aks holda False
ichida d	d da k not ga ekvivalent
emas d.has_key(k)	d dagi k ga ekvivalent , yangi kodda ushbu shakldan foydalaning
d.items()	d ning (kalit, qiymat) juftliklari ro'yxatining nusxasi
d.keys()	d kalitlari ro'yxatining nusxasi
d.update([b])	b dan kalit/qiymat juftlarini yangilaydi (va ustiga yozadi).
Seq kalitlari va qiymatga o'rnatilgan qiymatlar bilan yangi lug'at yaratadi	
d.qiymatlari ro'yxatining nusxasi	
a[k] agar k bo'ysa d, aks holda x a [k]	
bo'ysa d y k, aks holda x (shuningdek uni	
o'yrnatish) d[k] bo'ysa d y k , aks holda x (va kyni olib tashlang)	
d.setdefault(k, x) d.pop(k, None) qiyomat()qiymat) juftlikni olib tashlang va qaytaring	

Agar siz x ni o'tkazib yuborsangiz va lug'atda k tugmasi **bo'lmasa** , usul Yo'q ni qaytaradi.

```
>>> iupac.get('Z')
Yo'q
```

### Elementlarni o'chirish

Lug'atdan elementlarni o'chirish uchun del ko'rsatmasidan foydalaning:

```
>>> iupac = {'E': 'Glu', 'X': 'Xaa', 'C': 'Cys', 'A': 'Ala'} >>> del iupac['A'] >>> iupac
```

```
{'C': 'Cys', 'X': 'Xaa', 'E': 'Glu'}
```

**3.2-jadvalda** lug'atlarning umumiyl xususiyatlari keltirilgan.

## 3.6 TOPLAR

---

### 3.6.1 Ob'ektlarning tartibsiz to'plami

Ushbu turdag'i ma'lumotlar boshqa dasturlash tillarida ham tez-tez uchramaydi. To'plam matematikada tez-tez uchraydigan tuzilmadir. Bu ro'yxatga o'xshaydi, ikkita ajoyib farqi bor: uning elementlari nazarda tutilgan tartibni saqlamaydi va har bir element noyobdir.

To'plamlarning eng keng tarqalgan qo'llanilishi a'zolikni tekshirish, dublikatlarni olib tashlash va matematik operatsiyalarni qo'llashdir: kesishmalar, birlashmalar, farqlar va simmetrik farqlar.

#### To'plam yaratish

To'plamlar ko'rsatmalar seti() yordamida yaratiladi:

```
>>> first_set = {'CP0140.1','XJ8113.5','EF3616.3'}
```

Bundan tashqari, bo'sh to'plam yaratish va kerak bo'lganda elementlarni qo'shish mumkin:

```
>>> first_set = set() >>>
first_set.add('CP0140.1') >>>
first_set.add('XJ8113.5') >>>
first_set.add('EF3616.3') >>>
birinchi_to'plam
{'CP0140.1','XJ8113.5','EF3616.3'}
```

Ro'yixatni tushunish kabi to'plamni tushunish orqali ham belgilashingiz mumkin (46-betga qarang):

```
>>> {2*x x uchun [1,2,3]} {2, 4, 6}
```

To'plam takroriy elementlarni qabul qilmagani uchun, to'plamda mavjud bo'lgan elementni qo'shishga harakat qilganiningizda hech qanday ta'sir bo'lmaydi:

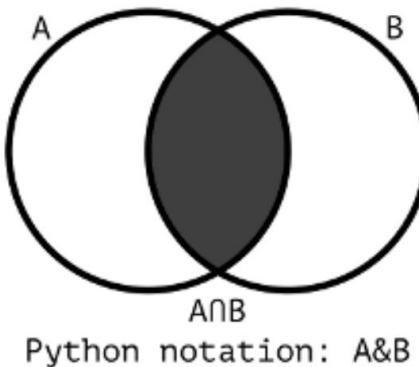
```
>>> first_set.add('CP0140.1') >>>
first_set {'CP0140.1','XJ8113.5','EF3616.3'}
```

Belgilangan tushunish holatida:

```
>>> {2*x x uchun [1,1,2,2,3,3]} {2, 4, 6}
```

Ushbu xususiyat takrorlangan elementlarni ro'yxatdan olib tashlash uchun ishlatalishi mumkin:

```
>>> noyob = {2,2,3,4,5,3} >>> noyob
{2, 3, 4, 5}
```



### 3.1-rasm kesishma.

#### 3.6.2 Operatsiyalarni sozlash

Chorraha

Umumiy elementlarni ikkita to'plamda olish uchun (3.1-rasmda ko'rsatilganidek ) kesishma () operatoridan foydalaning:

```
>>> first_set = {'CP0140.1','XJ8113.5','EF3616.3'} >>> other_set =
{'EF3616.3'} >>> umumiy = first_set. kesishishi (boshqa_set)

>>> umumiy
{'EF3616.3'}
```

Bu & ga teng:

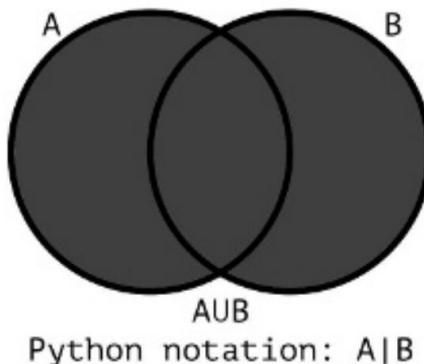
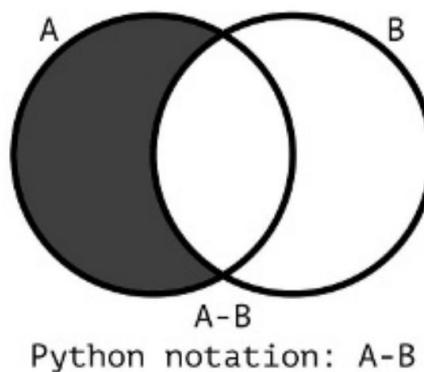
```
>>> umumiy = birinchi_to'plam va boshqa_to'plam
>>> umumiy
{'EF3616.3'}
```

ittifoq

Ikki (yoki undan ko'p) to'plamlarning birlashmasi operator birlashmasi (3.2-rasmda ko'rsatilganidek ) va uning qisqartirilgan shakli |:

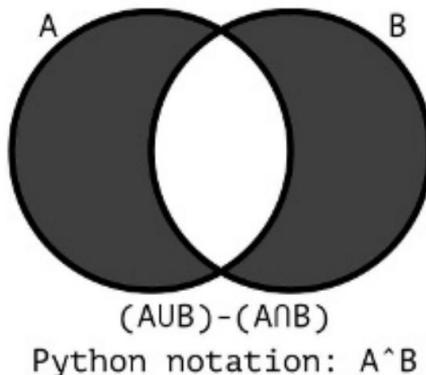
```
>>> first_set = {'CP0140.1','XJ8113.5','EF3616.3'} >>> other_set =
{'AB7416.2'} >>> first_set.union(other_set)

{'CP0140.1', 'XJ8113.5', 'EF3616.3', 'AB7416.2'} >>>
birinchi_to'plam | other_set {'CP0140.1', 'XJ8113.5', 'EF3616.3',
'AB7416.2'}
```

3.2-rasm **Birlashma.**3.3-rasm **Farq.****Farq**

Farq - bu bir to'plamga tegishli bo'lgan, lekin ikkinchisiga tegishli bo'lmasagan elementlarning natijaviy to'plamidir ([3.3-rasmga qarang](#)). Uning qisqartmasi :-

```
>>> birinchi_to'plam.farq (boshqa_to'plam)
{'CP0140.1', 'XJ8113.5', 'EF3616.3'}
>>> first_set - other_set
{'CP0140.1', 'XJ8113.5', 'EF3616.3'} >>>
other_set - first_set {'AB7416.2'}
```



### 3.4-rasm Simmetrik farq.

#### Simmetrik farq

Nosimmetrik farq kesishmaning bir qismi bo'limgan elementlarga tegishlidir ([3.4-rasmga qarang](#)); uning operatori simmetrik\_farq va u qisqartirilgan

<sup>^</sup> sifatida:

```
>>> birinchi_to'plam.simmetrik_farq(boshqa_to'plam)
{'CP0140.1', 'XJ8113.5', 'EF3616.3', 'AB7416.2'} >>> birinchi_to'plam
^ boshqa_to'plamlar(['EF3616.2', 'CP0140.1', 'CP0140.2 ', 'EF3616.1'])
```

#### 3.6.3 Boshqa ma'lumotlar turlari bilan birlgilidagi operatsiyalar

##### Maksimal, minimal va uzunlik

To'plamlar maks, min, len, in va boshqalar kabi ketma-ketliklar bilan ba'zi xususiyatlarni baham ko'yradi. Kutganimizdek, bu xususiyatlar xuddi shunday ishlaydi.

##### To'plamni ro'yxatga aylantirish

Satrlarda bo'lgani kabi, to'plamlar list() funksiyasi bilan ro'yxatlarga aylantirilishi mumkin:

```
>>> first_set
{'CP0140.1', 'XJ8113.5', 'EF3616.3'} >>> ro'yxati
(birinchi_to'plam)
['CP0140.1', 'XJ8113.5', 'EF3616.3']
```

To'plamlar bilan bog'liq barcha usullarni ko'rish uchun konsolda help(set()) buyrug'ini kriting.

### 3.6.4 O'zgarmas to'plam: Frozenset

Frozenset to'plamning o'zgarmas versiyasidir. Uning mazmunini o'zgartirib bo'lmaydi, shuning uchun add() va remove() kabi usullar mavjud emas. U frozenset ob'ekti bilan yaratiladi, u kirish sifatida iteratsiyani oladi:

```
>>> fs = frozenset(['a','b']) >>> fs
```

```
frozenset({'a', 'b'}) >>> fs.remove('a')
```

Traceback (eng oxirgi qo'ng'iroq):

Fayl "<stdin>", 1-qator, <modul> da

```
AttributeError: 'frozenset' obyektida 'olib tashlash' atributi yo'q >>> fs.add('c')
```

Traceback (eng oxirgi qo'ng'iroq):

Fayl "<stdin>", 1-qator, <modul> da

```
AttributeError: "frozenset" ob'ektiida "add" atributi yo'q
```

Frozensests o'zgarmas bo'lgani uchun ular lug'at kaliti sifatida ishlatalishi mumkin.

## 3.7 OB'YEKTLARNI NOM BERISH

---

Yaroqli nomlarda harflar, raqamlar va pastki chiziq (...) bo'lishi kerak, lekin ular raqamlar bilan boshlanmaydi. Shuningdek, ular "til uchun ajratilgan so'zlar" bo'lishi mumkin emas, masalan:

Yolg'on	sinf	nihoyat lambda	qaytish
Yo'q	uchun davom etadi		
To'g'ri	def	dan	nolocal while harakat qilib ko'ring
va	del	global emas, agar	bilan
kabi	elif	yoki	Yo'il bering
asser	boshqa	import passni oshirish	
tanaffus	bundan mustasno		

Bu erda noto'g'ri nomlar namunasi, izohda tushuntirish bilan:

```
>>> 23cm = "1"           # Raqam bilan boshlang
>>> 23 = "1"            # Raqam bilan boshlang
>>> Varmi? = "value" # Noto'g'ri belgiga ega (?).>>> $besh = 5
                                         # Belgisi yaroqsiz ($)
>>> uchun = 123          # Zaxiralangan so'z bor
>>> agar = "ma'lumotlar" # Zaxiralangan so'z bor
```

Biz shu paytgacha bir nechta nom topshiriqlarini ko'rdik:

```
>>> my_sequence = 'MRVLLVALALLALAASATS' >>> first_list =
[1,2,3,4,5]
```

## 64 Bioinformatika uchun Python

```
>>> d= {1:'a',2:'b',3:'c'} >>> k =
d.keys() >>> nuqta = (23,56,11)
>>> first_set =
{'CP0140.1','XJ8113.5','EF3616.3'} >>> fs = frozenset(['a','b'])
```

Bu haqiqiy nomlar. Kodni o'qishni yaxshilash uchun amal qilish kerak bo'lgan nomlash qoidalari ham mavjud. Ushbu konvensiyalar Python uslubi **bo'yicha** qo'llanmaning bir qismidir.<sup>14</sup> Ushbu qo'llanmaga ko'ra, o'qishni yaxshilash uchun ismlar kichik harflardan iborat bo'lishi kerak, so'zlar kerak bo'lganda pastki chiziq bilan ajratilishi kerak.

### 3.8 O'ZGARGANCHAGA QIYMAT BERISH VA NOMNI BOG'LASH OB'YEKT

---

**Quyidagi bayonotni o'zgaruvchan topshiriq sifatida ko'rish mumkin:**

```
>>> a = 3
>>> b = [1,2,a]
```

Shuningdek, bu:

```
>>> b = [1,2,a]
```

Ingliz tiliga tarjima qilinganda ular: "a o'zgaruvchisi 3 qiymatiga ega bo'lsin" va "b o'zgaruvchisi uchta elementdan iborat ro'yxatga ega bo'lsin: 1, 2 va a (3 qiymatiga ega)" degan ma'noni anglatadi.

Chop etish b ikkala bayonotni tasdiqlaydi:

```
>>> b
[1, 2, 3]
```

Shunday qilib, a qiymatini o'zgartirib, b qiymati ham o'zgarishi kerak:

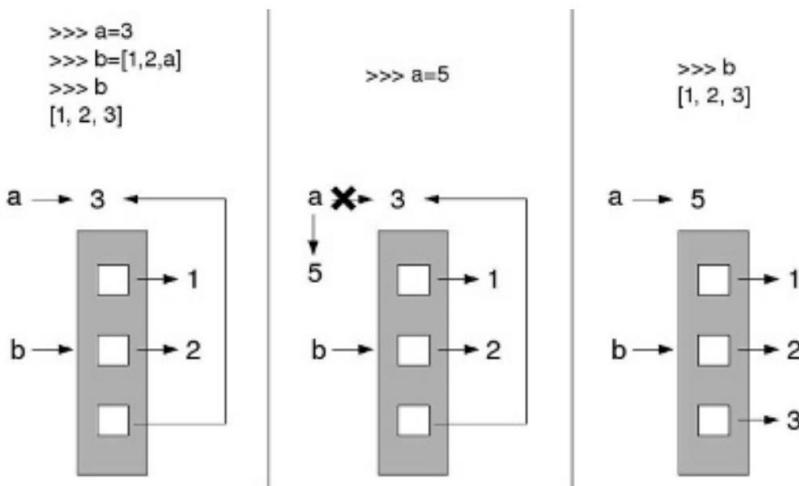
```
>>> a = 5
>>> b
[1, 2, 3]
```

Bu erda nima bo'ldi? Agar siz boshqa dasturlash tilini bilsangiz, "Python qiymatga havola o'rniga qiymatni saqlaydi" deb o'ylashingiz mumkin. Aynan shunday emas, shuning uchun men sizni o'qishni davom ettirishingizni so'rayman.

Quyidagi bayonotlar boshqacha tarzda ishlaydi:

```
>>> c = [1, 2, 3] >>> d
= [5, 6, c]
```

14 <https://www.python.org/dev/peps/pep-0008> manzilida mavjud



### 3.5-rasm 1-holat.

Ingliz tiliga tarjima qilinganda ular: “c o‘zgaruvchisi uchta elementli ro‘yxat bo‘lsin: 1, 2 va 3” va “d o‘zgaruvchisi uchta elementli ro‘yxat bo‘lsin: 5, 6 va c (bu ro‘yxat bo‘lsin) uchta element: 1, 2 va 3”.

Buni ikkala o‘zgaruvchini chop etish orqali tasdiqlash mumkin:

```
>>> c
[1, 2, 3]
>>> d
[5, 6, [1, 2, 3]]
```

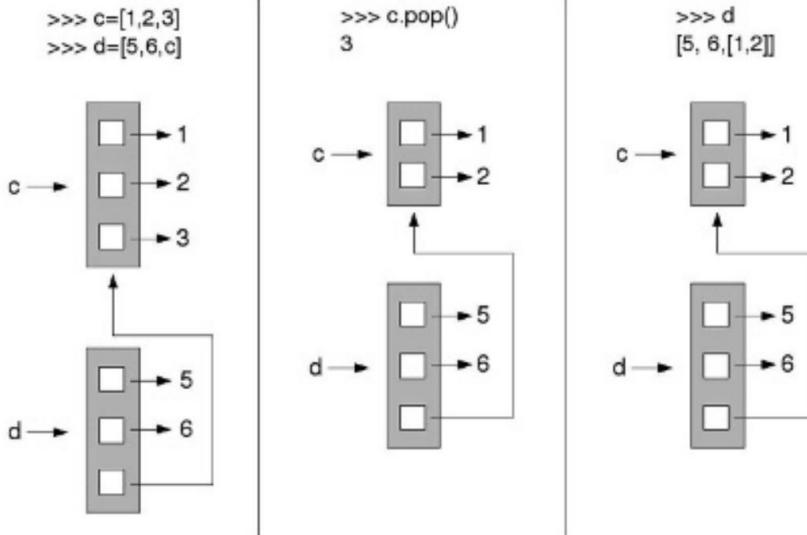
d bilan nima sodir bo‘lishini ko‘rish uchun c qiymatini o‘zgartiramiz:

```
>>> c.pop() 3
>>> c
[1, 2]
>>> d
[5, 6, [1, 2]]
```

Bunday holda, bitta o‘zgaruvchini o‘zgartirish boshqa o‘zgaruvchini o‘zgartiradi. Bu mos kelmaydigan xatti-harakatlarga o‘xshaydi. Agar biz ushbu o‘zgaruvchan topshiriqlarning barchasini ob’ektlar bilan bog‘langan nomlar deb hisoblasak, nomuvofiq bo‘lib tuyulgan narsa mantiqiy bo‘la boshlaydi. 3.5-rasmdan foydalanib, keyingi tushuntirishga amal qilib ko‘ring .

```
>>> a = 3
>>> b = [1, 2, a]
```

## 66 Bioinformatika uchun Python



## 3.6-rasm 2-holat.

Ingliz tiliga tarjima qilinganda ular: "3-ob'ekt a deb nomlansin" va "uchta element (1, 2 va a) bo'lgan ro'yxat b deb nomlansin" degan ma'noni anglatadi.

Chop etish b ikkala bayonotni tasdiqlaydi:

```
>>> b
```

```
[1, 2, 3]
```

Keyin yangi ob'ektni (5) yaratamiz va uni a deb nomlaysiz. Shunday qilib, oldingi mos yozuvlar (a=3) yo'q qilinadi (bu a dan 3 gacha bo'lgan o'qdagi xoch bilan ifodalanadi). a nomi endi 3 ga bog'lanmagan, endi a 5 ga bog'langan. b haqida nima deyish mumkin?

```
>>> a = 5
```

```
>>> b
```

```
[1, 2, 3]
```

b deb nomlangan ro'yxatdagi uchinchi o'rinni o'zgartirilmagan uchun b o'zgartirilmagan bo'lib qoladi. Biz faqat a va 3 orasidagi bog'lashni o'zgartirdik.

Keyingi namunaviy holat Python-da o'zgaruvchilarni belgilash emas, balki ob'ektlarni bog'laydigan nomlar mavjudligini hisobga olgan holda ham tushuntirilishi mumkin. Bunday holda siz [3.6-rasmga amal qilishingiz kerak](#).

```
>>> c = [1, 2, 3] >>> d
```

```
= [5, 6, c]
```

Ingliz tiliga tarjima qilinganda, ular "uch elementli ro'yxat: 1, 2 va 3" deb nomlansin degan ma'noni anglatadi va "Uch elementli ro'yxat: 5, 6 va c (bu a elementning nomi).

uchta element ro'yxati: 1, 2 va 3) d deb nomlanadi. Bu tarkibni yoki ikkala nomni so'rash orqali tasdiqlanishi mumkin:

```
>>> c
[1, 2, 3]
>>> d
[5, 6, [1, 2, 3]]
```

Keyingi bosqichda oxirgi elementni olib tashlash orqali c deb nomlangan ro'yxatni o'zgartiring va qarang d bilan nima sodir bo'ladi:

```
>>> c.pop() 3
>>> chop
etish c [1, 2]
>>> chop
etish d [5, 6, [1, 2]]
```

Bu safar, c o'zgartirildi (va faqat munosabatlar emas). c ning haqiqiy qiymati o'zgartirilganligi sababli, bu har safar chaqirliganda aks etadi. Shubha tug'ilganda **3.6-rasmga** qarang .

Agar nomlar ob'ektlarga bog'langan bo'lsa va Python-da o'zgaruvchan tayinlanmagan bo'lsa ham, odat kuchi kuchli va aksariyat matnlar (hatto bu kitob) o'zgaruvchilar va nomlarni bir -birining o'rnila ishlataladi.

### 3.9 QO'SHIMCHA RESURSLAR

---

- Python yordamida dasturlashni o'rganing: o'zgaruvchilar va identifikatorlar.  
<http://www.developer.com/lang/other/article.php/626321>
- Python 101 — Python tiliga kirish. <http://ascii-world.wikidot.com/python-101>
- Bioinformatika uchun boshlang'ich Python. <http://www.onlamp.com/pub/a/python/2002/10/17/biopython.html>
- Matnni qayta ishslash xizmatlari. <https://docs.python.org/3/library/text.html>
- Python 3 Unicode QANDAY. <https://docs.python.org/3/howto/unicode.html>
- O'rnatilgan ob'ekt turini qo'shish. <http://www.python.org/dev/peps/pep-0218/>
- Python o'rnatilgan turlari. <https://docs.python.org/3/library/stdtypes.html>

## 68 Bioinformatika uchun Python

- dict.keys(), .values() va .items() ni yangilash.  
<http://www.python.org/dev/peps/pep-3106/>
- Rekursiv nuqta belgilaridan foydalanish imkoniyatiga ega Python lug'atlari. <https://github.com/cdgriffith/Box>

### 3.10 O'Z-O'ZI BAHOLASH

---

1. Pythonda qaysi asosiy ma'lumotlar turlari mavjud?
2. Ro'yxat va kortej o'rtaqidagi farq nima? Har biridan qachon foydalanasiz bitta?
3. To'plam nima va siz undan qachon foydalanasiz?
4. Ro'yxat ichida element mavjudligini qanday tekshirish mumkin?
5. Lug'at nima?
6. Lug'atda kalit sifatida qanday ma'lumotlar turidan foydalanish mumkin?
7. "Lug'at ko'rinishi" nima?
8. Tartibsiz ketma-ketlikni takrorlay olasizmi?
9. Quyidagi ma'lumotlar turlarini quyidagi mezonlarga ko'ra tartiblang:
  - O'zgaruvchan – o'zgarmas
  - Saralangan – saralanmagan
  - Ketma-ketlik – xaritalash

Saralash uchun ma'lumotlar turlari: ro'yxatlar, kortejlar, lug'atlar, to'plamlar, satrlar.

10. To'plam va frozenset o'rtaqidagi farq nima?
11. Har qanday takrorlanadigan ma'lumotlar turini ro'yxatga qanday aylantirasiz?
12. Ro'yxat bo'yicha lug'at qanday tuziladi?
13. Lug'atdan ro'yxat qanday tuziladi?

# Dasturlash: oqimni boshqarish

---

## MAZMUNI

4.1 If-Else. ....	69	4.1.1 O'tish bayonoti. ....	
<b>74</b> 4.2 Loop uchun .....			
<b>75</b> 4.3 While Loop. ....			
<b>77</b> 4.4 Tanaffus: halqani buzish .....			<b>78</b>
4.5 Uni oyrash .....			<b>80</b> 4.5.1
Proteinning sof zaryadini hisoblang. ....			80
degeneratsiya zonasini qidirish .....			<b>81</b> Birinchi
versiya. ....			81
While bilan versiya. ....			
<b>82</b> Quyi zanjirlar royxatishiz versiya .....			
<b>82</b> 4.6 Qo'shimcha manbalar .....			
<b>83</b> 4.7 O'z-o'zini baholash. ....			<b>83</b>

Foydali biror narsa qilish uchun dasturlarda ko'rsatmalar qanday va qachon bajarilishini boshqarish mexanizmi bo'lishi kerak. Xuddi shu tarzda, svetoforlar ko'chada avtomobil oqimini boshqaradi, oqimni boshqarish tuzilmalari ma'lum bir vaqtda kod qismi bajarilishini boshqaradi.

Python faqat uchta oqimni boshqarish tuzilmalariga ega. Bitta shartli va ikkita iteratsion tuzilmalar mavjud. Shartli tuzilma (*agar*) ifodani baholashdan so'ng kod bloki bajarilgan yoki bajarilmaganligini aniqlaydi. Iteratsiya tuzilmalari (*for* va *while*) bir xil kod qismini bir necha marta bajarishga imkon beradi. Iteratsiya tuzilishi bilan bog'langan kod necha marta bajariladi? A *for* cycle kod blokini ko'p marta bajaradi, chunki elementlar ma'lum bir takrorlanadigan elementda mavjud bo'lib, ma'lum bir davrdagi kod berilgan shart noto'g'ri bo'lguncha bajariladi.<sup>1</sup>

---

## 4.1 AGAR-BOSHQA

Eng klassik boshqaruv tuzilmasi shartli hisoblanadi. U baholash natijasiga ko'ra harakat qiladi. Agar siz boshqa kompyuter tilini bilsangiz, *if-else* bilan tanish bo'lishingiz mumkin.

<sup>1</sup>Bu shart to'g'ri bo'lganda shart bajarilishini aytishga teng.

## 70 Bioinformatika uchun Python

Ifodani baholasa. Agar ifoda rost bo'lsa, kod bloki faqat keyin if bandi bajariladi. Aks holda, else ostidagi blok bajariladi.

If-else shartining asosiy sxemasi,

agar ifoda:

BLOK1

boshqa:

BLOK2

EXPRESSION to'g'ri yoki noto'g'ri qaytaruvchi ifoda bo'lishi kerak. Barcha taqqoslash operatorlari kabi:  $x < y$  (kichik),  $x > y$  (katta),  $x == y$  (teng),  $x != y$  (teng emas),  $x <= y$  (kichik yoki teng),  $x >= y$  (katta yoki teng).

Keling, misolni ko'rib chiqaylik:

---

### Listing 4.1: ifelse1.py: asosiy if-else namunasi

---

```
1 balandlik = float(input('Balandlik nima? (metrda): '))
2 agar balandlik >
1,40 bo'lsa: 3
    chop etish ("Siz kirishingiz mumkin")
4 ta boshqa:
5     chop etish ('Bu sayohat siz uchun emas')
```

---

Dastur chiqishi,

balandlik nima? (metrda): 1.80 Siz kirishingiz  
mumkin

---

Maslahat: Ushbu kitobdag'i kod haqida.

**Listing 4.1** (yoki ushbu kitobdag'i boshqa) dagi kodni kiritishingiz shart emas .  
Uni <https://github.com/> Serulab/Py4Bio manzilidagi GitHub omboridan yuklab olish mumkin .  
Bundan tashqari, Microsoft Azure noutbuklarida onlayn foydalanish mumkin (<https://notebooks.azure.com/library/py3.us>). Ikkala havola ham kitobning veb-saytida mavjud (<http://py3.us/>).

Kodni shunchaki o'qishdan ko'ra (mahalliy yoki onlayn) bajarishga harakat qiling  
bu kitob.

---

Yana bir misol,

**Listing 4.2:** ifelse2.py: if-else amalda

---

```
1 three_letter_code = {'A':'Ala','N':'Asn','D':'Asp','C':'Cys'} 2 aa = kiritish('Bir harf
kirititing:') 3 bo'lsa aa uch_harfli_kodda: 4

    print('{0} uchun uch harfli kod: {1}'.format(aa, uch_harfli_kod[aa]))
5
Yana 6:
7     print('Kechirasiz, bu mening lug'atimda yo'q')
```

---

Dastur chiqishi,

Bitta harf kriting: A

A uchun uchta harfli kod: Ala

Bir nechta shartlarni baholash uchun elifdan foydalaning:

agar IFOTO 1:

    BLOK1

elif EXPRESSION2:

    BLOK2

elif EXPRESSION3:

    BLOK 3

boshqa:

    BLOK 4

Siz o'zingiz baholamoqchi bo'lgan shartlar qadar elifdan foydalanishingiz mumkin.

Biror shart rost deb baholangandan keyin qolgan shartlar tekshirilmasligini hisobga oling.

Quyidagi dastur elif yordamida bir nechta shartlarni baholaydi:

**Listing 4.3:** elif1.py: elifdan foydalanish

---

```
1 dna = input('Asosiy ketma-ketlikni kriting: ') 2 seqsize =
len(dna) 3, agar ketma-ketlik o'lchami < 10: 4 bo'lsa
```

```
    print('Primer kamida o'n nukleotidga ega bo'lishi kerak') 5 elif seqsize
< 25: print('Bu o'lcham yaxshi') 6
```

7 boshqa:

```
8     chop etish ('Past juda uzun')
```

---

Ushbu dastur (elif1.py) ish vaqtida klaviatura bilan kiritilgan DNK ketma-ketligiga ega qatorni so'raydi. Bu ketma-ketlik DNK deb ataladi . 2-qatorda uning o'lchami hisoblanadi

## 72 Bioinformatika uchun Python

va bu natija seqsize nomi bilan bog'langan . 3-qatorda baholash mavjud. Agar ketma-ketlik o'ndan past bo'lsa, "Primerda kamida o'n nukleotid bo'lishi kerak" xabari chop etiladi. Dastur oqimlari if operatorida boshqa hech qanday shartni baholamasdan, bu if operatorining oxirigacha boradi. Lekin agar u to'g'ri bo'lmasa (masalan, ketma-ketlik uzunligi 15 bo'lsa), u shart rost deb baholangan taqdirda keyingi shartni va unga bog'langan blokni bajaradi. Agar ketma-ketlik uzunligi 10 dan katta bo'lsa, dastur 4-qatorni (birinchi shart bilan bog'langan kod bloki) o'tkazib yuboradi va 5-qatordagi ifodani baholaydi. Agar bu shart bajarilsa, u "Ushbu o'lcham" ni chop etadi. yaxshi". Agar rost deb baholanadigan ifoda bo'lmasa, else bloki bajariladi.

---

Maslahat: Nima haqiqat?

Esda tutingki, if shartidan **keyingi gap faqat ifoda True deb baholanganda bajariladi** . Shunday qilib, "**To'g'ri nima?**" Degan savollar. (va "Nima noto'g'ri?") tegishli.

**To'g'ri nima ?:**

- Bo'sh bo'limgan ma'lumotlar tuzilmalari (ro'yxatlar, lug'atlar, kortejlar, satrlar, to'plamlar). Bo'sh ma'lumotlar tuzilmalari noto'g'ri deb hisoblanadi.
- **0 va None False (boshqa qiymatlar True) deb hisoblanadi .**
- True kalit so'zi **True va False - False .**

Agar ifoda rost yoki noto'g'ri ekanligiga shubhangiz bo'lsa , **bool()**:

**>>> bool(1=='1')**

Yolg'on

---

**Shartlarni joylashtirish mumkin:**

**Listing 4.4: nested.py: Nested if**

---

```
1 dna = raw_input('DNK ketma-ketligini kiriting: ')
2 seqsize =
len(dna)
3, agar ketma-ketlik < 10: 4 bo'lsa
5     print('Sizning astaringizda kamida o'n nukleotid bo'lishi kerak') if seqsize == 0:
6         print('Siz biror narsa kiritishingiz kerak!') 7 elif seqsize < 25: 8
7
8     print('Bu o'lcham yaxshi') 9 ta
9 boshqa:
10    chop etish ("Sizning astaringiz juda uzun")
```

---

5-qatorda boshqa bir shart mavjud.

Yagona teng o'rniga ikki barobar belgisiga ("==" ) e'tibor bering. Ikki barobar qiymatlarni solishtirish uchun, teng belgisi esa qiymatlarni belgilash uchun ishlataladi:

```
>>> javob=42
>>> javob
42
>>> javob==3
Yolg'on
>>> javob==42
To'g'ri
```

Agar [ro'yxat 4.4da kiritilgan bo'lса](#), ichki o'rnlardan qochish mumkin:

#### Listing 4.5: elif2.py: Agar ichki o'rnatilgan

---

```
1 dna = raw_input('DNK ketma-ketligini kriting: ')
2 seqsize = len(dna)
3 agar ketma-ketlik o'lchami == 0: 4 bo'lса
4
5     print('Siz biror narsa kiritishingiz kerak!')
6 elif 0 < seqsize < 10: print('Sizning astaringizda kamida o'n
7     nukleotid bo'lishi kerak')
8 elif seqsize < 25: 9
9     print('Bu o'lcham yaxshi')
10    boshqa:
11        chop etish ("Sizning astaringiz juda uzun")
```

5-qatorda ifoda qanday baholanishini ko'ring. Bu bizni bittaga bir nechta iboralarni kiritish haqida o'yylashga olib keladi, agar, [4.6. Ro'yxatdagи kabi](#):

#### Listing 4.6: multipart.py: Ko'p qismli shart

---

```
1 x = 'Yo'q'
2 agar x != 'N/A' va 5 < float(x) < 20 bo'lса:
3     chop etish ('OK')
4 ta boshqa:
5     chop etish ('Yaxshi emas')
```

Bu ifoda chapdan o'ngga qarab baholanadi. Agar ifodaning bir qismi noto'g'ri bo'lса, keyingi qismilar baholanmaydi. x='N/A' bo'lgani uchun dastur 'Not OK' chop etadi (chunki birinchi shart noto'g'ri). Qarang, bir xil ifoda teskari tartibda yozilsa nima bo'ladi.

## 74 Bioinformatika uchun Python

Ushbu ro'yxat (`multiplepart2.py`),

**Listing 4.7: `multiplepart2.py`: Ko'p qismli shart, teskari**

---

```
1 x = 'Yo'q' 2,  
agar 5 < float(x) < 20 va x != 'N/A' bo'lsa:  
3     chop etish ('OK')  
4 ta boshqa:  
5     chop etish ('Yaxshi emas')
```

---

qaytaradi:

**Traceback (eng oxirgi qo'ng'iroq):**

```
Fayl "<pyshell#2>", 1-qator, <modul> da, agar 5 <  
    float(x) < 20 va x != 'N/A' bo'lsa: ValueError: satrni  
floatga aylantirib bo'lmasdi: 'N/A'
```

ValueError qaytarildi, chunki "N/A" qatorini floatga aylantirib bo'lmasdi. **4.7 ro'yxatida** x ham "Yo'q" sifatida baholanadi, ammo ValueError mavjud emas, chunki ifodaning bu qismi baholashdan oldin o'tkazib yuborilgan.

### 4.1.1 O'tish bayonoti

Ba'zan if iborasida muqobil tanlovga ehtiyoj qolmaydi. Bunday holda siz boshqasini ishlatalishdan ochishingiz mumkin:

agar ifoda:

**BLOK**

# Dasturning qolgan qismi...

Xuddi shu kodni yanada o'qilishi uchun Python o'tish bayonotini taqdim etadi. Bu gap to'ldiruvchiga o'xshaydi; gap sintaktik jihatdan talab qilinganda biror narsani qo'yishdan boshqa maqsadi yo'q. Quyidagi kod oldingi kod bilan bir xil natijani beradi:

agar SHART:

**BLOK**

boshqa:

**o'tish**

# Dasturning qolgan qismi ...

---

Kengaytirilgan maslahat: shartli ifodalar.

Ba'zan foydali bo'ladi: bitta satrda if shartini yozish uchun maxsus sintaksis mavjudligi

ifoda1 if shart else ifoda2 **Agar** shart rost bo'lsa , bu satr

ifoda1 qiymatini oladi ; aks holda u ifoda 2 qiymatini oladi .

Ushbu sintaksis bizga yozishga imkon beradi:

```
>>> jami = 5
>>> elementlar = 2
>>> chop ('O\'rtacha = {0}'.format(jami/elementlar, agar elementlar != 0 bo\'lsa, "Yo'q"))
O'rtacha = 2,5

o'rniga,

>>> jami = 5
>>> elementlar = 2
>>> agar elementlar != 0 bo\'lsa:
...      chop etish ('O\'rtacha = {0}'.format (jami/elementlar))
... boshqa:
...      chop etish ('O\'rtacha = N/A')
...
O'rtacha = 2,5
```

---

## 4.2 FOR LOOP

Bu boshqaruv tuzilmasi o'zgaruvchini takrorlanuvchi ob'ekt qiymatiga ega bo'lgan holda saqlagan holda kodni qayta-qayta bajarishga imkon beradi.<sup>2</sup> For tsiklining umumiy shakli:

**ITERABLE da VAR uchun:**

**BLOK**

**Masalan:**

**ba'zi\_ro'yxatdagi har bir\_element uchun:**

```
# Har bir elementni chop etish (har
bir_element) bilan biror narsa qiling
```

Birinchi qator oxiridagi ikki nuqtaga e'tibor bering. Bu majburiy. Kod blokining chegarasi sifatida ikki nuqta for tsiklining bir qismidir. Ushbu tuzilma BLOCK ning takrorlanadigan ob'ektdagi elementlar qanchalik ko'p bo'lsa, shuncha takrorlanishiga olib keladi. Har bir iteratsiyada V AR IT ERABLE-dagi joriy elementning qiymatini oladi. Quyidagi kodda to'rt elementli ro'yxat (asoslar) bo'ylab yurish uchun. Har bir iteratsiyada x ro'yxatdagi elementlardan birining qiymatini oladi.

```
>>> asoslar = ['C', 'T', 'G', 'A'] >>>
```

**asoslardagi x uchun:**

```
...      chop etish(x)
...
...
```

<sup>2</sup>Eng keng tarqalgan takrorlanadigan ob'ektlar ro'yxatlar, kortejlar, satrlar va lug'atlardir. Fayllar va moslashtirilgan ob'ektlar ham takrorlanishi mumkin.

## 76 Bioinformatika uchun Python

C  
T  
G  
A

Takrorlanayotgan iteratsiya holatini bilish uchun enumerate usuli qiymat bilan birga iteratsiya indeksini qaytaradi.

```
>>> asoslar = ['C', 'T', 'G', 'A'] >>> uchun
n, x uchun sanab(asos): print(n, x)
...
...
0 C
1 T
2 G
3 A
```

Boshqa tillarda for tsikli hisoblagich o'zgaruvchisini o'zgartirganda kod blokining bir necha marta ishlashiga ruxsat berish uchun ishlatalidi. Ushbu xatti-harakat Python-da raqamlar ro'yxatini takrorlash orqali takrorlanishi mumkin:

```
>>> [0, 1, 2, 3, 4] dagi n uchun: print(n)
...
...
0
1
2
3
4
```

Raqamlar ro'yxatini takrorlashning yana bir muqobil varianti ularni o'rnatilgan funksiya3 diapazoni (n) yordamida yaratishdir. Bu funksiya takrorlanadigan ob'ektni qaytaradi. Har safar qo'ng'iroq qilganingizda, funktsiyaga kiritilgan 0 dan birinchi parametrgacha minus bir (ya'ni, n-1) gacha bo'lgan raqamni qaytaradi.

```
>>> diapazondagi x uchun(4):
...     chop etish(x)
...
0
1
2
3
```

---

<sup>3</sup>Barcha o'rnatilgan funksiyalar <https://docs.python.org/3.6/library/functions.html>.

Quyidagi kod oqsilning molekulyar og'irligini uning alohida aminokislotalariga qarab hisoblab chiqadi.<sup>4</sup> Aminokislota satrda saqlanganligi sababli, dastur har bir harfni for yordamida bosib o'tadi.

#### **Listing 4.8: protwfor.py: Proteinning og'irligini aniqlash uchun for dan foydalaning**

---

```
1 prot_seq = kiritish ('Oqsil ketma-ketligini kiriting: ') 2 prot_weight
= {'A':89, 'V':117, 'L':131, 'I':131, 'P':115, 3 'F ':165, 'W':204, 'M':149, 'G':75,
'S':105, 'C':121, 'T':119, 'H':130}, pNot_132 'Q':146 'D':133 'E':147 'K':146, 'R':174,
4 total_weight = umumiy_og'irlik + prot_weight.get(aa.upper()),
5 0) 8
```

9 umumiy\_og'irlik = umumiy\_og'irlik - (18 \* (len(prot\_seq) - 1)) 10 ta chop  
etish('Sof vazn: {0}'.format(jami\_og'irlik))

Kod tushuntirishi: Birinchi qatorda foydalanuvchidan proteinlar ketma-ketligini kiritish so'raladi (masalan, MKTFVLHIFIFALVAF). Kirish orqali qaytariladigan satr protseq deb nomlanadi. 2-qatordan 5-qatorgacha aminokislota og'irliliklari bilan lug'at (protweight) ishga tushiriladi. Protseqdagi har bir elementni takrorlash uchun 7-qatorda for tsikli ishlatiladi. Har bir iteratsiyada aa protseq elementidan qiymat oladi. Bu qiymat protweight lug'atda qidirish uchun ishlatiladi. Tsikldan keyin jami W barcha aminokislotalarning og'irligi yig'indisi bilan yakunlanadi. 9-qatorda har bir bog'lanish suv molekulasini (molekulyar og'irligi 18) yo'qotishni o'z ichiga olganligi sababli tuzatish mavjud. Oxirgi satr aniq vaznni chop etadi.

### **4.3 WHILE LOOP**

---

Loop for ga juda o'xshaydi, chunki u kod qismini ham takroriy tarzda bajaradi. Bu holda ob'ekt ustida iteratsiya bo'lmaydi, shuning uchun bu tsikl takrorlash ob'ekti o'tganda tugamaydi, lekin berilgan shart to'g'ri bo'lmasa ganda.

while tsikli modeli:

Ifoda esa:

BLOK

Blok ichida while shartini noto'g'ri qilish uchun ko'rsatma bo'lishi kerakligini hisobga olish juda muhimdir. Aks holda, biz cheksiz tsiklga kirishimiz mumkin.

<sup>4</sup>Aminokislolar oqsillarning qurilish bloklaridir. Har bir aminokislota (bitta harf bilan ifodalangan) individual vaznga ega. Har bir aminokislota aloqasi suv molekulasini (og'irligi 18 iu) chiqaradiganligi sababli, chiqarilgan barcha suv molekulalarining og'irligi jamidan chiqariladi.

## 78 Bioinformatika uchun Python

```
>>> a = 10
>>> esa < 40:
...     chop
...     etish(a) a += 10
...
10
20
30
```

Vaqt oralig'idan chiqish yo'li break dan foydalanishdir. Bu holda halqa holatini baholamasdan, halqa buziladi. break ko'pincha har doim to'g'ri bo'lgan shart bilan birga ishlataladi:

```
>>> a = 10
>>> rost bo'lsa:
...     a < 40 bo'lsa:
...         chop etish(a)
...     boshqa:
...         tanaffus
...     a += 10
...
10
20
30
```

Bu pastadir ichidagi blokning kamida bir marta bajarilishini ta'minlash uchun amalga oshiriladi. Boshqa tillarda bu holatlar uchun alohida sikl turi mavjud (do while), lekin u Python.5 da mavjud emas.

#### 4.4 BUZISH: LOOPNI UZISH

---

**break** loop strukturasidan qochish uchun ishlataladi. Biz hozirgina **while** bilan foydalanish misolini ko'rdik, lekin u for ostida ham ishlatalishi mumkin.

Avvaliga **break** iborasi qayerda ekanligini tushunish oson emas tuyg'u.

Masalan, [4.9 ro'yxatini olaylik](#):

---

##### Listing 4.9: seachinlist.py: Kortejlar ro'yxatida qiymat qidirish

---

```
1 rang_kodi = [('qizil', 1), ('yashil', 2), ('ko'k', 3), ('qora', 4)]
2 ism = "ko'k"
```

**rang\_kodidagi rang\_juftligi** uchun  
bo'lsa: **3:** agar nom == rang\_jufti[0]

---

5A ushbu tuzilmani Python-ga qo'shish taklifi 2013 yilda rad etilgan.

---

```
5         kod = rang_jufti[1] 6
bosib chiqarish(kod)
```

Ushbu kodda `color_code` ro'yxatini takrorlash uchun for tsikli mavjud. Har bir element uchun, ya'ni har bir kortej uchun birinchi elementni tekshiradi. Bizning so'rovimizga (*ism*) mos kelganda, dastur tegishli kodni kodda saqlaydi.

Shunday qilib, ushbu dasturning natijasi "3" dir.

Ushbu dastur bilan bog'liq muammo shundaki, biz kerak bo'lmasa ham, butun ketma-ketlikni bosib o'tishadi. Bunday holda, 4-qatordagi shart rang kodidagi har bir element uchun bir marta baholanadi, agar moslik ijobiy bo'lsa, sinovni davom ettirishning hojati yo'qligi aniq bo'lsa. Ijobiy o'yindan so'ng halqani uzib, biroz vaqt va ishlov berish quvvatini tejashingiz mumkin:

#### **Listing 4.10: seachinlist2.py: To'plamlar ro'yxatida qiymat qidirish**

---

```
1 rang_kodi = [('qizil',1), ('yashil',2), ('ko'k',3), ('qora',4)] 2 nom = 'ko'k'
rang_kodidagi rang_juftligi uchun 3:
4     agar nom == rang_jufti[0] bo'lsa:
5         kod = rang_jufti[1] tanaffus
6
7 bosib chiqarish (kod)
```

Bu kod [Listing 4.9](#) bilan bir xil, 6-qatordagi break bayonoti bundan mustasno. Chiqish avvalgidek bo'ladi, lekin bu safar element topilgandan so'ng ketma-ketlik bo'yicha takrorlanadigan CPU sikllarini behuda sarflamaysiz. Ushbu misolda saqlangan vaqt ahamiyatsiz, lekin agar dastur buni katta ro'yxat yoki faylda bir necha marta bajarishi kerak bo'lsa (siz faylni takrorlashingiz mumkin), break uni sezilarli darajada tezlashtirishi mumkin.

Tanaffusdan foydalanishning oldini olish mumkin, ammo natijada olingan kod [4.10 Listingdagi kabi o'qilishi mumkin emas](#):

#### **Listing 4.11: seachinlist3.py: To'plamlar ro'yxatida qiymat qidirish**

---

```
1 rang_kodi = [('qizil',1), ('yashil',2), ('ko'k',3), ('qora',4)]
2 ism = "ko'k"
3 i = 0
4 while name != color_code[i][0]:
5     i += 1
6 kod = rang_kodi[i][1] 7 bosib
chiqarish(kod)
```

---

Bunday holatda, xotiraga osongina sig'adigan ro'yxat bilan, bu yaxshi fikr lug'at yaratish va uni so'rash uchun:

---

**Listing 4.12: seachindict.py: Dictio yordamida kortejlar ro'yxatida qiymatni qidirish nari**


---

```
1 rang_kodi = [('qizil',1), ('yashil',2), ('ko'k',3), ('qora',4)] 2 nom = "ko'k" 3
rang_kodi_d = dict(rang_kodi) 4 chop etish (rang_kodi[nom])
```

---

## 4.5 UNNI QO'YISH

---

Endi biz if, for, while va shu nuqtagacha ko'rilgan ma'lumotlar turini birlashtiramiz. Bu erda men o'zimiz o'rgangan vositalar yordamida yaratilgan bir nechta kichik dasturlarni taqdim etaman:

### 4.5.1 Proteinning sof zaryadini hisoblang

Ruxsat etilgan pHda oqsilning aniq zaryadini uning alohida aminokislotalarining zaryadini yig'ib hisoblash mumkin. Bu taxminiy taxmindir, chunki u aminokislotalarning oqsil tuzilishiga ta'sir qilishini yoki ko'milganligini hisobga olmaydi.

Ushbu dastur, shuningdek, sistein faqat disulfid ko'priginining bir qismi bo'limganida zaryad qo'shishini hisobga olmaydi. Bu taxminiy qiymat bo'lganligi sababli, olingan qiymatlarni baholash sifatida qabul qilish kerak. Mana protnetcharge.py ning birinchi versiyasi:

---

**Listing 4.13: protnetcharge.py: Proteinning aniq zaryadi**


---

```
1 prot_seq = kiritish('Protein ketma-ketligini kiriting: ').yuqori() 2
zaryad = -0,002 3 aa_charge = {'C':-045,'D':-999,'E':-998,'H':091,
4 'K':1,'R':1,'Y':-001} 5 prot_seq aa uchun: 6 bo'lsa aa aa_chargedan:
zaryad += aa_charge[aa] 8 bosib chiqarish(zaryad) )
```

7

---

6-qatordagi if iborasidan get() bilan qochish mumkin:

---

**Listing 4.14: protnetcharge2.py: Get yordamida oqsilning aniq zaryadi**


---

```
1 prot_seq = kiritish('Protein ketma-ketligini kiriting: ').yuqori() 2
zaryad = -0,002 3 aa_charge = {'C':-045,'D':-999,'E':-998,'H':091,
'K':1,'R':1,'Y':-001} prot_seqdagi aa uchun 4 5:
```

---

```
7      zaryad += aa_charge.get('aa', 0) 8 bosib
chiqarish(zaryad)
```

---

#### 4.5.2 Past degeneratsiya zonasini qidirish

PCR primerlarini topish uchun kamroq degeneratsiyaga (yoki ko'proq konservatsiyaga) ega bo'lgan DNK mintaqasidan foydalanish yaxshiroqdir. Bu bizga maqsadli ketma-ketlikni topish uchun yaxshi imkoniyat beradi. Ushbu dasturning maqsadi ushbu mintaqani qidirishdir. PCR primeri taxminan 16 nukleotidga ega bo'lganligi sababli, primer dizayni uchun joy berish uchun qidiruv maydoni kamida 45 nukleotid bo'lishi kerak. Biz kirish ketma-ketligida 15 aminokislota hududini topishimiz kerak. 15-aminokislotalar 45 ta nukleotidni qidirish hududini ta'minlaydi (har bir aminokislota uchun 3 nukleotid).

Har bir aminokislota ma'lum miqdordagi kodonlar bilan kodlangan. Misol uchun, valin (V) to'rt xil kodon (GTT, GTA, GTC, GTG) bilan kodlanishi mumkin, triptofan (W) esa faqat bitta kodon (TGG) bilan kodlanadi. Shunday qilib, valinlarga boy mintaqaga triptofan ko'p bo'lgan mintaqaga qaraganda ko'proq o'zgaruvchanlikka ega bo'ladi.

Past degeneratsiyali hududni topadigan dastur:

#### Birinchi versiya

**Listing 4.15: lowdeg.py: Past degeneratsiya zonasini qidiring**

---

```
1 prot_seq = kiritish('Proteinlar ketma-ketligi: ').yuqori() 2
prot_deg = {'A':4, 'C':2, 'D':2, 'E':2, 'F':2, 'G':4, 3 'H':2, 'I':3, 'K':2, 'L':6, 'M':1,
'N':2, 'P':4, 'Q':2, 'R':6, 'S':6, 'T':4, 'V':4} 4 M(leh(prot_seq))
4
5

    segment = prot_seq[aa:aa + 15] degen
9     = 0, agar len(segment)==15: segmentdagi
10    x uchun: degen += prot_deg.get(x, 3.05)
11        segs_values.append(degen) 14
12            min_value = min(segs_values) ) 15
13            minpos = segs_values.index(min_value) 16
chop (prot_seq[minpos:minpos + 15])
```

---

**Kod tushuntirishi:** Foydalanuvchi tomonidan kiritilgan qatorni (prot\_seq) oladi. Dastur har bir aminokislota mos keladigan kodonlarning SONini saqlash uchun lug'atdan (prot\_deg) foydalanadi. 7 dan 9 gacha bo'lgan qatorda biz uzunligi 15 bo'lgan toymasin oynalarini hosil qilamiz. Har bir 15 aminokislota segmenti uchun kodonlar soni baholanadi, keyin biz

## 82 Bioinformatika uchun Python

degeneratsiyasi kamroq bo'lgan segmentni tanlang (14-qator). E'tibor bering, 10-qatorda segmentning o'lchamini tekshirish mavjud, chunki prot\_seq ketma-ketligi siljib ketganda, pastki zanjirda 15 dan kam aminokislotalar mavjud.

While bilan versiya

**Listing 4.16:** lowdeg2.py: Past degeneratsiya zonasini qidirish; while bilan versiya

```

1 prot_seq = kiritish('Proteinlar ketma-ketligi: ').yuqori() 2
prot_deg = {'A':4, 'C':2, 'D':2, 'E':2, 'F':2, 'G':4, 3 'H':2, 'I':3, 'K':2, 'L':6,
'M':1, 'N':2, 'P':4, 'Q':2, 'R':6, 'S':6} 7 segs_values = []
8 segs_seqs = [] 9 segment =
4 prot_seq[:15]

9 a = 0
10 esa len(segment)==15:
segmentdaideyeshun=11 degen
12     prot_deg.get(x, 3.05)
13         segs_values.append(degen)
14     segs_seqs.append(segment) a += 1
15
16
17     segment = prot_seq[a:a+15] 18
chop (segs_seqs[segs_values.index(min(segs_values))])

```

Kod tushuntirishi: Bu versiyada prot\_seq ustidan yurish uchun for ishlatilmaydi; o'rniga while dan foydalanadi. Sürgülü oyna prot\_seq ichida ekan, kod bajariladi.

Pastki zanjirlar ro'yxatisiz versiya

**Listing 4.17:** lowdeg3.py: pastki zanjirlarsiz past degeneratsiya zonasini qidirish

```

1 prot_seq = kiritish('Proteinlar ketma-ketligi: ').yuqori() 2
prot_deg = {'A':4, 'C':2, 'D':2, 'E':2, 'F':2, 'G':4, 3 'H':2, 'I':3, 'K':2, 'L':6,
'M':1, 'N':2, 'P':4, 'Q':2, 'R':6, 'S':6} 7 max_degen_tmp = 11
8 degen_tmp = 0 9 segment =
4 prot_seq[:15]
5

```

```

n diapazon uchun 7 (len(prot_seq) - 15):
8     prot_seq[n:n
9     + 15] ichida x uchun degen = 0:
10    degen += prot_deg.get(x, 3.05) agar degen
11    <= degen_tmp: degen_tmp = degen seq =
12        prot_seq[n:n + 15]
13
bosma (ketma-ket)

```

**Kod tushuntirish:** Bu holda har bir degeneratsiya qiymati oxirgisi bilan taqqoslanadi (10-qator) va agar joriy qiymat pastroq bo'lsa, u saqlanadi. E'tibor bering, degeneratsiya qiymati birinchi marta baholanganda, uni solishtirish uchun hech qanday qiymat yo'q. Ushbu muammo 6-qatorda tartiblangan, bu erda maksimal nazariy qiymat berilgan.

## 4.6 QO'SHIMCHA RESURSLAR

---

- Python qo'llanmasi: Boshqarish oqimining ko'proq vositalari.  
<https://docs.python.org/3.6/tutorial/controlflow.html>
- Python dasturlash: Oqimni boshqarish.  
[http://en.wikibooks.org/wiki/Python\\_Programming/Flow\\_control](http://en.wikibooks.org/wiki/Python_Programming/Flow_control)
- Python asoslari: oqimni boshqarish bayonotlarini tushunish. <https://goo.gl/ss6uNh>
- Qisqacha aytganda Python, Ikkinchi nashr. Aleks Martelli tomonidan. 4 -bob .  
Ko'chirma <http://www.devshed.com/c/a/Python/The-Python-Language>
- Python sindirish, davom ettirish va topshirish bayonotlari.  
[http://www.tutorialspoint.com/python/python\\_loop\\_control.htm](http://www.tutorialspoint.com/python/python_loop_control.htm)

## 4.7 O'Z-O'ZI BAHOLASH

---

1. Boshqaruv tuzilmasi nima?
2. Python nechta boshqaruv strukturasiga ega? Ularga nom bering.
3. Qachon uchun, while ni qachon ishlatardingiz?
4. Ba'zi tillarda do while boshqaruv tuzilishi mavjud. Qanday qilib shunga o'xshash narsani olishingiz mumkin Python da funksiyasi bormi?
5. Qachon pass va qachon breakdan foydalanishingizni tushuntiring .
6. **4.16-listning 6-qatorida while ostidagi shart len(P rotSeq[i : i + 15]) == 15 dan i < (len(P rotSeq) ý 7) ga o'zgartirilishi mumkin. Nega?**

## 84 Bioinformatika uchun Python

7. Barcha mumkin bo'lgan IP manzillarni, ya'ni 0.0.0.0 dan chiqaradigan dastur tuzing  
255.255.255.255 gacha.
8. Ikki o'zgaruvchili chiziqli tenglamani yechish dasturini tuzing. Tenglama  
bu shaklga ega bo'lishi kerak:  
$$a1.x + a2.y = a3$$
$$b1.x + b2.y = b3$$

Dastur a1, a2, a3, b1, b2 va b3 ni so'rashi va x va y qiymatini qaytarishi kerak.
9. Berilgan sonning palindrom ekanligini tekshirish dasturini tuzing (ya'ni raqamlari teskari  
aylantirilganda ham o'zgarmaydi, masalan, 404).
10. Farengeyt haroratini Selsiyga aylantirish dasturini tuzing va natijani faqat bitta kasrli qiymat  
bilan yozing. Konvertatsiya qilish uchun ushbu formuladan foydalaning:  $T_c = (5/9) \circ C + 32$
11. Quyidagi ekvivalentdan foydalanib, siz kiritgan hamma narsani Leetspeak-ga aylantiradigan  
dastur tuzing: O uchun 0, I (yoki L uchun 1), Z uchun 2 (yoki R), E uchun 3, A uchun 4, S  
uchun 5, G (yoki B) uchun 6, T (yoki L) uchun 7, B uchun 8 va P (yoki G va Q) uchun 9.  
Shunday qilib, "Salom dunyo!" "H3770 w02ld!" sifatida ko'rsatilgan.
12. Ikki so'z berilgan bo'lsa, dastur ularning qofiya yoki qofiya emasligini aniqlashi kerak. Bu savol  
uchun "qofiya" oxirgi uchta harfning sehrgar va kaltakesak kabi bir xil ekanligini anglatadi.
13. Bir harfli kodda oqsil ketma-ketligini hisobga olib, metionin (M) va sistein (C) foizini hisoblang.  
Misol uchun, MFKFASAVILCLVAASSTQA dan natija 10% bo'lishi kerak (20 ta aminokislotadan  
1 M va 1 C).
14. [Listing 4.17](#) kabi dastur tuzing , lekin oldindan belgilangan maksimaldan foydalanmasdan  
qiymat.

# Fayllar bilan ishlash

---

## MAZMUNI

<b>5.1 Fayllarni o'qish .. . . . .</b>	<b>86</b>
5.1.1 Fayllar bilan ishlash misoli. ....	87 5.2
<b>Fayllarni yozish. ....</b>	<b>89</b>
5.2.1 Fayllarni o'qish va yozishga misollar. ....	90 5.3
<b>CSV fayllari. ....</b>	<b>90 CSV</b>
modulidan boshqa funksiyalar .. . . . .	92 Pickle: fayllario'zgaruvchilar.tarkibini.saqlash va olish....
	94 5.4.5.5.JS0Fayl bilan ishlash: os, os.path, shutil va path.py moduli .. . . . .
5.6 moduli .. . . . .	98 5.6.1 path.py 100 5.6.2 Bir nechta DNK ketma-ketliklarini bitta FASTA faylida birlashtirish. ... 102 5.7
Qo'shimcha manbalar .. . . . .	
<b>102 5.8 O'z-o'zini baholash. ....</b>	
<b>103</b>	

Fayllarni o'qish va saqlash ko'pgina dasturlarning muhim qismidir. Ushbu bobda har qanday matn faylini o'qish va yozish ko'rsatilgan. Ushbu kitobning maqsadlari uchun "matnli faylni o'qish" fayldan ma'lumotlarni dasturga kiritish jarayonidir. Keyinchalik tahlil qilish uchun ma'lum bir qismni olish uchun ifodaning sintaktik tuzilishini aniqlash jarayoni tahlil deb ataladi.

Masalan, quyidagi faylni oling:

1867864656,1,BOT,[T/C],0009803928,Homo sapiens  
1867864658,10,BOT,[A/T],0021792978,Homo sapiens  
1867864660,100,BOT,[A/G],0069608915,Homo sapiens

Har bir satrda vergul bilan ajratilgan turli xil ma'lumotlar birliklari mavjud (ko'pincha ma'lumotlar nuqtalari deb ataladi). Fayl o'qilganda, Python har bir satrni bitta satr sifatida taniydi, shuning uchun undagi oltita ma'lumot nuqtasining har birini tanib olish uchun qo'shimcha qadam kerak bo'ladi. Bu bosqich tahlil qilinadi. Tahlil qilish bosqichi ma'lumotlarning formatiga bog'liq, shuning uchun matnni tahlil qilishning universal usuli mavjud emas. Ushbu bob 90-betda vergul, nuqtali vergul yoki yorliq belgisi (odatda CSV fayllari deb ataladi) kabi maxsus belgi bilan ajratilgan ma'lumotlarni qanday tahlil qilishni ko'rsatadi. JSON va XML kabi ma'lumotlar almashinuvni uchun boshqa mos

## 5.1 FAYLLARNI O'QISH

---

Pythonda faylni o'qish uch bosqichli jarayondir:

**1. Faylni oching:** Open deb nomlangan o'rnatilgan funksiya mavjud bo'lib, u dle faylini yaratadi. Ushbu fayl ishlovchisi faylning ishlash muddati davomida faylga murojaat qilish uchun ishlatiladi. Ochiq funksiya ikkita parametri oladi: fayl nomi va ochilish rejimi. Fayl nomi ko'p hollarda tizim yo'lini o'z ichiga olgan fayl nomiga ega bo'lgan qatordir. Tizim yo'li kiritilganda, bu mutlaq yo'l dastur tomonidan ishlatiladi. Agar siz faqat fayl nomini (hech qanday yo'lsiz) kirtsangiz, nisbiy yo'l qabul qilinadi.<sup>1</sup> Ikkinchisi parametr quyidagi haqiqiy parametrlarga ega: o'qish uchun "r", yozish uchun "w" va ma'lumotlarni qo'shish uchun "a" faylning oxiri. Standart qiymat "r". Agar siz faylni o'qish va yozish uchun ochmoqchi bo'lsangiz, "r+" dan foydalaning.

Faylni o'qish uchun ochiq faylni yarating:

```
>>> file_handle = open('readme.txt', 'r')
```

Bu erda ko'rib turganingizdek, file\_handle fayl emas, balki unga havola::

```
>>> file_handle
<_io.TextIOWrapper name='readme.txt' mode='r' encoding='UTF-8'>
```

**2. Faylni o'qing:** Fayl ochilgandan so'ng biz uning mazmunini o'qiy olamiz. Fayl tutqichi faylni o'qishning bir necha usullariga ega; bu erda eng ko'p ishlatiladiganlar:

**read(n) :** Fayldan n bayt o'qiydi. Parametrlarsiz u butunni o'qiydi fayl.2

**readline() :** Fayldan faqat bitta satrli satrni qaytaradi, shu jumladan qator oxiri belgisi sifatida "\n". Faylning oxiriga yetganda, u bo'sh qatorni qaytaradi.

**readlines() :** Har bir element qatordan iborat bo'ylgan ro'yixatni qaytaradi fayl.

**read() bilan seqA.fas deb nomlangan faylni o'qish :**

```
>>> file_handle = open('seqA.fas', 'r') >>>
file_handle.read()
>O00626|INSON Kichkina induksiyalanuvchi sitokin A22.
```

<sup>1</sup>Agar joriy yo'lni bilishingiz kerak bo'lsa, os.getcwd() dan foydalaning.

<sup>2</sup>Un olishi mumkin bo'lgan xotira miqdori tufayli, agar fayl hajmiga ishonchingiz komil bo'lmasa, butun faylni shu tarzda o'qish tavsiya etilmaydi. Katta fayllarni qayta ishlash uchun bir vaqtning o'zida bir qatorni o'qish kabi yaxshiroq strategiyalar mavjud.

```
MARLQTALLVVLVLLAVALQATEAGPYGANMEDSVCCRDYVRYRLPLRVVKHFYWTSDS<=
CPRPGVVLLTFRDKEICADPR
VPWVKMILNKLSQ
```

**3. Faylni yoping.** Fayl bilan ishlashni tugatgandan so'ng, biz uni quyidagi yordamida yopamiz: `filehandle.close()`. Agar biz uni yopmasak, dastur bajarilgandan keyin Python uni yopadi. Ammo ko'p hollarda faylni kerak bo'limganda darhol yopish yaxshiroqdir, chunki ochiq fayllar soni cheklangan manbadir. Fayl yopilishini ta'minlashning bir usuli - bilan foydalanish. O'rniqa:

```
file_handle = open('readme.txt', 'r') #
file_handle.read() file_handle.close() fayli bilan
biror narsa qilish
```

Buni bajaring:

```
file_handle sifatida open('readme.txt', 'r') bilan: #
file_handle.read() # fayli bilan biror narsa qiling,
bundan buyon fayl yopiladi.
```

### 5.1.1 Fayllarni qayta ishlashga misol

Faraz qilaylik, bizda `seqA.fas` nomli fayl bor, unda quyidagilar mavjud:3

```
>O00626|INSON Kichik induksiyalanuvchi sitokin A22.
MARLQTALLVVLVLLAVALQATEAGPYGANMEDSVCCRDYVRYRLPLRVVKHFYWTSDS<=
CPRPGVVLLTFRDKEICADPR
VPWVKMILNKLSQ
```

Ushbu fayldan bizga nom va ketma-ketlik kerak. Birinchi yondashuv o'qishdir faylni `read()`:

```
>>> file_handle sifatida open('seqA.fas', 'r') bilan:
```

```
...     file_handle.read()
...
```

```
'>O00626|INSON kichik induksiyali sitokinA22.\nMARLQTALLVVLV<=
LAVALQATEAGPYGANMEDSVCCRDYVRYRLPLRVVKHFYWTSDSCPRPGVVLLTFRDK<=
EICADPR\nVPWQVKMILNKLS
```

---

3Bu bitta yozuvli FASTA fayli. Birinchi qatorda > keyin ketma-ketlik nomi va tavsifi mavjud. Quyidagi qatorlar ketma-ketlikka ega (DNK yoki aminokislotalar). FASTA fayllari haqida qo'yshimcha ma'yumot olish uchun <http://www.ncbi.nlm.nih.gov/BLAST/fasta.shtml> ga qarang .

## 88 Bioinformatika uchun Python

Bu holda mening maqsadim ikkita o'zgaruvchiga ega bo'l shdir, biri ketma-ketlik nomi bilan, ikkinchisi esa ketma-ketlikning o'zi. Listing 5.1 da biz buni read() yordamida amalga oshirish usulini ko'rishimiz mumkin:

**Listing 5.1: firstread.py: Avval FASTA faylini o'qishga harakat qiling**


---

```
1 fh sifatida open('seqA.fas') bilan:
    my_file = fh.read() 2
3 nom = my_file.split('\n')[0][1:] 4 sequence =
".join(mening_fayl.split('\n')[1:]) 5 print('Nomi:
{ 0}'.format(nom)) 6 ta chop etish('Tartib: {0}'.format(ketma-
ketlik))
```

Birinchi qator faylni o'qish rejimida ochadi va biz fh deb ataydigan fayl tutqichini yaratadi. Ikkinci qatorda butun fayl read() yordamida o'qiladi va natijada olingan satr tizim xotirasida my\_file nomi bilan saqlanadi. Keyingi qadam nomlarni ketma-ketliklardan ajratishdir. Ism ">" belgisidan keyin va "\n" dan oldin bo'lgani uchun bu ma'lumotlardan biz kerakli ma'lumotlarni olish uchun foydalanish mumkin (3-qator). Ketma-ketlik my\_file qatorini bo'l sh natijasida hosil bo'lgan elementlarni birlashtirish orqali olinadi, lekin birinchi elementsiz.

Ushbu kod bilan bog'liq muammo shundaki, u barcha faylini bir vaqtning o'zida o'qish uchun read() funksiyasidan foydalanadi. Agar fayl mazmunini joylashtirish uchun yetarli xotira boylmasa, bu mumkin boylgan muammo. Shuning uchun fayl ob'ektidagi satrlar bo'y lab aylanish uchun readline() dan foydalangan ma'qul.

**Listing 5.2: fastaRead.py: FASTA faylini ketma-ket o'qiydi**


---

```
1 ketma-ketlik      "
= 2 ochiq('seqA.fas') bilan fh: 3
    nom = fh.readline()[1:-1]
4    fh qatori uchun:
5        sequence += line.replace("\n",") 6 print('Nomi:
{0}'.format(nom)) 7 print('Tartib: {0}'.format(ketma-ket ) )
```

---

Kod tushuntirishi: Ushbu dastur bizning protein sof zaryadini hisoblash dasturimizga ([4.14 ro'yxati](#)) qo'lda kiritish o'rniga kirish ma'lumotlari, FASTA formati ketma-ketligi sifatida foydalanish imkoniyatini qo'shamdi. 3-qatorda biz ketma-ketlik nomini (O00626|INSON Kichik inducible sitokineA22.) olish uchun readline() bilan birinchi qatorni ushlaymiz va uni nomlaysiz. Fayl boshqaruvidagi x formulasi (4-qator) faylning barcha qatorlarini takrorlashning eng samarali usuli hisoblanadi.

**Listing 5.3: netchargefile.py: fayldan kiritilgan ma'lumotlarni o'qib, aniq to'lovni hisoblang**

```

1 ketma-ketlik "
= 2 zaryad = -0,002
3 aa_charge = {'C':-045, 'D':-999, 'E':-998, 'H':,091, 4
    'K':1, 'R':1, 'Y':-001} 5 bilan
open('prot.fas') fh sifatida: fh.readline() 6

7      fhdagi satr
8          uchun: ketma-ketlik +=
satr[:-1].upper() 9 aa uchun ketma-ket: 10
zaryad +ma'lumotlarnima'lumotlari 11 bosib

```

---

Kod tushuntirishi: Kod asosan **4.14 ro'yxatidagi** bilan bir xil bo'lib , oqsil ma'lumotlarini o'qishdagi farq bilan; kiritishdan foydalanish o'rniغا, ma'lumotlar 5.2 Ro'yxatdagi kabi fayldan o'qiladi (5-8 qatorlar) . E'tibor bering, 6-qatorda biz birinchi qatorni o'qiymiz va qaytarilgan qiymat saqlanmaydi, chunki u sarlavha va aniq to'lovni hisoblash uchun kerak emas. Dastur faylni 7-qatordan iteratsiya qilishni boshlaganda, u ikkinchi qatordan boshlanadi.

## 5.2 FAYLLARNI YOZISH

---

Fayl yozish uni o'qishga juda o'xshaydi. 1 va 3-bosqichlar o'xshash. O'zgarish ikkinchi holatda. Keling, butun jarayonni ko'rib chiqaylik:

1. Faylni oching. Bu o'qish uchun faylni ochishga o'xshaydi, faqat biz bajarmoqchi bo'lgan operatsiyaga mos keladigan ochiq rejimdan foydalanishni hisobga olish kerak. Yangi fayl yaratish uchun ochiq rejim sifatida "w" dan foydalaning. Fayl oxiriga ma'lumotlarni qo'shish uchun "a" dan foydalana olasiz.

Yangi fayl uchun fayl dastagini yaratish:

```
>>> fh = ochiq('newfile.txt','w')
```

Faylga ma'lumot qo'shish uchun yangi fayl tutqichini yaratish:

```
>>> fh = ochiq('error.log','a')
```

2. Faylga ma'lumotlarni yozish. Faylga ma'lumotlarni yozish usuli write() deb ataladi. U parametr sifatida funktsiya qo'llaniladigan fayl tutqichi bilan ko'rsatilgan faylga yoziladigan satrni qabul qiladi. Sxematik: fayl handle.write(string). Yozish satrlarni qo'shmasligini hisobga oling, kerak bo'lganda qo'shilishi kerak.

3. Faylni avvalgidek yoping: filehandle.close(). Fayllarni o'qishda bo'lgani kabi, siz faylni yozish uchun ochish va uni yashirin, ammo xavfsiz tarzda yopish uchun foydalanishingiz mumkin. **5.4 Ro'yxatga** qarang .

### 5.2.1 Fayllarni o'qish va yozishga misollar

Quyidagi kod 1 dan 5 gacha bo'lgan raqamlarni har biri alohida qatorda faylga saqlaydi. Har bir raqam o'rtaida tegishli qatorlar ko'rsatilgan.

**Listing 5.4:** Newfile.py: Faylga raqamlarni yozish.

```
1 ochiq ('numberlar.txt', 'w') fh sifatida:  
2     fh.write('1\n2\n3\n4\n5')
```

**Listing 5.3** dagi dastur natijani ekranda ko'rsatish o'rniga faylga yozish uchun o'zgartirilishi mumkin:

**Listing 5.5:** nettofile.py Net to'lovni hisoblash, natijalarni faylda saqlash

```
1 ketma-ketlik      "  
= 2 zaryad = -0,002 3  
aa_charge = {'C':-045, 'D':-0999, 'E':-0998, 'H':091, 4  
              'K':1, 'R':1, 'Y':-001} 5  
ochiq('prot.fas') bilan fh: keyingi(fh)  
6  
7     fh qatori uchun:  
8         sequence += line[:-1].upper() 9 aa  
uchun ketma-ketlikda: zaryad += aa_charge.get(aa,  
          0) 10  
12 open('out.txt','w') bilan file_out:  
        file_out.write(str(charge)) 13
```

Kod tushuntirishi: Kod **5.3 ro'yxatiga** o'xshaydi , qo'shilishi bilan natijani faylga yozish uchun ikkita oxirgi satrda (12 va 13) funksionallik.

### 5.3 CSV FAYLLARI

Ma'lumotlarni qayta ishlash jarayonida CSV deb nomlangan fayl turiga kirish juda keng tarqalgan. CSV "vergul bilan ajratilgan qiymatlar" degan ma'noni anglatadi. Bu ma'lumotlar vergul bilan ajratilgan fayllar, garchi ba'zida boshqa ajratgichlar (masalan, ikki nuqta, yorliqlar va boshqalar) ishlatiladi. Ayniqsa, ushbu matn fayl formatining yana bir xususiyati shundaki, har bir satr alohida yozuvni ifodalaydi. Barcha elektron jadvalarni ushbu fayl formatida o'qish va yozish mumkin, bu ularning mashhurligini tushuntirishga yordam beradi. Masalan, quyidagi faylni oling (B1.csv):

**MarkerID, LenAmp, MotifAmpForSeq**

**TKO001,119,AG(12)**

**TKO002,255,TC(16)**

TKO003,121,AG(5)  
 TKO004,220,AG(9)  
 TKO005,238,TC(17)

Satrda har bir maydonning tavsifi mavjud. U saqlaydigan ma'lumotlar singari, tavsiflar ham vergul bilan ajratiladi. Quyidagi satrlar tavsifning bir xil tartibiga riyox qilgan holda ma'lumotlarni o'z ichiga oladi. Ikkinchisi ustundagi qiymatning o'rtacha qiymatini olish uchun biz shunday qilishimiz mumkin:

#### **Listing 5.6: csvwocsv.py: CSV faylidan ma'lumotlarni o'qish**

---

```
1 total_len = 0 2
bilan open('B1.csv') fh sifatida:
qator (fh) keydingi(f1) nechchi(l, l), qator
4      total_len += int(data[1 ]) 7 ta chop
5          etish (jami_len/n)
6
```

---

Kod tushuntirishi: Bu [Listing 5.5](#) kabi fayl bo'ylab yuradigan dastur , ammo bu safar split() usuli har bir qator komponentlarini ajratish uchun ishlataladi. 6-qatorda ikkinchi maydonning yig'indisi (LenAmp) saqlanadi (bu maydon ketma-ketlikning uzunligiga ega).

Ushbu fayllar shu qadar mashhurki, Pythonda ular bilan ishlash uchun modul mavjud: csv.

#### **Listing 5.7: csv1.py: csv moduli yordamida CSV faylidan ma'lumotlarni o'qish**

---

```
1 import csv 2
total_len=0 3
satr = csv.reader(open('B1.csv')) n uchun 4
keyingi(satr) 5, raqamdagisi satr(satr): total_len
+= int(satr[1]) 6 7 ta chop etish (jami_tasvir / n)
```

---

Kod tushuntirishi: Bu dastur avvalgisiga juda o'xshaydi, farqi shundaki, csv modulidan foydalanish bizga bo'linish usulidan foydalanmasdan har bir satr tarkibiga kirish imkonini beradi.

Csv modulidan foydalanishning bir usuli o'quvchi usuli bilan qaytarilgan ob'ektni ro'yxatga aylantirishdir. Buni amalga oshirib, biz bir qator kodli csv faylidan matritsaga o'xshash narsani yaratamiz:

```
>>> ma'lumotlar = ro'yixat(csv.reader(open('B1.csv')))
>>> ma'lumotlar[0][2]
```

## 92 Bioinformatika uchun Python

```
'MotifAmpForSeq'
>>> ma'lumotlar[1][1]
'119'
>>> ma'lumotlar[1][2]
'AG(12)'
>>> ma'lumotlar[3][0]
'TKO003'
```

Shunday qilib, bizda nomi[satr, ustun] turdag'i ikki o'lchovli massiv mavjud . Tak Buni hisobga olib, biz dasturni **5.7 ro'yxatidan qayta yozishimiz mumkin**:

CSV modulidan qo'shimcha funksiyalar

Maydon chegaralagichi ajratuvchi atributi bilan o'zgartiriladi. Odatiy bo'lib, u "," dir, lekin maydonlarni chegaralash uchun istalgan satrdan foydalanish mumkin:

```
satrlar = csv.reader(open('/etc/passwd'), ajratuvchi =':')
```

Ba'zi fayllar uchun bizni qiziqtirgan CSV "dialektini" ko'rsatgan ma'qul. Bu muhim, chunki barcha csv fayllari bir xil tuzilishga ega emas. CSV rasmiy standart emas, shuning uchun har bir sotuvchi ba'zi o'zgarishlar kiritishi mumkin. Bizning ma'lumotlarni qayta ishlashimizni buzishi mumkin bo'lgan bu nozik farqlar. Ba'zi hollarda ma'lumotlar kotirovkalar orasiga qo'yilgan, boshqalarida esa kotirovkalar faqat matnli ma'lumotlar uchun ajratilgan. Excel tomonidan yaratilgan csv fayllar uchun bizda Excel "dialekti" mavjud:

```
qatorlar = csv.reader(open('data.csv'), dialekt='excel')
```

Bundan tashqari, Excel csv fayllari uchun dialekt mavjud bo'lib, u ma'lumotlarni ajratish uchun vergul o'rninga "yorliq" dan foydalanadi. Agar biz kodimiz qanday dialektga mos kelishiga ishonchimiz komil bo'lmasa, csv modulida uni taxmin qilishga harakat qiladigan sinf mavjud: Sniffer():

```
dialekt = csv.Sniffer().sniff(open('data.csv').read()) qatorlar =
csv.reader(open('data.csv'), dialekt=dialekt)
```

Csv modulida ko'proq usullar mavjud. Bu haqda ko'proq ma'lumot olish uchun men <https://docs.python.org/3/library/csv.html> manzilidagi modul hujjalarni tavsiya qilaman. va PEP-305 (<https://www.python.org/dev/peps/pep-0305/>), eski, lekin hali ham haqiqiy hujjat.

CSV fayllari juda qulay, lekin ular ierarxik ma'lumotlarni taqdim eta olmaydi, shuning uchun JSON va XML kabi boshqa formatlar ma'lumotlarni saqlash uchun ishlataladi. Ikkalasi ham ushbu kitobda yoritilgan.

	A	B	C	D	E
1	Epitope ID	Object Type	Description	Starting Position	Ending Position
2	6273	Linear peptide	CGAELNHFL	379	387
3	14101	Linear peptide	ERYLKDOQL	584	592
4	22030	Linear peptide	GREGKLVLY		
5	25569	Linear peptide	IDFPKTFGW		
6	26070	Linear peptide	IIFPKTFGW		
7	26790	Linear peptide	IKFPKTFGW		
8	27049	Linear peptide	ILFPKTFGW		
9	27636	Linear peptide	INFPKTFGW		
10	28419	Linear peptide	IRYPKTFGW	162	170
11	33140	Linear peptide	KRGILTLY	63	71
12	33170	Linear peptide	KRKKAAYADF		

Shakl 5.1 Excel formatlangan elektron jadval sampledata.xlsx.

Maslahat: Excel fayllarini o'qish va yozish.

Csv moduli, agar fayl birinchi navbatda csv ga aylantirilgan bo'lsa, Excel fayllarini o'qish imkonini beradi. Ushbu qadamni xlrd deb nomlangan uchinchi tomon moduli yordamida oldini olish mumkin. Ushbu modulni pip yoki conda bilan o'rnatish mumkin (tashqi paketlarni o'rnatish haqida batafsil ma'lumot uchun 117-betga qarang).

[Listing 5.8](#), sampledata.xlsx deb nomlangan Excel faylidan ma'lumotlarni oladi (5.3-rasmga qarang). Biz A ustunidan (kalitlar) va C ustunidan (qiymatlar) lug'at (iedb) yaratmoqchimiz, shuning uchun bu dastur ikkala ustunni bosib o'tadi va lug'atni to'ldiradi:

**Listing 5.8: excel1.py: xlsx faylini xlrd bilan o'qish**

```

1 import xlrd 2
iedb = {} 3 kitob
= xlrd.open_workbook('..../samples/sampledata.xlsx') 4 sh =
book.sheet_by_index(0) 5 diapazondagi satr_indeks (1, sh.nrows) : #musht
qatorini oytkazib yuboradi. 6
    iedb[int(sh.cell_value(rowx=row_index, colx=0))] = \
7        sh.cell_value(satr=satr_indeks, colx=2)
8 ta chop etish (iedb)

```

excel1.py quyidagi lug'atni qaytaradi:

## 94 Bioinformatika uchun Python

```
{6273: 'CGAELNHFL', 14101: 'ERYLKDQQL', 22030: 'GRFKLIVLY', <=
25569: 'IDFPKTFGW', 26070: 'IFFPKTFGW', 26790: 'IKFPKTFGW', <=
27049: 'ILFPKTFGW', 27636: 'INFPKTFGW', 28419: 'IRYPKTFGW', <=
33140: 'KRGILTLY', 33170: 'KRKKAYADF'}
```

E'tibor bering, bu namunaviy chiqish, haqiqiy chiqish kattaroq, ammo qisqalik uchun qisqartirilgan. Ushbu chiqishni 5.3-rasm bilan solishtiring.

Excel fayllarini yozish uchun siz xlrd ga o'xshash tarzda ishlaydigan xlwt dan foydalanishingiz mumkin.

**Listing 5.9** ro'yxat1 va ro'yxat2ni xlwt yordamida A va B ustunlariga yozadi.

**Listing 5.9: excel2.py: XLS faylini xlwt bilan yozing**

---

```
1 import xlwt
2
3 list1 = [1,2,3,4,5]
4 list2 =
5 [234,267,281,301,331]
6 wb =
7 xlwt.Workbook()
8 ws =
9
10 wb.add_sheet('Birinchi varaq')
11 ws.write(0, 0, 'ustun A')
12 ws.write(0, 1, 'ustun B')
13
14 i = 1
15 for x, y in zip(list1, list2):
16     ws.write(i, 0, x)
17     ws.write(i, 1, y)
18     i += 1
19
20 wb.save('mynewfile.xls')
```

---

PyExcelerator-dan foydalanish namunasi uchun 336-betdagি [18.2 ro'yxatiga](#) qarang.

---

## 5.4 PICKLE: VARI MAZMUNINI SAQLASH VA QAYTA QILISH QOLIB

Dasturning amal qilish muddati davomida yaratilgan barcha o'zgaruvchilar vaqtincha xotirada saqlanadi va dastur tugashi bilan yo'qoladi. Python ushbu o'zgaruvchilar tarkibini diskdan yoki boshqa har qanday vositadan saqlash va olish uchun modulni taqdim etadi: Pickle<sup>4</sup> moduli. pickle har qanday Python ma'lumotlar strukturasini bayt oqimiga seriyalizatsiya qiladi. Ushbu bayt oqimi diskda saqlanishi yoki tarmoq orqali yuborilishi mumkin. Boshqa tomonidan, tuzlama bayt oqimini asl ichki tuzilishga ega ob'ektga aylantirishi mumkin. Quyidagi skript lug'at (sp\_dict) hosil qiladi va uni boshqa dasturda mavjud bo'lishi uchun faylga (spdict.data deb nomlangan) saqlaydi.

---

4Pickle ushbu kitobda tasvirlanganlardan boshqa xususiyatlarga ega; tuzlangan bodring nimani taklif qilishi haqida kengroq ma'lumotga ega bo'lish uchun <https://docs.python.org/3/library/pickle.html#module-pickle> ga qarang.

**Listing 5.10: picklesample.py: Asosiy tuzlangan bodring namunasi**

```

1 import tuzlangan
2 sp_dict = {'bir':'uno', 'ikki':'dos', 'uch':'tres'} 3 ochiq('spdict.data', 'wb')
bilan fh: tuzlangan. dump (sp_dict, fh) 4

```

pickle.dump() yordamida sp\_dict lug'ati fayl tutqichi (fh) tomonidan havola qilingan faylga saqlanadi. pickle.dump() to'rtta parametrni qabul qiladi. Birinchi parametr siz saqlamoqchi bo'lgan ob'ektdir. Ikkinchisi, ob'ektingizni saqlamoqchi bo'lgan faylga o'xshash ob'ekt. Uchinchi argument (misolda ishlatilmagan) protokol bo'lib, ma'lumotni kodlash usulini ifodalovchchi butun sondir. Hech qanday protokol ko'rsatilmagan bo'lsa (bu holatda bo'lgani kabi), u sukut bo'yicha 3 ga o'rnatiladi, bu Python 3 uchun mo'ljallangan ikkilik orqaga mos kelmaydigan protokol. Pike protokollari haqida ko'proq ma'lumot olish uchun Pickle uchun **Infobox Protocols-ga** qarang. To'rtinchi parametr (fix\_imports) "True" ga o'rnatilganda va protokol 3 dan kam bo'lsa, Python 2 uchun orqaga qarab muvofiqligini olish uchun foydalaniladi.

spdict.data da saqlangan ma'lumotlarni pickle.load() yordamida olish mumkin:

```

>>> import bodring
>>> pickle.load(open('spdict.data','rb')) {'bir':'uno',
'ikki':'dos', 'uch':'tres'}

```

Yuklash usuli biz olmoqchi bo'lgan ob'ektning fayl dastagini talab qiladi.

E'tibor bering, ikkala holatda ham (dump va yuk) men faylni ochish uchun b (ikkilik uchun) dan foydalanaman, chunki men foydalanadigan standart protokol ikkilik faylni yozadi. Agar siz faylni ikkilik sifatida ochmasangiz, Python uni Unicode ga aylantirishga harakat qiladi va muvaffaqiyatsiz.

#### Pickle uchun protokollar

Bu erda tuzlash uchun ishlatalishi mumkin bo'lgan besh xil protokollar ro'yxati keltirilgan. Foydalanilgan protokol qanchalik baland bo'lsa, Python-ning yangi versiyasi ishlab chiqarilgan tuzlangan bodringni o'qish uchun kerak edi.

1. 0 asl "odam tomonidan o'yqiladigan" protokol bo'ylib, u bilan orqaga qarab mos keladi Pythonning oldingi versiyalari.
2. 1 eski ikkilik format bo'lib, uning oldingi versiyalari bilan ham mos keladi Python.
3. 2 yangi uslubdagi sinflarni ancha samarali tuzlashni ta'minlaydi.
4. 3 Python 3 da standart protokol hisoblanadi. U bayt ob'yeqtalarini aniq qo'llab-quvvatlaydi va Python 2.x tomonidan uni ajratib bo'lmaydi. Bu boshqa Python 3 versiyalari bilan moslik zarur bo'lganda tavsiya etilgan protokol.

## 96 Bioinformatika uchun Python

5. Python 3.4 dan 4 ta. U juda katta ob'ektlarni qo'llab-quvvatlaydi, ko'proq turdag'i ob'ektlarni tanlaydi va ba'zi ma'lumotlar formatini optimallashtiradi .

---

Ogohlantirish Hech qachon ishonchsiz manbadan olingan ma'lumotlarni olib tashlamang, chunki pickle moduli noto'g'ri yoki zararli tarzda tuzilgan ma'lumotlardan xavfsiz emas.

## 5.5 JSON FAYLLARI

---

**JSON (JavaScript Object Notation)** - bu JavaScript-dan ilhomlangan, inson o'qiy oladigan ma'lumotlar almashinuvni formati bo'lib, u veb-ilovalarda keng qo'llaniladi. Siz lug'at, ro'yxat yoki deyarli barcha turdag'i ma'lumotlarni JSON ob'ektiga aylantirishingiz va keyin ushbu ob'ektni tarmoq orqali saqlashingiz yoki uzatishingiz mumkin. Barcha ob'ektlarni JSONga aylantirib bo'lmaydi, chunki ba'zi ob'ektlar Python-ga xos va boshqa kompyuter tillarida mavjud emas. Masalan, siz to'plamni JSONga aylantira olmaysiz. Nima uchun sizda tuzlangan bo'lsa, JSON kabi kamroq qobiliyatli serializatorдан foydalanasz? Chunki siz ma'lumotlarni almashishingiz kerak va bu ma'lumotlar har qanday kompyuter tili uchun mavjud bo'lishini xohlaysiz. Agar siz tuzlangan ob'ektni baham ko'rsangiz, qabul qiluvchini Python-dan foydalanishga majbur qilasiz. Xo'sh, nima uchun juda mashhur bo'lgan CVS dan foydalanmaslik kerak? CVS ustunli ma'lumotlar uchun yaxshi, lekin ichki va lug'atga o'xshash ma'lumotlar uchun emas. Murakkab munosabatlarga ega bo'lgan va har qanday dasturlash tilida

Bu JSON fayliga misol:

```
{
    "contactPoint": {
        "fn": "PREUSCH, PITER\u00a0",
        "hasEmail": "mailto:preuschp@nigms.nih.gov"
    },
    "description": "<p>Protein ma'lumotlar banki (PDB) arxivi yirik biologik molekulalarning, jumladan, barcha organizmlarda topilgan oqsillar va nuklein kislotalarning 3D tuzilmalari haqidagi ma'lumotlarning butun dunyo bo'ylab yagona omboridir</p>\n",
    "identifier": "d9f3932a-9c55-41b3-ad3a-0b4e18ee4752",
    "kalit so'z": [ "milliy-sog'liqni saqlash-institutlari-nih"
],
    "til": [ "en"
],
    "license": "http://opendefinition.org/licenses/odc-odbl/",
    "modified": "18-07-2016",
    "dastur kodi": [ "009:000"
]
```

```

    ],
    "publisher":
        { "@type":"org:Organization",
          "name":"Milliy Sog'liqni saqlash institutlari (NIH)"
        },
    "title":"Protein ma'lumotlar banki (PDB)"
}

```

JSON tuzlangan bilan bir xil interfeysga ega, ya'ni siz shunga o'xshash tarzda o'qishingiz va yozishingiz mumkin.

```

>>> import json >>>
sp_dict = {'bir':'uno', 'ikki':'dos', 'uch':'tres'} >>> open('spdict.json', 'w'
bilan ) fh sifatida: json.dump(sp_dict, fh)
...

```

Saqlangan fayl quyidagicha ko'rindi:

```
{'uch': 'tres', 'bir': 'uno', 'ikki': 'dos'}
```

JSON faylidan lug'atni olish uchun:

```

>>> json import >>>
open('spdict.json') bilan fh sifatida: sp_dict =
...         json.load(fh)

```

Oldingi kodni ishga tushirgandan so'ng, sp\_dict **lug'ati asl tarkibga ega**:

```

>>> sp_d
{'uch': 'tres', 'bir': 'uno', 'ikki': 'dos'}

```

E'tibor bering, JSON to'plamlarni va boshqa maxsus Python obyektlarini seriyalashtira olmaydi:

```

>>> json.dump({1,2,3,4,3,2,7}, open('test.json','wb'))
Traceback (eng oxirgi qo'ng'iroq):
Fayl "<stdin>", 1-qator, <modul> faylida
"python3.5/json/__init__.py", 178-qator, dump
takrorlanadigan bo'lak uchun:
"python3.5/json/encoder.py" fayli, 436-qator, _iterencode o = _default(o)

"python3.5/json/encoder.py" fayli, 180-qator, sukut bo'yicha
TypeError(repr(o) +) ko'tariladi      JSON seriyali emas")
TypeError: {1, 2, 3, 4, 7} JSON seriyali emas

```

Bu yerda serializatsiya qilinadigan ob'ektlar ro'yxati: int, float, str, list, dict, True, False va None.

## 98 Bioinformatika uchun Python

---

5.6 FAYLLARNI ISHLATISH: OS, OS.PATH, SHUTIL VA PATH.PY MODULI

---

O'qish va yozishdan tashqari, fayllar bilan ko'proq harakatlar mavjud. Nusxa ko'chirish, ko'chirish, o'chirish, ro'yxatga olish, katalogni o'zgartirish, fayl xususiyatlarini o'rnatish va boshqalarni os, shutil va path.py modullari yordamida bajarish mumkin.

**OS** moduli Operatsion tizim bilan interfeysi boshqaradi. Keling, ushbu modul tomonidan taqdim etilgan ba'zi muhim usullarni ko'rib chiqaylik: `getcwd()`: Joriy ishchi katalogni ifodalovchi qatorni qaytaring.

```
>>> import os
>>> os.getcwd() '/'
home/sb'
```

**chdir(yo'l)**: Joriy ishchi katalogni berilgan yo'lga o'zgartiring.

```
>>> os.getcwd() '/'
home/sb' >>>
os.chdir('docs') >>>
os.getcwd() '/home/sb/
docs' >>> os.chdir('..')
>>> os.getcwd() '/home/
sb'
```

**listdir(dir)**: Katalogdagi yozuvlar nomlarini o'z ichiga olgan ro'yxatni qaytaring. `listdir` dan qaytarilgan nom fayl yoki katalog ekanligini bilish uchun `os.path.isdir()` yoki `os.path.isfile()` dan foydalaning.

```
>>> os.listdir('/home/sb/bioinfo/seqs') ['readme.txt',
'ms115.ab1','.atom', 'loyihalar', '.bash_history']
```

**path.isfile(string)** va **path.isdir(string)**: Argument sifatida uzatilgan satr fayl yoki katalog ekanligini tekshiring. Rost yoki Noto'g'ri qaytaradi.

```
>>> os.path.isfile('/home/sb')
Yolg'on
>>> os.path.isdir('/home/sb')
To'g'ri
```

**olib tashlash (fayl)**: faylni o'chirish. Fayl mavjud bo'lishi kerak va siz unga yozish ruxsatiga ega bo'lishingiz kerak.

```
>>> os.remove('/home/sb/bioinfo/seqs/ms115.ab1')
```

**rename(manba, destination)**: Fayl yoki katalog manbasini manzil nomiga o'zgartiring .

```
>>> os.rename('/home/sb/seqs/readme.txt','/home/sb/Readme')
```

**mkdir(yo'l):** yo'l nomli katalog yarating .

```
>>> os.mkdir('/home/sb/processed-seqs')
```

OS moduli ichida yo'l moduli joylashgan. U yo'lni manipulyatsiya qilish bilan bog'liq usullarni o'z ichiga oladi. path.join(katalog1,katalog2,...): Ikki yoki undan ortiq yo'l nomi komponentlarini birlashtirib, kerak bo'ylganda operatsion tizim yo'l ajratuvchisini kriting. Windows-da u "\" qo'shami, Linux va macOS-da esa "/" qo'shami. path.join yaratilgan yo'l to'g'ri yoki yo'qligini tekshirmaydi.

```
>>> os.path.join(os.getcwd(), 'rasmlar') '/home/images'
```

**path.exists(yo'l):** Berilgan yo'l mavjudligini tekshiradi.

```
>>> os.path.mavjud(os.path.join(os.getcwd(), "rasmlar"))  
Yolg'on
```

**path.split(yo'l):** fayl yoki katalog nomini bo'linadigan kortejni qaytaradi oxiri va yo'lning qolgan qismi.

```
>>> os.path.split('/home/sb/seqs/ms2333.ab1') ('/home/sb/  
seqs', 'ms2333.ab1')
```

**path.splitext(yo'l):** Fayl kengaytmasini ajratadi. U nuqtali kengaytmali kortejni va nuqtagacha bo'lgan asl parametrni qaytaradi.

```
>>> os.path.splitext('/home/sb/seqs/ms2333.ab1') ('/home/sb/  
seqs/ms2333', '.ab1')
```

Nusxa ko'chirish va o'chirish kabi fayl bilan bog'liq boshqa operatsiyalarni shutil modulida topish mumkin:

**Eng muhim funksiyalar** nusxa ko'chirish, nusxa ko'chirish2 **va** nusxa ko'chirishdir. nusxa ko'chirish (manba, maqsad): **Fayl** manbasini belgilangan joyga nusxalash . copy2 (manba, maqsad): **Oxirgi kirish vaqtini va oxirgi modifikatsiyani ham nusxalaydi** (masalan, Unix buyrug'i cp -p). copytree(manba, maqsad): **dan butun katalog daraxtini rekursiv ravishda nusxalash** manba katalogini allaqachon mavjud bo'limgan maqsad katalogiga.

Shutil haqida qo'shimcha ma'lumot olish uchun <http://docs> -dagi hujjatlarga qarang . [python.org/lib/module-shutil.html](http://python.org/lib/module-shutil.html) (yoki Python qobig'idagi yordam (shutil) bilan).

### 5.6.1 path.py moduli

os.path uchun o'rash vazifasini bajaradigan path.py deb nomlangan tashqi modul mavjud. Ushbu modul sizga boshqa barcha modular kabi bir xil vazifalarni bajarishga imkon beradi, ammo ulardan foydalanish oson dasturlash interfeysi bilan. Tashqi modul bo'lgani uchun u oddiy Python o'rnatilishi bilan mavjud emas (agar sizda anaconda kabi Python distrosi mavjud bo'lmasa, u va boshqa tashqi modular bilan birga keladi). Agar path.py ni o'rnatishingiz kerak bo'lsa, pip dan foydalaning:

```
# pip install path.py path.py
to'plami keshlangan
path.py-9.0-py2.py3-none-any.whl-dan foydalanish
Yig'ilgan paketlar o'rnatalmoqda: path.py
Muvaffaqiyatli o'rnatildi path.py >>> import yo'li
```

```
>>>
```

**Import xatosi bo'lmasa, path.py muvaffaqiyatlari o'rnatildi. Uchinchi tomon modularini o'rnatish haqida qo'shimcha ma'lumot olish uchun 117-betga qarang.**

Bu erda path.py bilan bajarilishi mumkin bo'lgan ba'zi narsalar mavjud. Ushbu misollar quyidagi katalog tuzilishini nazarda tutadi:

```
/home
-- /sb
    -- xx.py --
    py4bio --
        ch1.pdf --
        ch1.tex
        -- ch2.pdf --
        ch2.tex
    -- /imgs --
        fig1.png --
        fig2.png
```

- Yangi fayl yaratish: Path obyektini yaratish va teginish usulini chaqiring. Path obyektini yaratishda foydalanilgan satr hech qanday faylga mos kelmasa, bu yangi fayl yaratadi.

```
>>> yo'ldan import Yo'li >>> f =
Path('/home/sb/newfile.text') >>> f.touch()
```

- Yo'li obyekti fayl yoki katalog (isfile()) ekanligini tekshiring:

```
>>> f.isfile()
To'g'ri
```

- **Ism, kengaytma va ota-katalogni oling (qo'shimcha, ism va ota-onas):**

```
>>> f = Path('/home/sb/xx.py') >>> f.ext
'.py'
>>> f.name
Path('xx.py') >>>
f.parent Path('/home/sb') >>>
f.parent.parent Path('/home')
```

- **Katalogdagi barcha fayl va kataloglarni oling (files() va dirs()):**

```
>>> d = Path('/home/sb/py4bio') >>> d.files()
[Path('/home/sb/py4bio/ch1.tex'), Path('/home/sb/py4bio/ch2.pdf'), Path('/home/sb/py4bio/ch2.tex'), Yo'l('/home/sb/py4bio/ch1.pdf')] >>> d.dirs()
[Yo'l('/home/sb/py4bio/imgs')]
```

Siz shuningdek filtrlarni qo'llashingiz mumkin:

```
>>> d.files('*pdf')
[Path('/home/sb/py4bio/ch2.pdf'), Path('/home/sb/py4bio/ch1.pdf')]
```

- **walk() yordamida katalogdagi barcha fayllar (jumladan, ota-katalogdagi katalog ichidagi fayllar) bo'ylab yuring:**

```
d = d.walk(): if f.isfile(): print(f) da f
uchun yo'l('/home/sb/py4bio')
```

qaytib keladi:

```
/home/sb/py4bio/ch1.tex /home/
sb/py4bio/imgs/fig1.png /home/sb/py4bio/
imgs/fig3.png /home/sb/py4bio/imgs/
fig4.png /home/sb/py4bio/imgs/fig2.png /
/home/sb/py4bio/ch2.pdf /home/sb/py4bio/
ch2.tex /home/sb/py4bio/ch1.pdf
```

### 5.6.2 Bir nechta DNK ketma-ketliklarini bitta FASTA faylida birlashtirish

Quyidagi dastur bizda FASTA formatida bir nechta DNK ketma-ketliklari bo'lgan katalog (bioinfo/seqs/) borligini va biz ularni outfile.fasta deb nomlangan bitta FASTA faylida birlashtirmoqchimiz deb taxmin qiladi. Bu fayl, masalan, BLAST ishga tushirish uchun kirish fayli sifatida ishlatalishi mumkin.

**Listing 5.11: consolidate.py: Bir nechta fayllarni birlashtirish**

---

```
1 yo'l importidan 2-yo'l d =
Path('bioinfo/seqs/') 3 bilan
open('outfile.fasta', 'w') f_out sifatida: d.walk('.fasta') da
ma'lumofayl=fайл=foimireab()rf_4tiblanife(mazmunida)ish(fayl_nomi):
5
6
7
```

---

Kod tushuntirishi: Dastur bioinfo/seqs/ katalogi bilan yo'lni belgilaydi. 3-qatorda barcha ketma-ketliklarning mazmunini saqlash uchun faylni (outfile.fasta) ochamiz. 4-qatordan boshlab biz \*.fasta naqshiga mos keladigan har bir fayl ustida yurishni boshlaymiz. Har bir fayl uchun biz uni ochamiz, tarkibni o'qing (6-qator) va uni outfile.fasta-ga yozing (7-qator). Hech qanday ochiq faylni yopishning hojati yo'q, chunki ular with bayonoti ichida.

## 5.7 QO'SHIMCHA RESURSLAR

---

- Fayl va katalogga kirish.  
<https://docs.python.org/3/library/filesys.html>
- Vaqtinchalik fayllar va kataloglarni yaratish. <https://docs.python.org/3/library/tempfile.html>
- CSV fayl API.  
<http://www.python.org/dev/peps/pep-0305/>

- Tad, CSV ma'lumotlarini ko'rish va tahlil qilish dasturi.  
<http://tadviewer.com>
- Pythonda Excel fayllari bilan ishlash.  
<http://www.python-excel.org>
- “bilan” iborasi.  
<http://www.python.org/dev/peps/pep-0343/>
- JSON  
formatlashtiruvchi. <https://jsonformatter.curiousconcept.com/>
- Py YAML.  
<http://pyyaml.org/>

## 5.8 O'Z-O'ZINI BAHOLASH

---

1. Agar ikkalasi ham yozishga imkon bersa, "w" va "a" rejimlari o'rtaсидagi farq nima fayllar?
2. Nima uchun endi foydalanimayotgan barcha fayllarni yopishimiz kerak?
3. Nima uchun fayllarni with yordamida ochamiz?
4. Ism so'raydigan dastur tuzing va keyin uni nomli faylga yozing  
MyName.txt.
5. csv modulisiz csv fayllarni tahlil qilish mumkinmi? Agar shunday bo'lsa, u qanday amalga oshiriladi?
6. Nima uchun faylni read() yordamida o'qish tavsiya etilmaydi?
7. Faylni satr bo'ylab yurishning eng samarali usuli qanday?
8. Pythonda Pickle nima?
9. JSON nima ekanligini va uning Picklega nisbatan qanday cheklovlari borligini tushuntiring.
10. Excel faylining ikkinchi ustunidagi barcha raqamlarni o'qiydigan va bu qiymatlarning o'rtacha qiymatini chop etadigan dastur tuzing.



**Taylor & Francis**  
Taylor & Francis Group  
<http://taylorandfrancis.com>

# Kodni modullashtirish

---

## MAZMUNI

<b>6.1 Kodni modullashtirishga kirish.</b>	105	6.2
<b>Funktsiyalar</b>	106	6.2.1
Python kodini modulli qilishning standart usuli.	106	Funksiya
doirasi.	108	6.2.2
Funktsiya parametrlari parametrlari.	110	
Argumentlarni joylashtirish.	110	
<b>Standart qiymatli argumentlar</b>	111	
Argumentlarning aniqlanmagan sonlari	111	
Kalit so'yz argumentlarining aniqlanmagan soni	112	6.2.3
Generatorlar.	113	
Generator yaratish	114	6.3
<b>Modullar va paketlar.</b>	114	
6.3.1 Modullardan foydalanish.		
<b>115</b> 6.3.2 Paketlar		
<b>116</b> 6.3.3 Uchinchi tomon modollarini o'yrnatish		
<b>117</b> Pip afzal qilingan usul	117	
Tizim paketlarini boshqarishdan foydalanish.		
<b>118</b> PYTHONPATH ga nusxalash		
<b>119</b> 6.3.4 Virtualenv: Izolyatsiya qilingan Python muhitlari.		
<b>119</b> 6.3.5 Konda: Anakonda virtual muhiti	121	Qo'lda
yaratish va o'rnatish.	124	
6.3.6 Modullarni yaratish.	124	
6.3.7 Sinov modullari	125	
Doctest, Sinov modollarini avtomatik tarzda	125	6.4
<b>Qo'shimcha manbalar</b>	127	6.5 O'z-
o'zini baholash.	128	

## 6.1 KODLARNI MODULARLASHTIRISHGA KIRISH

Biz hozirgacha ko'rgan narsalarimiz bilan Python dasturlash uchun qiziqarli resurslar portfeliga egamiz.<sup>1</sup> Biz fayllarni o'qishimiz, ma'lumotlarni qayta ishlashimiz va ularni saqlashimiz mu-

<sup>1</sup> Agar siz shu paytgacha o'rghanlariningizni amalda qo'llashga qiziqsangiz, men quyidagi mashqlarni tavsiya qilaman bu sahifa: <https://github.com/karan/Projects>.

natijalar. Hozirgacha yaratilgan dasturlar juda qisqa bo'lsa-da, ularni boshqarish qiyin bo'lgan hajmgacha o'sishi mumkinligini tasavvur qilish oson.

Dastlabki kod bloklarini chaqiradigan kichik dastur bilan yakunlanishimiz mumkin bo'lgan tarzda manba kodini modullashtirish uchun ishlatalishi mumkin bo'lgan bir nechta manbalar mavjud. Ushbu yondashuv kodni qayta ishlatalish va o'qish qulayligini ta'minlaydi. Ikkala xususiyat ham parvarishlashda yordam beradi, chunki bu kod necha marta ishlatalishidan qat'i nazar, faqat bitta kodni amalga oshirishda disk raskadrovska qilishingiz kerak. Qo'shimcha afzallik sifatida, bu ishslashni yaxshilashga yordam beradi, chunki modullashtirilgan koddagi har qanday optimallashtirish uni chaqiradigan barcha kodlarga foyda keltiradi.

Ba'zi mualliflar uchun kodni modullashtirish "Kompyuter fanidagi eng buyuk ixtiro"<sup>2</sup> dir. Bu "eng buyuk ixtiro"mi yoki yo'qmi, bilmayman, lekin, albatta, bu asosiy tushuncha bo'lib, siz biron bir jiddiy dasturlashni rejalashtirmoqchi bo'lsangiz, ularsiz yashay olmaysiz.

Python manba kodini modullashtirishning bir necha usullarini taqdim etadi: funktsiyalar, modular, paketlar va sinflar. Bu bob o'z bo'limiga ega bo'lgan sinflar bundan mustasno, ularning barchasini qamrab oladi.

## 6.2 FUNKSIYALAR

---

### 6.2.1 Python kodini modulli qilishning standart usuli

Funktsiyalar kodni modullashtirishning an'anaviy usulidir. Funktsiya qiymatlarni (argumentlar yoki parametrler deb ataladi) oladi, ular asosida ba'zi operatsiyalarni bajaradi va qiymatni qaytaradi. Biz allaqachon bir nechta Python o'rnatilgan funktsiyalarini ko'rganmiz.<sup>3</sup> Masalan, 9-sahifada birinchi bo'ylib eslatib oytigilan len() parametr sifatida takrorlanuvchini qabul qiladi va raqamni qayta

```
>>> len('Salom')
5
```

Keling, o'z funktsiyalarimizni qanday qilishni ko'rib chiqaylik. Funktsyaning umumiy sintaksisi:

```
def funktsiya_nomi(argument1, argument2, ...):
    """ Majburiy emas funktsiya tavsifi (Docstring)
    ...
    DATA qaytaring
```

**4.14 ro'yxitidagi** kod funktsiya sifatida qayta yozilishi mumkin:

**Listing 6.1: netchargefn: Proteinning aniq zaryadini hisoblash funktsiyasi**

---

<sup>2</sup> <http://www.stevemcconnell.com/ieesoftware/bp16> manzilidagi Stiv Makkonnell ustunini o'qing .htm.

<sup>3</sup> Python-dagi barcha mavjud funktsiyalar ro'yxitatini quyidagi manzilda topish mumkin: <https://docs.python.org/3/library/functions.html>.

```

1 def protcharge(aa_seq):
2     """Protein ketma-ketligining aniq zaryadini qaytaradi"""\n    protseq =
3     aa_seq.upper() zaryad = -0,002 aa_charge = {'C':-045, 'D':-999 ,
4     'E':-998, 'H':091, 'K':1, 'R':1, 'Y':-001} protseqdagi aa uchun: zaryad
5     += aa_charge.get(aa, 0)
6
7
8
9     qaytarish to'lovi

```

---

Funktsiyadan "foydalanish" uchun uni parametr bilan chaqirish kerak:

```
>>> protcharge('EEARGPLRGKGDQKSAVSQKPRSRGILH')
4.094
```

Agar parametrni o'tkazishni unutib qo'ysak yoki noto'g'ri parametr sonini o'tkazsak eters, biz xatoga duch kelamiz:

```
>>> protcharge()
```

Traceback (eng oxirgi qo'ng'iroq):

```
Fayl "<pyshell#3>", 1-qator, <module> protcharge()
da
TypeError: protcharge() aynan 1 ta argumentni oladi (0 berilgan)
```

Ushbu misolda funktsiya raqamni qaytaradi (float turi). Agar biz uning bir nechta qiymatlarni qaytarishini istasak, biz uni ro'yixat yoki kortejni qaytarishimiz mumkin.<sup>4</sup> Funksiya protzaryati (6.1-listda kodlangan ) aniq zaryaddan tashqari, zaryadlangan aminokislotalarning nisbatini qaytarish uchun o'yzgartirilishi mumkin:

**Listing 6.2: netchargefn: Ikki qiymatni qaytaruvchi funksiya**

---

```

1 def charge_and_prop(aa_seq): 2
    """\n        Protein ketma-ketligining aniq zaryadini va zaryadlangan\n3     aminokislotalarning nisbatini qaytaradi\n4
5     protseq = aa_seq.upper() zaryad\n6     = -0,002 cp = 0 aa_charge =\n7     {'C':-045, 'D':-999, 'E':-998,\n8     'H':091, 'K':1, 'R':1, 'Y':-001} protseqdagi aa uchun:\n9
10

```

<sup>4</sup> Ro'yxat o'rniiga kortejni qaytarish mantiqan to'g'ri keladi, chunki ma'lum bir funktsiya uchun fiksatsiya mavjud qaytarilgan parametrlar soni.

## 108 Bioinformatika uchun Python

```

11      zaryad += aa_charge.get(aa,0) agar aa
12      aa_chargededa: cp += 1 tayanch = 100.*cp/
13          len(aa_seq) qaytish (zaryad, prop)
14
15

```

---

Agar funktsiyani oxirgi misoldagi bir xil parametrlar bilan chaqirsak, biz boshqa natijaga erishamiz:

```
>>> charge_and_prop('EEARGPLRGKGDQKSAVSQKPRSRGILH')
(4.094000000000003, 39.285714285714285)
```

Bitta qiymatni olish uchun indeksdan foydalaning:

```
>>> charge_and_prop('EEARGPLRGKGDQKSAVSQKPRSRGILH')[1]
39.285714285714285
```

Barcha funktsiyalar nimanidir qaytaradi. Funktsiya qiymatni qaytarish o'rniga "biror narsa qilish" uchun ishlatalishi mumkin. Bu holda qaytarilgan qiymat Yo'q. Masalan, quyidagi funktsiya ro'yxat mazmunini matn faylida saqlaydi:<sup>5</sup>

#### **Listing 6.3: convertlist.py: Ro'yxitni matnli faylga aylantiradi**

---

```

1 def save_list(kirish_royyxati, fayl_nomi): """Ro'yxit
saqlanaqdi kirish royyxati faylga (fayl_nomi) shuningda faylning nomi \n
3
4
5
6     Qaytish Yo'q

```

return None iborasi ixtiyoriy. Funktsiya usiz None qaytaradi, lekin Python ishlab chiquvchilari yashirin taxminlardan ko'ra aniq bayonotlarni afzal ko'radilar.

Python 3 dan boshlab, Listing 6.3 chop etish funktsiyasi bilan yozilishi mumkin. Chop etish uchun faqat 5 qatorni almashtiring (element, fayl = fh). 4-qatordagi "for loop" dan hali ko'rilmagan xususiyatdan foydalanishdan qochish mumkin. 112-betdagি 6.6 ro'yxati halqasiz muqobilni ko'rsatadi.

### Funktsiya doirasi

Funktsiya ichida e'lon qilingan o'zgaruvchilar faqat funktsiya ichida amal qiladi. Bu shuni anglatadiki, agar siz o'zgaruvchiga funktsiyadan tashqaridan kirishga harakat qilsangiz, Python uni topa olib qo'shasi mumkin. Funktsiya o'zgaruvchisi tarkibiga funktsiyadan tashqaridan kirish uchun o'zgaruvchini return operatori yordamida asosiy dasturga qaytarish kerak. Quyidagi misolda takroriy funktsiya ichida aniqlangan y o'zgaruvchisi funktsiyadan tashqarida ishlatilmaydi:

---

<sup>5</sup>Python ma'lumotlar tuzilmalarining barcha turlarini saqlash usuli uchun 94-betdagи Pickle-ga qarang.

```
>>> def duplicate(x): y = 1
...     print('y =
...     {0}'.format(y)) return(2*x)
...
...
...
>>> dublikat (5) y = 1
10
```

```
>>> y
Traceback (eng oxirgi qo'ng'iroq):
  Fayl "<stdin>", 1-qator, <modul> da
NameError: "y" nomi aniqlanmagan
```

Bunda y ning qamrovi dublikat funksiyasi ichida joylashgan. Aytishimiz mumkinki, funktsiya y nomi "yashaydigan" nom maydonini taqdim etadi.

Agar nom funktsiya ichida chaqirilsa, lekin u yerda aniqlanmagan bo'lsa, Python uni funktsiyadan tashqarida qidiradi; agar u erda uni topa olmasa, NameErrorni qayt E'tibor bering, ismlarni qidirishda afzallik tartibi mavjud. Avval doirada, keyin esa global miqyosga etgunga qadar tashqarida chaqirildi:

```
>>> def duplicate(x): print('y
...     = {0}'.format(y)) return (2*x)
...
...
...
>>> dublikat(5)
Traceback (eng oxirgi qo'ng'iroq):
  Fayl "<stdin>", 1-qator, <modul> da
  Fayl "<stdin>", 2-qator, ikki nusxada
NameError: "y" nomi aniqlanmagan
```

Agar nom ota-onada doirasida aniqlangan bo'lsa, u uni topadi:

```
>>> y = 3
>>> def duplicate(x): print('y
...     = {0}'.format(y)) return(2*x)
...
...
...
>>> dublikat (5) y = 3
10
```

Agar nom funktsiya tomonidan taqdim etilgan nomlar maydonida va tashqarisida aniqlangan bo'lsa, Python birinchi mavjud nomdan, ya'nii funktsiya ichidagi nomdan foydalanadi:

## 110 Bioinformatika uchun Python

```
>>> y = 3
>>> def duplicate(x): y = 1
...     print('y =
...     {0}'.format(y)) return(2*x)
...
...
...
>>> dublikat (5) y = 1
10
```

Funktsiya ichida o'zgaruvchi global turdag'i ekanligini ko'rsatish mumkin, shuning uchun uning ishlash muddati u belgilangan joy bilan chegaralanmaydi. Global o'zgaruvchilardan foydalanish yaxshi fikr emas, chunki ular kutilmagan joylarda o'zgartirilishi mumkin. Global o'zgaruvchilar bilan bog'liq yana bir muammo shundaki, Python butun ish vaqtiga uchun o'z qiymatini kuzatib borishi kerak, shuning uchun u xotirada samarali emas.

```
>>> def test(x): global
...     zz = 10
...
...     print('z = {0}'.format(z)) qaytarmoq
...     x*2
...
...
>>> z = 1
>>> test (4)
z = 10
8
>>> z
10
```

### 6.2.2 Funktsiya parametrlari parametrlari

#### Argumentlarni joylashtirish

Shu paytgacha argumentlar dastlab belgilangan tartibda joylashtirildi.

**Funktsiyani saqlash ro'yxatini shunday deb  
atash mumkin: save\_list([1,2,3], 'list.txt').**

Agar argumentlar tartibini o'zgartirsak (save\_list('temp.txt',[1,2,3])) biz xato xabarini oling:

```
save_list('list.txt', [1,2,3])
Traceback (eng oxirgi qo'ng'iroq):
```

```
Fayl "<ipython-input-5-fe7756f18e74>", 1-qator, <modul> da
save_list('list.txt', [1,2,3,4,5])
```

```
"save_list1.py" fayli, 10-qator, saqlash_ro'yxatida ochiq
(fayl_nomi, "w") fh sifatida:
```

`TypeError: noto'g'ri fayl: [1, 2, 3, 4, 5]`

Ushbu `TypeError`, bu funksiya birinchi parametr sifatida ro'yxatni va ikkinchi parametr sifatida qatorni kutganligi sababli yuzaga keladi. Funksiyani parametrlari bilan dastlab belgilanganidan boshqacha tartibda chaqirish uchun funksiyani chaqirishda parametr nomini berish kerak:

```
>>> saqlash ro'yxati (file_name='list.txt', input_list=[1,2,3])
```

O'zgaruvchilar nomlaridan foydalangan holda parametrlar tartibi ahamiyatsiz.

### Standart qiymatlar bilan argumentlar

Python argumentlarda standart qiymatlarga ruxsat beradi. Bu funksiya ta'rifiiga standart qiymatni kiritish orqali amalga oshiriladi:

```
def nomi (arg1=default_value, arg2=default_value, ... ):
```

Masalan, ro'yxat mazmunini faylga saqlaydigan `save_list` funksiyasi standart fayl nomiga ega boylishi mumkin:

---

#### Listing 6.4: list2textdefault.py: Standart parametrga ega funksiya

---

```
1 ta saqlash_ro'yxati (kirish_ro'yxati, fayl_name='temp.txt'):
2     """Ro'yxat (kirish_ro'yxati) faylda (fayl_nomi)"""" ochiq (fayl_nomi, 'w') fh
3     sifatida saqlanadi: kirish_ro'yxatidagi element uchun:
4         fh.write('{0}\n'.format( element))
5
6     Qaytish Yo'q
```

---

Endi funksiyani faqat bitta parametr bilan chaqirish mumkin:

```
>>> save_list(['MS233','MS772','MS120','MS93','MS912'])
```

### Argumentlarning aniqlanmagan soni

Yakuniy parametr oldidan "\*" belgisi bo'lsa, funksiyalar o'zgaruvchan sonli argumentlarga ega bo'lishi mumkin. Har qanday ortiqcha argumentlar oxirgi parametrga kortej sifatida tayinlanadi:

---

#### Listing 6.5: getaverage.py: Parametr sifatida kiritilgan qiymatlarning oyrttacha qiymatini hisoblash funksiyasi

---

## 112 Bioinformatika uchun Python

```

1 def o'rtacha(*raqamlar): agar
2     len(raqamlar)==0 bo'lса:
3         Qaytish Yo'q
4     boshqa:
5         jami = summa (raqamlar)
6         jami qaytarish / len (raqamlar)

```

Shu tarzda o'rtacha funktsiyani aniqlanmagan sonli argumentlar bilan chaqirish mumkin:

```
>>> o'rtacha (2,3,4,3,2) 2.8
```

```
>>> o'rtacha(2,3,4,3,2,1,8,10) 4.125
```

Pythonda yulduzcha (\*) dan yana bir foydalanish mavjud. Python 3 dan oldin “\*\*” qo'yilgan o'zgaruvchi ro'yxatda jihatlangandigan elementlar mavjud. Bayxolsasiyatjan yedihik etma-ro'yxatdagi 5-satr) tsikldan foydalansmaslik uchun ishlataladi. L ning barcha elementlari bo'ylab yuring:

**Listing 6.6: list2text2.py: Chop etish va \* dan foydalanib ro'yxatni matnli faylga o'yzgartiradi.**

```

1 ta saqlash_ro'yxati (kirish_ro'yxati, fayl_name='temp.txt'): 2
    """Ro'yxat (kirish_ro'yxati) faylga (fayl_nomi)"""
    ochiq(fayl_nomi, 'w') fh
3     sifatida saqlanadi: print(*input_list, sep='\n', file=fh)
4
5     Qaytish Yo'q

```

### Kalit so'z argumentlarining aniqlanmagan soni

Funktsiyalar kalit so'zlar bilan ixtiyoriy sonli argumentlarni ham qabul qilishi mumkin. Bunday holda biz “\*\*” (ikki yulduzcha) oldidan yakuniy parameterlarni foydalashtirishga qaratiladi:

**Listing 6.7: list2text2.py: Argumentlarning o'zgaruvchan sonini qabul qiluvchi funktsiya**

```

1 def buyruq qatori (nom, ** parametrlar): 2 qator =
2
3     parametrlardagi element
4         uchun: qator += ' -{0} {1}'.format(element, parametrlar[element])
5     qaytish nomi + qator

```

6Bu PEP-3132 da batafsil tushuntirilgan (<http://www.python.org/dev/peps/pep-3132>) va bu Stackoverflow posti: <http://stackoverflow.com/questions/6967632>.

---

Ushbu funksiyani kalit so'z parametrlarining o'zgaruvchan soni bilan chaqirish mumkin:

```
>>> buyruq satri('formatdb', t='Caseins', i='indata.fas')
'formatdb -t Caseins -i indata.fas'
>>> buyruq satri('formatdb', t='Kazeinlar', i='indata.fas', p='F') 'formatdb -t Kazeinlar
-p F -i indata.fas'
```

---

Maslahat: Docstrings haqida ba'zi so'zlar.

Funksiyalar funksiya ta'rifidan so'ng darhol matn qatoriga ega bo'lishi mumkin.

Ushbu qator (yoki satrlar) "docstring" deb ataladi. 112-betdag'i [6.6 ro'yixatda](#) bir qatorli hujjat qatori mavjud.

Ushbu satrlar onlays yordam, hujjatlarni avtomatik yaratish tizimlari va manba kodini o'qishni istagan har bir kishi uchun ishlataladi. Docstring ichida istalgan narsani yozishingiz mumkin , lekin hujjatlar qatori tuzilishini standartlashtirish uchun yozma ko'rsatmalar mavjud. Iltimos, PEP-257 ga qarang (<http://www.python.org/dev/peps/pep-0257>) Docstring format konvensiyalari haqida qo'shimcha ma'lumot olish uchun.

Docstrings nafaqat funktsiyalarga ega bo'lishi mumkin; modular va sinflar birinchi bayonot sifatida o'z hujjatlariga ega bo'lishi kutilmoqda.

---

### 6.2.3 Generatorlar

Generatorlar - bu alohida turdag'i funksiya. Funktsiyalar mahalliy nomlar maydonidagi o'zgaruvchilar yordamida ba'zi amallarni bajaradi. Ushbu o'zgaruvchilar funksiya bajarilgandan so'ng o'chiriladi. Bu jarayon har safar funksiya chaqirilganda sodir bo'ladi. Bunga yo'l qo'ymaslik uchun generator deb ataladigan maxsus funksiya mavjud. Generator bajarilganda uning ichki holati saqlanadi, shuning uchun keyingi safar chaqirilganda o'zgaruvchilar qiymatlariga kirish mumkin. Ba'zan ular qayta tiklanadigan funktsiyalar deb ataladi. Bu bir vaqtning o'zida katta ob'ektni (masalan, katta ro'yxat, tuple va hokazo) qaytarmaslik uchun

Masalan, fayldagi yozuvlarni o'qiydigan va ushbu fayldagi ma'lumotlar bilan ma'lumotlar strukturasini qaytaradigan funksiyani olaylik. Agar fayl juda katta bo'lsa (masalan, mavjud xotiradan bir necha marta), natijada olingan ma'lumotlar strukturasi xotiraga sig'masligi mumkin. Ushbu muammoni hal qilish funksiyani bir vaqtning o'zida bitta yozuvni qaytarish uchun o'zgartirishdir. Funktsiya buni qila olmaydi, chunki u holatni saqlamaydi, shuning uchun u har safar bajarilganda, barcha ma'lumotlarni qayta ishlashga to'g'ri keladi. Generator - bu o'zinin Ular yangi kalit so'zni kiritadilar: rentabellik. `yield EXPRESSION` bayonoti topilganda, u `EXPRESSION`ni chaqirilgan joyiga qaytaradi (yoki beradi) (funksiya sifatida), lekin uning ichki qiymatlarini kuzatib boradi, shuning uchun keyingi safar chaqirilganda, u avvalgidek qiymatlar bilan ishslashni davom ettiradi. qiymat berishdan oldin.

## Generator yaratish

**6.8 ro'yxatida** berilgan qiymatgacha mavjud bo'lgan barcha tub sonlarni qaytaruvchi funksiya (`all_primes()`) mavjud. Bu ularning barchasini ro'yxatda qaytaradi:

**Listing 6.8: allprimes.py: berilgan qiymatgacha barcha tub sonlarni qaytaruvchi funksiya**

---

```

1 def is_prime(n): 2
    (2,n-1) oralig'idagi i uchun """To'g'ri n asosiy, bo'lmasa
3      noto'g'ri"" ni qaytaradi: agar n%i == 0 bo'lsa:
4
5          False ni qaytaring
6      Qaytish Haqiqiy
7
8 def all_prime(n): tub
diapazon oralig'sozi(9 ,n):
10     agar asosiy(raqam):
11         primes.append(raqam) p
12             qaytariladi
13

```

**6.8 ro'yxatidagi** `all_primes()` funksiyasi `g_all_primes()` generatori bilan almashtirilishi mumkin:

**Listing 6.9: allprimesg.py: 6.8 kodidagi putn() o'rnnini bosuvchi generator.**

---

```

1 def g_all_prime(n):
2 agar bo'saz oralidagi(sapuchun(1,n):
3     rentabellik raqami
4

```

E'tibor bering, **6.9 ro'yxatidagi** kod ro'yxatni ishlatmaydi, chunki unga ehtiyoj yo'q, chunki u bir vaqtning o'zida bitta natija beradi. Natijalar ustida yurish uchun ikkala funksiyadan ham foydalanish mumkin, lekin `all_primes()` ro'yxatni yaratadi, `g_all_primes()` esa yo'

## 6.3 MODULLAR VA PAKETLAR

---

Modul - bu funksiya ta'riflari, konstantalar yoki boshqa modullardan yoki asosiy dasturingizdan foydalanishingiz mumkin bo'lgan har qanday turdag'i ob'ektga ega fayl. Modullar nomlar bo'shilqlarini ham ta'minlaydi, shuning uchun ikkita funksiyaga bir xil nom berilishi mumkin, agar ular turli modullarda aniqlangan bo'lsa. Modul nomi fayl nomidan olingan. Agar modul fayl nomi `my_module.py` bo'lsa, modul nomi `my_module` bo'ladi.

### 6.3.1 Modullardan foydalanish

**Modul tarkibiga kirish uchun importdan foydalaning.** Odatda import dasturning boshida chiqariladi. Importlarni faylning boshida joylashtirish majburiy emas, lekin modulning biron bir elementini chaqirishdan oldin uni joylashtirish kerak.

Import bayonotini dasturning boshida joylashtirish odatiy holdir. Importdan foydalanishning ko'plab usullari mavjud. Eng ko'p ishlataladigan shakl modulni o'z nomi bilan chaqirishdir. O'rnatilgan OS moduliga qo'ng'iroq qilish uchun quyidagilardan foydalaning:

```
>>> import os
```

Modul birinchi marta import qilinganda uning mazmuni bajariladi. Agar modul bir necha marta import qilinsa, ketma-ket importlar hech qanday ta'sir ko'rsatmaydi.

Bu bizga import bayonotini funksiya ichiga qo'yishimiz va agar u qayta-qayta chaqirilsa, tashvishlanmasligimizga kafolat beradi.

Modul import qilingandan so'ng, funksiya yoki o'zgaruvchiga kirish uchun modul nomini prefiks sifatida ishlating:

```
>>> os.getcwd() '/  
mnt/hda2'  
>>> os.sep '/'
```

Shuningdek, moduldan faqat kerakli funksiyani import qilish mumkin. Bu yerga modul nomini prefiks sifatida ishlatmasdan uni chaqirishimiz mumkin.

```
>>> OS import getcwd >>>  
getcwd() '/mnt/hda2'
```

Modulning barcha mazmunini import qilish uchun "\*" operatoridan foydalaning (yulduzcha):

```
>>> OS importidan * >>>  
getcwd() '/mnt/hda2'  
  
>>> sen  
'/'
```

**Ogohlantirish:** Agar nima qilayotganingizni bilmasangiz, import modulidan \* foydalanmang. Modulning barcha elementlarini import qilish bilan bog'liq muammo shundaki, u asosiy dasturda allaqachon aniqlangan (yoki boshqa modullarda aniqlangan va xuddi shunday import qilingan) nomlar bilan ziddiyatlarni keltirib chiqarishi mumkin. Python dasturlash standartlarida joker belgilari importi kuchning qorong'u tomoniga teng. Ular tezroq, osonroq va jozibali, ammo xavfli.

Boshqa nom yordamida modulni import qilish ham mumkin:

```
>>> import xml.etree.ElementTree kabi et >>>
daraxt = et.parse('/home/sb/bioinfo/smallUniprot.xml')
```

Agar siz `xml.etree.ElementTree` nima ekanligini bilmasangiz, tashvishlanmang, biz buni XML bobida ko'rib chiqamiz, ammo shu paytdan boshlab bu nom (`xml.etree.ElementTree`) "" deb nomlanganligini hisobga oling. ET."

### 6.3.2 Paketlar

Paket - bu umumiy xususiyatlarga ega bo'lgan modullar guruhi. Ular ichida modullar yoki boshqa kataloglar joylashgan kataloglar. Shuningdek, `__init__.py` nomli maxsus fayl mavjud. Bu fayl uning tarkibidagi katalog Python paketi ekanligini va modul sifatida import qilinishi mumkinligini bildiradi.

Bio/	Yuqori darajadagi paket
__init__.py	Ovoz paketini ishga tushiring
Align/	Alignment bilan bog'liq dasturiy ta'minot uchun kichik paket
__init__.py	
AlignInfo.py	
Alphabet/ __init__.py	Aminokislota alifbolari uchun kichik paket
IUPAC.py	
Reduced.py	
Blast/	Blast tahlilchilari uchun kichik paket
__init__.py	
Applications.py	
NCBIStandalone.py	
NCBIWWW.py	
NCBIXML.py	
ParseBlastTable.py	
Record.py	

`__init__.py` fayllari Python kataloglarni o'z ichiga paketlar sifatida ko'rishi uchun talab qilinadi. Ko'pgina hollarda `__init__.py` bo'sh fayldir, lekin u paketni ishga tushirish kodini ham bajarishi mumkin.

Paket foydalanuvchilari paketdan alohida modullarni import qilishlari mumkin, masalan:

```
Bio.Blast.Applications import
```

Modullar va paketlar o'tasida farqlar mavjud bo'lsa ham, ikkala atama bir-birining o'rnidagi ishlataladi.

### 6.3.3 Uchinchi tomon modullarini o'rnatish

Python bir nechta modullar (o'rnatilgan modullar) bilan birga keladi. Bu modullar Python bilan birlashtirilgan, shuning uchun sizda ishlaydigan Python tarjimoni boylgan zahoti foydalanishga tayyor boyladi.<sup>7</sup> 11-sahifada aytib o'ytilganidek, Python funksiyalarini kengaytiruvchi uchinchi tomon modullari ham mavjud. O'rnatish nusxa olish kabi oson boylishi mumkin. bir nechta dasturlarni oldindan belgilangan tartibda bajarishgacha ma'lum bir joyga bitta fayl. Bu modullarning murakkabligiga bog'liq. Modullar boshqa dasturlar bilan o'zaro aloqada bo'lgan bir nechta kataloglarda joylashgan bir fayldan bir nechta fayllargacha bo'ladi; bu holda u paket deb ataladi. Shunday qilib, Python uchun mavjud bo'lgan har bir tashqi modulni o'rnatishning yagona usuli yo'q.

#### Pip - afzal qilingan usul

Aksariyat paketlar pip o'rnatishni qo'llab-quvvatlaydi. pip Python paketlarini o'rnatishning mahalliy usuli bo'lib, afzal qilingan usul hisoblanadi.<sup>8</sup> Bunday o'rnatish uchun sizga pip va setuptools kerak bo'ladi. Ehtimol, siz ularni allaqachon o'rnatgan bo'lishingiz mumkin (bu Python 3.4 binaries bilan birga keladi), agar bo'lmasa; uni o'rnatishing:

**sudo apt o'rnatish python-pip**

Esda tutingki, Ubuntu-da paket python3-pip deb ataladi.

Keyin uni eng so'nggi versiyaga yangilang.

Linux yoki macOS da:

**\$ pip o'rnatish -U pip o'rnatish vositalari**

Ubuntu'da buyruq quyidagicha:

**\$ pip3 o'rnatish -U pip o'rnatish vositalari**

Windowsda:

**\$ python -m pip o'rnatish -U pip o'rnatish vositalari**

O'rnatilgan va yangilangandan so'ng, Python modullari quyidagilar bilan o'rnatilishi mumkin:

**MODULE\_NAME*n* \$ pip o'yrnating**

Masalan, Excel fayllarini o'qish uchun xlrd paketini o'rnatish uchun:

---

7 <http://docs.python.org/library/index.html> ni tekshiring o'rnatilgan modullarni o'z ichiga olgan Python standart kutubxonasining to'liq tavsifi uchun.

8 Ushbu kitobning birinchi nashrida ko'rsatilgan easy\_install deb nomlangan yana bir usul bor, lekin pip ko'proq xususiyatlarga ega, shuning uchun easy\_install bu kitobdan olib tashlandi. Ikkala tizimni solishtirish uchun [https://packaging.python.org/pip\\_easy\\_install/](https://packaging.python.org/pip_easy_install/) manzilini tekshiring .

```
$ pip install xlrd
xlrd-1.0.0 yig'ilmoqda
  xlrd-1.0.0.tar.gz yuklanmoqda
Yig'ilgan paketlar uchun qurilish g'ildiraklari: xlrd
  xlrd uchun setup.py bdist_wheel ishga tushirilmoqda ...
  bajarildi Katalogda saqlangan: /home/sb/.cache/pip/wheels/55/e2/c6f97024749<=
ea24f67400fb4c55eab7f2b49cbf39379805ef
Muvaffaqiyatli qurilgan xlrd
Yig'ilgan paketlar o'rnatilmoqda: xlrd
Muvaffaqiyatli o'rnatildi xlrd-1.0.0
```

Yuqoridagi buyruqni bajarish uchun sizga ishlaydigan Internet aloqasi kerak. Pip paketni <https://pypi.python.org> manzilidagi PyPi omboridan oladi .

Pip yordamida qanday paketlarni oýrnatish mumkinligini bilish uchun <https://pypi.python.org/pypi?%3Aaction=browse> roýyxatiga qarang .

E'tiborga olish kerak bo'lgan yana bir ogohlantirish - o'rnatilgan paketni kim ishlatsihi. Paketni mashinadagi barcha Python foydalanuvchilari uchun mavjud qilish uchun siz administrator foydalanuvchi bo'lisingiz yoki uni sudo bilan o'rnatishingiz kerak:

```
$ sudo pip install xlrd
```

Lekin o'rnatishning afzal usuli foydalanuvchi sifatida va virtual muhit ichida (119-betga qarang).

### Tizim paketlarini boshqarishdan foydalanish

Ba'zi Python modullari kompyuteringizga o'rnatgan har qanday boshqa dastur kabi o'rnatilishi mumkin, masalan, o'rnatuvchini ikki marta bosish (Windows/macOS) yoki Ubuntu'da apt-get kabi tizim paketini boshqarish.

Tizim paketlarini boshqarishdan foydalanishning afzalligi shundaki, siz o'rnatilgan Python modullarini tizimingizdagi boshqa dasturlarni kuzatib borganingiz kabi kuzatib borishingiz mumkin. Yangilash va o'chirish oson va hech qanday yomon oqibatlarsiz, masalan, etim fayllar yoki buzilgan o'rnatishlarsiz. Ushbu usulning kamchiliklari ham bor, masalan joriy paket versiyasi va Linux tarqatish omborida mavjud versiya o'rtasidagi bo'shliq. Ba'zi modullar tez sur'atlar bilan rivojlanadi, ba'zida esa shu qadar tezki, paket menejerlari yangilanib turolmaydi. Misol uchun, Biopython-ni apt-get yordamida o'rnatmoqchi bo'lgan Ubuntu foydalanuvchilari yozish paytida 1.66 versiyasi bilan cheklangan, agar 1.69 Biopython veb-saytida mavjud bo'lgan oxirgi versiya bo'lsa. Yana bir muammo shundaki, ba'zi tizimlarda paket boshqaruvchisidan foydalanish uchun sizga ma'muriy huquqlar kerak bo'ladi. Windows o'rnatuvchilari barcha kerakli dasturlarni taqdim etmaydi va uni avtomatik tarzda qidirmaydi, shuning uchun o'rnatuvchini ishga tushirishdan oldin ba'zi bir zarur dasturlarni o'rnatishingiz kerak bo'lishi mumkin. Paketlarni boshqarish bilan bog'liq asosiy muammo shundaki, ba'zida kerakli paket mavjud emas. Shu sabablarga ko'ra, bu yangi paketlarni o'rnatishning tavsiya etilgan usuli emas.

## PYTHONPATHga nusxa olinmoqda

Bu modulni o'rnatishning eng tez-tez uchraydigan tartibi emas, lekin bu juda oddiy bo'lgani uchun birinchi navbatda eslatib o'tiladi. Python modullarni qidiradigan modulni faqat nusxa ko'chiring. Python modullarni qayerda qidiradi? Uchta joy bor:

- Modulni chaqiradigan dastur joylashgan katalogda.
- Python bajariladigan fayl joylashgan katalogda. Bu katalog har bir operatsion tizimda har xil.<sup>9</sup>
- Bizning modularimiz uchun maxsus yaratilgan katalogda. Bunday holda, u PYTHONPATH muhit o'zgaruvchisida yoki sys.path o'zgaruvchisida ko'rsatilishi kerak. Ushbu yakuniy o'zgaruvchi Python modulni izlashi kerak bo'lgan barcha yo'llarni sanab o'tadi. Katalogni sys.path ga qo'shish uchun uni qo'shish usulidan foydalangan holda istalgan ro'yxatdag'i kabi o'zgartirishingiz kerak:

```
>>> import sys
>>> sys.path [ '/usr/lib/python3.5', '/usr/lib/python3.5/plat-x86_64-<= linux-gnu', '/usr/lib/python3.5/lib-dynload', '/usr/local/lib/python3.5/dist-packages', '/usr/lib/python3/dist-<= -packages' ] >>> sys.path.append('/home/sb/MyPyModules')
```

### 6.3.4 Virtualenv: Izolyatsiya qilingan Python muhitlari

`virtualenv` izolyatsiyalangan Python muhitini yaratuvchi dasturdir. `Virtualenv` bilan yaratilgan har bir muhit o'zining tashqi (uchinchi tomon) modullariga ega.

Bu har birining o'z muhitida bir nechta mustaqil loyihalarga ega bo'lislis imkonini beradi, shuning uchun mos kelmaydigan bog'liqliklar bilan ziddiyat yo'q. Loyiha X versiyada modulni talab qilishi mumkin va boshqa loyihamaga bir xil modul kerak bo'lishi mumkin, a Xuddi shu Python o'rnatishda bir xil modulning ikki xil versiyasini o'rnatolmaysiz, shuning uchun har bir o'rnatishni izolyatsiya qilish usuli kerak. Bu `virtualenv` taqdim etadi. `Virtualenv` dan qachon foydalinish kerak? Qisqa javob: Har doim. Uzoq javob: Har safar yangi loyiha bilan ishlashni boshlaganingizda, maxsus Python muhitiga ega bo'lganingiz ma'qul. Bu, shuningdek, kerak bo'lganda, ma'lum bir dastur uchun sozlamalarni boshqa mashinada qayta ishlab chiqarish imkoniyatiga ega. Agar sizda Python3.6 yoki undan kattaroq bo'lsa, `virtualenv` mustaqil dastur sifatida kerak emas, chunki u Python-ga kiritilgan.

3.6 dan oldingi Python versiyasi bilan `virtualenv` ni pip yordamida o'rnating:

---

<sup>9</sup>Windowsda odatda C:\program files\Python da topiladi, Linuxda esa topiladi. /usr/bin/python. \*nix da Python bajariladigan faylga yo'llni topish uchun qaysi pythondan foydalaning.

## 120 Bioinformatika uchun Python

# pip virtualenv o'rnatish

Windows-da pip skriptlar katalogida joylashgan bo'lishi mumkinligini unutmang.

O'rnatilgandan so'ng virtualenv shu tarzda yaratiladi:

\$ virtualenv <DIRECTORY>

Agar sizda Python 3.6 yoki undan yuqori versiya mavjud bo'lsa, virtualenv ni o'rnatishingiz shart emas Python 3.6 yoki undan yuqori versiyalari bilan yangi virtual muhit yarating. Buning o'rniغا, shunchaki bajaring:

python3 -m venv <DIRECTORY>

Pythonning eski versiyasiga misol:

\$ virtualenv bioinfo '/usr'

asosiy prefiksidan foydalanish /

home/sb/bioinfo/bin/python3 da bajariladigan yangi python. Shuningdek, /home/sb/bioinfo/bin/python da bajariladigan fayl yaratish. Setuptools, pip, wheel o'rnatish... bajarildi. \$

yoki Python 3.6 bilan:

python3 -m venv bioinfo

Bu joriy katalog ichida bioinfo katalogini yaratadi.

Yaratilgandan so'ng, uni faollashtirish vaqtি keldi. macOS va Linux:

\$ manba <DIRECTORY>/bin/activate

Oldingi misolda faollashtirish buyrug'i quyidagicha bo'ladi:

\$ manba bioinfo/bin/activate

Windows-da virtual muhit shu tarzda faollashtirilganligini unutmang:

<DIRECTORY>\Scripts\activate

Virtual muhitni faollashtirgandan so'ng, so'rov quyidagicha o'zgaradi:

(bioinfo)\$

Bu virtualenv faollashtirilganligini va har bir paketni ko'rsatish uchun ishlataladi o'sha nuqtadan o'rnatish faqat ushbu virtualenv ichida mavjud bo'ladi.

Paketni o'rnatish uchun avvalgidek (pip-dan foydalanib), lekin muhit ichida davom eting, masalan:

```
(bioinfo)$ pip install xlrd
```

Shunday qilib, xlrd to'plami faqat bioinfo muhitida mavjud bo'ladi va boshqa Python o'rnatilishiga xalaqtirishga bermaydi.

Virtual muhit bilan ishlashni tugatgandan so'ng, siz o'chirib qo'yishingiz kerak standart so'rovningizga qaytish uchun:

```
(bioinfo)$ $ o'chirish
```

Windowsda:

```
(bioinfo)> \path\to\env\bin\deactivate.bat
>
```

### 6.3.5 Conda: Anaconda virtual muhiti

Agar siz Anaconda Python tarqatishdan foydalananayotgan bo'lsangiz, virtualenv o'rniغا conda create dan foydalaning. Agar siz oddiy Python-dan foydalananayotgan bo'lsangiz, ushbu bo'limni o'tkazib yuboring.

Yangi muhit yaratish uchun quyidagi buyruqdan foydalaning:

```
$ conda create -n NAME
```

Bu erda NAME yangi muhit uchun foydalanoqchi bo'lgan nom bo'lsa, bioinfo dan foydalanoqchi bo'lsangiz, buyruq quyidagicha bo'ladi:

```
$ conda create -n bioinfo Paket
```

metamaýlumotlari olinmoqda .....

.Paket spetsifikatsiyalarini hal qilish: .

Atrof muhitda o'rnatish uchun paket rejasini /home/sb/anaconda3/<= envs/bioinfo:

Quyidagi bo'sh muhitlar YARATILADI:

```
/home/sb/anaconda3/envs/bioinfo
```

Davom etish ([y]/n)?

```
#  
# Ushbu muhitni faollashtirish uchun quyidagilardan foydalaning:  
# > manba bioinfoni faollashtiradi  
#  
# Ushbu muhitni o'chirish uchun: # > manba bioinfoni  
o'chirish  
#
```

Atrof muhitni faollashtirish uchun quyidagilarni yozing:

```
$ manba faollashtirish bioinfo
(bioinfo) $
```

Virtualenv muhitida bo'lgani kabi, ushbu muhitga o'rnatgan hamma narsa atrof-muhit uchun mahalliy bo'ladi va boshqa Python o'rnatilishiga ta'sir qilmaydi.

Paketni faol muhitda o'rnatish uchun conda install dan foydalanish afzalroqdir. Yostiq to'plamini o'rnatish uchun:

```
(bioinfo) $ conda o'rnatish yostiq
```

Agar paket konda omborida mavjud bo'lmasa, siz pip dan foydalanshingiz mumkin o'rnatish:

```
(bioinfo)$ pip install beautifulsoup4
```

Mening maslahatim Anaconda bilan ishlashda conda va agar foydalansangiz pip dan foydalaning standart Python taqsimoti.10

Conda shuningdek, virtual muhit yaratish va paketlarni o'rnatish imkonini beradi bitta buyruq. Faqat conda create buyrug'idan keyin paket nomini qo'shing:

```
$ conda create -n excelprocessing xlrd. Paket
metamaylumotlarini olish .....
Paket spetsifikatsiyalarini hal qilish: .....
```

/sb/anaconda3/envs/exc<= elprocessing muhitida o'rnatish uchun paket rejasi:

Quyidagi paketlar yuklab olinadi:

paket		qurmoq
	----- python-3.6. 0 0	
setuptools-27.2.0 g'ildirak-0.29.0		16,3 MB
	py36_0	523 KB
	py36_0	88 KB
xlrd-1.0.0	py36_0	185 KB
pip-9.0.1	py36_1	1,7 MB
	Jami:	18,8 MB

Quyidagi YANGI paketlar O'R NATILADI:

```

openssl:      1.0.2j-0
pip:          9.0.1-py36_1
python:        3.6.0-0
readline:     6.2-2
setuptools:   27.2.0-py36_0 sqlite:
              3.13.0-0 tk:
              8.5.18-0
g'ildirak:    0.29.0-py36_0
xlrd:         1.0.0-py36_0
xz:           5.2.2-1
zlib:         1.2.8-3

```

Davom etish ([y]/n)? y

Paketlar olinmoqda ...

```

python-3.6.0-0 100% #####| Vaqt: 0:00:01 10,99 MB/s setuptools-27. 100% |
#####| Vaqt: 0:00:00 5,72 MB/s (...)
```

Paketlar chiqarilmoqda ...

```

[ TO'LIQ Delgahamoqda ... ]#####| 100%
[ TO'LIQ #
]#####| 100%
```

# Ushbu muhitni faollashtirish uchun

quyidagilardan foydalaning: # > source activate  
excelprocessing #

# Ushbu muhitni o'chirish uchun quyidagilardan  
foydalaning: # > manba excelprocessingni o'chirish  
#

Paketni faollashtiring va o'rnatilganligini tekshiring:

```

$ source activate excelprocessing
(excelprocessing) $ python Python 3.6.0 |
Continuum Analytics, Inc.| (2016 yil 23 dekabr, 12:22:00)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] Linuxda Qo'shimcha
ma'lumot uchun "yordam", "mualliflik huquqi", "kreditlar" yoki "litsenziya" ni kriting. >>>
import xlrd
>>>
```

Barcha o'rnatilgan muhitlarni (nom va yo'l) ko'rish uchun conda info --envs ni ishga tushiring:

```

$ conda info --envs
# konda muhiti:
#
```

```

AEML           /home/sb/anaconda3/envs/AEML /home/
bioinfo        sb/anaconda3/envs/bioinfo /home/sb/
biopy1         anaconda3/envs/biopy1 /home/sb/anaconda3/
py4bio-ni qayta ishlash  envs/excelprocessing /home/sbassi/anaconda4/envs/ * /
                           home/sbassi/anaconda3
ildiz

```

Yulduzcha (\*) faol muhitni ko'rsatadi.

**Virtualenv** (yoki Anaconda ishlatisa, conda) dan foydalanish shunchalik muhimki, shunday bo'ladi uni qanday ishlatish haqida ushbu kitobda bir nechta havolalar bo'lishi kerak.

### Qo'lida qurish va o'rnatish

Agar siz tizim paketlaridan foydalana olmasangiz va pip-dan foydalanishni xohlamasangiz (yoki foydalana olmasangiz), paketlarni o'rnatishning har doim qo'lida usuli mavjud. Modul fayllarini yuklab oling (odatda ".tar.gz" formatida), ularni oching va setup.py faylini qidiring. Ko'pgina hollarda, uni o'rnatish ishga tushirish masalasiadir:

```
python setup.py o'rnatish
```

Agar biron bir muammo bo'lsa, README fayliga qarang. Aslida, dasturni o'rnatishdan oldin README faylini tekshirish tavsiya etiladi (buni kim qiladi?). Aksariyat hollarda muammo siz bajarishingiz kerak bo'lgan bog'liqliklar etishmasligidan kelib chiqadi (masalan, Y modulini ishga tushirish uchun X moduli kerak). Shuning uchun Python modullarini pip yoki tizim paketini boshqarish bilan o'rnatish yaxshiroqdir.

#### 6.3.6 Modullarni yaratish

Modul yaratish uchun siz fayl yaratishingiz va uni ".py" kengaytmasi bilan saqlashingiz kerak. U PYTHONPATH o'zgaruvchisi kabi Python tarjimonini uni qidiradigan katalogda saqlanishi kerak (qo'shimcha ma'lumot uchun 119-betga qarang).

Masalan, save\_list funksiyasini modulda saqlang va uni utils deb chaqiring. Buning uchun quyidagi tarkibga ega utils.py faylini yaratish:

```

# utils.py fayli def
save_list(input_list, file_name='temp.txt'):
    """Ro'yixat (kirish_ro'yixati) faylga (fayl_nomi)"""" ochiq(fayl_nomi, 'w') fh sifatida
    saqlanadi: print(*input_list, sep='\n', file=fh)

```

Qaytish Yo'q

Shunday qilib, ushbu funktsiyadan (save\_list) har qanday dasturdan foydalanish mumkin, agar ushbu fayl Python-dan foydalanish mumkin bo'lgan joyda saqlangan bo'lsa:

```

>>> import utils >>>
utils.save_list([1,2,3])

```

### 6.3.7 Sinov modullari

Yaxshi dasturlash amaliyoti kodingizning to'g'ri ishlashini tekshirish uchun testlarni yaratishni o'z ichiga oladi.

Modular dastur ichidan foydalanish uchun mo'ljallanganligi sababli, bu testlar faqat buyruq satridan chaqirliganda bajarilishi kerak. Shunday qilib, testlar dasturning normal ishlashiga xalaqit bermaydi.

Bunga erishish uchun biz kod qachon mustaqil dastur sifatida bajarilayotganini va qachon boshqa dasturdan modul sifatida bajarilayotganini farqlay olishimiz kerak.

Kod dastur sifatida bajarilganda `__name__` o'zgaruvchisi "`__main__`" qiymatini oladi. Natijada, test kodini kiritish usuli dastur mustaqil ravishda bajarilishini tekshirgandan so'ng uni amalga oshirishdir.

```
agar __name__ == '__asosiy__':
    #Biror narsa qilmoq
```

Ushbu turdag'i test odatda modul oxirida kiritiladi. Ro'yxatda [6.10](#)  
(125-bet) biz sinovni amalda ko'rishimiz mumkin.

Python bizning kodimiz biz kutgandek ishlashini tekshirish vazifasini osonlashtiradigan modulni taqdim etadi. Ushbu modul doctest deb ataladi.

#### Doctest, modullarni avtomatik tarzda sinovdan o'tkazish

Doctest - bu docstring ichida Python kodining qismlarini qidiradigan modul. Ushbu kod interaktiv Python sessiyasi kabi bajariladi. Modul ushbu kodning docstring yoki tashqi faylda ko'rsatilgandek ishlayotganligini tekshiradi.

[Listing 6.10](#) da bizda `is_prime()` funksiyasi berilgan raqam mavjudligini tekshiradi  
(n) asosiy hisoblanadi. Keling, qanday qilib sinov blokini o'rnatishimiz va uni ishga tushirishimiz mumkinligini ko'rib chiqamiz:

#### Listing 6.10: prime5.py: doctest bilan modul

---

```
1 def is_prime(n): 2
    """ n tub son ekanligini tekshiring.
3     Foydalanish
4     namunasi: >>> is_prime(0)
5     Yolg'on
6     >>> is_prime(1)
7     To'g'ri
8     >>> is_prime(2)
9     To'g'ri
10    >>> is_prime(3)
11
12    To'g'ri >>> is_prime(4)
13    Yolg'on
14    >>> is_prime(5)
```

## 126 Bioinformatika uchun Python

```

15      To'g'ri
16      """
17
18      agar n <= 0 bo'lса:
19          # Bu faqat > 0 raqamlari uchun. False
20          qaytaring
21      (2, n) diapazonidagi x uchun:
22          agar n%x == 0 bo'lса:
23              False ni qaytaring
24      Qaytish Haqiqiy
25
26 def _test():
27     import doctest
28     doctest.testmod()
29
30 agar __name__ == '__main__':
31     _test()

```

---

Kod tushuntirishi: `is_prime(n)` funksiyasi 1-qatordan 24-qatorgacha aniqlanadi, lekin haqiqiy funksionallik 18-qatordan boshlanadi. Bu qatorgacha baъzi testlar mavjud.

Agar dastur 30-qatorda tekshirilgan boshqa dasturdan chaqirilsa, bu testlar bajarilmaydi.

Agar dastur mustaqil dastur sifatida bajarilsa, barcha testlar bajarilmaydi.

boshqariladi:

`$ python prime5.py $`

Chiqish yo'q. Ya'ni, hech qanday yangilik yaxshi yangilik emas. Keling, 21-qatorni “`x in diapazonda(1,n):`” ga o'zgartirganimizda nima sodir bo'lishini ko'rib chiqamiz: Bu holda test muvaffaqiyatsiz tugadi:

`$ python prime5.py`

```
*****
"./prime5.py" fayli, 10-qator, __main__.is_prime ichida
Muvaffaqiyatsiz misol: is_prime(2)
```

Kutilgan:

To'g'ri

Olingan:

Yolg'on

```
*****
"./prime5.py" fayli, 12-qator, __main__.is_prime ichida
Muvaffaqiyatsiz misol: is_prime(3)
```

Kutilayotgan:

To'g'ri

Olingen:

Yolg'on

"./prime5.py" fayli, 16-qator, \_\_main\_\_.is\_prime ichida Muvaffaqiyatsiz misol:

is\_prime(5)

Kutilgan:

To'g'ri

Olingen:

Yolg'on

1 ta elementda xatolik yuz berdi:

\_\_main\_\_.is\_prime ichida 6 tadan 3 tasi

\*\*\*Test muvaffaqiyatsiz tugadi\*\*\* 3 ta xato.

Sinov shunchalik muhimki, testga asoslangan rivojlanish deb ataladigan metodologiya mavjud. U kod yozishni boshlashdan oldin har bir funksiya uchun test ishlab chiqishni taklif qiladi.

Sinovni dastur uchun asosiy ehtiyoj sifatida qabul qilish mumkin emas, lekin uni sinab ko'rmaguningizcha, funksiya ishlashiga ishonch hosil qilib bo'lmaydi. Sinov koddagi o'zgarishlarning kutilmagan oqibatlarga olib kelmasligiga ishonch hosil qilish uchun ham foydalidir.

Python dasturiy ta'minotni sinovdan o'tkazish uchun keng ko'lamli yordamga ega (doctest va unittest modullari bilan), ammo bu ushbu kitobning doirasiga kirmaydi. Sinov haqida ko'proq ma'lumot olish uchun "Qo'shimcha manbalar" ga qarang.

## 6.4 QO'SHIMCHA RESURSLAR

---

- Modullar, Python darsligi. <http://docs.python.org/tutorial/modules.html>
- Python'da standart parametr qiymatlari, Fredrik Lund. <http://effbot.org/zone/default-values.htm>
- Python kutubxonasi ma'lumotnomasi. Unittest API. <https://docs.python.org/3.6/library/unittest.html>
- Python modullarini o'rnatish. <http://docs.python.org/install/index.html>
- PIP. <https://pip.pypa.io/en/stable/>
- Ekstremal dasturlash. Vikipediya maqolasi. [http://en.wikipedia.org/wiki/Extreme\\_Programming](http://en.wikipedia.org/wiki/Extreme_Programming)

## 6.5 O'Z-O'ZINI BAHOLASH

---

1. Funksiya nima?
2. Funksiyani nechta qiymat qaytarishi mumkin?
3. Funksiyani hech qanday parametrsiz chaqirish mumkinmi?
4. Docstring nima va u nima uchun funksiyalar va modular bilan bog'liq?
5. Har bir funksiya qancha parametr larga ega bo'lishini oldindan bilishi kerakmi  
qabul qiladimi?
6. Generator funksiyasini yozing.
7. Nima uchun funksiyadagi barcha ixtiyoriy argumentlar ning oxirida joylashtirilishi kerak  
funksiya chaqiruviga?
8. Modul nima?
9. Nima uchun modular dastur boshida chaqiriladi?
10. Modulning barcha mazmunini qanday import qilasiz? Ushbu protsedura tavsiya etiladimi?
11. Kodingiz mustaqil dastur sifatida bajarilayotganligini qanday tekshirishingiz mumkin yoki  
modul sifatida chaqiriladimi?
12. Virtualenv nima va undan qachon foydalanasiz?

# Xato bilan ishlash

---

## MAZMUNI

---

<b>7.1 Xatolarni boshqarishga kirish.</b>	<b>129</b>
7.1.1 Sinab ko'ring va bundan mustasno .....	131
mustasno .....	7.1.2 Istisno
turlari. ....	134 Turli istisnolarga
qanday javob berish kerak .....	134 7.1.3 Istisnolarni
ishga tushirish. ....	135 7.2 Moslashtirilgan
istisnolarni yaratish. ....	136 Barcha
istisnolar istisnolar sinfidan kelib chiqadi. ....	137 7.3
Qo'shimcha manbalar ..	137 7.4 O'z-o'zini ba

Siz uni aql bovar qilmaydigan qilib qo'yishingiz mumkin, lekin siz uni la'natsiz qila olmaysiz.

Naeser qonuni

## 7.1 XATOLARNI QO'LLANISHGA KIRISH

---

Dastur kamdan-kam hollarda kutilganidek ishlaydi, hech bo'limganda birinchi urinishda.

An'anaga ko'ra, dasturchi ish vaqtidagi dastur xatolariga duch kelganida ushbu ikkita strategiyadan birini tanlaydi. Muammo e'tiborga olinmaydi yoki xatolik yuzaga kelishi mumkin bo'lgan har bir shart tekshiriladi va keyin u ketma-ket kod yozadi. Juda mashhur bo'lgan birinchi variant, agar biz dasturimizni o'zimizdan boshqa hech kim ishlashini xohlasak, tavsiya etilmaydi. LBYL (Look Before You Leap) nomi bilan ham tanilgan ikinchi variant ko'p vaqt talab qiladi va kodni o'qib bo'lmaydigan qilib qo'yishi mumkin. Keling, har bir strategiyaning misolini ko'rib chiqaylik.

Quyidagi dastur yorliqlar bilan ajratilgan faylni (myfile.csv) o'qiydi va birinchi qatorning birinchi ustunida joylashgan raqamni qidiradi. Bu qiymat 0,2 ga ko'paytiriladi va bu natija boshqa faylga (otherfile.csv) yoziladi.

Ushbu versiya hech qanday turdag'i xatolarni tekshirmaydi va o'zini asosiy funksiyalari bilan cheklaydi.

**Listing 7.1: wotest.py: Xatolarni tekshirmaydigan dastur**

---

```
1 ochiq('myfile.csv') bilan fh sifatida:  
2     line = fh.readline()
```

## 130 Bioinformatika uchun Python

```
3 qiymat = line.split('\t')[0] 4 bilan
open('other.txt', "w") fw sifatida:
5     fw.write(str(int(value)*.2))
```

---

Agar kutilmagan hodisalar bo'limasa, bu dastur o'z vazifasini bajarishi mumkin. Ushbu kontekstda "kutilmagan hodisalar" nimani anglatadi? Birinchi qator xatolikka moyil. Masalan, u mavjud bo'Imagan faylni ochmoqchi bo'lishi mumkin. Bunday holda, dastur ishga tushganda, u birinchi qatorni bajargandan so'ng darhol to'xtaydi va foydalanuvchi xatoga duch keladi:

Traceback (eng oxirgi qo'ng'iroq):

```
"wotest.py" fayli, 1-qator, <module> ichida
    open('myfile.csv') fh sifatida: FileNotFoundError:
[Errno 2] Bunday fayl yoki katalog yo'q: 'myfile.csv'
```

Bu muammo, chunki dastur to'xtaydi va uni ko'rsatish professional emas oxirgi foydalanuvchi tizim xatosi.

Ushbu dastur turli joylarda muvaffaqiyatsiz bo'lishi mumkin. Faylda yorliqlar bo'Imasligi mumkin, raqamlar or'niga harflar bo'lishi mumkin va biz chiqish faylini yozmoqchi bo'lgan katalogda yozish ruxsatiga ega bo'Imasligimiz mumkin.

Fayl mavjud bo'lganda, lekin ichida yorliqlar bo'limasa, shunday bo'ladi.

Traceback (eng oxirgi qo'ng'iroq):

```
"wotest.py" fayli, 6-qator, <module>
    fw.write(str(int(value)*.2))
ValueError: int() uchun yaroqsiz literal 10 asosli: '12,dsa\n'
```

Natija avvalgisiga o'xshaydi. Bu dasturni to'xtatishga olib keladi va tarjimon bizga boshqa xato xabarini ko'rsatadi. Shunday qilib, biz muvaffaqiyatsizlikka moyil bo'lgan barcha kod bloklari bilan davom etishimiz mumkin.

Keling, xato yuzaga kelishining oldini olish uchun har bir holatni tekshirish strategiyasini ko'rib chiqaylik (LBYL).

**Listing 7.2: LBYL.py: LBYL versiyasini qayta ishlashda xato**

---

```
1 import os
2
3 iname = input("Kiritish fayl nomini kriting: ")
4 oname = input("Chiqish fayl nomini kriting: ")
5 agar os.path.mavjud bo'lsa(iname): fh sifatida open(iname)
6     bilan: 5
7         line = fh.readline(), agar '\t' qatorda bo'lsa: qiymat =
8             line.split('\t')[0] if os.access(name,
9                 os.W_OK) == 0:
```

```

10         fw sifatida open(name, 'w') bilan: if
11             value.isdigit():
12                 fw.write(str(int(value)*.2))
13             boshqa:
14                 print("int-ga aylantirib bo'lmaydi")
15             boshqa:
16                 print ("Chiqish fayli yozilmaydi")
17             boshqa:
18                 print("Tab yo'q. Kirish faylini tekshiring")
19 boshqa:
20     chop etish ("Fayl mavjud emas")

```

---

Ushbu dastur deyarli barcha mumkin bo'lgan xatolarni ko'rib chiqadi. Agar foydalanuvchi kiritgan fayl mavjud bo'lmasa, dasturda g'ayritabiyy tugatish bo'lmaydi. Buning o'rniiga, u foydalanuvchiga kirish fayli nomini qayta kiritish imkonini beradigan dasturchi tomonidan ishlab chiqilgan xato xabarini ko'rsatadi.

Ushbu variantning kamchiligi shundaki, kodni o'qish va saqlash qiyin, chunki xatolarni tekshirish uni qayta ishlash va dasturning asosiy maqsadi bilan aralashtiriladi. Aynan shuning uchun yangi dasturlash tillari istisno sharoitlarni boshqarish uchun maxsus tizimni o'z ichiga olgan. LBYLdan farqli o'laroq, bu strategiya EAFF sifatida tanilgan (ruxsatdan ko'ra kechirim so'rash osonroq). Python bilan iboralar harakat qiladi, bundan mustasno, aks holda y nihoyat ishlatiladi.

#### 7.1.1 Sinab ko'ring va istisno qiling

try biz bajarmoqchi bo'lgan kodni chegaralaydi, istisno esa try bloki ostidagi kodda xatolik bo'lsa, bajariladigan kodni chegaralaydi. Amalga oshirish jarayonida aniqlangan xatolar istisnolar deb ataladi . Keling, umumiy sxemani ko'rib chiqaylik:

```

harakat
    qilib ko'ring: kod bloki 1
    # ...ba'zi xato kodlari...
bundan
    mustasno: kod bloki 2
    # ...xato bilan biror narsa qiling... [boshqa:

        kod bloki 3
        # ...xato bo'lmaganda qilish uchun...

nihoyat:
    kod bloki 4
    #...ba'zi tozalash kodi...]

```

Bu kod birinchi bo'lib 1-blokdagи kodni bajarishga harakat qiladi. Agar kod muammosiz bajarilsa, bajarish oqimi 3-blokdagи kod orqali va nihoyat 4-blok orqali davom etadi. 1-blokdagи kod xatolik (yoki) hosil qilgan taqdirda ko'taradi a

## 132 Bioinformatika uchun Python

jargonga ko'ra istisno), 2-blokdagi kod, so'ngra 4-blokdagi kod bajariladi. Ushbu mexanizm ortidagi g'oya xatoga olib kelishi mumkin bo'lgan kod blokini (1-blok) sinab ko'rish ichiga qo'yishdir. bandi. Istisno mavjud bo'lganda ishga tushiriladigan kod istisno blokiga joylashtiriladi. Ushbu kod (kod bloki 2) istisno bilan shug'ullanadi yoki boshqacha qilib aytganda, istisno bilan shug'ullanadi. Istisno holatlar ko'rib chiqilmasa, foydalanuvchi xato xabarlarini oladi:

```
>>> 0/0
```

Traceback (eng oxirgi qo'ng'iroq):

Fayl "<stdin>", 1-qator, <modul> da

ZeroDivisionError: nolga bo'linish

Ixtiyoriy ravishda, try (kod bloki 1) ichidagi kod muvaffaqiyatli bajarilgan taqdirdagina bajariladigan boshqa bayonotni qo'shish mumkin. E'tibor bering, quyida keltirilgan boshqa kodni sinab ko'rish blokiga qo'yish mumkin, chunki u bir xil ta'sirga ega bo'ladi (xato bo'lmasa, u ishlaydi). try ichidagi blok faqat istisnoga olib kelishi mumkin bo'lgan kodni o'z ichiga olishi kerak, biz esa blok ichidagi blokda qoldirishimiz kerak, aks holda try ichidagi ko'rsatmalar xatosiz bajarilganda bajarilishi kerak bo'lgan ko'rsatmalar. E'tibor bering, nihoyat ichidagi kod har doim bajariladi.

Masalan; misol uchun:

urinib

ko'ring:

print(0/0) bundan

mustasno: print("Xyuston, bizda muammo bor...")

Natijada:

Xyuston, bizda muammo bor...

Biz e'tiborga oladigan birinchi narsa shundaki, na boshqasi, na oxiri kiritilmagan, chunki ular ixtiyoriy bayonotlardir. Bunday holda, print(0/0) bayonoti istisno keltirib chiqaradi. Ushbu istisno faqat ichidagi kod tomonidan "ushlangan". Shunday qilib, xatolikdan keyin ham dastur oldindan aytib bo'ladigan tarzda oqishiga ishonch hosil qilamiz.

Ushbu kodda istisnolardan foydalanish [Listing 7.2 kodiga nisbatan qo'llaniladi](#):

[Listing 7.3: exception.py: 7.2 ga o'xhash, ammo istisnolar bilan ishlash.](#)

---

```

1 ta
onamei=inpolt('Qo''yishda yaxshi?') open('kiritilay') blaj
3     fh:
4
5         line = fh.readline(), agar '\t'
6         qatorda bo'lsa: qiymat =
7             line.split('\t')[0]
```

```

8     fw sifatida open(name, 'w') bilan:
9         fw.write(str(int(qiymat)*.2))

Nom xatosidan tashqari
10: 11    print("TAB yo'q. Kirish faylini tekshiring") 12 bundan
mustasno FileNotFoundError: print("Fayl mavjud emas") 13 14 bundan
mumkin emas. 15 PermissionError: print("Ochiq faylga yozish

```

print("Qiymatni int ga aylantirib bo'lmaydi") 18 boshqa:

```
19    print("Rahmat!. Hammasi yaxshi bo'ldi.")
```

Birinchi qarashda bu kodni amal qilish avvalgi versiyaga (7.2) qaraganda osonroq ekanligi seziladi. Hech bo'limganda kod mantig'i xatolarni qayta ishlashdan ajratilgan. 10-qatorдан istisnolar bilan ishlash boshlanadi. Istisno turiga ko'ra, quyida bajariladigan koddir. Har xil turdag'i istisnolarni qanday ajratish mumkinligini keyinroq ko'rib chiqamiz.

**7.3 ro'yxati** istisnolardan foydalanishni qanday qo'llashning kirish namunasidir  
**Listing 7.1** va istisnolarni qanday hal qilish bo'yicha aniq qo'llanma emas.

**Listing 7.4: nested.py: Ichki istisnolar bilan kod**

```

1 iname = input("Kirish fayl nomini kriting: ") 2 oname
= input("Chiqish fayl nomini kriting: ") 3 urinib ko'ring:
open(iname) bilan fh: 4

5         line = fh.readline() 6
bundan mustasno FileNotFoundError:
agar "\t" qaynot(dengyob'ishavjudemas") 7 8
line.split("\t")[0] 10 urinib ko'ring:
9         open(name) bilan , 'w') fw sifatida:
11

12         fw.write(str(int(qiymat)*.2))

Nom xatosidan tashqari
13: 14    print("TAB yo'q. Kirish faylini tekshiring") 15 bundan
mustasno PermissionError: print("Ochiq faylga yozish mumkin emas.")
16    17 bundan mustasno ValueError: 18

print("Qiymatni int ga aylantirib bo'lmaydi") 19 boshqa:
20    print("Rahmat!. Hammasi yaxshi bo'ldi.")

```

Biz umumiy ma'noda try/except bandining qanday ishlashini ko'rdik va endi buni qila olamiz istisnolar turlarini muhokama qilish uchun biroz chuqurroq boring.

### 7.1.2 Istisno turlari

Istisnolar individual bo'lishi mumkin. Mavjud bo'limgan o'zgaruvchi va mos kelmaydigan ma'lumotlar turlarini aralashtirish bir xil turdag'i xato emas. Birinchi istisno NameError turiga, ikkinchisi esa TypeError turiga tegishli. Istisnolarning to'yliq ro'yxatini <https://docs.python.org/3.6/library/exceptions.html> sahifasida topish mumkin.

Turli xil istisnolarga qanday javob berish kerak

Xatoni odatda parametrsiz ishlatish mumkin:

```
d = {"A":"Adenin", "C":"Sistein", "T":"Timin", "G":"Guanin"} harakat qilib  
ko'ring: chop etish d[input("Harf kriting: ")]
```

bundan

```
mustasno: print("Bunday nukleotid yo'q")
```

Barcha xatolarga umumiy javob bera olishimiz bu yaxshi fikr degani emas. Bu bizning kodimizni disk raskadrovska qilishni qiyinlashtiradi, chunki kutilmagan xatolik e'tibordan chetda qolishi mumkin. Ushbu kod har qanday xatolik uchun "Bunday nukleotid yo'q" ni qaytaradi. Agar biz EOF signalini kirtsak (faylning oxiri, ba'zi terminallarda CONTROL-D), dastur "Bunday nukleotid yo'q" deb chiqaradi. Har xil turdag'i g'ayritabiyy hodisalarni ajratib ko'rsatish va natijada reaksiyaga kirishish foydalidir. Masalan, EOFni mavjud bo'limgan lug'at kalitidan farqlash uchun:

```
d = {"A":"Adenin", "C":"Sistein", "T":"Timin", "G":"Guanin"} urinib ko'ring:  
print(d[input("Harf kriting: ")]) ) EOFErrorдан tashqari: print("Xayr!") KeyError  
bundan mustasno: print("Bunday nukleotid yo'q")
```

Shunday qilib, foydalanuvchi kalitni kiritganda dastur "Bunday nukleotid yo'q" ni chop etadi. Bu d lug'atda yo'q va "Xayr, xayr!" EOF olganida.

Hozirda ko'rib chiqilayotgan istisno haqida ma'lumot olish uchun sys.exc\_info() dan foydalaning:

**Listing 7.5: sysexc.py: sys.exc\_info() dan foydalanish**

1 ta import tizimi

```

2
3 ta
urinib
ko'ring: 4
6      0/0 5 bundan mustasno: a,b,c
7      = sys.exc_info() print('Xato nomi: {0}'.format(a.__name__))
8      print('Xabar: {0}'.format (b)) chop etish ('Satrdagi xato:
9      {}'.format(c.tb_lineno))

```

---

Ushbu dastur chop etadi:

Xato nomi: ZeroDivisionError  
 Xabar: butun son yoki modul nolga bo'linish  
 Qatordagi xato: 4

#### **Listing 7.6: sysexc2.py: sys.exc\_info() dan boshqa foydalanish**

---

```

1 import tizimi 2
3 urinib
5 bundahotiring asosiy faydalishni o'qish uchun
[::2] print('Xato nomi: {}'.format(a.__name__))
print('Xato kodi: {}'.format(b.args[0]))
print('Xato xabari: {}'.format(b.args[1]))
7
8
9

```

---

Ushbu dastur chop etadi:

Xato nomi: FileNotFoundError  
 Xato kodi: 2  
 Xato xabari: Bunday fayl yoki katalog yo'q

#### **7.1.3 Istisnolarni ishga tushirish**

Istisnolar, ularning paydo bo'lishini kutmasdan, oshirish yordamida qo'lida faollashtirilishi mumkin. Nima uchun istisnoni ishga tushirishni xohlayotganingizni qiziqtirgan bo'lishingiz mumkin. Tegishli tarzda ko'tarilgan istisno dasturchi yoki foydalanuvchi uchun nazoratsiz tarzda o'chirilgan istisnodan ko'ra ko'proq foydali bo'lishi mumkin. Bu, ayniqsa, dasturlarni disk raskadrova qilishda to'g'ri keladi.

Buni misol bilan yaxshiroq tushunish mumkin. Quyidagi avg funktsiyasi raqamlar ketma-ketligining o'rtacha qiymatini hisoblaydi:

```

def avg(raqamlar):
    qaytariladigan summa(raqamlar)/len(raqamlar)

```

Ushbu turdag'i funksiya bo'sh ro'yxat bilan bog'liq muammolarga duch keladi:

```
>>> o'rtacha([])
Traceback (eng oxirgi qo'ng'iroq):
  Fayl "<stdin>", 1-qator, <modul> da
  Fayl "<stdin>", 2-qator, o'rtacha
ZeroDivisionError: nolga bo'linish
```

Ushbu xato xabari bilan bog'liq muammo shundaki, u bizga bo'sh ro'yxat sabab bo'lganligini aytmaydi, lekin u qo'zg'atilganligini, lekin nolga bo'linishga harakat qilganini aytadi. Funktsiya qanday ishlashini bilib, bo'sh ro'yxat bu xatoga sabab bo'lgan degan xulosaga kelish mumkin. Biroq, bu xato funktsiyaning ichki tuzilishini bilmasdan turib, buni ko'rsatsa, qiziqroq bo'ladi. Buning uchun biz o'zimiz xato qilishimiz mumkin.

```
def o'rtacha (raqamlar):
    raqamlar bo'lmasa:
        oshirish ValueError("Iltimos, kamida bitta element kriting")
    qaytariladigan summa(raqamlar)/len(raqamlar)
```

Bunday holda, xato turi haqiqiy muammoga yaqinroq bo'ladi.

```
>>> o'rtacha([])
Traceback (eng oxirgi qo'ng'iroq):
  Fayl "<stdin>", 1-qator, <modul> da
  Fayl "<stdin>", 3-qator, o'rtacha
ValueError: kamida bitta element kriting
```

Agar satrni xatoni ko'tarmasdan chop qilsak, xatodan qochishimiz mumkin edi, ammo bu pitonik tamoyillarga zid bo'ladi ("xatolar e'tibordan chetda qolmasligi kerak"). Amalda, bu muammolarni keltirib chiqarishi mumkin, chunki funktsiya kutilmagan qiymatni qaytarsa, ta'sirlarni oldindan aytib bo'lmaydi. Istisnoni ko'tarib, xato e'tibordan chetda qolmasligiga ishonch hosil qilamiz.

Ba'zi matnlarda yoki eski kodda siz "Bu xato" ko'tarilishining sintaksisini topasiz. Ushbu turdag'i istisnolar (zanjirli istisnolar deb ataladi) Python 2.6 va undan keyingi versiyalar bilan mos kelmaydi. Shakl ValueError ko'taradi, "Xabar" ham eskirgan va afzal ko'rilgan shakl ValueError('Xabar')ni ko'tarishdir. Python 3.0 dan oxirgi shakl majburiydir.<sup>1</sup>

## 7.2 SOZLANGAN ISTISOZLARNI YARATISH

---

Istisno tizimining afzalligi shundaki, biz o'zimizni Python tomonidan taqdim etilganlar bilan cheklashimiz shart emas. Biz ehtiyojlarimizni qondirish uchun yangi istisnolarni belgilashimiz mumkin. In

<sup>1</sup>PEP 3109 ga qarang (<http://www.python.org/dev/peps/pep-3109>) Buning mantiqiy asoslari haqida.

istisno yaratish uchun biz ob'ektga yo'naltirilgan dasturlash (OOP) bilan ishlashimiz kerak, bu mavzu hali o'rganilmagan. Natijada, agar siz ushbu kitobni boshidan o'qiyotgan bo'lsangiz va o'zingizning istisnolariningizni yaratishingiz kerak bo'lsa, mening tavsiyam shuki, ushbu bobning qolgan qismini o'tkazib yuborib, to'g'ridan-to'g'ri 8-bobga o'ting. 8-bobni o'qib bo'lgach, **ushbu bo'limga** qayting .

### Barcha istisnolar istisno sinfidan kelib chiqadi

Barcha istisnolar Exception sinfidan kelib chiqqanligi sababli, biz Exception sinfini kichik sinflarga ajratish orqali o'zimizni istisno qilishimiz mumkin. Masalan, men NotDNAException deb atagan ushbu istisnoni olaylik. "a", "c", "t" yoki "g" ga tegishli bo'limgan belgilarga ega bo'lgan DNK ketma-ketligi mavjud bo'lganda uni ko'tarish kerak. Keling, aniqlangan maxsus istisnoni ko'r

sinf NotDNAException (istisno):

```
"""Foydalanuvchi tomonidan belgilangan
istisno"""
def __init__(self, DNA):
    self.dna = DNA
def __str__(self):
    self.dna da nt uchun:
        agar nt "atcg" da bo'lmasa:
            qaytish nt
```

Dasturchi istisnoni aniqlash uchun kod yaratishi kerak:

```
dnaseq = 'agctwtacagt' agar
o'rnatilgan(dnaseq) != set('atcg'):
    NotDNAException(dnaseq)ni ko'taring
boshqas:
    chop etish ('OK')
```

Agar dnaseq 'a', 'c', 't' yoki 'g' bilan takrorlanadigan ob'ekt bo'lsa, bu kod OK chop etadi. Ammo agar dnaseqda DNK bo'limgan belgi bo'lsa, istisno ko'tariladi. Bu avvalgi kodning natijasi, ammo dnaseqda "w" bilan:

Traceback (eng oxirgi qo'ng'iroq):

```
"7_25.py" fayli, 22-qator, <modul> da
NotDNAException(dnaseq) ko'tariladi
__main__.NotDNAException: w
```

---

## 7.3 QO'SHIMCHA RESURSLAR

- PEP 3134 “Istisno zanjirlar va o'rnatilgan kuzatuvalar”. <https://www.python.org/dev/peps/pep-3134/>
- Python hujjatlari. O'rnatilgan istisnolar.
   
<https://docs.python.org/3.6/library/exceptions.html>

- Python hujjatlari. Standart xato tizim belgilari. <https://docs.python.org/3.6/library/errno.html>
- C H. Swaroop. Python istisnolari. <https://python.swaroopch.com/exceptions.html>
- Ian Biking. Istisnolarni qayta ko'tarish. <http://www.ianbicking.org/blog/2007/09/re-raising-exceptions.html>

## 7.4 O'Z-O'ZI BAHOLASH

---

1. LBYL va EAfp nimani anglatadi? Qaysi biri Pythonda ishlataladi?
2. Istisno nima?
3. "Ko'rib chiqilmaydigan istisno" nima?
4. Qachon oxirgi marta foydalanasiz va qachon boshqasini ishlatasiz?
5. Istisnolar ko'pincha fayllarni qayta ishlash bilan bog'liq. Nega?
6. Diskning to'liq holatidan kelib chiqqan xatoni faqat o'qish uchun fayl tizimiga yozishga urinishdan qanday qilib tartiblash mumkin?
7. Why is not advisable to use except: to catch all kind of exceptions, instead foydalanish, masalan, IOError:?
8. Istisnolar o'z xohishiga ko'ra ko'tarilishi mumkin. Nega bunday qilardingiz?
9. sys.exc\_info() ning maqsadi nima?
10. Ushbu funktsiyaning maqsadini tushuntiring:

```
def formatExceptionInfo():
    """ Muallif: Arturo 'Buanzo' Busleiman
    cla, exc = sys.exc_info()[:2]
    excName = cla.__name__
    """
    urinib ko'ring: excArgs = exc.__dict__["args"]
    bundan mustasno, KeyError: excArgs =
        str(exc) qaytish (excName, excArgs)
```

# Ob'ektga kirish

## Yo'naltiruvchi dasturlash

### (OOP)

#### MAZMUNI

8.1 Ob'ektlar paradigmasi va Python .....	.....
<b>139</b> 8.2 Jargонни ойрганиш .....	.....
<b>140</b> Sinflar: Obyekt generatorlari .....	.....
<b>140</b> Misol: Sinfning alohida amalga oshirilishi .....	..... <b>140</b>
Atributlar yoki misol oýzgaruvchilari: obyektlarning xarakteristikalari .....	..... <b>141</b>
Usullar: Ob'ektlarning xatti-harakati .....	..... <b>141</b>
Sinf atributlari: Sinf larning xususiyatlari .....	..... <b>141</b>
Meros: xossalalar turdosh sinflar orasida uzatiladi <b>141</b>	.....
Polimorfizm .....	..... <b>141</b>
Inkapsulyatsiya .....	..... <b>141</b>
8.3 Sinflar yaratish. ....	..... <b>142</b>
Meros .....	..... <b>145</b> Ayrim
Biopython ob'ektlari bilan tanishish. ....	..... <b>146</b>
8.5 Maxsus usullar. ....	..... <b>149</b>
8.5.1 Oýrnatilgan ma'lumotlar turidan foydalanib yangi ma'lumotlar turini yaratish .....	<b>154</b>
8.6 Kodeksimizni shaxsiy qilish .....	..... <b>154</b>
8.7 Qo'shimcha manbalar .....	..... <b>155</b>
8.8 O'z-o'zini baholash. ....	.....

#### 8.1 OBYEKT PARADIGMASI VA PYTHON

Kitobning kirish qismida aytib o'tilganidek, Python ob'ektga yo'naltirilgan tildir. Ob'ektlar bilan ishlaydigan boshqa tillardan farqli o'laroq, Python bizga ob'ektlar paradigmasini hisobga olmagan holda klassik protsessual tarzda dasturlash imkonini beradi. Ba'zan bu "ko'p paradigmali til" deb ataladi.

Biz allaqachon ob'ektlardan foydalanganmiz, hatto uni aniq ko'rsatmasdan ham. Python-ga kiritilgan ma'lumotlar turlari ob'ektlardir. Satrlar, lug'atlar va ro'yxatlar ob'ektlarning amalga oshirilishidir. Ularning har biri o'ziga xos funktsiyalarga ( jargondagi usullar ) va uning atributlariga (bog'langan ma'lumotlarga) ega. Biz `Lower()` ning satrni q

kichik harfda. Buning sababi shundaki, sinf satrining barcha ob'ektlari ular bilan bog'langan low() usuliga ega.

Haqiqiy dunyoning bir qismini ifodalash odatda dasturlashning maqsadlaridan biridir. Bank tranzaksiyasidan tortib DNK ketma-ketligini qayta tiklashgacha hammasini dasturlash tilida ifodalash mumkin. Python-ga kiritilgan ma'lumotlar turlari juda ko'p va xilma-xil bo'lsa-da, uning barcha axborotni modellashtirish ehtiyojlarini o'z ichiga olishi cheklangan. Sinf yangi turdag'i ma'lumotlar turini aniqlash uchun ishlatalishi mumkin.

Masalan, lug'at nukleotidlari va aminokislotalari orasidagi tarjima jadvalini, qator DNK ketma-ketligini va kortej oqsildagi atomning fazoviy koordinatalarini ifodalashi mumkin. Ammo hujayraning metabolik holatini ko'rsatish uchun qaysi ma'lumotlar turidan foydalanamiz? Proteindagi turli domenlar? BLAST yugurish natijasimi? Ekotizim haqida nima deyish mumkin?

Biologik yoki boshqa turdag'i har qanday tizimni modellash imkoniyatiga ega bo'lish uchun o'zimizning ma'lumotlar turlarini aniqlashga ehtiyoj bor. Funktsiyalar kodni modullashtirish uchun foydali bo'lsa-da, ular bu rolni bajarish uchun mo'ljallanmagan. Funktsiyalar holatlarni saqlay olmaydi, chunki o'zgaruvchilar qiymatlari faqat funksiya bajarilayotga Boshqa tillarda C tilidagi "structs" yoki Paskalda "record" kabi shaxsiylashtirilgan ma'lumotlar turlari mavjud, ammo ular OOP asosidagi tillar ob'ektlari (Java, C++ yoki Python kabi) kabi moslashuvchanlikka ega emas. Ob'ektlar har qanday turdag'i tizimni va uning boshqa tizimlar bilan bo'lishi mumkin bo'lgan munosabatlarini modellashtirish uchun etarli

## 8.2 JARGONNI O'rganish

---

OOP dunyosi o'z lug'atiga ega. Ushbu bo'limda men sinf, usul, misol, atributlar, polimorfizm, meros va boshqalar kabi ko'plab yangi so'zlarning bir nechtasini tushuntirishga harakat qilaman. Ta'riflar to'liq bo'lmaydi. Ulardan ba'zilari hatto aniq bo'lmaydi, lekin ortiqcha rasmiy emas, balki mavzuni tushunish ustuvor bo'ladi. Esda tutingki, ushbu kitobning maqsadi biologik muammolarni hal qilish uchun dasturlash vositalarini taqdim etishdir. Shuni hisobga olib, quyidagi ta'riflar va ularga tegishli misollar yozildi.

### Darslar: Obyekt generatorlari

Sinf - bu ob'ektlarni yaratish uchun ishlataladigan shablon. Ob'ektlar ma'lumotlarni o'z ichiga olishi va ular bilan bog'liq funktsiyalarga ega bo'lishi mumkin. Sinf qator yoki to'plam kabi ma'lumotlar turi bo'lishi mumkin, shuningdek, genom, odamlar, ketma-ketliklar va boshqalar kabi murakkabroq narsa bo'lishi mumkin. Abstraktsiyalash mumkin bo'lgan har qanday ob'ekt sinf bo'lishi mumkin.

### Misol: Sinfning alohida amalga oshirilishi

Misol - bu sinfni amalga oshirish. Misol uchun, agar bizda Orca sinfi bo'lsa, misol Willy bo'lishi mumkin. Bitta sinfdan bir nechta misollar yaratilishi mumkin (masalan, Shamu) va barchasi bir-biridan mustaqil.

**Atributlar yoki misol o'zgaruvchilari: ob'ektlarning xarakteristikalari**

Har bir ob'ektning o'ziga xos xususiyatlari (yoki atributlari), masalan, vazni bo'ladi. Villi Shamudan farqli vaznga ega bo'lishi mumkin, ammo ularning atributlaridagi o'zgarishlarga qaramay, ikkala misol ham bir xil Orca sinfiga tegishli. Ular hech bo'limganda "atributlar turini" baham ko'rishadi. Biz Lassie, Laika va Rin-tin-tin misollari bilan sind itini yaratishimiz mumkin. Bu sindda hair\_color atributi bo'ylishi mumkin, bu atribut Orca sinfi misollari tomonidan baham ko'yrilmaydi.

**Usullari: Ob'ektlarning xatti-harakati**

Metod - bu ob'ekt bilan bog'langan funktsiya. Usullar ob'ektlarning qanday "o'zini tutishini" belgilaydi. Misol uchun, DNK klassi aminokislotalar ketma-ketligini oqsilga aylantirish imkonini beruvchi translatsiya usuli bilan plazmid namunasiga ega bo'lishi mumkin. Buning uchun Pythondagi yozuv: plasmid.translate(). Bu usul sind bilan bog'langan funktsiyadir. Bu parametr sifatida DNK ketma-ketligi va tarjima jadvali bo'lган lug'atni talab qilishi mumkin. Orca sinfiga mos ravishda, u ovqatlanish usuliga ega bo'lishi mumkin.

**Sinf atributlari: Sinfning xususiyatlari**

Atributlar sinfning barcha ob'ektlari bilan bog'langan o'zgaruvchilardir. Ob'ekt sindan yaratilganda, bu ob'ekt sind o'zgaruvchisini meros qilib oladi. Orca sindida vazn sind atributi bo'lishi mumkin.

**Meros: Xususiyatlar tegishli sinflar o'rtasida uzatiladi**

Sinflar bir-biri bilan bog'liq bo'lishi mumkin va ular alohida ob'ektlar emas. Orca sinfi va Dog sinfi bilan umumiy xususiyatlarga ega Sutemizuvchilar sinfiga ega bo'lish mumkin . Masalan, sutemizuvchilar sinfi uchun takror ishlab chiqarish usuli aniqlanishi mu... Biz Dog va Orca sinflarini yaratganimizda va ularni sutemizuvchilarning "bolalari" deb belgilaganimizda, ular uchun ko'payish usulini yaratish kerak bo'lmaydi. Bu usul ota-onasindan meros bo'lib qoladi. Bolalar sinflarida suzish va yugurish kabi o'ziga xos usullar bo'lishi mumkin.

**Polimorfizm**

Polimorfizm - bu har xil turdag'i ob'ektlarning bir xil usulga boshqa xatti-harakatlar bilan javob berish qobiliyati. Xuddi shu usul, masalan, ozuqa, Orca sinfida va Dog sinfida juda farq qiladi. Ikkalasi ham bir xil deb ataladi, ammo natija boshqacha bo'lishi mumkin. Misol uchun, siz ro'yxat, lug'at, fayl va boshqalarni xuddi shu tarzda takrorlashingiz mumkin, lekin Python iteratsiyani boshqarish usuli har bir ob'ekt turi uchun o'zgaradi.

## Inkapsulyatsiya

Inkapsulyatsiya - bu ob'ektning ichki ishlashini yashirish va dasturchilarga faqat umumiy usullar orqali kirish huquqini qoldirish qobiliyati. Encapsulation atamasi Python bilan bog'lanmagan, chunki bu tilda haqiqiy inkapsulyatsiya mavjud emas. Muayyan usullarga kirishni qiyinlashtirish mumkin, ammo buni oldini olish mumkin emas. Dasturchining yo'lida bo'lish Python falsafasiga kirmaydi. Python-da nima qilish mumkin, qaysi usullar va xususiyatlar faqat sinfga tegishli ekanligini va qaysi biri baham ko'rinishini aniq ko'rsatishdir.

Ushbu xatti-harakat psevdo-kapsulyatsiya yoki shaffof inkapsulyatsiya deb ham ataladi. Bu variantdan oqilona foydalanish dasturchiga bog'liq. Bu Pythonda deyiladi: qonun bilan emas, balki konvensiya bilan himoya qilish. Ushbu xususiyatdan foydalanish uchun 154-betdag'i "Kodimizni shaxsiy qilish" bo'limiga qarang.

### 8.3 SINFLAR TUZISH

---

Sinflar ob'ektlarning shablonidir. Python-da sinflarni yaratish sintaksisi juda oddiy:

**sinf nomi:**

[tana]

Keling, namunaviy sinfni ko'rib chiqaylik:

**sinf kvadrati:**

```
def __init__(self):
    o'z tomoni = 1
```

Bu sinfda (Kvadrat) `__init__` deb nomlangan usul mavjud. Bu hech qanday qiymatni qaytarmaydigan maxsus usul. Kvadrat nusxasi yaratilganda (yoki yaratilganda) u bajariladi. U ma'lum bir boshlang'ich holatni sozlash uchun ishlatiladi. Bunday holda, u atribut tomonining qiymatini belgilaydi. Ko'rib chiqilishi kerak bo'lgan yana bir o'ziga xoslik - bu o'z-o'zidan so'z bo'lib, u usulning parametri sifatida va atribut nomining bir qismi sifatida takrorlanadi. Self - bu Square misolini ko'rsatish uchun ishlatiladigan o'zgaruvchidir. O'z o'rniga boshqa ismni ishlatish mumkin, lekin o'z-o'zidan an'anaviy ravishda ishlatiladi. Konvensiyaga amal qilish tavsiya etiladi, chunki bu bizning dasturimizni boshqa dasturchilarga tushunishni osonlashtiradi.<sup>1</sup>

Sinfni yaratish uchun siz funksiya belgilardan foydalanishingiz kerak. Bu funktsiyaga o'xshaydi sinfning yangi namunasini qaytaradigan parametrлarsiz.

Keling, misolni, Bob misolini yaratish bilan Square sinfidan foydalanishni ko'rib chiqaylik:

```
>>> Bob = Square() # Bob - Square misoli.
>>> Bob.side #Keling, tomonning qiymatini ko'rib chiqaylik
```

1

<sup>1</sup>Shuningdek, ishslash uchun ushbu konvensiyaga bog'liq bo'lgan kod analizatorlari ham mavjud.

**Bob misolining atribut tomonining qiymatini o'zgartirish mumkin:**

```
>>> Bob.side = 5 #Yanga yangi qiymatni belgilash
>>> Bob.side #Keling, tomonning yangi qiymatini ko'rib chiqaylik
5
```

Bu o'zgarish Bob misoli uchun xosdir. Yangi misollar yaratilganda, yon qiymatni yangisiga belgilash uchun `__init__` usuli yana bajariladi. misol:

```
>>> Krusty = Kvadrat()
>>> Krusty.side 1
```

Agar o'zgaruvchi tomoni sinfning barcha misollaridan foydalanish mumkin bo'lgan o'zgaruvchi bo'lsa, sinf o'zgaruvchisidan foydalanish tavsiya etiladi. Ushbu o'zgaruvchilar bir xil sinfning barcha ob'ektlari tomonidan taqsimlanadi.

**Kvadrat sinf:**

```
tomoni = 1
```

Shunday qilib, tomonning **qiymati biz misol yaratishdan oldin ham aniqlanadi**  
Kvadrat:

```
>>> Kvadrat tomoni
1
```

Albatta, agar biz Square misollarini yaratsak, ular ham ushbu tomon qiymatiga ega bo'ladi:

```
>>> Qisqichbaqa = Kvadrat()
>>> Crab.side
1
```

Sinf o'zgaruvchilari misollar haqida ma'lumotga ega bo'lishi mumkin. Masalan, sinfning nechta nusxasi yaratilganligini nazorat qilish uchun ulardan foydalanish mumkin.

```
class Square:
    count = 0
    def __init__(self):
        Square.count += 1
        print("Ob'ekt muvaffaqiyatlari yaratildi")
```

Kvadratning ushbu versiyasi yaratilgan misollar sonini hisoblashi mumkin. E'tibor bering, count o'zgaruvchisi o'zini self.name prefiksi bilan qayd etilgan misol o'zgaruvchisidan farqlash uchun sinf ichida Square.count sifatida qo'shiladi .

Keling, ushbu ob'ekt qanday ishlatalishini ko'rib chiqaylik:

```
>>> Bob = kvadrat()
Ob'ekt muvaffaqiyatlari yaratildi >>>
Patrik = Square()
Ob'ekt muvaffaqiyatlari yaratildi >>>
Square.count 2
```

Keling, boshqa sinfni ko'rib chiqaylik:

**sinf Ketma-ketligi:**

```
transcription_table = {'A':'U', 'T':'A', 'C':'G'} def __init__(self, , 'G':'C')
seqstring): self.seqstring = seqstring.upper() def
transkripsiya(self):
    tt =
    self.seqstringdagi harf uchun: agar
        'ATCG'dagi harf:
            tt += self.transscription_table[harf]
    qaytish tt
```

Bu sinf ikkita usul va bitta atributga ega. `__init__` usuli har bir misolda seqstring qiyamatini belgilash uchun ishlataladi :

```
>>> xavfli_virus = Ketma-ket ('atggagagccttgttggtgtcaa') >>>
xavfli_virus.seqstring 'ATGGAGAGCCTTGTCTTGGGTCAA'
```

```
>>> zararsiz_virus = Sequence('aatgctactactattagtagaaattgatgcca') >>>
zararsiz_virus.seqstring 'AATGCTACTATTAGTAGAATTGATGCCA'
```

Sequence sinfida transkripsiya deb ataladigan usul ham mavjud bo'lib, uning yagona parametri misolning o'zi (`self` bilan ifodalanadi). Funktsiya chaqirilganda bu parametr ko'rinxaydi, chunki u yashirindir. E'tibor bering, transkripsiya funksiyasi ketma-ketlik qatorini transkripsiya ekvivalentiga aylantirish uchun transkripsiya\_jadvalining (ya'ni lug'at) sinf o'zgaruvchisidan foydalanadi :

```
>>> xavfli_virus.transcription()
'GCUAAGAGCUCGCGUCCUCAGAGUUUAGGA'
```

Usullar ham parametrlarga ega bo'lishi mumkin. Buni ko'rsatish uchun Sequence sinfida yangi usul (cheklov) mavjud . Bu usul ma'lum bir ferment uchun ketma-ketlikda qancha cheklash joylari borligini hisoblab chiqadi.2 Shuning uchun bu usul

---

**2A cheklash fermenti ma'lum bir DNK ketma-ketligini taniyadigan va tanib olish zonasida kesik hosil qiluvchi oqsildir.**

parametr sifatida cheklovchi ferment nomini talab qiladi. Yana bir farq shundaki, bu sinf ferment nomini tanib olish ketma-ketligi bilan bog'laydigan lug'atni o'z ichiga oladi:

---

**Listing 8.1: seqclass.py: Sequence klassi**


---

sinf ketma-ketligi:

```
transcription_table = {'A':'U', 'T':'A', 'C':'G'} enz_dict = , 'G':'C'}
{'EcoRI':'GAATTC', 'EcoRV':'GATATC'} def __init__ (self,
seqstring): self.seqstring = seqstring.upper() def cheklovlar(self,
enz): urinib ko'ring: enz_target = Sequence.enz_dict[enz]
self.seqstring.count(enz_target) qaytish KeyErrordan tashqari:
0 qaytaring
```

```
def transkripsiysi (o'z-o'zidan):
    tt =
        self.seqstringdagi harf uchun:
            agar "ATCG" da harf bo'lса:
                tt += self.transscription_table[harf]
    qaytish tt
```

---

Sequence sinfidan foydalanish:

```
>>> other_virus = Sequence('atgatatcgggagaggatatcggtgtcaa') >>>
other_virus.restriction('EcoRV')
2
```

---

## 8.4 MEROS

---

Sinflarning merosi yangi (bola) sinf asosiy sinfning usullari va atributlarini "meros olishini" anglatadi. Quyidagi sintaksis boshqa sinfdan meros bo'lgan sinf yaratish uchun ishlataladi:

sinf DerivedClass(BaseClass): [tana]

Sut emizuvchilar sinfidan kelib chiqqan Orca sinfiga misol keltiramiz :

---

**Listing 8.2: orca.py: Orca klassi**


---

**sinf Orca (sutemizuvchilar):**

```
"""Sinf tavsifi bilan docstring"""
# Xususiyatlар bu yerda
# Usullar bu yerda
```

Misol sifatida Sequence sinfiga asoslangan Plasmid3 deb nomlangan sinfni ko'rib chiqaylik . Plazmid DNK ketma-ketligining bir turi bo'lganligi sababli, biz ketma-ketlikdan usullar va xususiyatlarni meros qilib olgan Plazmid sinfini yaratdik. Shuningdek, biz AbResDict va ABres kabi ushbu yangi sinfga xos usullar va atributlarni aniqladik. ABres usuli bizning plazmidimiz ma'lum bir antibiotikga chidamliliginini bilish uchun ishlatalidi, AbResDict atributi esa turli antibiotiklarga qarshilikni tavsiflovchi hududlar haqida ma'lumotga ega.

#### **Listing 8.3: plasmid.py: Plazmid sinfi**

**Plazmid sinfi (ketma-ket):**

```
ab_res_dict = {'Tet':'ctagcat', 'Amp':'CACTACTG'} def
__init__(self, seqstring): Sequence.__init__(self, seqstring) def
    ab_res(self, ab):

        agar self.ab_res_dict[ab] self.seqstringda:
            Qaytish Haqiqiy
        False ni qaytaring
```

E'tibor bering, Plazmidning \_\_init\_\_ usuli doirasida biz ketma-ketlikning \_\_init\_\_ usulini chaqirdik. Bu bizning sinfimiz "ota" sinfining atributlari va usullarini meros qilib oladi. Keling, Plazmid sinfi o'zining va otasining usullaridan qanday foydalinishini ko'rib chiqaylik (Sequence). ABres usuli Cheklovga o'xshash tarzda ishlaydi, farqi shundaki, u biz izlayotgan pozitsiyani qaytarish o'rniiga, u mavjud yoki yo'qligini bizga ma'lum qiladi.

#### **Ba'zi Biopython ob'ektlari bilan tanishtirish**

Ushbu kitobda oldinda Biopython uchun maxsus bo'lim mavjud bo'lsa-da, biz ular bilan tanishish uchun bu erda ba'zi Biopython tuzilmalarini ko'ramiz.

**IUPACambiguousDNA klassi:** IUPACambiguousDNA4 klassi IUPAC modulida joylashgan .  
Bu alifboden olingan va DNK ketma-ketligi uchun IUPAC5 tomonidan tasdiqlangan harflar bilan bog'liq shaklga ega bo'lgan sinfdir . Bunda

3A plazmid mikroorganizmning xromosoma DNKsidan mustaqil bo'lgan DNK molekulasiadir.

<sup>4</sup><http://biopython.org/DIST/docs/api/Bio.Alphabet.IUPAC-module.html> ga qarang. shakllantirishda ko'proq uchun.

<sup>5</sup>IUPAC so'f va amaliy kimyo xalqaro ittifoqini anglatadi; bu xalqaro kimyoda qo'llaniladigan nomenklaturani tartibga soluvchi federatsiya.

The IUPAC nucleic acid notation			
	Symbol	Meaning	Mnemonic
DNA Bases	G	Guanine	<u>G</u> uanine
	T	Thymine	<u>T</u> hymine
	A	Adenine	<u>A</u> denine
	C	Cytosine	<u>C</u> ytosine
Ambiguity Characters	R	G + A	p <u>R</u> ine
	Y	T + C	p <u>Y</u> rimidine
	S	G + C	<u>S</u> trong interactions (3 H bonds)
	W	T + A	<u>W</u> eak interactions (2 H bonds)
	K	G + T	<u>K</u> eto
	M	A + C	a <u>M</u> ino
	D	G + T + A	Not-C ( <u>D</u> follows C in alphabet)
	H	T + A + C	Not-G ( <u>H</u> follows G)
	B	G + T + C	Not-A ( <u>B</u> follows A)
	V	G + A + C	Not-T or U ( <u>V</u> follows U)
	N	G + A + T + C	a <u>N</u> y

Shakl 8.1 IUPAC nuklein kislotasi belgilari jadvali.

case (IUPACambiguousDNA) noaniqlik hisobga olinadi, ya'ni ma'lum bir pozitsiyada to'liq aniqlanmagan nukleotidlarni kodlash uchun belgilari mavjud.

Misol uchun, agar ma'lum bir pozitsiyadagi nukleotid A yoki G bo'lishi mumkin bo'lsa, u R bilan kodlanadi ( IUPAC nuklein kislotasining to'liq yozuvlari jadvali uchun [8.1-rasmga](#) qarang). Shuning uchun IUPACambiguousDNA "GATCRYWSMKHBVDN" qatorini o'z ichiga olgan sind o'zgaruvchan harflarga ega. Bir qarashda bu unchalik foydali sind emasdek ko'rindi, lekin Seq sindida uning foydaliligi.

IUPACAbiguousDNK klassi: IUPACAmbiguousDNK singari, IUPACunambiguousDNK mavjud. Bu sind avvalgisidan kelib chiqadi, shuning uchun u o'z xususiyatlarini saqlab qoladi. Yagona farq shundaki, bu sind yana harflar atributini belgilaydi, mazmun sifatida "GATC".

6 Class Seq: Seq modulida Seq deb nomlangan sind mavjud. Ushbu sindagi ob'ektlar ketma-ketlik ma'lumotlarini saqlaydi. Shu nuqtaga qadar biz ketma-ketliklarni satrlar sifatida taqdim etdik. Ushbu yondashuv bilan bog'liq muammo shundaki, satr faqat ketma-ketlik ma'lumotlarini saqlaydi va uning qanday ketma-ketlik (DNK, RNK, aminokislotalar) ekanligini aytadigan metama'lumotlar yo'q. Seq sindida ikkita parametr

<sup>6</sup><http://biopython.org/DIST/docs/api/Bio.Seq.Seq-class.html> ga qarang. qo'shimcha ma'lumot uchun.

va alifbo. Ma'lumotlar ketma-ketlikka ega bo'lgan satr va alifbo alifbo tipidagi ob'ektdir. Unda ketma-ketlik alifbosи turi haqida ma'lumotlar mavjud. Bu klassning yana bir xususiyati shundaki, u "o'zgarmas", ya'ni ketma-ketlik aniqlangandan so'ng uni o'zgartirib bo'lmaydi (faqat satr kabi). Shunday qilib, biz ketma-ketlik bir nechta manipulyatsiyalardan keyin ham bir xil bo'lib qolishiga ishonch hosil qilamiz. Tartibni o'zgartirish uchun biz `MutableSeq` ob'ektidan foydalanishimiz kerak.

Seq klassi eng muhim sifatida bir nechta usullarni belgilaydi: to'ldiruvchi (`to'ldiruvchi` ketma-ketlikni qaytaradi), `reverse_complement` (`teskari` to'ldiruvchi ketma-ketlikni qaytaradi), `tomutable` (`MutableSeq` ob'ektini qaytaradi) va `tostring` (`ketma-ketlikni qator sifatida` qaytaradi). Keling, buni amalda ko'rib chiqaylik: 7

```
>>> Bio.Alphabet dan import IUPAC >>>
Bio.Seq import Seq >>> first_seq =
Seq('GCTATGCAGC', IUPAC.unambiguous_dna) >>> first_seq
Seq('GCTATGCAGC', IUPACUNambiguousDNA()) >>> first_seq.complement()
```

```
Seq('CGATACGTCG', IUPACUNambiguousDNA())
>>> first_seq.tostring()
"GCTATGCAGC"
```

Bu obyekt dasturchiga a bilan ishslash imkonini beruvchi maxsus usullarga ega Seq tipidagi ob'ekt xuddi satr kabi:

```
>>> first_seq[:10] # ketma-ketlikni kesib oling
Seq('GCTAT', IUPACUNambiguousDNA()) >>>
len(first_seq) # 10-ketlik uzunligini oling

>>> first_seq[0] # bitta belgi olish
'G'
```

Sinf `MutableSeq`: Bu Seq ga juda o'xshash ob'ekt bo'lib, asosiy farqi uning ketma-ketligini o'zgartirish mumkin. U Seq bilan bir xil usullarga ega, ba'zi usullar o'zgaruvchan ketma-ketliklarni boshqarish uchun moslashtirilgan.

Biz uni noldan yaratishimiz mumkin yoki uni tomutable usuli yordamida Seq ob'ektidan yaratishimiz mumkin :

```
>>> first_seq
Seq('GCTATGCAGC', IUPACUaniguousDNA())
>>> AnotherSeq=first_seq.tomutable()
>>> AnotherSeq.extend("TTTTTTTT")
```

```
>>> chop etish (Boshqa qator)
MutableSeq('GCTATGCAGCTTTTTT', IUPACUanbiguousDNA())
>>> AnotherSeq.pop()
'T'
>>> AnotherSeq.pop()
'T'
>>> chop etish (Boshqa qator)
MutableSeq('GCTATGCAGCTTTT', IUPACUanbiguousDNA())
```

## 8.5 MAXSUS USULLAR

---

Ba'zi usullar alohida ma'noga ega. Biz allaqachon har safar yangi namuna yaratilganda (yoki yangi ob'ekt yaratilganda) bajariladigan `__init__` usulini ko'rdir. Har bir maxsus usul oldindan belgilangan shartda bajariladi.

Ishlab chiquvchi ob'ekt ushbu oldindan o'rnatilgan shartlarning har biriga qanday javob berishini o'zgartirishi mumkin.

Masalan, `__len__` usulini olaylik. Ushbu usul ob'ektda `len(nasol)` funksiyasi har safar chaqirilganda faollashtiriladi. Ushbu usul nimani qaytarishi ishlab chiquvchiga bog'liq. Sequence sinfini eslang ([8.1 ro'yxati](#)) va ketma-ketlikning uzunligini bilmoqchi bo'lganingizda nima sodir bo'lismeni ko'ring:

```
>>> len(Sequence("ACGACTCTCGACGGCATCCACCCTTGAGA"))
```

Traceback (eng oxirgi qo'ng'iroq):

```
Fayl "<stdin>", 1-qator, <modul> da
AttributeError: ketma-ketlik misolida " __len__ " atributi yo'q
```

Bu qandaydir tarzda kutilgan edi. Biz Sequence uzunligi nimani anglatishini aniqlamadik. Bu ob'ekt bir nechta atributlarga ega va tarjimon `len(Sequence)` talab qilinganda qaysi atribut qaytishini bilishning imkoniy yo'q. Xato xabari bizga muammo haqida ma'lumot beradi: "Sequence instance" atributi yo'q "`__len__`". Shunday qilib, agar biz `len()` funksiyasi uchun xatti-harakatni o'rnatmoqchi bo'lsak, `__len__` maxsus metod atributini aniqlashimiz kerak:

```
def __len__(self):
    qaytarish len (self.seqstring)
```

Ushbu usul sinf ta'rifiiga kiritilishi kerak ([8.1](#)).

**Listing 8.4: seqclass2.py: Sequence klassi**

---

```
sinf ketma-ketligi:
    transcription_table = {'A':'U', 'T':'A', 'C':'G' enz_dict =
                           , 'G':'C'}
    {'EcoRI':'GAATTC', 'EcoRV':'GATATC'}
```

## 150 Bioinformatika uchun Python

```

def __init__(self, seqstring): self.seqstring
    = seqstring.upper() def __len__(self): qaytish
len (self.seqstring)

def cheklash(self, enz): urinib
    ko'ring: enz_target =
        Sequence.enz_dict[enz] qaytish
        self.seqstring.count(enz_target) KeyError bundan
    mustasno: qaytish 0

def transkripsiysi (o'z-o'zidan):
    tt =
        self.seqstringdagi harf uchun: agar
        'ATCG'dagi harf:
            tt += self.transscription_table[harf]
    qaytish tt

```

---

Endi biz \_\_len\_\_ usulini aniqladik, biz funktsiyani qo'llashimiz mumkin  
Sequence obyektlariga len :

```

>>> M13 = Sequence("ACGACTCTCGACGGCATCCACCCTCTGAGA")
>>> len(M13)
32

```

Xuddi shu tarzda, biz len() tomonidan qaytariladigan narsalarni boshqarishimiz mumkin, biz buni sinfda dasturlashtirilishi mumkin bo'lgan boshqa usullar bilan amalga oshirishimiz mumkin. Keling, ulardan ba'zilarini ko'rib chiqaylik.

- \_\_str\_\_ Bu usul ob'ektning satrli tasviri zarur bo'lganda chaqiriladi. Bu tasvir str(obyekt) yoki chop etish obyekti bilan olinadi. Shunday qilib, dasturchi o'z ob'ekti qanday "ko'rinishini" tanlashi mumkin. Masalan, Biopython tomonidan taqdim etilgan tarjima jadvali, Bio.Data.CodonTable, lug'at sifatida saqlanadi, lekin uning ko'rinishi jadval sifatida ko'rinaldi:

```

>>> import Bio.Data.CodonTable >>>
chop etish (Bio.Data.CodonTable.standard_dna_table)
1-jadval Standart, SGC0

```

T	C	A	G	
T   TTT F   TCT S		TAT Y   TGT C   T		
T   TTC F	TCC S	TAC Y	TGC C	C
T   TTA L	TCA S	TAA Stop  TGA Stop  A		

<sup>8</sup><https://docs.python.org/3.6/reference/datamodel.html#special-method-names> da U yerda maxsus usullar ro'yxati.

T   TTG L(lar)	TCG S	TAG Stop  TGG W   G
C   CTT L	CCT P	CAT H   CGT R   T
C   CTC L	CCC P	CAC H   CGC R   C
C   CTA L	CCA P	CAA Q   CGA R   A
C   CTG L(lar)	CCG P	CAG Q   CGG R   G
A   ATT I	ACT T	AAT N   AGT S   T
A   ATC I	ACC T   AAC N   AGC S	C
A   ATA I	ACA T   AAA K   AGA R   A	
A   ATG M(lar)	ACG T   AAG K   AGG R   G	
G   GTT V	GCT A	GAT D   GGT G   T
G   GTC V	GCC A   GAC D	GGC G   C
G   GTA V	GCA A	GAA E   GGA G   A
G   GTG V	GCG A	GAG E   GGG G   G

- `__repr__` o'rnatilgan `repr()` funksiyasi bilan va ob'ekt interaktiv qobiqqa kiritilganda chaqiriladi. U bir xil qiymatga ega bo'lgan ob'ektni qayta yaratish uchun ishlatalishi mumkin bo'lgan haqiqiy Python ifodasiga o'xshab ko'rinishi kerak, agar imkonim bo'lmasa, <...ba'zi foydali tavsif...> shaklidagi qator. U asosan disk raskadrovdaka qo'llaniladi. Yuqoridagi kabi bir xil ob'ektga qarang, lekin `print()` o'rniغا `repr()` bilan:

```
>>> repr(Bio.Data.CodonTable.standard_dna_table)
'<Bio.Data.CodonTable.NCBICodonTableDNA nusxasi 0xb7da0c>'
```

- `__getitem__` ob'ektga ketma-ket kirish uchun yoki `ob'ekt[n]` kabi pastki skript yordamida foydalaniladi. Har safar ob'ektga `ob'ekt [n]` sifatida kirishga harakat qilganingizda, `ob'ekt.__getitem__(n)` bajariladi. Bu usul ikkita parametrni talab qiladi: `ob'ekt (odatda o'zini ) va indeks.` **8.6 ro'yxatda** foydalanish namunasi mavjud .
- `__iter__` ketma-ketlik ustida yurish imkonini beradi. `__iter__` yordamida biz lug'atlar, ro'yxatlar, fayllar, satrlar va boshqalar kabi ko'plab turli ob'ektlarni bir xil tarzda takrorlashimiz mumkin. For bayonoti takrorlanayotgan ob'ektga o'rnatilgan funksiya iterini chaqiradi. `__iter__ __next__` maxsus usulidan foydalanganda elementlar qanday qaytarilishini belgilaydi. Birinchi misolda **biz Straight sinfini yaratamiz**, bu erda uning elementlari saqlanadigan tartibda qaytariladi, Reverse **klassi esa o'z elementlarini teskari tartib** yordamida qaytaradi:

**Listing 8.5: straight.py: To'g'ri va teskari sinflar**

---

To'g'ridan-to'g'ri sinf:

## 152 Bioinformatika uchun Python

```

def __init__(self, data): self.data
    = data
    self.index = 0
def __iter__(o'zini):
    o'zini qaytarish
def __keyingi__(self):
    agar self.index == len(self.data):
        StopIteration ko'taring
    javob = self.data[self.index]
    self.index += 1
    javob qaytarish

```

Teskari sinf:

```

def __init__(self, data): self.data
    = data
    self.index = len (ma'lumotlar)
def __iter__(self):
    self def
    __keyingi__(self) ni qaytarish:
        agar self.index == 0:
            StopIteration self.index
            ni ko'taring -= 1
            self.data[self.index]ni qaytarish

```

---

Keling, ularni amalda ko'rib chiqaylik:

>>> a = Straight("123") >>> a  
in i uchun:

chop etish (i)

```

1
2
3
>>> b = teskari ("123")
>>> i in b uchun:
    chop etish (i)

3
2
1

```

- \_\_setitem\_\_ kalitga qiymat berish uchun ishlatiladi ( `self[key]=value` shakli bilan). Odatda lug'at kalitining qiymatini o'zgartirish uchun ishlatiladi. Bu holda u satrdagi belgini almashtirish uchun ishlatiladi:

```

def __setitem__(self, kalit, qiymat): agar
    len(qiymat) == 1: self.seq = self.seq[:key]
        + qiymat + self.seq[key+1:] qaytish Yo'q

boshqa:
ValueError-ni ko'tarish

```

- \_\_delitem\_\_ self[kalit] shaklidagi ob'ektlarni o'chirishni amalga oshiradi . U o'z elementlarini o'chirishni qo'llab-quvvatlaydigan har qanday ob'ekt bilan ishlatalishi mumkin.

Ba'zi maxsus usullar bilan ketma-ketlik klassi:

**Listing 8.6: seqwitsm.py: Maxsus usullar atributlari bilan ketma-ketlik klassi**

---

sinf ketma-ketligi:

```

transcription_table = {'A':'U', 'T':'A', 'C':'G', 'G':'C'} comp_table = {'A':'T', 'T':'A', 'C':'G', 'G':'C'}
def __init__(self, seqstring): self.seqstring =
seqstring.upper() def cheklovlari (self, enz): enz_dict = {'EcoRI':'ACTGG',
'EcoRV':'AGTGC'} urinib ko'ring: target = enz_dict[enz] bundan mustasno
KeyError: oshirish ValueError('Enzim ma'lumotlar bazasida bunday ferment
yo'q') qaytish self.seqstring.count(target) def __getitem__(self, indeks):

```

```

    return self.seqstring[index] def
__getslice__(self, low, high): return
    self.seqstring[low:high] def __len__(self):
return len(self.seqstring) def __str__(self): if
    len(self. seqstring) >= 28: qaytarish '{0}...
{1}'.format(self.seqstring[:25], self.seqstring[-3:])

```

```

boshqa:
    self.seqstring def
transkripsiyasini qaytarish(self):
    tt =
        self.seqstringda x uchun: agar
        x "ATCG" da: tt +=
            self.transcription_table[x]
qaytish tt

```

```

def to'ldiruvchi (o'zini):
    tt =
        self.seqstring da x uchun: agar
        x "ATCG" da:
            tt += self.comp_table[x]
qaytish tt

```

---

### 8.5.1 O'rnatilgan ma'lumotlar turidan foydalanib yangi ma'lumotlar turini yaratish

Biz o'rnatilgan ma'lumotlar turlaridan olingan o'z sinflarimizni yaratishimiz mumkin. Ushbu fikrni tushuntirish uchun dict turining variantini qanday yaratishni ko'ring . Zdict - lug'atga o'xhash ob'ekt; u bitta farq bilan lug'at kabi ishlaydi: mavjud bo'limgan kalit bilan qiymatni olishga urinayotganda istisno ko'tarish o'rniiga 0 (nol) ni qaytaradi.

#### Listing 8.7: zdict.py: Lug'at sinfini kengaytirish

---

```

1-sinf Zdic(dict):
2     """ Lug'atga o'xhash ob'ekt, foydalanuvchi qachon 0 qaytaradi
3     """ mavjud bo'limgan kalitni so'rash.
4
5
6     def __missing__(self,x):
7         0 qaytaring

```

Kod tushuntirishi: 1-qatorda biz sinf nomini beramiz va ma'lumotlar turini ar gument (dict) sifatida beramiz. Bu shuni anglatadiki, hosil bo'lgan sinf (Zdic) dict turidan meros oladi. 7-dan 8-qatorgacha maxsus usulning ta'rifi mavjud: \_\_missing\_\_. Ushbu usul foydalanuvchi mavjud bo'limgan kalit bilan qiymatni olishga harakat qilganda ishga tushiriladi. Argument sifatida kalitning qiymatini oladi, lekin bu holda dastur bunday qiymatdan (x) foydalanmaydi, chunki u kalit qiymatini hisobga olmaganda 0 ni qayta

```

>>> a = Zdic() >>>
a['blue'] = 'azul' >>> a['red']

0

```

## 8.6 KODIMIZNI XUSUSIY QILISH

---

Ushbu bobning boshida OOPning xususiyatlaridan biri inkapsulyatsiya ekanligi ta'kidlangan. Inkapsulyatsiya - dasturchilarning ob'ektlarning ichki ishlashiga e'tibor bermasliklari va faqat ularning mavjud usullarini ko'rishlari.

Biz yaratadigan ushbu usullarning ba'zilari "tashqi iste'mol" uchun bo'lmaydi, lekin ular biz xohlagan sinfning boshqa usullarini qo'llab-quvvatlaydi.

dasturning boshqa bo'limlaridan foydalanish uchun. Ba'zi tillar usullar va xususiyatlarni yashirishga imkon beradi. Jargonda bu "uslubni shaxsiy qilish" deb ataladi.

Python usulni yashirishga yo'l qo'ymaydi, chunki u dasturchining yo'liga to'sqinlik qilmaslik uning binolaridan biridir. Ammo uning sintaksisi mavjud bo'lib, u sinfdan tashqaridagi usul yoki xususiyatga kirishni qiyinlashtiradi. Bu mangling deb ataladi va uning sintaksisi biz maxfiy bo'lishni xohlagan usul yoki atribut nomining boshida (lekin oxirida emas) ikkita pastki chiziq qo'shishdan iborat. Keling, ikkita usulni, a va `__b` ni belgilaydigan sinf misolini ko'rib chiqaylik:

Sinf sinfi:

```
""" "Shaxsiy" metodli sinf (b) """ def a(self):
```

```
    pass
def __b(self): #
    _TestClass__b ga o'tish
    o'tish
```

`__b()` ga kirishga urinish xatoga olib keladi:

```
>>> my_object = TestClass() >>>
my_object.a() >>> my_object.__b()
```

Traceback (eng oxirgi qo'ng'iroq):

```
Fayl "<pyshell#14>", 1-qator, <modul> my_object.__b()
```

AttributeError: TestClass misolida "`__b`" atributi yo'q

A usuliga kirish mumkin, lekin `__b` emas, hech bo'limganda bevosita emas. Belgilash ob'ekti.`_Class__method` ishlatalishi kerak. Masalan:

```
>>> my_object._TestClass__b()
```

Shaxsiy ittifoqdosh bo'limgan maxfiylik usulidan maqsad nima ekanligiga hayron bo'lishingiz mumkin. Bir tomondan, ushbu "yarim himoya" ga ega bo'lgan usullar bolalar sinflari tomonidan meros qilib olingan / bog'langan (va nom maydoni ifloslanmagan). Boshqa tomondan, ob'ekt dir yordamida o'rganilganda, ob'ektlarning bu sinfi ko'rinxaydi. Ko'rib chiqilishi kerak bo'lgan muhim narsa shundaki, ushbu belgi tomonidan taqdim etilgan himoya samarali himoyadan ko'ra ko'proq davom etish bo'yich

## 8.7 QO'SHIMCHA RESURSLAR

---

- Python dasturlash/OOP.

[http://en.wikibooks.org/wiki/Python\\_Programming/OOP](http://en.wikibooks.org/wiki/Python_Programming/OOP)

## 156 Bioinformatika uchun Python

- Python bilan OOP ga kirish. <http://www.voidspace.org.uk/python/articles/OOP.shtml>
- Pythonga sho'ng'in, Mark Pilgrim. 5 -[bob](http://diveintopython.org/object_oriented_framework), Ob'ektlar va ob'ektlar Orientatsiya.  
[http://diveintopython.org/object\\_oriented\\_framework](http://diveintopython.org/object_oriented_framework)
- Python ob'ektlari, Fredrik Lund tomonidan.  
<http://www.effbot.org/zone/python-objects.htm>
- Java darsligi: dars: ob'ektga yo'naltirilgan dasturlash tushunchalari.  
<http://java.sun.com/docs/books/tutorial/java/concepts/>
- Seq obyekti:  
<http://biopython.org/wiki/Seq>

### 8.8 O'Z-O'ZINI BAHOLASH

---

1. Nima uchun Python ko'pincha ko'p paradigmali til sifatida tavsiflanadi?
2. Ob'ektga yo'naltirilgan dasturlashning (OOP) asosiy xarakteristikalarini ayting.
3. Quyidagi tushunchalarni izohlang: Meros, Inkapsulyatsiya va Polimorfizma.
4. Sinf atributlari va misol atributlari o'rtaqidagi farq nima?
5. Maxsus usul atributi nima? Kamida to'rttasini nomlang.
6. `__str__` va `__repr__` o'rtaqidagi farq nima?
7. Shaxsiy usul nima? Ular haqiqatan ham Pythonda shaxsiymi?
8. O'zlik nima? Sinf ta'rifi ichida `self.var` o'rtaqidagi farq nima  
= 0 va var = 0?
9. Qancha misollar yaratilganligini kuzatib boruvchi sinfni aniqlang.
10. O'rnatilgan turga asoslanib yangi turni aniqlang.

# Biopythonga kirish

---

## MAZMUNI

<b>9.1 Biopython nima? .....</b>	<b>158</b>	<b>9.1.1 Loyihani tashkil etish .....</b>	<b>158</b>
<b>9.2 Biopython dasturini o'rnatish. ....</b>		<b>159 macOS/Linux tizimida .....</b>	<b>159</b>
		<b>Windowsda. ....</b>	<b>162</b>
<b>9.3 Biopython komponentlari .....</b>	<b>162</b>		
<b>9.3.1 Alifbo. ....</b>	<b>162</b>	<b>9.3.2 Seq .....</b>	<b>163</b>
<b>Seq .....</b>		<b>Seq</b>	
<b>Ob'ektlar satr sifatida .....</b>		<b>165</b>	
<b>9.3.3 MutableSeq. ....</b>		<b>165</b>	
<b>9.3.4 SeqRecord .....</b>		<b>166</b>	
<b>9.3.5 Tegishlash. ....</b>			
<b>167 9.3.6 AlignIO. ....</b>			
<b>169 AlignInfo .....</b>			
<b>170 9.3.7 ClustalW. ....</b>			
<b>171 Parametrlarni ClustalW ga o'tkazish. ....</b>			
<b>173 9.3.8 SeqIO. ....</b>			
<b>173 Ketma-ket fayllarni o'qish .....</b>			
<b>174 Fayllarni ketma-ket yozish. ....</b>			
<b>175 9.3.9 AlignIO. ....</b>			
<b>176 9.3.10 PORTLASH .....</b>			
<b>177 BLAST Biopython bilan ishlash va qayta ishslash .....</b>	<b>178</b>		
<b>BLAST ishni boshlash. ....</b>	<b>178</b>		
<b>BLAST chiqishini o'qish .....</b>		<b>180</b>	
<b>BLAST Record obyektida nima bor? .....</b>		<b>180</b>	
<b>9.3.11 Biologik bog'lilq ma'lumotlar. ....</b>			
<b>187 9.3.12 Entrez .....</b>			
<b>Bir qarashda 190 eUtils .....</b>			
<b>190 Biopython va eUtils .....</b>			
<b>191 eUtils: Bibliografiyani olish. ....</b>		<b>191</b>	
<b>eUtils: Gen ma'lumotlarini olish. ....</b>		<b>192</b>	<b>9.3.13 Bio.PDB</b>
<b>PDB. ....</b>		<b>194</b>	
<b>moduli. ....</b>		<b>195</b>	<b>9.3.14 PROZIT</b>
<b>PROZIT .....</b>		<b>196</b>	<b>9.3.15</b>
<b>Cheklash. ....</b>		<b>197</b>	
<b>Bio.Cheklash moduli .....</b>			<b>198</b>

Tahlil sinfi: Hammasi bitta .. . . . .	<b>199</b>	9.3.16
SeqUtils .. . . . .	<b>200</b>	DNK
Utils. .... . . . .	<b>200</b>	Protein
Utils. .... . . . .	<b>202</b>	9.3.17 Ketma-ketlik.
fayllari .. . . . .	<b>202</b>	PhD
fayllari .. . . . .	<b>203</b>	Ace
SwissProt. .... . . . .	<b>203</b>	9.3.18
Xulosa .. . . . .	<b>205</b>	9.4
manbalar .. . . . .	<b>207</b>	9.5 Qo'shimcha
baholash. .... . . . .	<b>209</b>	207 9.6 O'z-o'zini

## 9.1 BIOPITON NIMA?

---

Biopython1 - bioinformatika dasturlarini ishlab chiqish uchun foydali modullar to'plami. Har bir bioinformatika tahlili o'ziga xos bo'lsa-da, takrorlanadigan ba'zi vazifalar, dasturlar va standart fayl formatlari o'rtasida almashinadigan doimiylar mavjud. Bu holat biologik muammolarni hal qilish uchun paket zarurligini ko'rsatadi.

Biopython g'oya sifatida 1999 yil avgust oyida boshlangan; bu Jeff Chang va Endryu Dalke tashabbusi edi. Ular bu g'oyani o'ylab topishgan bo'lsa-da, tez orada hamkorlar loyihaga qo'shilishdi. Eng faol ishlab chiquvchilar orasida Bred Chapman, Peter Kok, Mishel de Xun va Iddo Fridberg ajralib turadi. Loyerha 2000-yilning fevral oyida kod shaklini olishni boshladi va o'sha yilning iyul oyida birinchi nashri chiqarildi. Asl g'oya BioPerl-ga ekvivalent paketni yaratish edi, u o'sha paytda bioinformatikaning asosiy to'plami edi. BioPerl Biopythonning ilhomni bo'lishi mumkin bo'lsa-da, Perl va Python o'rtaqidagi kontseptual farqlar Biopythonga ishlarni bajarishning o'ziga xos usulini berdi. Biopython ochiq biologiyalar oilasining bir qismidir (shuningdek, Bio<sup>\*</sup> nomi bilan ham tanilgan), u institutsional jihatdan Ochiq Bioinformatika jamg'ýarmasining aýzosi hisoblanadi.<sup>2</sup>

### 9.1.1 Loyihani tashkil etish

Bu ochiq manbali hamjamiyat loyihasidir. Ochiq Bioinformatika jamg'armasi ma'muriy, iqtisodiy va huquqiy jihatlarga g'amxo'rlik qilsa-da, uning mazmuni dasturiy ta'minot ishlab chiquvchilari va foydalanuvchilari tomonidan boshqariladi.

Kod jamoat mulki hisoblanadi va uning Github omborida <https://github.com/biopython/> biopython manzilida mavjud . Loyihada har kim ishtirok etishi mumkin. Biopython-da hamkorlik qilish uchun amal qilish kerak bo'lgan protsedura boshqa ochiq manbali loyihalarga o'xshaydi. Siz dasturiy ta'mindan foydalanishingiz kerak, keyin esa o'zimniki unga qo'shimcha funksiyalar kerakmi yoki mavjud xususiyatlardan birini o'zgartirishni xohlaysizmi yoki yo'qligini aniqlashingiz kerak. Har qanday kod yozishdan oldin, mening tavsiyam sizning fikringizni muhokama qilishdir

<sup>1</sup> <http://www.biopython.org> saytida mavjud .

<sup>2</sup> <http://www.open-bio.org>

rivojlanish pochta ro'yxati<sup>3</sup> birinchi. U erda siz ushbu xususiyat allaqachon muhokama qilinganligini va rad etilganligini yoki o'sha vaqtgacha hech kimga kerak bo'limgani uchun kiritilmaganligini bilib olasiz. Xatolar tuzatilgan bo'lса, so'rashingiz shart emas; shunchaki muammo kuzatuvchisida xabar bering,<sup>4</sup> va iloji bo'lса, yechim taklifini qo'shing.

Loyihaning ochiq tabiatи tufayli, o'nlab odamlar bioinformatikaning turli sohalaridan, axborot nazariyasidan populyatsiya genetikasigacha bo'lган kodlarni o'z hissalarini qo'shdilar.

Men Biopython dasturida 2002 yildan beri foydalanuvchi sifatida ishtirot etdim va birinchi hissamni 2003 yilda ketma-ketlikning mahalliy kompozitsion murakkabligini hisoblash funksiyasi lcc.py bilan topshirdim. 2004 yilda men oligonu kleotidlarining erish nuqtasini hisoblash uchun kodni taqdim etdim. 2007-yilda men CheckSum moduli uchun ba'ysi funksiyalar bilan o'yз hissamni qo'yshdim.<sup>5</sup> Mening so'nggi taqdimnomam Bio.Restriction moduliga (2017) yamoq bo'yldi. Har bir holatda men qo'llab-quvvatlovchi hamjamiyatni topdim, ayniqsa kodlash ko'nikmalarim boshlang'ich darajada bo'lган birinchi topshirishda.

Biopython loyihasida qanday ishtirot etish haqida ko'proq ma'lumot olish uchun, <http://biopython.org/wiki/Contributing> manzilidagi maxsus ko'rsatmalarga qarang .

Biopython kodi Biopython litsenziysi ostida ishlab chiqilgan.<sup>6</sup> U juda erkin va undan foydalanishda deyarli hech qanday cheklovlар yo'q.<sup>7</sup>

## 9.2 BIOPYTHONNI O'R NATISH

---

### MacOS/Linux-da

Quyidagi buyruqlar Biopythonni virtualenv ostida qanday o'rnatishni ko'rsatadi. Virtualenv-ni birinchi o'rnatishing (agar sizda allaqachon mavjud bo'ilmasa):

```
$ pip virtualenv o'rnatish
```

Virtualenv yig'ilmoqda

Keshlangan virtualenv-15.1.0-py2.py3-none-any.whl-dan foydalanish

Yig'ilgan paketlarni o'rnatish: virtualenv Muvaffaqiyatli o'rnatilgan

virtualenv-15.1.0 Siz 8.1.1 pip versiyasidan foydalanyapsiz, ammo 9.0.1

versiyasi mavjud.

Siz "pip install --upgrade pip" <= buyrug'i orqali yangilashni ko'rib chiqishingiz kerak.

```
$ pip install --upgrade pip pip
```

yig'ilmoqda (...)

---

<sup>3</sup>[http://biopython.org/wiki/Mailing\\_lists](http://biopython.org/wiki/Mailing_lists)

<sup>4</sup><https://github.com/biopython/biopython/issues>

<sup>5</sup>Bassi, Sebastian va Gonsales, Virjiniya. Biopython uchun yangi nazorat summasi funksiyalari. Mavju Nature Precedings dan <<http://dx.doi.org/10.1038/npre.2007.278.1>> (2007).

<sup>6</sup>Litsenziya Biopython paketiga kiritilgan va <http://www.biopython.org/DIST/LICENSE>.

<sup>7</sup>Biopython-dan foydalanish uchun qo'yilgan yagona shart mualliflik huquqi to'g'risidagi bildirishnomani nashr etish bilan bog'liq va reklamada hissa qo'shuvchilar nomidan foydalanmaslik.

## 160 Bioinformatika uchun Python

**Biopython uchun virtualenv yarating (bu holda virtualenv py4biovirtualenv deb ataladi)**

```
$ virtualenv py4biovirtualenv '/usr' asosiy
prefksidan foydalanish /home/sb/
py4biovirtualenv/bin/python3 da bajariladigan yangi python. Shuningdek, /home/sb/
py4biovirtualenv/bin/python da bajariladigan fayl yaratish SetupTools, wheels...don o'rnatish, pip.
```

**Virtualenv-ni faollashtiring**

```
$. py4biovirtualenv/bin/activate (py4biovirtualenv)
$
```

**Virtualenv ichida numpy va keyin biopythonni o'mnating**

```
(py4biovirtualenv) $ pip install numpy numpy yig'ilmoqda
(...)
```

**Yig'ilgan paketlar o'rnatilmoxda: numpy Muvaffaqiyatli  
o'rnatildi numpy-1.11.2 (py4biovirtualenv) \$ pip install  
biopython Biopython yig'ilmoqda (...)**

```
Biopython-1.68 (py4biovirtualenv) $ python Python
3.5.2 muvaffaqiyatli o'rnatildi (standart, 2016 yil 17-
noyabr, 17:05:23)
[GCC 5.4.0 20160609] Linuxda Qo'shimcha
ma'lumot uchun "yordam", "mualliflik huquqi", "kreditlar" yoki "litsenziya" ni kriting. >>> Bio import >>>
Bio.__versiya__
```

'1.68'

Agar siz Anaconda-dan foydalanayotgan bo'lsangiz, virtualenv-ni ishlatalish o'rniga, conda create-dan foydalaning.  
Siz buni faqat bir marta qilishingiz kerak.

```
$ conda create -n biopy python Paket
metama'lumotlarini olish: ....
Paket spetsifikatsiyalarini hal qilish: .....
/sb/anaconda3/envs/biopy muhitida o'rnatish uchun paket rejasি:
```

**Quyidagi paketlar yuklab olinadi:**

paket		qurmoq
----- -----		

pip-9.0.1		py35_1	1,7 MB
-----------	--	--------	--------

```
(...)
#
# Ushbu muhitni faollashtirish uchun
quyidagilardan foydalaning: # $ source activate biopsy
#
# Ushbu muhitni o'chirish uchun quyidagilardan
foydalaning: # $ manba o'chirish
#
```

Conda muhiti (virtualenvga teng) yaratilgandan so'ng, sizga kerak  
uni faollashtirish uchun. Har safar atrof-muhitdan foydalanish kerak bo'lganda buni qilasiz:

```
$ manba faollashtirish biopyiani
tashlab, /sb/anaconda3/bin dan PATH oldidan /sb/
anaconda3/envs/biopy/bin PATH (biopiya)$
```

**Konda muhitiga kirganingizdan so'ng, Biopythonni pip o'rninga conda yordamida o'rnatating:**

```
(biopy)$ biopython o'rnatish uchun paket
metama'lumotlarini olish: ....
Paket spetsifikatsiyalarini hal qilish: ....
/sb/anaconda3/envs/biopy muhitida o'rnatish uchun paket rejasি:
```

Quyidagi paketlar yuklab olinadi:

(...)

Quyidagi YANGI paketlar O'R NATILADI:

```
biopython: 1.68-np111py35_0 mkl:
11.3.3-0
numpy: 1.11.2-py35_0
```

Davom etish ([y]/n)?

(...)

Biopython paketi o'rnatilganligini tekshiring:

```
(biopiya)$ python
Python 3.5.2 |Continuum Analytics, Inc.| (standart, 2016 yil 2 iyul)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] Linuxda
```

## 162 Bioinformatika uchun Python

Qo'shimcha ma'lumot uchun "yordam", "mualliflik huquqi", "kreditlar" yoki "litsenziya" ni kiriting. >>> Bio import >>> Bio.\_versiya\_\_

'1,68'

## Windowsda

64-bitli Windows (eng keng tarqalgan Windows arxitekturasi) uchun rasmiy Biopython paketlari mavjud emas, shuning uchun Windows-ni o'rnatish unchalik oson emas. Norasmiy paketni <http://www.lfd.uci.edu/~gohlke/> pythonlibs manzilidan yuklab olish uchun birinchi qadam . Ushbu sahfada juda ko'p fayllar mavjud, shuning uchun diqqat bilan tanlang. U siz foydalanayotgan Python versiyasi va mikroprotsessor arxitekturasiga mos kelishi kerak. 64 bitli mashinada Python 3.6 uchun biopython-1.68-cp36-cp36m-win\_amd64.whl yuklab oling. Fayl yuklab olingandan so'ng, uning qayerda ekanligiga e'tibor bering, chunki keyingi bosqichda sizga kerak bo'ladi. Python 2.7.9 versiyasidan boshlab pip oldindan o'rnatilganligi sababli uni buyruq satridan foydalanishingiz mumkin:

```
c:\Users\sb\AppData\Local\Programs\Python36\Scripts> pip install c:<=
\Users\sb\Downloads\biopython-1.68-cp36-cp36m-win_amd64.whl
```

Bu Biopython dasturini o'rnatadi.

## 9.3 BIOPITON KOMPONENTLARI

---

**Biopython** bir nechta modullarga ega. Ba'zilari ko'pgina molekulyar biologiya laboratoriyalarda har kuni bajariladigan vazifalarni osonlashtiradi, boshqalari esa juda aniq maqsadlarga ega. "Umumiy ishlataladigan" narsa o'quvchining ish muhitiga bog'liq bo'ladi, shuning uchun keyingi kompilyatsiya mening shaxsiy nuqtai nazarimga asoslanadi, men u eng ko'p qo'llaniladi.

Barcha ro'yxtarlarda bo'lgani kabi, bu o'zboshimchalikdir va u barcha o'quvchilarning manfaatlarini aks ettirmasligi mumkin. Birinchi elementlar qolganlarini tushunishga yordam beradi degan maqsadda didaktik tarzda tartiblangan.

### 9.3.1 Alifbo

Bioinformatikada biz alifbolar bilan shug'ullanamiz. DNK 4 harfli alifboga (A, C, T, G) ega, oqsillarda esa 20 ta aminokislotalar mavjud bo'lib, ularning har biri alifbo harfi bilan ifodalanadi. Noma'lum pozitsiyalarni o'ylaydiganlar kabi maxsus "alifbolar" ham mavjud. Bular bir nechta nukleotidlar bo'lishi mumkin bo'lgan pozitsiyalardir.

Masalan, S harfi C yoki G nuklein kislotalarini, H harfi esa A, C yoki T ni ifodalashi mumkin. Biopython tilidagi bu noaniq alifbo ambigu ous\_dna deb ataladi. Proteinlarga kelsak, kengaytirilgan lug'at ham mavjud bo'lib, u odatda oqsillarda topilmaydigan aminokislotalarni o'z ichiga oladi<sup>8</sup> (ExtendedIUPACProtein). Xuddi shunday, uchun kengaytirilgan alifbo mavjud

---

<sup>8</sup>Selenosistein va pirolizin tipik misollardir.

o'zgartirilgan asoslarga ega harflarga ruxsat beruvchi nukleotidlar (ExtendedIUPACDNA). Oqsillarga qaytsak, umumiy fizik-kimyoviy xususiyatlarni hisobga olgan holda bir nechta aminokislotalarni bir harfga birlashtiradigan qisqartirilgan alifbo ham mavjud.

Hatto DNK yoki aminokislotalarga asoslangan bo'limgan bitta alifbo mavjud: SecondarySTuzilmasi. Bu alifbo Helix, Turn, Strand va Coil kabi domenlarni ifodalaydi.

IUPAC tomonidan belgilangan alifbolar Biopython-da IUPAC modulining sinflari sifatida saqlanadi. Ota-modul (Bio.Alphabet) ko'proq umumiy/umumiy holatlarni o'z ichiga oladi. Bu erda alifboning ba'zi atributlari:

```
>>> import Bio.Alphabet >>>
Bio.Alphabet.ThreeLetterProtein.harflar ['Ala', 'Asx', 'Cys',
'Asp', 'Glu', 'Phe', 'Gly', 'Uning', '<= 'Ile', 'Lys', 'Leu', 'Met', 'Asn', 'Pro', 'Gln', 'Arg',
'<= 'Ser', 'Thr', 'Sec', 'Val ', 'Trp', 'Xaa', 'Tyr', 'Glx'] >>> Bio.Alphabet import IUPAC
>>> IUPAC.IUPACProtein.letters
```

```
'ACDEFGHIKLMNPQRSTVWY'
>>> IUPAC.unambiguous_dna.letters 'GATC'

>>> IUPAC.ambiguous_dna.letters
'GATCRYWSMKHBVDN'
>>> IUPAC.ExtendedIUPACProtein.harflar
'ACDEFGHIKLMNPQRSTVWYBXZJUO'
>>> IUPAC.ExtendedIUPACDNA.letters
"GATCBDSW"
```

Alifbolar ketma-ketlikning mazmunini aniqlash uchun ishlataladi. "CCGGGTT" dan tashkil topgan ketma-ketlik bir nechta sistein, glitsin va treonindan iborat kichik peptid yoki sitozin, guanin va timinning DNK qismi ekanligini qaerdan bilasiz? Agar ketma-ketliklar satrlar sifatida saqlangan bo'lisa, uning qanday ketma-ketlik ekanligini bilishning hech qanday usuli bo'lmaydi. Shuning uchun Biopython Seq obyektlarini taqdim etadi.

### 9.3.2 Seq

Bu ob'ekt ketma-ketlikning o'zi va ketma-ketlikning tabiatini belgilaydigan alifbodan iborat.

Keling, DNK fragmenti sifatida ketma-ketlik ob'ektini yarataylik:

```
>>> dan Bio.Seq import Seq >>>
import Bio.Alphabet >>> seq =
Seq('CCGGGTT', Bio.Alphabet.IUPAC.unambiguous_dna)
```

Ushbu ketma-ketlik (seq) DNK sifatida aniqlanganligi sababli, siz DNK ketma-ketlikda ruxsat etilgan operatsiyalarni qo'llashingiz mumkin. Seq ob'ektlari transkripsiya va tarjima qilish usullariga ega:

## 164 Bioinformatika uchun Python

```
>>> seq.transcribe()
Seq('CCGGGUU', IUPACUnambiguousRNA()) >>>
seq.translate()
BiopythonOgohlantirish: qisman kodon, len(ketma-ketlik) th ning karrali emas<=
ree.
```

Tarjima qilishdan oldin ketma-ketlikni aniq qisqartiring yoki N ni qo'shing.<= Bu kelajakda xato bo'lishi mumkin.

```
Seq('PG', IUPACProtein())
```

RNK ketma-ketligini transkripsiya qilib bo'lmaydi, lekin uni tarjima qilish mumkin:

```
>>> rna_seq = Seq('CCGGGUU',Bio.Alphabet.IUPAC.anambiguous_rna) >>>
rna_seq.transcribe()
Traceback (eng oxirgi qo'ng'iroq):
  Fayl "<stdin>", 1-qator, <modul> fayli "/home/
  sb/Seq.py", 520-qator, transkripsiya ValueError("RNKni
  transkripsiya qilib boylmaydi!")
ValueError: RNKni transkripsiya qilib bo'lmaydi!
>>> rna_seq.translate()
Seq('PG', IUPACProtein())
```

**Back\_transcribe** usuli yordamida RNK dan DNK ga qaytishingiz mumkin.

```
>>> rna_seq.back_transcribe()
Seq('CCGGGT', IUPACUnambiguousDNA())
```

Maslahat: Biopython-dagi Transkripsiya funksiyasi.

Esda tutingki, transkripsiya funksiyasi ko'pchilik biologlar kutganidek ishlamasligi mumkin. Bu funksiya ketma-ketlikdagi har bir "T" hodisasini "U" bilan almashtiradi. Biologiyada transkripsiya har bir DNK nukleotidini o'zining komplementar nukleotidiga almashtirish va hosil bo'lgan qatorni teskari aylantirishni anglatadi. Transkripsiya funksiyasi shu tarzda ishlaydi, chunki barcha biologik nashrlar shablon bo'Imagan ipni ko'rsatadi. Biopython siz funksiyaga shablon bo'Imagan qatorni berayotganingizni taxmin qiladi. Bio.Seq modulida shuningdek, Seq obyektlari yoki satrlarida ishlatalishi mumkin bo'lgan transkripsiya, orqaga ko'chirish va tarjima qilish funksiyalari mavjud:

```
>>> Bio.Seq dan import tarjima qilish, transkripsiya qilish, orqaga_ko'chirish >>>
dnaseq = 'ATGGTATAA' >>> translate(dnaseq)
```

'MV\*'

```
>>> transkripsiya (dnaseq)
'AUGGUUAUA'
>>> rnaseq = transkripsiya (dnaseq) >>>
tarjima qilish (rnaseq)
```

```
'MV*'
```

```
>>> back_transcribe(rnaseq)
"ATGGTATAA"
```

---

## Ob'ektlarni qator sifatida

Seq ob'ektlari deyarli string kabi ishlaydi, shuning uchun ko'plab string operatsiyalariga ruxsat beriladi:

```
>>> seq = Seq('CCGGGTTAACGTA',Bio.Alphabet.IUPAC.anambiguous_dna) >>> seq[:5]
```

```
Seq('CCGGG', IUPACUanbiguousDNA()) >>>
len(seq) 13
```

```
>>> chop etish (ketma-ket)
CCGGGTTAACGTA
```

Ushbu xatti-harakat doimiy ravishda rivojlanib boradi, shuning uchun keyingi Biopython relizlarida ko'proq qatorga o'xshash xususiyatlarni kuting.

Agar sizga Seq ob'ektining satrli tasviri kerak bo'lsa, Python o'rnatilgan str() funksiyasidan foydalaningiz mumkin. Hali ham ishlayotgan tostring() usuli ham mavjud, lekin kodingizni eski Biopython versiyalari bilan moslashtirmoqchi bo'lsangizgina tavsiya etiladi.

### 9.3.3 MutableSeq

Seq ob'ektlari o'zgaruvchan emas. Bu sizning ma'lumotlaringizni o'zgartirmasdan saqlashni xohlishingiz uchun mo'ljallangan. Shunday qilib, o'zgarmas ketma-ketlik Python string harakati bilan mos keladi:

```
>>> ketma-ket [0] = 'T'
Traceback (eng oxirgi qo'ng'iroq):
  Fayl "<stdin>", 1-qator, ?
TypeError: "Seq" ob'ekti elementni belgilashni qo'llab-quvvatlamaydi
```

Bu muammoni to mutable() usuli bilan MutableSeq yaratish orqali hal qilish mumkin:

```
>>> mut_seq = seq.tomutable() >>>
mut_seq MutableSeq('CCGGGTTAACGTA',
IUPACUnambiguousDNA())
```

Uning o'zgaruvchanligini tekshirish uchun o'zgartirish kriting:

```
>>> mut_seq[0] = 'T'
>>> mut_seq
MutableSeq('TCGGGTTAACGTA', IUPACUanambiguousDNA())
```

Siz ketma-ketlikni ro'yxat kabi o'zgartirishingiz mumkin, `append()`, `insert()`, `pop()` va `remove()`. Shuningdek, DNK ketma-ketligini manipulyatsiya qilish uchun maxsus usullar mavjud:

```
>>> mut_seq.reverse()
>>> mut_seq
MutableSeq('ATGCAATTGGGCT', IUPACUNambiguousDNA())
>>> mut_seq.complement() >>> mut_seq
MutableSeq('TACGTTAACCCGA', IUPACUNambiguousDNA())
>>> mut_seq () >>> mut_seq MutableSeq('TCGGGTTAACGTA',
IUPACUanambiguousDNA())
```

### 9.3.4 SeqRecord

`Seq` klassi muhim ahamiyatga ega, chunki u bioin formatidagi asosiy o'rGANISH mavzusini saqlaydi: ketma-ketlik. Ba'zan bizga ism, identifikator, tavsif va tashqi ma'lumotlar bazalari va izohlarga o'zaro havolalar kabi oddiy ketma-ketliklardan ko'ra ko'proq ma'lumot kerak bo'ladi. Bu ketma-ketlik bilan bog'liq barcha ma'lumotlar uchun `SeqRecord` klassi mavjud. Boshqacha qilib aytganda, `SeqRecord` bog'langan metama'lumotlarga ega `Seq` ob'ektiidir:

```
>>> Bio.SeqRecord dan SeqRecord importi
>>> SeqRecord (seq, id='001', name='MHC geni')
SeqRecord(seq=Seq('CCGGGTTAACGTA', IUPACUNambiguousDNA()), id='001'<=
, name='MHC geni', tavsif='<noma'lum tavsif>', dbxrefs=[])
```

`SeqRecord` ikkita asosiy atributga ega:

`id` Identifikatorli qator. Bu atribut ixtiyorli, lekin juda tavsiya etiladi.

`seq` A `Seq` obyekti. Bu atribut talab qilinadi.

Ba'zi qo'shimcha atributlar mavjud:

`nom` Ketma-ketlik nomiga ega satr.

`tavsif` Qo'shimcha ma'lumotga ega qator.

`dbxrefs` Satrlar ro'yxati; har bir satr ma'lumotlar bazasi o'zaro mos yozuvlar identifikatoridir.

xususiyatlar SeqFeature ob'ektlari ro'yxati. Bu Genbank yozuvlarida topilgan ketma-ketlik xususiyatini ifodalaydi. Ushbu atribut odatda GenBank faylidan (masalan, SeqIO tahlilchisidan foydalangan holda) ketma-ketlikni qidirganimizda to'ldiriladi. U ketma-ketlik joylashuvi, turi, strand va boshqa o'zgaruvchilarni o'z ichiga oladi.

izohlar Butun ketma-ketlik haqida qo'shimcha ma'lumotga ega lug'at.

SeqRecord obyektini ishga tushirishda ushbu atributni o'rnatib bo'lmaydi.

SeqRecord obyektini noldan yaratish:

```
>>> Bio.SeqRecord dan SeqRecord >>> Bio.Seq dan
import Seq >>> Bio.Alphabet dan import
generic_protein >>> rec =
SeqRecord(Seq('mdstnvrsgmksrkkpkttviddddcmtnacsacqs' 'klvkisditkvsldyintmrgn'
              =klvkisditkvsldayintmrgn2), _generic_protein .1',
          nomi = 'P20994', tavsifi = 'Protein A19', dbxrefs = ['Pfam:PF05077',
          'InterPro:IPR007769', 'DIP:2186N']) >>> tavsiyalar
izohlari['eslatma'] = 'Oddiy eslatma' >>> chop etish (rec)
```

**ID: P20994.1**

**Nomi: P20994**

**Tavsif: Protein A19**

**Ma'lumotlar bazasi oyzaro havolalari: Pfam:PF05077, InterPro:IPR007769, DIP<= : 2186N**

**Funktsiyalar soni: 0**

**/note=Oddiy eslatma**

**Seq('mdstnvrsgmksrkkpkttviddddcmtnacsacqsklvkisditkvsldyint...kl<= l', ProteinAlphabet())**

GenBank faylidan SeqRecord yaratish uchun 174-betga qarang.

### 9.3.5 Tegishlash

Align moduli hizalamalar bilan ishlash uchun kodni o'z ichiga oladi. Ushbu modulning markaziy ob'ekti MultipleSeqAlignment sinfidir. Ushbu ob'ekt ketma-ketlikni moslashtirishni saqlaydi. U tekislash uchun mo'ljallanmagan; unda hizalanishlarni saqlashdan oldin ketma-ketliklar allaqachon tekislangan deb taxmin qilinadi.

Bu erda oddiy ikkita kichik peptid ketma-ketligi hizalanishi:

MHQQAIFIYQIGYPLKSGYIQSIRSPEYDNW

|| |||||||\*||||||| ||

MH--IFIYQIGYALKSGYIQSIRSPEY-NW

Ushbu hizalama 9.1 Listingdagi kabi Biopython yordamida bitta ob'ektda saqlanishi mumkin :

#### Listing 9.1: Align modulidan foydalanish

---

```
1 Bio.Alphabet import generic_protein 2 Bio.Align import
MultipleSeqAlignment 3-dan Bio.Seq-dan import 4-seq
Bio.SeqRecord import SeqRecord 5 seq1 =
'MHQAIIFIYQIGYPLKSGYIQSIRSPEYDNW' 6 seq2 =
'IQSIRSPEYDNW' 6 seq2- = 'FIIRSPEYDNW' Seq(seq1,
generic_protein), id = 'asp') 8 seq_rec_2 = SeqRecord(Seq(seq2,
generic_protein), id = 'unk') 9 align = MultipleSeqAlignment([seq_rec_1, seq_rec_2])
10)
```

---

Kod tushuntirishi: MultipleSeqAlignment klassi 9-satrda yaratilgan. align - MultipleSeqAlignment obyektining nomi. Ikkala ketma-ketlik MultipleSeqAlignment ob'ektini ishga tushirishda SeqRecord ob'ektlari sifatida qo'shiladi.

Oldingi kodning chiqishi:

```
ProteinAlphabet() ni 2 satr va 30 ustun bilan tekislash
MHQAIIFIYQIGYPLKSGYIQSIRSPEYDNW asp
MH--IFIYQIGYALKSGYIQSIRSPEY-NW unk
```

MultipleSeqAlignment ketma-ketliklar ro'yxati (yoki SeqRecord ob'ektlari) sifatida ko'rib chiqilishi mumkin, u o'zining ba'zi usullarini baham ko'radi. Hizalamaga yangi ketma-ketlikni qo'shish uchun append va bir nechta ketma-ketlikni qo'shish uchun kengaytmani qo'llab-quvvatlaydi:

```
>>> seq3 = 'M---IFIYQIGYAAKSGYIQSIRSPEY--W' >>>
seq_rec_3 = SeqRecord(Seq(seq3, generic_protein), id = 'cas') >>> align.append(seq_rec_3)
>>> print(align)
```

```
ProteinAlphabet() 3 qator va 30 ustunli MHQAIIFIYQIGYPLKSGYIQSIRSPEYDNW
asp MH--IFIYQIGYALKSGYIQSIRSPEY-NW unk M---
IFIYQIGYAAKSGYIQSIRSPEY--
```

E'tibor bering, yangi SeqRecord ob'ektlari asl nusxa bilan bir xil uzunlikka ega bo'lishi kerak hizalama va hizalanish alifbosiga mos keladigan alifbolarga ega bo'ling.

Ro'yxatlar bilan umumiy bo'lgan yana bir xususiyat shundaki, siz butun son indeksidan foydalanib, elementni (qator yoki SeqRecord ob'ekti) olishingiz mumkin:

```
>>> tekislash[0]
```

```
SeqRecord(seq=Seq('MHQAIIFIYQIGYPLKSGYIQSIRSPEYDNW', ProteinAlphabet()), <= id='asp',
          name='<noma'lum nom>', tavsif='<noma'lum tavsif>', <= dbxrefs=[])
```

Pastki hizalamani olish uchun butun son indeksi o'rniga Python tilim belgisidan foydalaning:

```
>>> chop etish (tekislash[:2,5:11])
ProteinAlphabet() ni 2 satr va 6 ustun bilan tekislash
FIYQIG asp
FIYQIG unk
```

Har qanday Python ketma-ketligida bo'lgani kabi, len() yordamida tekislash uzunligini olishingiz mumkin:

```
>>> len (tekislash)
3
```

Shuningdek, u har bir ketma-ketlik uchun SeqRecord ob'ektini qaytarib, uning barcha elementlarini takrorlashni qo'llab-quvvatlaydi.

Quyidagi kod hizalanishdagi har bir ketma-ketlikning izoelektrik nuqtasini hisoblab chiqadi:

```
>>> Bio.SeqUtils.ProtParam dan ProteinAnalysis importi >>> ketma-
ketligi uchun: print(ProteinAnalysis(str(seq.seq)).izoelektrik_point())
...
6.50421142578125
8.16033935546875
8.13848876953125
```

### 9.3.6 AlignIO

Bitta tekislash bilan faylni o'qish uchun AlignIO.read() dan foydalaning. Bu ikkita parametrni talab qiladi: fayl nomi (yoki fayl dastagi ob'ekti) va hizalama formati. Matlar uchun amal qiladi: clustal, emboss, fasta, fasta-m10, ig, maf, nexus, phylip, philip-sequential, phylip-relaxed va stockholm. AlignIO.read() usuli Multiple SeqAlignment obyektini qaytaradi.

```
>>> Bio import AlignIO dan >>>
AlignIO.read('cas9al.fasta', 'fasta') chop etish(align)
```

```
8 satr va 1407 ustunli SingleLetterAlphabet() hizalamasi
MDKKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIK...GGD J7M7J1
MDKKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIK...GGD A0A0C6FZC2
MDKKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIK...GGD A0A1C2CVQ9
MDKKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIK...GGD A0A1C2CV43
```

## 170 Bioinformatika uchun Python

```
MDKKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIK...GGD Q48TU5
MDKKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIK...GGD M4YX12
MKKPYSIGLDIGTNSVGWAVVTDDYKVPACKMKVLGNTDKSHIK...GGD A0A0E2EP65
-----...GED A0A150NVN1
```

Bir nechta hizalamali fayllarni o'qish uchun AlignIO.parse() dan foydalaning: U AlignIO.read() bilan bir xil argumentlarni oladi va bu faylda mavjud barcha hizalamalar bilan iteratorni qaytaradi. U loopda foydalanish uchun mo'ljallangan:

```
>>> Bio import AlignIO dan >>>
AlignIO.parse('example.aln', 'clustal'):
...     chop etish (tegishlash)
```

1098

233

Diskka tekislashni yozish uchun AlignIO.write() dan foydalaning. Bu usul birinchi parametr sifatida MultipleSeqAlignment ob'ektini talab qiladi, keyin u AlignIO.read() bilan bir xil ikkita parametrni talab qiladi (fayl nomi va formati). Qabul qilingan formatlar: clustal, fasta, maf, nexus, phylip, phylip-sequential, phylip-relaxed, stockholm. AlignIO.write() usuli saqlangan tekislashlar sonini qaytaradi:

```
>>> Bio import AlignIO dan >>>
AlignIO.write(align, 'cas9al.phy', 'phylip') 1
```

Hizalama fayllarini bir qadamda aylantirish uchun yordamchi funksiya mavjud: AlignIO.convert(). To'rtta parametr kerak: o'qish uchun fayl nomi, o'qiladigan fayl formati, yozish uchun fayl nomi va yozish uchun fayl formati. Shuningdek, saqlangan hizalamalar sonini qaytaradi:

```
>>> Bio import AlignIO dan >>>
AlignIO.convert('cas9al.fasta', 'fasta', 'cas9al.aln', 'clustal') 1
```

## AlignInfo

AlignInfo moduli tekislash ob'ektlaridan ma'lumot olish uchun ishlataladi. U print\_info\_content funksiyasini hamda SummaryInfo va PSSM ni taqdim etadi sinflar:

- print\_info\_content():

Keling, ularni amalda ko'rib chiqaylik:

```
>>> Bio import AlignIO dan >>>
```

```
Bio.Align.AlignInfo dan import SummaryInfo >>> Bio.Alphabet dan  
import ProteinAlphabet >>> align = AlignIO.read('cas9align.fasta', 'fasta')  
>>> align._alphabet = ProteinAlphabet() >>> xulosa = SummaryInfo(align) >>>  
print(summary.information_content()) 4951.072487965924
```

```
>>> summary.dumb_consensus(consensus_alpha=ProteinAlphabet())  
Seq('MDKKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIKKNL...GGD',<=  
ProteinAlphabet()) >>> summary.gap_consensus(consensus_alpha=ProteinAlphabet())
```

```
Seq('MDKKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIKKNL...GGD',<=  
ProteinAlphabet()) >>> chop etish (summary.alignment)
```

```
ProteinAlphabet() 8 satr va 1407 ustunli hizalama
```

```
MDKKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIK...GGD J7M7J1
```

```
MDKKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIK...GGD A0A0C6FZC2
```

```
MDKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIK...GGD A0A1C2CVQ9
```

```
MDKKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIK...GGD A0A1C2CV43
```

```
MDKKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIK...GGD Q48TU5
```

```
MDKKYSIGLDIGTNSVGWAVITDDYKVPSKKFKVLGNTDRHSIK...GGD M4YX12
```

```
MKKPYSIGLDIGTNSVGWAVITDDYKVPAKKMKVLGNTDKSHIK...GGD A0A0E2EP65
```

```
-----...GED A0A150NVN1
```

```
>>> chop etish (summary.pos_specific_score_matrix())
```

```
- ACDEFGHIKLMNPQ
```

```
M 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0
```

```
D 1,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0
```

```
K 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0
```

```
K 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,0,0,0,0,0,0,1,0
```

```
Y 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

```
S 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

```
I 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

```
G 1,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

```
L 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0
```

```
D 1,0,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

### 9.3.7 ClustalW

9 Ushbu modulda ClustalW bilan o'zaro ishlash uchun sinflar va funksiyalar mavjud. Muallifi  
bo'lgan mashhur grafik ko'p tekislash dasturini ClustalX bilishingiz mumkin

9Ushbu dastur <http://www.clustal.org/download/current> saytida mavjud .

## 172 Bioinformatika uchun Python

Juli Tompson va Fransua Janmujin. ClustalX - bu ClustalW uchun grafik old qism, buyruq qatorini bir nechta tekislash dasturi.

Biopython ClustalwCommandline paketi bilan ClustalW-ni qo'llab-quvvatlaydi boshiga. Ushbu sinf ClustalW buyruq qatorini yaratish uchun ishlatalishi mumkin:

```
>>> Bio.Align.Applications dan import ClustalwCommandline >>> clustalw_exe =
'clustalw2' >>> ccli = ClustalwCommandline(clustalw_exe, <=
infile="input4align.fasta", outfile='../../aoutput.aln') >>> print(ccli) clustalw2
-infile=input4align.fasta -outfile=../../aoutput.aln
```

Agar clustalw dasturi tizim yo'lida bo'lmasa, ob'ektni ishga tushirishda uning joylashuvini belgilashingiz kerak. Masalan, agar clustalw c:\windows\program file\clustal\clustalw.exe da bo'lsa, qatorni almashtiring.

```
>>> clustalw_exe = 'clustalw2'
```

uchun

```
>>> clustalw_exe='c:\\windows\\dastur fayli\\clustal\\clustalw.exe'
```

Dasturni ishga tushirish uchun yaratilgan misolni chaqiring:

```
>>> Bio.Align.Applications dan import ClustalwCommandline >>> clustalw_exe =
'clustalw2' >>> ccli = ClustalwCommandline(clustalw_exe, infile="input4align.fasta",
outfile='../../aoutput.aln') >>> ccli() ('\\n\\n\\n CLUSTAL 2.1 Bir nechta ketma-ketlikni
tekislash\\n\\n\\n Kartlik <= formati Pearson\\n1-ketlik: AGA92859.1 106 aa\\n2-ketlik:
<= AML31452.1 116 aa \\n3-ketma-ket: AAH03888.1 473 aa\\n4-ketlik<= : BAE71953.1
118 aa\\nJuftlarni tekislashning boshlanishi\\nTezalash...<= \\n\\nKetliklar (1:2) tekislangan. Bal:
88\\n1qu :3) Aligned<=. Bal: 93\\nSequences (1:4) Hizalangan. Bal: 82\\nSequences (2:<= (...)[alignoutput.txt]\\n\\n', '')
```

Funktsiya ikkita qiymatlari kortejni qaytaradi. Birinchi qiymat dastur qaytaradigan qiymat (standart chiqish deb ham ataladi), ikkinchisi esa xato xabari, agar

har qanday.

Chiqishni qayta ishlash uchun faylni AlignIO.read() bilan o'qing:

```
>>> Bio import AlignIO dan >>> seqs
= AlignIO.read('../../aoutput.aln', 'clustal')
```

```
>>> ketma-ketliklar[0]
SeqRecord(seq=Seq('-----QVQLQQSDAELVKPGASVKISCKVSG<= YTFTDHTIH...---',
SingleLetterAlphabet()), id='AGA92859.1' , nomi<= ='<noma'lum nom>',
description='AGA92859.1', dbxrefs=[]) >>> ketma-ketliklar[1]

SeqRecord(seq=Seq('MEWSWVFLFFLSVTGVHSQVQLQQSDAELVKPGASVKISCKVSG<=
YTFTDHTIH...PGK', SingleLetterAlphabet()), id='AAH03888.1', ism<= ='<noma'lum nomi
d8bAx1>'s.' =[]) >>> ketma-ketliklar[2]

SeqRecord(seq=Seq('-----EVQLQESDAELVKPGASVKISCKVSG<= YTFTDHSIH...---',
SingleLetterAlphabet()), id='AML31452.1' , nomi<= ='<noma'lum nom>',
description='AML31452.1', dbxrefs=[])
```

### Parametrlarni ClustalW ga o'tkazish

Clustalw-ga qo'shimcha parametrlarni o'tkazish uchun ularni ClustalwCom mandline-ni yaratishda o'tkazing. Masalan , bo'shlqn ochish jazosini o'zgartirish uchun pwgapopen dan foydalaning :

```
>>> Bio.Align.Applications import ClustalwCommandline >>> clustalw_exe =
'clustalw2' >>> ccli = ClustalwCommandline(clustalw_exe,
infile="input4align.fasta", outfile='./aoutput.aln', pwgapopen=5) >>> print(ccli)
clustalw2 -infile=input4align.fasta -outfile=./aoutput.aln <=
-pwgapopen=5
```

Qolgan mavjud parametrlarni ko'rish uchun quyidagilarni bajaring:

```
>>> Bio.Align.Applications dan import ClustalwCommandline >>> ccli =
ClustalwCommandline() >>> help(ccli)
```

yoki <https://goo.gl/dJwoJx> saytidagi onlayn qo'llanmani ko'ring yoki API sahifasida: <http://biopython.org/DIST/docs/api/>.

### 9.3.8 SeqIO

Bio.SeqIO fayl formatlarini kiritish va chiqarish uchun umumiyl interfeysdir. Ushbu interfeys yordamida olingan ketma-ketliklar dasturingizga SeqRecord obyektlari sifatida uzatiladi. Bio.SeqIO shuningdek, tekislash fayl formatlarini o'qiy oladi va u har bir yozuvni SeqRecord ob'ekti sifatida qaytaradi. Alignment obyekti sifatida tekislashni olish uchun Bio.AlignIO modulidan foydalaning.

## Ketma-ket fayllarni o'qish

Ketma-ketlikni o'qish uchun ishlataladigan usul parse (fayl\_handle, format). Bu erda format "fasta", "genbank" yoki [10.1-jadvalda](#) ko'rsatilgan boshqa har qanday bo'lishi mumkin . Ushbu tahlilchi generatori qaytaradi. Ushbu generator tomonidan qaytarilgan elementlar SeqRecord turiga kiradi:

```
>>> Bio import SeqIO >>> f_in =
open('..../samples/a19.gp') >>> seq =
SeqIO.parse(f_in, 'genbank') >>> keyingi(seq)
```

```
SeqRecord(seq=Seq('MGHHHHHHHHHHSSGHIDDDKHMLEDSTNRSGMKSRRKKPKT<=
TVIDDDDC...FAS', IUPACProtein()), id='AAX78491.1', name='AAX784',<=sconstruksiya
ma'lum [']f =[])
```

Faylda faqat bitta ketma-ketlik mavjud bo'lsa, Se qI0.parse() o'rniqa SeqIO.read() dan foydalaning:

```
>>> f_in = ochiq('..../samples/a19.gp')
>>> SeqIO.read(f_in, 'genbank')
SeqRecord(seq=Seq('MGHHHHHHHHHHSSGHIDDDKHMLEDSTNRSGMKSRRKKPKT<=
TVIDDDDC...FAS', IUPACProtein()), id='AAX78491.1', name='AAX784',<=sconstruksiya
ma'lum [']f =[])
```

[Listing 9.2](#) da FASTA formatidagi ketma-ketliklarga to'la faylni o'qiydigan skript mavjud. va har bir yozuvning sarlavhasi va uzunligini ko'rsatadi.

### Listing 9.2: readfasta.py: FASTA faylini o'qing

---

Bio import SeqIO 2 dan 1

```
3 FILE_IN = '..../samples/3seqs.fas' 4
```

```
5 ochiq(FILE_IN) fh sifatida:
record.SeqIO.parse(fh[0].read(), 'fasta').read(1).record.id
6 record.seq =
7 len(seq)))
8
9
```

---

Kirish faylining mazmuni (3seqs.fas):

```
>Protein-X [Simian immunitet tanqisligi virusi]
NYLNNLTVDPDHNKCDNTTGRKGNAAPGPCVQRTYVACH
> Protein-Y [Homo sapiens]
MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLSQAMDDLMLSPDDIEQWFTEDPGPDA
> Protein-Z [Rattus norvegicus]
MKAAVLAVALVFLTGCQAWEFWQQDEPQSQWDRVKDFATYYVDAVKDSGRDYVSQFESST
```

**Listing 9.2** 3seqs.fas faylini tahlil qiladi va quyidagi natijani hosil qiladi:

```
(biopy169) $ python readfasta.py lsm: Protein-
X, hajmi: 38 lsm: Protein-Y, hajmi: 62 lsm:
Protein-Z, hajmi: 60
```

## 9.1-JADVAL Ketma-ketlik va tekislash formatlari

Format nomi	Tavsif	Hizalama - Ketma-ketlik
ace	ACE yig'ish faylidan kontig sekanslarini o'qiydi.	S
klastal	Clustal W yoki X dan chiqish	A
embl	EMBL tekis fayl formati.	S
bo'rtma	EMBOSS vositalaridan "juftlik" va "oddiy" tekislash formati.	A
fasta	Har bir yozuv ">" belgisidan boshlanadigan identifikator qatoridan, keyin ketma-ketlik satrlaridan boshlanadigan oddiy format.	A/S
fasta-m10 -m 10	buyruq qatori opsiyasi bilan foydalanilganda Bill Pearsonning FASTA vositalari tomonidan chiqarilgan hizalamalar. genbank GenBank yoki GenPept tekis fayl formati.	A
qo'llaniladi. A aloqa	Single M sequence of DNA or protein is taken and then matches to a database, ushbu fayllardagi har qanday filogenetik daraxtlarni o'qiy oladigan Bio.Nexus moduliga qarang. Phd PHRED dan olingan.	S
	S phylip PHYLIP vositalari tomonidan ishlatiladi.	
	PFAM tomonidan ishlatiladigan stokholm.	
	ikkita ustun yordamda ketma-ketlik fayl formati. Sariya S yorlig'i Oddiy tahlil qilish.	

**Ketma-ket fayllarni yozish**

SeqIO-da ketma-ketliklarni yozish usuli mavjud: write (iterable, file\_handle, mat uchun). Bu funksiya qabul qiladigan birinchi parametr Se qRecord ob'ektlari (masalan, SeqRecord ob'ektlari ro'yxati) bilan takrorlanadigan ob'ektdir. Ikkinci parametr - ketma-ketliklarni yozish uchun ishlatiladigan fayl tutqichi. Format argumenti quyidagi kabi ishlaydi

tahlil qilish.

**9.3 ro'yxati** ketma-ketlikdagi faylni oddiy matn sifatida qanday o'qish va yozishni ko'rsatadi  
FASTA ketma-ketligi sifatida:

**Listing 9.3: rwmfasta.py: Faylni o'qing va uni FASTA ketma-ketligi sifatida yozing**

## 176 Bioinformatika uchun Python

```

1 Bio import SeqIO 2 Bio.Seq
dan import Seq 3 Bio.SeqRecord
import SeqRecord 4 bilan open('NC2033.txt') fh
sifatida: open('NC2033.fasta','w') sifatida f_out:
5     rawseq = fh.read().replace('\n','') yozuv =
6         (SeqRecord(Seq(rawseq),'NC2033.txt','','',))
7
8     SeqIO.write(record, f_out,'fasta')

```

Ko'pgina biologik fayl formatlarini o'qish va yozishni bilish o'qish imkonini beradi bir formatdagi ketma-ketliklarga ega fayl va ularni boshqa formatga yozing:

```

Bio importdan SeqIO
fo_handle = open('myseqs.fasta','w') readseq =
SeqIO.parse(open('myseqs.gbk'), 'genbank')
SeqIO.write(readseq, fo_handle, "fasta")
fo_handle.close()

```

15-bobda SeqIO-dan foydalanishning ko'proq misollari mavjud .

### 9.3.9 AlignIO

AlignIO - hizalamalar uchun kirish/chiqish interfeysi. U asosan SeqIO sifatida ishlaydi, lekin SeqRecord ob'ektini qaytarish o'rniiga Alignment ob'ektini qaytaradi. Uning uchta asosiy usuli bor: o'qish, tahlil qilish va yozish. Birinchi ikkita usul kiritish uchun, oxirgisi esa chiqish uchun ishlataladi.

- o'qing(tutqich,format[,sek\_hisoblash]): Fayl tutqichini va hizalashni oling argument sifatida formatlang va Alignment obyektini qaytaring.

```

>>> Bio import AlignIO dan >>> fn
= open('secu3.aln') >>> align =
AlignIO.read(fn, 'clustal') >>> print(align)

```

SingleLetterAlphabet() 3 satr va 1098 ustunli hizalama

```

-----...--- secu3
-----...--- AT1G14990.1-CDS

```

GCTTGCTATGCTATATGTTATTACATTGTGCCTCTG...CAC AT1G14990.1-SEQ

Sec\_count argumenti har qanday fayl formati bilan ishlatalishi mumkin , lekin u asosan FASTA hizalamalarida ishlataladi. U har bir hizalamadagi ketma-ketliklar sonini ko'rsatadi, bu fayl 15 ta ketma-ketlikdan iborat faqat bitta hizalanish yoki 5 ta ketma-ketlikning uchta hizalanishi ekanligini aniqlash uchun foydalidir.

- `write(iterable,handle,format)`: Faylga yozish uchun Alignment ob'ektlari to'plamini, faylni boshqarish va fayl formatini oling. Siz ushbu funksiyani takrorlanadigan barcha hizalamalar bilan chaqirishingiz va fayl tutqichini yopishingiz kutilmoqda. Quyidagi kod Clustal formatidagi tekislashni o'qiydi va uni Phylip formatida yozadi.

#### Listing 9.4: Hizalamalar

---

```
fi = open('..../samples/example.aln') bilan open('..../
samples/example.phy', 'w') sifatida fo: align = AlignIO.read(fi,
"klastal")
AlignIO.write([alig], fo, 'philip')
```

---

#### 9.3.10 PORTLASH

**Basic Local Alignment Search Tool (BLAST)** — foydalanuvchi soýrovini ketma-ketliklar maýlumotlar bazasi bilan solishtirish uchun foydalilanadigan ketma-ketlik oýxshashligini qidirish dasturi. DNK yoki aminokislotalar ketma-ketligini hisobga olgan holda, BLAST evristik algoritmi ikkita ketma-ketlik o'rtasida qisqa mosliklarni topadi va bu "issiq nuqtalar" dan tekislas BLAST shuningdek, "expect" qiymati kabi tekislash haqida statistik ma'lumotlarni taqdim etadi. 10 E'tibor bering, BLAST yagona dastur emas, balki dasturlar oilasi. Barcha BLAST dasturlari ketma-ketliklar orasidagi moslikni qidiradi, lekin ketma-ketlikni qidirishning har bir turi uchun maxsus BLAST dasturi mavjud. blastn , masalan, kirish sifatida nukleotidlardan ketma-ketligidan foydalangan holda nukleotid ma'lumotlar bazalarida qidirish uchun ishlataladi. Agar ma'lumotlar bazasi oqsilga (aminokislotalarga) asoslangan bo'lса va sizning kiritishingiz nukleotidlardan ketma-ketligi bo'lса, siz foydalaningiz kerak bo'lган BLAST dasturi blastx hisoblanadi . Ushbu dastur nukleotid kiritishni aa oqsiliga aylantiradi va oqsil ma'lumotlar bazasiga qarshi qidiruvni amalga oshiradi. BLAST dasturlari ro'yixati va ulardan qachon foydalanish kerakligi

BLAST eng keng tarqalgan bioinformatika tadqiqot vositalaridan biridir, chunki u bir nechta ilovalarga ega. Bu erda odatiy BLAST ilovalari ro'yxati:

- Bir turda ilgari noma'lum gen topilgandan so'ng, boshqa turlarda ham xuddi shunday gen borligini bilish uchun boshqa genomlarni qidiring.
- Ketma-ketliklar orasidagi funksional va evolyutsion munosabatlarni topish.
- Promouter signallari, splicing kabi konsensus tartibga solish modellarini qidiring saytlar va transkripsiya faktorini bog'lovchi saytlar.
- Ilgari kristallangan oqsillar asosida oqsil tuzilishi haqida xulosa chiqarish.
- Gen oilalari a'zolarini aniqlashga yordam bering.

10 Kutilayotgan qiymat (E) ko'rishni "kutish" mumkin bo'lган zARBALAR sonini tavsiflovchi parametrdir. ma'lum bir o'chamdagи ma'lumotlar bazasini qidirishda tasodifan.

Agar siz bioinformatika sohasida ishlayotgan bo'lsangiz, ehtimol siz BLAST so'rovlarini bajarishingiz yoki siz yoki boshqa shaxs tomonidan yaratilgan BLAST so'rovlarini qayta ishlash zarurligiga duch kelishingiz mumkin. Biopython ikkala vazifa uchun vositalarni taqdim etadi:

### 9.2-JADVAL Blast dasturlari

Dastur nomi	So'rov/ma'lumotlar bazasi kombinatsiyasi
blastn blastp	nukleotid va nukleotid. protein
blastx tblastn	va oqsil. tarjima qilingan
tblastx	nukleotid va oqsil. oqsil va tarjima nukleotid. tarjima qilingan nukleotid va tarjima qilingan nukleotid

### BLASTni Biopython bilan ishlash va qayta ishlash

BLAST NCBI veb-serverida onlayn yoki shaxsiy kompyuterindan mahalliy sifatida ishga tushirilishi mumkin. BLAST-ni Internet orqali ishga tushirish bir nechta ketma-ketlikni o'z ichiga olgan kichik ishlar uchun yaxshi imkoniyatdir. Kattaroq ishlar uzoq server tomonidan "CPU foydalanish chegarasi oshib ketdi" xabari bilan to'xtatiladi. NCBI BLAST davlat xizmati bo'lgani uchun ular o'z serverlarini ortiqcha yuklamaslik uchun protsessordan foydalanishga kvotalar qo'yishlari kerak. BLAST ning mahalliy versiyasidan foydalanishning yana bir jiddiy sababi - bu maxsus ma'lumotlar bazasini so'rash kerak bo'lganda. NCBI BLAST serverida foydalanishingiz mumkin bo'lgan ma'lumotlar bazasi(lar)iga nisbatan bir oz moslashuvchanlik mavjud, biroq u har qanday turdag'i va maxsus ma'lumotlar hajmini sig'dira olmaydi.

Shu sabablarga ko'ra, ko'pchilik tadqiqot laboratoriyaning ishlashi odatiy hol emas uy ichidagi BLAST qidiruvlari.

### BLAST ishni boshlash

Biopython-da har bir BLAST bajariladigan fayl uchun o'ram mavjud, shuning uchun siz BLAST dasturini skriptingiz ichidan ishga tushirishingiz mumkin. Blastn uchun o'ram - `Bio.Blast.Applications` moduli ichidagi `NcbiblastnCommandline` deb nomlangan funksiya . blastx uchun o'ram boshiga `NcbiblastxCommandline` va hokazo. Biz `NcbiblastnCommandline`-dan qanday foydalanishni ko'rib chiqamiz , ammo bu boshqa barcha o'ramlarga ham qo'llanilishi mumkin.

Mana NcbiblastnCommandline sintaksisi:

`NcbiblastnCommandline(portlash bajariladigan fayl, dastur nomi, ma'lumotlar bazasi,<= kiritish fayli, [align_view=7], [parametrlar])`

Bu funksiya ikkita fayl ob'ekti bo'lgan kortejni qaytaradi. Birinchisi haqiqiy natija, ikkinchisi esa BLAST xato xabari (agar mavjud bo'lsa). Ko'pgina parametrlar o'z-o'zidan tushunarli. [9.5 ro'yxati](#) buni aniq ko'rsatib beradi:

**Listing 9.5: runblastn.py: Mahalliy NCBI BLASTni ishga tushirish**

---

1 `Bio.Blast.Applications` dan `NcbiblastnCommandline` ni blastn sifatida import qiladi

```
2 BLAST_EXE = '/home/sb/opt/ncbi-blast-2.6.0+/bin/blastn' 3 f_in = 'seq3.txt' 4
b_db = 'db/samples/TAIR8cds' 5 blastn_cline = blastn(cmd=BLAST_EXE) ,
query=f_in, db=b_db, evalue=.0005, outfmt=5) 7 rh, eh = blastn_cline()
```

6

---

BLAST dasturi 5-qatorda ishlaydi. Natijani olish uchun siz o'qishingiz kerak qaytarilgan faylga o'xshash ob'ekt rh, allaqachon 5 -bobda ko'rib chiqilgan :

```
>>> rh.readline()
<?xml versiyasi="1.0"?>
>>> rh.readline()
'<!DOCTYPE BlastOutput PUBLIC "-//NCBI//NCBI BlastOutput/EN"=< "http://
www.ncbi.nlm.nih.gov/dtd/NCBI_BlastOutput.dtd">\n'
```

Chiqish XML formatida. Ushbu ma'lumotni 11-bobda o'rganilgan vositalar yoki Biopython tomonidan taqdim etilgan vositalar yordamida tahlil qilish mumkin (bu haqda keyingi bo'limda). NcbiblastnCommandline-ni chiqish sifatida oddiy matndan foydalanishga majburlash orqali XML chiqishi bilan shug'ullanishdan qochishning bir usuli ham mavjud. Bu buyruq satrida yoki Biopython funksiyasida ixtiyoriy parametr sifatida -outfmt 011 dan foydalanish orqali amalga oshiriladi . Bu o'qish oson (odam tomonidan), lekin tahlil qilish qiyin (kompyuter tomonidan) natijaga olib keladi. Agar oxirgi jumla g'alati tuyulsa, nima uchun "odam tomonidan o'qilishi mumkin bo'lgan" format avtomatlashtirilgan ishlov berish uchun mos kelmasligini tushunish uchun bir necha paragraflar uchun men bilan birga bo'ling.

Eh fayl dastagi NcbiblastnCommandline tomonidan qaytarilgan xato xabarini saqlaydi. Bu holda u bo'sh (chunki xatolik yo'q edi):

```
>>> eh.readline()
"
```

5-qatordagi funksiya chaqiruvi quyidagi bayonotni kiritish bilan tengdir buyruq qatorida:

```
$ ./blastn -query seq3.fasta -db db/TAIR8cds -outfmt 5
```

Ushbu buyruq qatoridagi barcha parametrlar Biopython NcbiblastnCommandline funksiyasidagi parametr bilan mos kelishi mumkin. Oxirgi parametr (-outfmt 5) natijani XML chiqarishga majbur qiladi. Oddiy matn va HTML kabi boshqa BLAST chiqish formatlari versiyadan versiyaga o'zgarib turadi, bu esa eng so'nggi tahlilchini saqlashni juda qiyinlashtiradi.<sup>12</sup> Inson tomonidan o'qilishi mumkin bo'lgan matn odatda tuzilmagan bo'ladi. Bu sabablarga ko'ra, o'qish oson bo'lgan natijani tahlil

<sup>0</sup> dan 4 gacha bo'lgan 11-outfmt odamlar tomonidan o'qilishi mumkin bo'lgan har xil turdag'i natijalarni yaratadi.

<sup>12</sup>Bu haqda NCBIning rasmiy bayonoti bor: "NCBI dan foydalanishni yoqlamaydi. ma'lumotlarni aniq tahlil qilish vositasi sifatida BLAST chiqishining oddiy matni yoki HTML.

Funktsiya chaqiruvining oxirida qo'shiladigan ixtiyoriy parametrlar orqali boshqarilishi mumkin bo'lgan portlash so'rovining ko'plab jihatlari mavjud. Ushbu ilovani NCBI BLAST foydalanuvchi qo'llanmasida <https://www.ncbi.nlm.nih.gov/books/NBK279675> orqali tekshiring. qo'shimcha ma'lumot uchun.

Fayl ob'ekti sifatida BLAST natijasini olgанингиздан so'ng, uni qayta ishлаshingiz kerak bo'lishi mumkin. Agar natijani keyinchalik qayta ishлаsh uchun saqlashni rejalashtirsangiz, uni saqlashingiz kerak:

```
>>> fh = open('testblast.xml','w') >>>
fh.write(rh.read()) >>> fh.close()
```

Ko'pincha siz BLAST chiqishidan ba'zi ma'lumotlarni olishingiz kerak bo'ladi. Shu maqsadda keyingi bo'limda keltirilgan NCBI XML tahlilchisi yordam beradi.

### BLAST chiqishini o'qish

BLAST faylining mazmunini tahlil qilish har qanday bioinformatika mutaxassisi bilan shug'ullanishi kerak bo'lgan narsadir. Biopython Bio.Blast.NCBIXML modulida (parse deb ataladi) foydali tahlilchini taqdim etadi. Ushbu tahlilchi yordamida dasturchi BLAST chiqish faylidan istalgan muhim bitni ajratib olishi mumkin. Ushbu tahlilchi kirish sifatida BLAST natijasiga ega fayl ob'ektini oladi va fayl ichidagi har bir yozuv uchun iteratorni qaytaradi. Shu nuqtai nazardan, yozuv har bir BLAST natijasi haqidagi barcha ma'lumotlarga ega bo'lgan ob'ektni ifodalaydi (agar BLAST faylida bir nechta BLAST so'rovlar natijasi bo'lisa,13). U iteratorni qaytargani uchun siz for tsikli yordamid

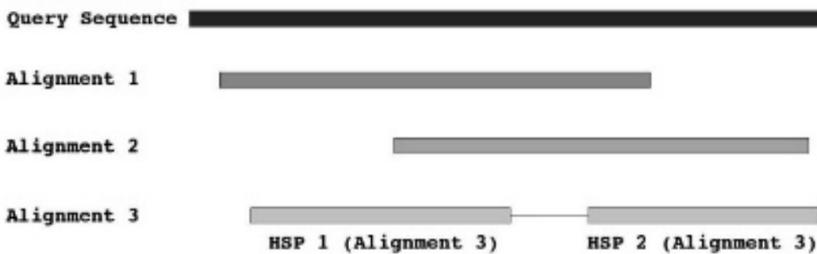
```
Bio.Blast dan NCBIXML.parse(rh)
da blast_record uchun NCBIXML import: #
    blast_record bilan biror narsa qiling
```

### BLAST Record obyektida nima bor?

BLAST faylida mavjud bo'lgan har bir ma'lumotni portlash yozuvi ob'ektidan olish mumkin. Mana katta rasm: BLAST yozuvi BLAST yugurish ma'lumotlarini o'z ichiga oladi. Ushbu ma'lumotlar ikki guruhga bo'lingan. Birinchidan, dasturning xususiyatlari, so'rovlar ketma-ketligi va ma'lumotlar bazasi (masalan, dastur nomi, dastur versiyasi, so'rov nomi, ma'lumotlar bazasi uzunligi, nomi) kabi sobit xususiyatlar mavjud. Boshqa ma'lumotlar guruhi hizalamalar (yoki xitlar) bilan bog'liq. Har bir zarba so'rovlar ketma-ketligi va topilgan maqsad o'rtaсидаги hizalanishdir. O'z navbatida, har bir hizalamada bir nechta HSP (High-scoring Segment Pairs) bo'lishi mumkin. HSP - bu hizalama segmenti. [9.1-rasmida](#) ushbu tushunchalarni yanada qulayroq qilish kerak.

---

13Bu 2.2.14 dan oldingi BLAST versiyalarida XML natijalarini formatlash usulidagi xatodir. bir nechta so'rovlar, shuning uchun siz yangiroq NCBI BLAST versiyalaridan foydalanishingiz kerak.



9.1-rasm BLAST natijasining anatomiyasi. Ushbu so'rovlar ketma-ketligi uchta hizalanishga ega. Har bir hizalamada kamida bitta HSP mavjud. E'tibor bering, hizalama (yoki urish) "Hizani 3" kabi bir nechta HSPga ega bo'lishi mumkin.

BLAST yozushi obyekti ushbu tuzilmani aks ettiradi. U hizalanish xususiyatiga ega, bu (BLAST) hizalama ob'ektlari ro'yxati. Har bir hizalama ob'ektida zarba ma'lumotlari (hit\_id, hit\_definition, sarlavha) va har bir HSP ma'lumotlari bilan ro'yxat (hsps) mavjud. Har bir HSP bilan bog'liq ma'lumotlar odatda BLAST yozuvidan eng ko'p so'raladigan ma'lumotdir (masalan, bit balli, E qiymati, pozitsiyasi). 9.6 Listingda oddiy matnli BLAST chiqishini ko'rib chiqamiz :

#### **Listing 9.6: BLAST chiqishi**

---

#### **BLASTX 2.6.0+**

Malumot: Stiven F. Altsxul, Tomas L. Madden, Alejandro A.

Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller va David J. Lipman (1997),

"Gapped BLAST va PSI-BLAST: oqsil ma'lumotlar bazasini qidirish dasturlarining yangi avlod", Nuklein kislotalari Res. 25:3389-3402.

Ma'lumotlar bazasi: ortiqcha bo'limgan UniProtKB/SwissProt ketma-ketliklari  
466 658 ta ketma-ketlik; Jami 175 602 800 harf

**So'rov = X namunasni**

**Uzunligi = 257**

Muhim hizalanishlarni keltirib chiqaradigan ketma-ketliklar:

Xol	E
(Bits)	Qiymat

## 182 Bioinformatika uchun Python

P04252.1 RecName: To'liq = Bakterial gemoglobin; ... 93.6 Q9RC40.1	1e-34
RecName: To'liq=Flavogemoprotein; AltN... 66.2 Q8ETH0.1 RecName:	2e-17
To'liq=Flavogemoprotein; AltN... 66.6	1e-16

>P04252.1 RecName: To'liq = Bakterial gemoglobin; AltName:  
To'liq=Eriydigan sitoxrom O Uzunligi=146

Bal = 93,6 bit (231), Kutish (2) = 1e-34, Usul: Kompozitsion  
matriksani sozlash.

Identifikatsiya = 45/45 (100%), Ijobiy = 45/45 (100%), Bo'shliqlar = 0/45  
(0%)  
Ramka = +3

123-so'rov VAAAHYPIVGQELLGAIKEVLGDAATDDILDAWGKAYGVIADVFI 257  
VAAAHYPIVGQELLGAIKEVLGDAATDDILDAWGKAYGVIADVFI  
Sbjct 90 VAAAHYPIVGQELLGAIKEVLGDAATDDILDAWGKAYGVIADVFI 134

Bal = 72,8 bit (177), Kutish (2) = 1e-34, Usul: Kompozitsion  
matriksani sozlash.  
Identifikatsiya = 36/36 (100%), Ijobiy = 36/36 (100%), Bo'shliqlar = 0/36  
(0%)  
Ramka = +2

2-so'rov PKALAMTVAAAQNENLPAILPAVKKIAVKHCQAG 109  
PKALAMTVAAAQNENLPAILPAVKKIAVKHCQAG  
Sbjct 54 PKALAMTVAAAQNENLPAILPAVKKIAVKHCQAG 89

>Q9RC40.1 RecName: To'liq=Flavogemoprotein; AltName:  
To'liq=Flavogemoglobin; AltName: To'liq=Gemoglobinga o'xshash  
protein; AltName: To'liq=Azot oksidi dioksigenaza; Qisqa=oksigenaza yo'q;  
Qisqa = NOD uzunligi = 411

Bal = 66,2 bit (160), Kutish (2) = 2e-17, Usul: Kompozitsiyaga  
asoslangan statistika.  
Identifikatsiya = 28/45 (62%), Ijobiy = 37/45 (82%), Bo'shliqlar = 0/45  
(0%)  
Ramka = +3

123-so'rov VAAAHYPIVGQELLGAIKEVLGDAATDDILDAWGKAYGVIADVFI 257

+ YPIVG+ LL A++EVLGDAASDDVLEAWREAYELIADVFI  
**Sbjct 94 IKPEQYPIVGENLLAAMREVLGDAASDDVLEAWREAYELIADVFI 138**

Bal = 41,6 bit (96), Kutish (2) = 2e-17, Usul:  
 Kompozitsiyaga asoslangan statistika.  
 Identifikasiya = 19/32 (59%), Ijobiy = 26/32 (81%), Bo'shliqlar =  
 0/32 (0%)  
 Ramka = +2

2-so'rov PKALAMTVLAAQNIENLPAILPAVKKIAVKH 97  
 P+ALA ++ AAA++I+NL AILP V +IA KH  
**Sbjct 58 PQALANSIYAAAEEHIDNLEAILPVVSRIAHKH 89**

>Q8ETH0.1 RecName: To'liq=Flavogemoprotein;  
 AltName: To'liq=Flavogemoglobin; AltName:  
 To'liq=Gemoglobinga o'xshash protein; AltName: To'liq=Azot oksidi  
 dioksigenaza; Qisqa=oksigenaza yo'q; Qisqa = NOD uzunligi = 406

Bal = 66,6 bit (161), Kutish (2) = 1e-16, Usul: Kompozitsiyaga  
 asoslangan statistika.  
 Identifikasiya = 31/45 (69%), Ijobiy = 37/45 (82%), Bo'shliqlar =  
 0/45 (0%)  
 Ramka = +3

123-so'rov VAAAHHYPIVGQELLGAIKEVLGDAATDDILDAGKAYGVIADVFI 257  
 + YPIVG+ LL AIKEVLGDAATD+I++AW KAY VIAD+FI  
**Sbjct 96 IKPEQYPIVGKYLLIAIKEVLGDAATDEIIAEWEEKYFVIADIFI 140**

Bal = 39,3 bit (90), Kutish (2) = 1e-16, Usul:  
 Kompozitsiyaga asoslangan statistika.  
 Identifikasiya = 22/31 (71%), Ijobiy = 23/31 (74%), Bo'shliqlar =  
 0/31 (0%)  
 Ramka = +2

5-so'rov KALAMTVLAAQNIENLPAILPAVKKIAVKH 97  
 KALA TV AAA NIE L ILP VK+IA KH  
**Sbjct 61 KALANTVYAAAANIEKLEEILPHVKQIAHKH 91**

## 184 Bioinformatika uchun Python

Lambda	K	H	a	alfa
0,318	0,134	0,401	0,792	4.96

Bo'shashgan

Lambda	K	H	a	alfa	sigma
0,267	0,0410	0,140	1.90	42.6	43.6

**Samarali qidiruv maydoni ishlataligan: 4334628608****Ma'lumotlar bazasi: ortiqcha bo'Imagan UniProtKB/SwissProt ketma-ketliklari****E'lon qilingan sana: 2017 yil 14-may, 12:32****Ma'lumotlar bazasidagi harflar soni: 175 602 800****Ma'lumotlar bazasidagi ketma-ketliklar soni: 466 658****Matritsa: BLOSUM62****Bo'shliq jazolari: mavjudligi: 11, uzaytirish: 1****Qo'shni so'zlar chegarasi: 12****Ko'p marta urish uchun oyna: 40**

**9.6 ro'yixati** bu sozlamalardan foydalangan holda SwissProt protein majlumotlar bazasiga nisbatan DNK sojrovlari ketma-ketligining blastx mahsulotidir :

```
./blastx -db db/swissprot -query sampleX.fas -evaluate 1e-15 -outfmt 5
```

standart dastur sozlamalari.14 E'tibor bering, bu natijada uchta tekislash mavjud.

Birinchi va oxirgi hizalamalarda faqat bitta HSP mavjud, ikkinchisida esa ikkita HSP mavjud.

Barcha maqsadli ketma-ketlik nomlari nomini qanday qilib olish mumkinligi **9.7 ro'yxitiga** qarang :

**Listing 9.7: BLASTparser1.py: BLAST chiqishidan hizalamalar sarlavhasini chiqarib oling**

```
1 Bio.Blast dan import NCBIXML 2 ochiq('..../
samples/sampleXblast.xml') bilan xmlfh sifatida: NCBIXML.parse(xmlfh) da yozib
olish uchun: 3 record.alignmentsda tekislash uchun: print(align. sarlavha)
4
5
```

**Listing 9.7 (BLASTparser1.py dasturi) quyidagi natijani beradi:**

```
gi|114816|sp|P04252.1|BAHG_VITST RecName: To'liq=Bakteriya gemoglobi<= in; AltName:
To'liq=Eriydigan sitoxrom O gi|52000645|sp|Q9RC40.1|HMP_BACHD RecName:
To'liq=Flavogemoprotein;<=
```

14Ushbu ro'yxat haqiqiy ishlab chiqarishning qisqartirilgan versiyasi bo'ylib, ba'ysi natijalar ataylab qoldirilgan ortiqcha ma'lumotlarni ko'rsatmaslik va o'quvchining tahlilchi qanday ishlashiga e'tibor qaratishini osonlashtirish.

AltName: To'liq=Flavogemoglobin; AltName: To'liq=Gemoglobinga o'xshash pro<= tein;  
 AltName: To'yliq=Azot oksidi dioksigenaza; Qisqa=oksigenaza yo'q;<=  
 Qisqa=NOD  
 gi|52000637|sp|Q8ETH0.1|HMP\_OCEIH RecName: To'liq=Flavogemoprotein;<= AltName:  
 To'liq=Flavogemoglobin; AltName: To'liq=Gemoglobinga o'xshash pro<= tein; AltName:  
 To'yliq=Azot oksidi dioksigenaza; Qisqa=oksigenaza yo'q;<=  
 Qisqa=NOD

Maqsad uzunligi kabi har bir hizalamadan ko'proq ma'lumot olishingiz mumkin  
 ketma-ketlik va boshqa tegishli ma'lumotlar:

```
>>> align.length 406
```

```
>>> align.hit_id 'gi|  

52000637|sp|Q8ETH0.1|HMP_OCEIH' >>>  

align.hit_def 'RecName: To'liq=Flavogemoprotein;  

AltName: To'liq=Flavogemoglobin; A<= ItName: To'liq=Gemoglobinga o'xshash oqsil;  

AltName: To'liq=Azot oksidi <= dioksigenaza; Qisqa=oksigenaza yo'q; Short=NOD' >>>  

align.hsps [<Bio.Blast.Record.HSP obyekti 0x7fa444665eb8>, <Bio.Blast.Reco<= rd.HSP  

obyekti 0x7fa444665ef0>]
```

hsps HSPlar ro'yxatini o'z ichiga oladi . Har bir HSP misolida, yuqorida aytib o'tilganidek, mavjud  
ko'pchilik foydalanuvchilar BLAST chiqishidan olishni xohlaydigan ma'lumot. HSP ga qarang:

>P04252.1 RecName: To'liq = Bakterial gemoglobin; AltName:  
To'yliq=Eriydigan sitoxrom O Uzunligi=146

Bal = 93,6 bit (231), Kutish (2) = 1e-34, Usul: Kompozitsion  
matriksani sozlash.

Identifikasiya = 45/45 (100%), Ijobiy = 45/45 (100%), Bo'shliqlar = 0/45  
(0%)  
Ramka = +3

123-so'rov VAAAHYPIVGQELLGAIKEVLGDAATDDILDWGKAYGVIADVFI 257  
VAAAHYPIVGQELLGAIKEVLGDAATDDILDWGKAYGVIADVFI  
Sbjct 90 VAAAHYPIVGQELLGAIKEVLGDAATDDILDWGKAYGVIADVFI 134

Ushbu ma'lumotni BLAST tahlilchisi yordamida olish mumkin:

```
>>> xmlfile = '../samples/sampleXblast.xml' >>> blast_records  

= NCBIXML.parse(open(xmlfile)) >>> blast_record = keyingi  

(blast_records)
```

```
>>> align = blast_record.alignments[0] >>>
align.hsps [<Bio.Blast.Record.HSP obyekti
0x7fa444677e80> da, <Bio.Blast.Re<= cord.HSP obyekti 0x7fa444677f28>] >>> hsp
= align.hsps[0] >>> hsp.bits 93.5893

>>> hsp.score
231.0
>>> hsp.expect
1.06452e-34
>>> hsp.identities 45

>>> hsp.align_length 45

>>> hsp.frame
(3, 0) >>>
hsp.query_start 123

>>> hsp.query_end
257
>>> hsp.sbjct_start 90

>>> hsp.sbjct_end
134
>>> hsp.query
'VAAAHYPIVGQELLGAIKEVLGDAATDDILDAWGKAYGVIADVFI'
>>> hsp.match
'VAAAHYPIVGQELLGAIKEVLGDAATDDILDAWGKAYGVIADVFI'
>>> hsp.sbjct
'VAAAHYPIVGQELLGAIKEVLGDAATDDILDAWGKAYGVIADVFI'
```

Shuni yodda tutgan holda, E qiymati chegara qiymatidan past bo'lgan hizalamalarning kirish raqamlari bilan bog'liq savollarga javob bera olamizmi? ([9.8 ro'yxat](#)) va BLAST chiqishidagi har qanday parametr bilan bog'liq boshqa savollar.

**Listing 9.8: BLASTparser2.py: E qiymati ma'lum chegaradan kichik bo'lgan ketma-ketliklarning kirish raqamlarini ajratib oling**

---

**Bio.Blast importidan 1 ta NCBIXML 2  
cheagarasi = 0,0001**

```
3 xmlfh = open('..../samples/other.xml') 4
blast_record = keyingi(NCBIXML.parse(open(xmlfh))) 5
blast_record.alignmentsda tekislash uchun:
```

```

6     if align.hsps[0].expect < pol: print(alignment.accession)
7

```

---

Kod tushuntirildi: Ushbu dastur Listing 9.7 ga juda o'xshaydi . U xml faylidagi birinchi BLAST yozuvini oladi (4-qatordagi keyingi() o'rnatilgan funksiyasiga e'tibor bering). Ushbu usul Biopythonning eski versiyasida NCBIXML.read() usuli yo'qligi sababli ishlatiladi. Agar siz Biopython 1.50 dan foydalanayotgan bo'lsangiz va xml faylida faqat bitta BLAST yozuvi mavjud bo'lsa, NCBIXML.read(open(xmlfh)) dan foydalaning. Dastur blast\_record (5-qatordan boshlab) dagi barcha hizalamalar orqali o'tadi. Har bir tekislash uchun (5-qatorda tekislang), u birinchi HSP ning kutilgan qiymatini tekshiradi (6-qator). Agar E qiymati 2-qatorda belgilangan chegaradan kichik bo'lsa, dastur hizalanishning kirish raqamini chop etadi.

E'tibor bering, BLAST qidiruvini amalga oshirayotganda siz E qiymatini buyruq satridan yoki NcbiblastnCommandline o'ramidan o'rnatishingiz mumkin. Chiqish hosil bo'lgach, siz **9.8 ro'yxitidagi kabi filtrni qo'llashingiz mumkin**.

---

Maslahat: Biopython holda BLAST ishga tushirish va qayta ishlash.

Biopython BLAST qidiruvlarini bajarish va tahlil qilish uchun ishlatilishi mumkin bo'lsa-da, biz olishimiz mumkin Agar kerak bo'lsa, Biopython holda.

BLAST har qanday tashqi dastur sifatida os.system yoki sub process.Popen bilan bajarilishi mumkin. Chiqishni qanday qayta ishlashni rejalashtirganingizga qarab, "m" opsiyasini o'rnatishni unutmang.

BLAST chiqishini qayta ishlashning ikki yo'li mavjud. Agar BLAST chiqishni XML formatida (buyruqlar qatori "-m 7" opsiyasi bilan) ishlab chiqarish uchun sozlangan bo'lsa, natijani **11-bobda** ko'rsatilgan asboblar yordamida tahlil qilish mumkin . BLAST natijalarini tahlil qilishning yana bir oson yo'li CSV modulidan foydalanishdir ( 90-betda ko'rilgan). Buning uchun BLAST chiqishi mos keladigan tarzda formatlanishi kerak (" -m 8 " buyruq qatori opsiyasini o'rnatish).

---

### 9.3.11 Biologik bog'liq ma'lumotlar

Biopython shunchaki asboblar to'plami emas. Unda biologik ma'lumotlar mavjud. Bu majlumotlar Biopython ichiga ichki foydalanish uchun kiritilgan, masalan, tarjima qilish funksiyasi uchun tarjima jadvalari (CodonTable.unambiguous\_dna\_by\_name) va molekulyar\_ogyrilik funksiyasi uchun aminokislota ogyriliklari (oqsil\_ogyriliklari).

Sizning kodingiz ham ushbu ma'lumotlarga kirishi mumkin. Masalan, ushbu interaktiv seansdagi kod "noaniq DNK qiymatini" uning mumkin bo'lgan qiymatlariga aylantiruvchi lug'atga kiradi:

```

>>> Bio.Data dan IUPACData import >>>
IUPACData.ambiguous_dna_values['M']
'AC'
>>> IUPACData.ambiguous_dna_values['H']

```

## 188 Bioinformatika uchun Python

'ACT'

```
>>> IUPACData.ambiguous_dna_values['X']
"GATC"
```

77-betdagি 4.8 ro'yxatidagi protein vazni kalkulyatorini eslaysizmi ? Biopython bilan aminokislolar og'irligi bilan lug'atni aniqlashning hojati yo'q, chunki bunday lug'at allaqachon kiritilgan:

**Listing 9.9: protwwbiopy.py: Biopython bilan oqsil vazni kalkulyatori**


---

Bio.Data.IUPACData dan 1 ta protein\_og'irliklarini pw sifatida import qilish 2

```
protseq = raw_input('Oqsil ketma-ketligini kiriting: ') 3 jami_w = 0
```

Protseqda aa uchun 4:

```
5      total_w += pw.get(aa.upper(),0)
6 jami_w -= 18*(len(protseq)-1) 7 chop('Sof
vazn: {0}'.format(jami_w))
```

---

Olingen dastur asl nusxadan qisqaroq va mos yozuvlar jadvalidan olingen qiymatlar bilan lug'atni aniqlashning hojati yo'q; Biopython de velopers buni siz uchun hal qilsin.

Ko'pgina ma'lumotlar Bio.Data.IUPACData va Bio.Data.Codon saytlarida mavjud Jadval mos ravishda 9.10 va 9.11 ro'yxatlarda keltirilgan .

**Listing 9.10: Bio.Data.IUPACData maъlumotlari**


---

```
protein_letters
extended_protein_letters
ambiguous_dna_letters
unambiguous_dna_letters
ambiguous_rna_letters
unambiguous_rna_letters
ambiguous_dna_complement
ambiguous_dna_values
ambiguous_dna_weight_ranges
ambiguous_rna_complement
ambiguous_rna_values
ambiguous_rna_weight_ranges
avg_ambiguous_dna_weights
avg_ambiguous_rna_weights
avg_extended_protein_weights
extended_protein_values
extended_protein_weight_ranges
protein_weight_ranges
```

---

```
protein_og'irligi
aniq_dna_og'irligi_diapazonlari bir
ma'noli_dna_og'irliklari bir
ma'noli_rna_og'irliklari
```

---

**Listing 9.11: Bio.Data.CodonTable маълумотлари**

```
noaniq_dna_by_id
noaniq_dna_by_name
noaniq_generic_by_id
noaniq_generic_by_name
noaniq_rna_by_id noaniq_rna_by_id
noaniq_rna_by_ad generic_by_name
standart_dna_jadval
```

---

```
standart_rna_jadval
noaniq_dna_by_ad
noaniq_dna_by_nom bir
noaniq_rna_by_id bir
noaniq_rna_by_ism
```

---

Bakterial DNK tarjimasi jadvalini olish uchun:

```
>>> Bio.Data.CodonTable dan import unambiguous_dna_by_id >>>
bact_trans=anbiguous_dna_by_id[11] >>> bact_trans.forward_table['GTC']

"V"
>>> bact_trans.back_table['R']
"CGT"
```

Tarjima jadvalining grafik tasviriga ega bo'lish uchun:

```
>>> Bio.Data dan CodonTable importi >>> chop
CodonTable.generic_by_id[2]
2-jadval Umurtqali mitoxondrial, SGC1
```

U	C	A	G	
U   UUU F	UCU S	UAU Y	UGU C	U
U   UUC F	UCC S	UAC Y	UGC C	C
U   UUA L	UCA S	UAA Stop	UGA W	A

## 190 Bioinformatika uchun Python

U   UUG L	UCG S	UAG Stop	UGG W   G
C   CUU L	CCU P	CAU H	CGU R   U
C   CUC L	CCC P	CAC H	CGC R   C
C   CUA L	CCA P	CAA Q	CGA R   A
C   CUG L	CCG P	CAG Q   CGG R   G	
A   AUU I(lar)  ACU T		AAU N	AGU S   U
A   AUC I(lar)  ACC T		AAC N   AGC S	C
A   AUA M(lar)  MUSHUK		AAA K   AGA Stop  A	
A   AUG M(lar)  ACG T		AAG K   AGG Stop  G	
G   GUU V	GCU A	GAU D	GGU G   U
G   GUC V	GCC A	GAC D	GGC G   C
G   GUA V	GCA A	GAA E	GGA G   A
G   GUG V(lar)  GCG A		GAG E   GGG G	G

## 9.3.12 Entrez

Entrez - Milliy Biotexnologiya Axborot Markazi (NCBI) veb-saytida bir nechta sog'liqni saqlash fanlari ma'lumotlar bazalarini birlashtirgan qidiruv tizimi. Bitta veb-sahifadan siz "ilmiy adabiyotlar, DNK va oqsillar ketma-ketligi ma'lumotlar bazalari, 3D oqsil tuzilishi va oqsil domeni ma'lumotlari, ifoda ma'lumotlari, to'liq genomlar yig'indisi va taksonomik ma'lumotlar" kabi turli ma'lumotlar to'plamlarini qidirishingiz mumkin . <http://www.ncbi.nlm.nih.gov/sites/gquery>

Siz uni har qanday standart qidiruv tizimi sifatida onlayn ishlatishingiz mumkin, lekin uni brauzerdan foydalanish skriptlaringizga ma'lumotlarni kiritish uchun foydali emas. Shuning uchun NCBI "Entrez Programming Utilities" (eUtils) ni yaratdi. Bu Entrez ma'lumotlar bazasini veb-brauzersiz so'rash uchun server tomonidagi vositalar to'plami va qidiruv natijalarini ularni o'z dasturlaringizga kiritish uchun olish uchun ishlatilishi mumkin.

## eUtils bir qarashda

Foydalanuvchi maxsus yaratilgan URL manzilini yaratishi kerak. Ushbu URL NCBI veb-serverida foydalanish uchun dastur nomini va barcha kerakli parametrлarni (masalan, ma'lumotlar bazasi nomi va qidiruv so'zлari) o'z ichiga olishi kerak. Ushbu URL manzili joylashtirilgandan so'ng, NCBI olingan ma'lumotlarni foydalanuvchiga qaytarib yuboradi. Ushbu ma'lumotlar, aksariyat hollarda, XML faylida yuboriladi.

Ushbu protseduraning mantiqiy sababi shundaki, dastur avtomatik ravishda URL manzilini yaratishi, uni joylashtirishi, natijalarini olishi va qayta ishlashi kerak. URL qo'lda emas, balki dastur tomonidan tuzilishi kerak. Olingan XML fayl uchun ham xuddi shunday kutiladi.

15<https://www.ncbi.nlm.nih.gov/books/NBK3807/>

Moslashtirilgan ma'lumotlar quvurlarini yaratish uchun eUtils komponentlarini birlashtirish mumkin ushbu ilovalar ichida.

## Biopython va eUtils

Pythonda URL (urllib2) olish va XML fayllarni (masalan, miniDOM) tahlil qilish vositalari mavjud, shuning uchun u eUtils-ga kirish uchun ishlatalishi mumkin. Hatto eUtils bilan o'zaro ishlash uchun tegishli Python modullaridan foydalanish juda ko'p ishni o'z ichiga oladi. Shu sababli, Biopython Entrez modulini o'z ichiga oladi. Bio.Entrez moduli har bir eUtils dasturiga URL-manzil yaratish yoki XML faylini qanday tahlil qilishni bilmasdan turib qo'ng'iroq qilish funksiyalarini taqdim etadi.

Entrez ma'lumotlar bazasi bilan o'zaro ishlashning ikki yo'lli mavjud: ma'lumotlar bazasini so'rash va haqiqiy ma'lumotlarni olish. Birinchi amal bio.Entrez esearch va egquery funksiyalarini bilan amalga oshirilishi mumkin, efetch va esummary funksiyalarini esa ma'lumotlarni qidirish uchun ishlataladi. [9.3-jadvalda](#) eUtils modulida mavjud bo'lgan barcha funksiyalar jamlangan.

### 9.3-JADVAL eUtils

Ism	Tavsif
olib ketish	Bir yoki bir nechta asosiy identifikatorlar ro'yxatidan yoki foydalanuvchi muhitidan so'ralgan formatdagi yozuvlarni oladi.
einfo	Har bir ma'lumotlar bazasi uchun maydon indeksi atamalarini, oxirgi yangilanishni va mavjud havolalarni taqdim etadi.
egquery	Global Query yordamida bitta qidiruv uchun XML formatidagi Entrez ma'lumotlar bazasi hisobini ta'minlaydi.
elink	Bir yoki bir nechta asosiy identifikatorlar ro'yxatidan tashqi yoki Tegishli maqolalar havolasi mavjudligini tekshiradi.
epos	Keyingi qidiruv strategiyalarida foydalanish uchun foydalanuvchi muhitida kelajakda foydalanish uchun asosiy identifikatorlar ro'yxatini o'yz ichiga olgan faylni joylashtiradi.
izlanish	Asosiy identifikatorlarni qidiradi va oladi (EFetch, ELink va ESummary da foydalanish uchun).
espell	Imlo bo'yicha takliflarni oladi.
esummary	Asosiy identifikatorlar ro'yxatidan yoki foydalanuvchi muhitidan hujjat xulosalarini oladi.
o'qing	Yuqorida funksiyalardan biri tomonidan qaytarilgan XML natijalarini tahlil qiladi.

### eUtils: Bibliografiyani olish

Quyidagi skript PubMed-ni Entrez orqali so'rildi. PubMed - bu MEDLINE uchun qidiruv tizimi, hayot haqidagi fanlar va biotibbiyot ma'lumotlarining adabiyotlar bazasi.

**Listing 9.12: entrez1.py: PubMed'dan ma'lumotlarni olish va ko'rsatish**

```

1 Bio import Entrez 2 my_em =
'user@example.com' 3 db = "nashr
qilingan"
4 # eUtils'dan esearch yordamida Entrez veb-saytida qidiruv 5 # esearch
dastakni qaytaradi (h_search deb ataladi) 6 h_search = Entrez.esearch(db=db,
email=my_em, termin='python va bioinformatika') 7

8 # Natijani Entrez.read() bilan tahlil qiling 9 yozuv =
Entrez.read(h_search)
10 # Oldingi qidiruv orqali qaytarilgan identifikatorlar ro'yixatini oling 11
res_ids = rekord["IdList"]
12 # Ro'yxatdagi har bir identifikator uchun
res_idlarda r_id uchun 13:
14     # Har bir identifikator uchun xulosa ma'lumotlarini
15     oling h_summ = Entrez.esummary(db=db, id=r_id, email=my_em)
16     # Natijani Entrez.read() yordamida tahlil qiling summ
17     = Entrez.read(h_summ) print(summ[0]['Title'])
18     print(summ[0]['DOI']) print('=====
19     =====')
20

```

**Listing 9.12** ishga tushirilganda ishlaydigan Internet aloqasi mavjud bo'lsa , u quyidagicha chiqadi:

Qisqa ketma-ketlik o'qishlarining optimal birlashtirilgan hizalamalari.

10.1093/bioinformatika/btn300

---

Protein tuzilishi ansamblari uchun aralashma modellari. 10.1093/
bioinformatika/btn396

---

NMR rezonansini tayinlash uchun kontaktni almashtirish. 10.1093/
bioinformatika/btn167

---

## eUtils: Gen ma'lumotlarini olish

eUtils bir nechta ma'lumotlar bazalari uchun interfeys bo'lganligi sababli, bibliografik ma'lumotlarni olish uchun ishlataladigan bir xil dastur ([Ro'yxat 9.12](#)) gen ma'lumotlarini olish uchun ishlatalishi mumkin. [9.13 ro'yxatidagi](#) asosiy o'zgarish ma'lumotlar bazasi maydonidir (3-qator).

**Listing 9.13: entrez2.py: PubMed'dan ma'lumotlarni olish va ko'rsatish**

```

1 Bio import Entrez 2 my_em =
'user@example.com' 3 db = "gen" 4
termin = 'kobalamin sintaza homo
sapiens' 5 h_search = Entrez.esearch(db=db, email=my_em,
term=term) 6 rekord = Entrez.read(h_search) 7 res_ids = rekord["IdList"]
res_idsdagi r_id uchun 8: h_summ = Entrez.esummary(db=db, id=r_id,
email=my_em) summa = Entrez.read(h_summ) print(r_id) print(summa[0]['Tavsif'])
chop etish(summa[0]['Xulosa']) chop etish('=====
9      =====')
10
11
12
13
14

```

---

### **9.13 (entrez2.py) rojyxati quyidagi natijani beradi:**

**326625**

metilmalonik atsiduriya (kobalamin tanqisligi) cbIB turi Bu gen vitamin B(12) ni adenosilkobalaminga (AdoCb<= I), metilmalonil-KoA m< uchun vitamin B12 o'z ichiga olgan koenzimga aylantirishning <= yakuniy bosqichini katalizlovchi oqsilni kodlaydi. = utase. Gendagi mutatsiyalar vitamin B12-dep<= endent metilmalonik atsiduriyaning cbIB komplementat<= ion guruhi bilan bog'langan sababidir. [RefSeq tomonidan taqdim etilgan]

=====

**4524**

**5,10-metilentetrahidrofolat reduktaza (NADPH)**

Metilentetrahidrofolat reduktaza (EC 1.5.1.20) t<= 5,10-metilentetrahidrofolatning 5-metilta<= trahidrofolatga, homosisteining metioninga <= remetilatsiyasi uchun kosubstratga aylanishini katalizlaydi.[OM tomonidan taqdim etilgan.

=====

(...)

E'tibor bering, ushbu chiqishda 9.12-listing natijasida mavjud bo'limgan raqam mavjud . Bu raqam qidiruv funksiyasi tomonidan qaytarilgan identifikatordir. Bu identifikator esummary funksiyasi bilan xulosani olish uchun ishlataligan. Keyingi kod haqiqiy DNK ketma-ketligini olish uchun ushbu identifikatoridan foydalananadi:

```

>>> n = "nukleotid"
>>> tutqich = Entrez.efetch(db=n, id="326625", rettype='fasta') >>> print handle.read()
>gi|326625|gb|M77599.1|HIVED82FO Inson immunitet tanqisligi virusi< =

```

1 turdag'i gp120 (env) gen ketma-ketligi

```
TTAATAGTACTTGGATTCAACATGGGATTAAACACAACCTTAATAGTACTCAGAATAAAGA
AGAAAATATCACACTCCCAGTAGAATAAAAAAAATTATAAACATGTGGCAGGAAGTAGGA
AAAGCAATGTATGCCCTCCCATCAAAGGACAAATTAAATGTTCATCAAATATTACAGGGC
TACTATTAACAAGAGATGGTGGTAATAGTGGAACAAAAGCAACGACACCACCGAGACCTT
CAGACC
```

Retyupe parametrini "genbank" ga o'zgartirib, oddiy ketma-ketlik o'rniga GenBank yozuvini olishingiz mumkin. Ketma-ket GenBank formatida bo'lidan so'ng, u 173-betda ko'satilganidek, uni SeqIO moduli yordamida tahlil qilishi mumkin. Natijalarni tahlil qilishning muqobil usuli ularni XML formatida olish va keyin ularni Entrez.read() funksiyasi bilan tahlil qilishdir:

```
>>> tutqich = Entrez.efetch(db=n, id="326625", retmode='xml') >>> rekord[0]
['GBSeq_moltype']
"RNK"
>>> record[0]['GBSeq_sequence']
'ttaatagtacttggattcaacatgggatTTAACACAACCTTAATAGTACTCAGAATAAAGA<=
agaaaatatacacactcccAGTAGAATAAAACAAATTATAAACATGTGGCAGGAAGTAGGA<=
aagcaatgtatGCCCTCCCATCAAAGGACAAATTAAATGTTCATCAAATATTACAGGGC<=
ctattaacaagagatGGTGGTAATAGTGGAACAAAAGCAACGACACCACCGAGACCTTcag<= acc'
```

```
>>> rekord[0]['GBSeq_organism']
"Odamning immunitet tanqisligi virusi 1"
```

### 9.3.13 PDB

PDB fayllari Protein ma'lumotlar bankida saqlanadigan molekulalarning uch o'lchovli tuzilmalari haqidagi ma'lumotlarni saqlaydi.

Ellik mingdan ortiq yozuvlarga ega bo'lgan ushbu ma'lumotlar bazasi protein tarkibiy ma'lumotlarining mos yozuvlar ombori hisoblanadi. PDB fayli rentgen kristallografiyasi, NMR spektroskopiyasi va boshqa eksperimental usullar yordamida olingan atomlarning fazoviy pozitsiyalarini saqlaydi.

Bu ma'lumotlar Deep View,16 Cn3D17 va PyMol18 kabi molekula tuzilishini ko'yruvchilar, oqsil tahlillari va MakeMultimer19 va Modeller.20 kabi tuzilmalarni bashorat qilish dasturlari kabi bir qancha dasturlar tomonidan qo'llaniladi.

Ushbu ma'lumotlar bazasining yozuvlariga RCSB veb-sahifasi orqali http orqali kirish mumkin :  
[/www.rcsb.org/pdb/home/home.do](http://www.rcsb.org/pdb/home/home.do)

<sup>21</sup> Agar siz o'zingizning arizangizni yaratmoqchi bo'lsangiz

---

16<http://spdbv.vital-it.ch> 17<http://www.ncbi.nlm.nih.gov/Structure/CN3D/cn3d.shtml> 18<http://www.pymol.org> 19<http://watcut.uwaterloo.ca/cgi-bin/makemultimer>  
 20<http://www.salilab.org/modeller> 21RCSB - bu strukturaviy bioinformatika bo'yicha tadqiqot hamkorligi, mas'ul konsortsium

PDB boshqaruvi.

oqsil tuzilishi ma'lumotlarini tahlil qilish uchun dasturingiz PDB fayllaridan ma'lumotlarni tahlil qila olishi kerak. Bu Bio.PDB modulining roli.

Bio.PDB modulidan samarali foydalanish uchun avvalo PDB fayl tuzilishini tushunishingiz kerak. Protein tuzilishi yuqorida pastga ierarxiya bilan modellashtirilgan. U tuzilish sinfigacha, atom kichik sinfigacha. Oraliq buyurtmalar model, zanjir va qoldiqdir. Ushbu ierarxiya SMCRA sifatida ham tanilgan. Ba'zi oqsillar bu naqshga amal qilmaydi, lekin PDB fayllari.<sup>22</sup>

## Bio.PDB moduli

PDB moduli PDBParser sinfini taqdim etadi.<sup>23</sup> Bu sinf get\_structure usuliga ega. Ushbu usul kirish sifatida identifikator va fayl nomiga muhtoj va u struktura ob'ektini qaytaradi. Ushbu SMCRA ierarxiyasiga kalit sifatida identifikator orqali kirish mumkin:

```
>>> Bio.PDB.PDBParser dan import PDBParser >>>
pdbfn = ' ../../samples/1FAT.pdb' >>> parser =
PDBParser(PERMISSIVE=1) >>> structure =
parser.get_structure(" 1 yog ", pdf)
OGOHLANTIRISH: A zanjiri 7808-qatorda uzilib qolgan.

(... ba'zi ogohlantirishlar olib tashlandi ...)
OGOHLANTIRISH: D zanjiri 7870-qatorda uzuksiz.

>>> structure.child_list [<Model
id=0>] >>> model = structure[0]
>>> model.child_list [<Zanjir
identifikatori=A>, <Zanjir
identifikatori=B>, <Zanjir identifikatori=C> , <Zanjir identifikatori=D>, <=
<Zanjir identifikatori= >] >>> zanjir = model['B'] >>> chain.child_list[:5]

[<Rezidue SER het= resseq=1 icode= >, <Rezidue ASN het= <= resseq=2
icode= >, <Rezidue ASP het= resseq=3 icode= >, <= <Qoldiq ILE het=
resseq=4 icode = >, <Residue TYR het= <= resseq=5 icode= >] >>> qoldiq =
zanjir[4] >>> residue.child_list [<Atom N>, <Atom CA>, <Atom C>, <Atom
O>, <Atom CB>, <= <Atom CG1>, <Atom CG2>, <Atom CD1>] >>> atom =
qoldiq['CB']
```

>>> atom.bfaktor

---

**22**U erda noto'g'ri tuzilgan PDBlar mavjud. Bio.PDB moduli muammoni topganda, PERMISSIVE argumentiga qarab istisno yoki ogohlantirish hosil qilishi mumkin (tolerantlik uchun 0 va ogohlantirishlarni chiqarish uchun tahlilchi uchun 1).

23Ba'zi Linux o'rnatishlarida ushbu modul uchun python-numeric-ext paketini o'rnatishingiz kerak. yugurmoq.

## 196 Bioinformatika uchun Python

```
14.1300000000000001
>>> atom.coord
massiv([ 34.30699921, -1.57500005, 29.06800079], 'f')
```

Quyidagi dastur gzip bilan siqilgan PDB faylini ochadi. oqsilning barcha zanjirlari orqali va har bir zanjirda tartibsiz atom mavjud bo'lganda qoldiq va atom nomini chop etish uchun har bir qoldiqdagi barcha atomlar bo'ylab yuradi:

24

Skanerlaydi

**Listing 9.14: pdb2.py: gziplangan PDB faylini tahlil qilish**

```
Bio.PDB.PDBParser
dan 1 import gzip 2 import PDBParser 3 4 def
buzilish(tuzilma): strukturada zanjir uchun[0].get_list():
    chain.get_list()dagi qoldiq uchun:
5
6
7             residue.get_list()dagi atom uchun: agar
8                     atom.is_disordered(): chop
9                     etish(qoldiq, atom)
10            Qaytish Yo'q
11
12 pdbsfn = '../samples/pdb1apk.ent.gz' 13 tutqich =
gzip.GzipFile(pdbsfn) 14 parser = PDBParser() 15
struktura = parser.get_structure("test", tutqich) 16
tartibsizlik(struktura) )
```

### 9.3.14 PROZIT

**PROSITE** - bu protein domenlari, oilalar va funktsional saytlarni, shuningdek ularni aniqlash uchun ishlataladigan tegishli naqshlar va profilarni tavsiflovchi hujjat yozuvlari ma'lumotlar bazasi.

Ushbu ma'lumotlar bazasiga PROSITE sayti <http://www.expasy.org/> orqali kirish mumkin . [org/prosite](#) yoki bitta oddiy matnli fayl sifatida taqsimlanadi.25 Ushbu faylini Prosite modulidagi parse funksiyasi yordamida tahlil qilish mumkin :

```
>>> from Bio import Prosite >>>
handle = open("prosite.dat")
```

24gzip faylini siqish uchun \*nix tizimlarida qo'llaniladigan "standart" dasturdir, lekin u eng keng tarqalgan platformalarda ham mavjud. Bu misolda ko'rsatilgan, chunki hamma uchun ochiq bo'lgan molekulyar ma'lumotlar fayllari ushbu formatda siqilgan.

25.2008 yil 2-sentyabrdagi 20.36-versiyasi 22 Mb hajmli fayl bo'lib, <ftp://ftp.expasy.org/databases/> manzilida mavjud. prosite/prosite.dat.

```
>>> yozuvlar = Prosite.parse (tutqich)
>>> yozuvlardagi r uchun:
    print(r.accession)
    print(r.name)
    print(r.description)
    print(r.pattern) print(r.created)
    print(r.pdoc)
    print("=====")
    =====")
```

**PS00001**

**ASN\_GLYCOSYLATION**

N-glikosillanish joyi.

N-{P}-[ST]-{P}.

APR-1990

PDOC00001

=====

**PS00004**

**CAMP\_PHOSPHO\_SITE**

cAMP- va cGMP-ga bog'liq protein kinaz fosforillanish joyi.

[RK](2)-x-[ST].

APR-1990

PDOC00004

=====

**PS00005**

**PKC\_PHOSPHO\_SITE**

Protein kinaz C fosforillanish joyi.

[ST]-x-[RK].

APR-1990

PDOC00005

=====

### 9.3.15 Cheklash

Rekombinant DNK texnologiyasi odatda birga bo'lmaydigan DNK ketma-ketliklarini (odatda turli organizmlardan) birlashtirish imkoniyatiga asoslanadi.

Ushbu turdag'i biologik kesish va yopishtirish maxsus molekulyar qaychi sifatida ishlaydigan fermentlarning maxsus guruhi bo'lgan cheklovchi endonukleaza tomonidan amalga oshiriladi.

Ushbu fermentlarning asosiy xususiyati shundaki, ular nukleotidlarning ma'lum bir ketma-ketligini taniydiilar va ikkala DNK zanjirini kesib tashlaydilar. Tadqiqotchi ma'lum DNK ketma-ketligidagi kesmani kiritmoqchi bo'lganida, u birinchi navbatda qaysi fermentni tekshirishi kerak.

## 198 Bioinformatika uchun Python

ketma-ketlik ichidagi sayt uchun o'ziga xos xususiyatga ega. Barcha mavjud cheklash fermentlari REBASE.26 deb nomlangan ma'lumotlar bazasida saqlanadi

Mashhur cheklovchi ferment EcoRI hisoblanadi. Bu ferment "GAATTC" ketma-ketligini taniydi. Shunday qilib, bu ferment shu ketma-ketlikka ega bo'lgan har qanday ikki zanjirli DNKnii kesadi

**CGCGAATTCTGCG  
GCGCTTAAGCGC**

Bunday holda, cheklash joyi yuqori ipning o'rtasida joylashgan ('-' bilan belgilangan): CGC-GAATTC-GCG. Ajratilgan qismlar quyidagicha ko'rindi:

**CGC            GAATTCGCG  
GCGCTTAA        GCGC**

### Bio.Cheklash moduli

Biopython cheklash fermentlari bilan ishlash uchun vositalarni, shu jumladan REBASE-dan olingan ferment ma'lumotlarini taqdim etadi. Barcha cheklash fermentlari Cheklov modulida mavjud:

```
>>> dan Bio import cheklash >>>
```

```
Restriction.EcoRI
```

```
EcoRI
```

Cheklash fermenti ob'ektlari DNK ketma-ketligidagi cheklash joylarini qidirish uchun ishlatalishi mumkin bo'lgan qidiruv kabi bir nechta usullarga ega:

```
>>> Bio.Seq dan import Seq >>>
```

```
Bio.Alphabet.IUPAC dan import IUPACambiguousDNA >>> alfa =
```

```
IUPACambiguousDNA() >>> gi1942535 = Seq('CGCGAATTCTGCG', alfa)
```

```
>>> Restriction.EcoRI.search(gi1942535) [5]
```

E'tibor bering, qidiruv funktsiyasi ferment kesadigan barcha pozitsiyalar bilan ro'yxatni qaytaradi. Lavozim kesilgandan keyingi birinchi nukleotid bo'lib, 0 o'rniغا 1 dan boshlanadi (odatdagidek Pythonning boshqa qismlarida). Qidiruvdag'i yana bir parametr chiziqli. U sukul bo'yicha False bo'lib, ketma-ketlik aylana bo'lsa, True sifatida o'rnatilishi kerak.

Cheklovdan keyin hosil bo'lgan segmentlarni katalizlash funktsiyasi bilan ko'rish mumkin:

```
>>> Cheklov.EcoRI.catalyse(gi1942535)
```

```
(Seq('CGCG', IUPACambiguousDNA()), Seq('AATTCTGCG', <= IUPACambiguousDNA()))
```

Bir vaqning o'zida bir nechta fermentlarni tahlil qilish uchun **RestrictionBatch** klassi mavjud:

```
>>> enz1 = Cheklash.EcoRI
>>> enz2 = Cheklash.HindIII
>>> batch1 = Restriction.RestrictionBatch([enz1, enz2]) >>>
batch1.search(gi1942535)
{EcoRI: [5], HindIII: []}
```

Fermentlar to'plamida qo'llaniladigan qidiruv funktsiyasi lug'atni qaytaradi:

```
>>> dd = batch1.search(gi1942535) >>>
dd.get(Restriction.EcoRI) [5] >>>
dd.get(Restriction.HindIII) []
```

Fermentlar xuddi **RestrictionBatch** misoli to'plam kabi qo'shilishi yoki olib tashlanishi mumkin:

```
>>> batch1.add(Restriction.Earl)
>>> 1-to'plam
RestrictionBatch(['Earl', 'EcoRI', 'HindIII']) >>>
batch1.remove(Restriction.Earl)
>>> 1-to'plam
RestrictionBatch(['EcoRI', 'HindIII'])
```

Cheklash modulida **AllEn** kabi oldindan belgilangan to'plamlar ham mavjud zymes, **CommOnly** va **NonComm**:

```
>>> batch2 = Restriction.CommOnly
```

### Tahlil sinfi: Hammasi bitta

Tahlil klassi bir nechta fermentlar bilan ishlashni osonlashtiradi:

```
>>> an1 = Cheklov.Tahlil(part1,gi1942535) >>> an1.full()
{HindIII: [], EcoRI: [5]}
```

Shu nuqtaga qadar **Analysis** obyekti dagi **full()** usulining natijasi hisoblanadi **RestrictionBatch** orqali qidiruv bilan bir xil. Tahlil quyidagi larni ta'minlaydi:

## 200 Bioinformatika uchun Python

```
>>> an1.print_that()
```

EcoRI : 5.

Ketma-ketlikni kesmaydigan fermentlar.

HindIII

```
>>> an1.print_as('map') >>>
an1.print_that()
```

5 EcoRI  
|  
CGCGAATTGCG  
|||||||  
GCGCTTAAGCGC  
1 12

Ketma-ketlikni kesmaydigan fermentlar.

HindIII

```
>>> an1.faqat_orasida(1,8)
{EcoRI: [5]}
```

Bu Cheklash modulida mavjud bo'lgan ko'pgina funktsiyalarni qamrab oladi. Qo'shimcha ma'lumot olish uchun <http://biopython.org/DIST/docs/cookbook/Restriction.html>.

### 9.3.16 SeqUtils

Ushbu modul DNK va oqsil ketma-ketligi bilan shug'ullanish uchun bir nechta funktsiyalarga ega, masalan, CG, GC skew, molekulyar og'irlik, nazorat yig'indisi algoritmlari, Kodondan foydalanish, erish harorati va boshqalar. Barcha funktsiyalar to'g'ri hujjatlashtirilgan, shuning uchun men ulardan qanday foydalanishni tushunish uchun faqat bir nechta funktsiyalarni tushuntiraman.

#### DNK Utils

SeqUtils DNK ketma-ketligiga qo'llanilishi mumkin bo'lgan juda ko'p funktsiyalarga ega. Keling, ulardan ba'zilarini ko'rib chiqaylik: GC tarkibi: guanin yoki sitozin bo'lgan asoslar

foizi DNK molekulasining ba'zi jismoniy xususiyatlariga ta'sir qiluvchi parametrdir. U GC funktsiyasi bilan hisoblab chiqiladi:

```
>>> Bio.SeqUtils import GC dan
```

```
>>> GC('gacgatcggattcgttag')
50,0
```

DNKning erish harorati: Bu Erish Temp.Tm\_staluc funksiyasi bilan hisoblanishi mumkin. Bu funksiya "eng yaqin qo'shni usuli"<sup>27</sup> ni amalga oshiradi va DNK va RNK ketma-ketligi uchun ishlatalishi mumkin:

```
>>> dan Bio.SeqUtils import MeltingTemp >>>
MeltingTemp.Tm_staluc('tgcagttacgtatcgt') 42.211472744873447
>>> chop etish ("%2f"%MeltingTemp.Tm_staluc('tgcagttacgtatcgt')) 42.21
```

**CheckSum funktsiyalari:** Tekshirish summasi odatda kirish fayliga asoslangan qisqa harf-raqamli qator bo'lib, asosan ma'lumotlar yaxlitligini tekshirish uchun ishlatalidi. Har qanday turdag'i ma'lumotlardan (masalan, matnli fayl, DNK ketma-ketligi) algoritm yordamida siz asl ma'lumotlarni aks ettira oladigan kichik qatorni (odatda "imzo" deb ataladi) yaratishingiz mumkin. Ba'zi dasturlar ma'lumotlar yaxlitligini ta'minlash uchun ketma-ketlik ma'lumotlariga nazorat summasi ma'lumotlarini biriktiradi. Oddiy nazorat summasi GCG dasturi tomonidan amalga oshiriladi.

Bu gcg formatidagi ketma-ketlik:

ID AB000263 standarti; RNK; PRI; 368 BP.

XX

AC AB000263;

XX

DE Homo sapiens peptid kabi prepro kortistatin uchun mRNK.

XX

SQ Sequence 37 BP;

AB000263 Uzunlik: 37 Tekshirish: 1149 .. 1 acaagatgcc

attgtcccccc ggcctccgc tgctgt

Tekshirish raqami (bu holda 1149) ketma-ketlikdan olingan. Agar ketma-ketlik o'zgartirilsa, raqam (umid qilamanki) o'zgartiriladi. Har doim tasodifiy to'qnashuv ehtimoli bor, ya'ni ikki xil ketma-ketlik bir xil imzoni hosil qilganda. "Gcg nazorat summasi" zaifdir, chunki u faqat 10000 xil imzoga ruxsat beradi. Shuning uchun crc32, crc64 va seguid kabi kuchli nazorat summalarini mavjud.<sup>28</sup> Bu nazorat summalarining barchasi CheckSum modulida mavjud. Ular nazorat yig'indisi algoritmi zaifdan eng kuchligacha bo'lgan tartibda ko'rsatilgan.

---

<sup>27</sup>Eng yaqin qo'shni usuli haqida ko'proq ma'lumot olish uchun "Santalucia, et al. (1996) Biokimyo 35, 3555-3562.

<sup>28</sup>Bassi, Sebestyan va Gonza lez, Virjiniyadagi nazorat summalarini haqida qo'yshimcha ma'yumot olish uchun qarang. Biopython uchun yangi nazorat summasi funksiyalari." Nature Precedings dan mavjud <<http://dx.doi.org/10.1038/npre.2007.278.1>> (2007).

```
>>> Bio.SeqUtils dan import CheckSum >>> myseq
= 'acaagatgccattgtccccggcctcgtgcgtgc' >>> CheckSum.gcg(myseq)
1149

>>> CheckSum.crc32(myseq)
-2106438743

>>> CheckSum.crc64(myseq)
"CRC-A2CFDBE6AB3F7CFF"

>>> CheckSum.seguid(myseq)
'9V7Kf19tfPA5TntEP75YiZEm/9U'
```

### Protein vositalari

**Protein bilan bog'liq funktsiyalarga ProtParam sinfidan kirish mumkin. Mavjud protein xususiyatlari:** molekulyar ogl'irlik, aromatiklik, beqarorlik indeksi, moslashuvchanlik, izoelektrik nuqta va ikkilamchi tuzilish fraktsiyasi. **Funktsiya nomlari oddiy.**

Ularni 9.15 roýyxatida koýring :

**Listing 9.15: protparam.py: PropParam funktsiyalarini oqsillar guruhiga qo'llang**

---

```
1 Bio.SeqUtils.ProtParam import ProteinAnalysis 2 Bio.SeqUtils dan
ProtParamData 3 Bio import SeqIO 4 dan import
```

5 ochiq('..../samples/pdbaa') fh sifatida:

```
SeqIO.parse(fh,'fasta') da rec uchun: 6
7      myprot = ProteinAnalysis(str(rec.seq))
8      print(myprot.count_amino_acids())
9      print(myprot.get_amino_acids_percent())
10     print(myprot.molecular_weight())
11     print(myprot.aromaticity()) print(myprot.instability_index() )
12     print(myprot.flexibility()) print(myprot.izolektrik_nuqta())
13     print(myprot.secondary_structure_fraction())
14     print(myprot.protein_scale(ProtParamData.kd, 9, .4))
15
16
```

---

### 9.3.17 Sequencing

Sequencing loyihalari odatda .ace va .phd.1 fayllarini yaratadi.<sup>29</sup>

**29**Bu ketma-ketlik texnologiyasiga bog'liq; bu fayllar Phred, Phrap, CAP3 va Consed kabi mashhur ketma-ketlikni qayta ishlash dasturlari yordamida ketma-ketlik izi xromatogrammasini qayta ishlash orqali hosil qilinadi.

## Phd fayllar

DNK sequencer iz ma'lumotlari Phred dasturi tomonidan o'qiladi. Bu dastur bazalarni chaqiradi, bazalarga sifat qiymatlarini tayinlaydi va asosiy chaqiruvlar va sifat qiymatlarini chiqish fayllariga yozadi (.phd.1 kengaytmasi bilan).

Quyidagi kod ([9.16 roýyxati](#)) .phd.1 fayllaridan maylumotlarni qanday chiqarishni koýrsatadi :

---

### **Listing 9.16: phd1.py: .phd.1 faylidan maylumotlarni ajratib oling**

---

#### Bio.Sequencing

```
import Phd 3 dan 1 ta import pprint 2

4 fn = ' ../../sample/phd1' 5 fh =
ochiq(fn) 6 rp = Phd.RecordParser()

7 # Iterator yarating
8 it = Phd.Iterator(fh, rp) 9 undagi r
uchun:
10      # Barcha sharhlar pprint pprint(r.comments)
11      lug'atida
12      # Ketma-ketlik ma'lumotlarini
13      chop etish ('Tartib: %s' % r.seq)
14      # Har bir asosiy bosma uchun sifat
15      maylumoti('Sifat: %s' % r.saytlar) 16 fh.close()
```

---

Agar siz faqat ketma-ketlikni ajratib olishni istasangiz, SeqIO dan foydalanish osonroq:

```
>>> Bio import SeqIO >>> fn =
' ../../samples/phd1' >>> fh = open(fn)
>>> seqs = SeqIO.parse(fh,'phd') >>>
seqs = SeqIO.parse(fh,'phd') >>> s uchun
ketma-ketlikda: print(s.seq)
```

ctccgtcggAACATCATCGGATCCTATCACAGAGTTTGAAACGAGTTCTCG (...)

## Ace fayllar

Odatiy ketma-ketlik strategiyasida bir-biriga o'xshash bir nechta ketma-ketliklar (yoki "o'qishlar") elektron shaklda bitta uzun qo'shni ketma-ketlikda birlashtirilgan. Bu qo'shni ketma-ketlik

"contig" deb nomlanadi va CAP3 va Phrap kabi ixtisoslashtirilgan dasturlarda ishlab chiqariladi. Con tig fayllar ko'rish yoki keyingi tahlil qilish uchun ishlatiladi. Biopython-da Ace modulida ACEParser mavjud. Har bir .ace fayli uchun kontiglar sonini, o'qishlar sonini va ba'zi fayl ma'lumotlarini olishingiz mumkin:

```
>>> dan Bio.Sequencing import Ace >>>
fn='836CLEAN-100.fasta.cap.ace' >>>
acefilerecord=Ace.read(open(fn)) >>>
acefilerecord.ncontigs 87

>>> acefilerecord.nreads
277
>>> acefilerecord.wa[0].info ['phrap'
304_nucIsu.fasta.screen -new_ace -retain_duplicates', <= 'phrap versiyasi 0.990329']
>>> acefilerecord.wa[0].date

'040203:114710'
```

Ace.read shuningdek, 9.17 ro'yxatda ko'rsatilganidek, har bir kontigmaning tegishli ma'lumotlarini oladi.

**Listing 9.17: ace.py: ".ace" faylidan ma'lumotlarni olish**

---

#### Biodan 1. Sequencing import Ace 2

```
3 fn = '../samples/contig1.ace' 4
acefilerecord = Ace.read(open(fn))
5
6 # Har bir kontig uchun:
acefilerecord.contigsdagi ctg uchun 7: 8
    chop etish('=====')
9    chop etish(' Contig nomi: %s'%ctg.name) print('Asosiyalar:
10   %s'%ctg.nbases) print('O'qilganlar: %s'%ctg.nreads)
11   print('Segmentlar: %s'%ctg.nsegments) print('Sequence:
12   %s'%ctg.sequence) print('Sifat: %s'%ctg.quality)
13
14
15 # Contig ichida har bir o'qish
16 uchun: ctg.reads da o'qish uchun:
17     print('O'qish nomi: %s'%read.rd.name)
18     print('Align start: %s'%read.qa.align_clipping_start) print('Align end:
19     %s'%read.qa.align_clipping_end) print('Qual start:
20     %s'%read.qa.qual_clipping_start) print('Qual end:
21     %s'%read.qa.qual_clipping_end)
```

```
22      print('O'qish ketma-ketligi: %s'%read.rd.sequence) chop
23      etish('===== ======')
```

---

### 9.3.18 SwissProt

SwissProt30 - qo'lida izohlangan proteinlar ketma-ketligi ma'lumotlar bazasi. UniProt konsorsiumini tashkil etuvchi Shveytsariya bioinformatika instituti (SIB) va Yevropa bioinformatika instituti (EBI) tomonidan hamkorlikda olib borilmoqda. U yuqori darajadagi izoh bilan bog'liq ishonchli oqsil ketma-ketligi bilan mashhur va oqsillar uchun ma'lumot basasi hisoblanadi. 2008 yil sentyabr holatiga ko'ra u deyarli 400 000 ta yozuvga ega, butun UniProt ma'lumotlar bazasida 6 000 000 dan ortiq yozuvlar mavjud. Uning kichik o'lchami qo'lida ishlov berish jarayoni bilan bog'liq.

SwissProt fayllari inson o'quvchilarini, shuningdek, kompyuter dasturlari tomonidan foydalanan uchun tuzilgan matnli fayllardir. Ushbu fayl formati uchun texnik xususiyatlar <http://www.expasy.org/sprot/userman.html>, lekin uni Biopython bilan tahlil qilish uchun uning ichki xususiyatlarni bilişning hojati yo'q.

SwissProt faylining namunasi quyida ko'rsatilgan:<sup>31</sup>

ID 6PGL\_ECOLC    Ko'rib chiqilgan;                                  331 AA.

AC B1IXL9; DT 20-

MAY-2008, UniProtKB/Swiss-Prot-ga integratsiyalashgan.

DT 29-APR-2008, ketma-ketlik 1-versiyasi.

DT 02-SEP-2008, kirish versiyasi 5.

DE RecName: To'liq=6-fosfoglyukonolaktonaza; DE Short=6-P-glyukonolaktonaza; EC 3.1.1.31; GN nomieng; OS

DE    Escherichia coli (shtammi ATCC 8739 / DSM 1576 / Crooks).

OC bakteriyalari; Proteobakteriyalar; gammaproteobakteriyalar; Enterobacteriales; OC Enterobacteriaceae; Escherichia.

OX NCBI\_TaxID=481805; RN

[1]

RP nukleotidlardan ketma-ketligi [katta miqyosli genomik DNK].

RA Copeland A., Lukas S., Lapidus A., Glavina del Rio T., Dalin E., RA Tice X., Bryus D., Gudwin L., Pitluck S., Kiss H., Brettin T.; RT "Escherichia coli C str. ATCC 8739 ning to'liq ketma-ketligi"; RL EMBL/GenBank/DDBJ ma'lumotlar bazalariga taqdim etilgan (FEB-2008).

CC -!- FUNKSIYA: 6-fosfoglyukonolakton CC ning 6-fosfoglyukonatga gidrolizlanishini katalizlaydi (o'xshashligi bo'yicha).

CC -!- KATALITIK FAOLIYAT: 6-fosfo-D-glyukon-1,5-lakton + H<sub>2</sub>O

<sup>30</sup><http://www.expasy.org/sprot>

31Ushbu fayl ushbu sahifaga joylashtirish uchun biroz o'zgartirilgan; Asl faylni http dan olish mumkin : <http://www.expasy.org/uniprot/B1IXL9.txt>.

## 206 Bioinformatika uchun Python

CC = 6-fosfo-D-glyukonat.

CC -!- YO'L: Uglevodlar degradatsiyasi; pentoza fosfat yo'li; CC D-glyukoza 6-fosfatdan D-ribuloza 5-fosfat (oksidlanish bosqichi): bosqich 2/3.

CC

CC -!- O'XSHAGI: sikloizomeraz 2 oilasiga mansub.

CC -----

CC UniProt Konsortsiumi tomonidan mualliflik huquqi bilan

himoyalangan, CC <http://www.uniprot.org/terms> ga qarang Ijodkorlik ostida tarqatiladi

CC Commons Attribution-NoDerivs litsenziyasi

CC -----

DR EMBL; CP000946; ACA78522.1; -; Genomik\_DNK.

DR RefSeq; YP\_001725849.1; -.

DR GenelD; 6065358; -.

DR GenomeReviews; CP000946\_GR; EcolC\_2895.

DR KEGG; ekl: EcolC\_2895; -.

DR GO; GO: 0017057; F:6-fosfoglyukonolaktonaza faolligi; IEA: HAMAP.

DR GO; GO: 0006006; P: glyukoza almashinuvi jarayoni; IEA: HAMAP.

DR HAMAP; MF\_01605; -; 1.

DR InterPro; IPR015943; WD40/YVTN\_takrorlash kabi.

DR Gene3D; G3DSA: 2.130.10.10; WD40/YVTN\_takrorlash kabi; 1.

PE 3: Gomologiyadan olingan; KVt

Karbongidrat almashinuvi; To'liq proteoma; glyukoza almashinuvi; KVt gidrolaza.

FT zanjiri 1 331 6-fosfoglyukonolaktonaza.  
FT /FTId=PRO\_1000088029.

SQ SEQUENCE 331 AA; 36308 MVt; D731044CFCF31A8F CRC64;

MKQTVYIASP ESQQIHWNL NHEGALTQ VVDVPGQVQP MVVSPDKRYL YVGVRPEFRV  
LAYRIAPDDG ALTFAAESAL PGSPTHISTD HQGQFVFVGS YNAGNVSVTR LEDGLPVGVV  
DVVEGLDGCH SANISPDNRT LWVPALKQDR ICLFTVSDDG HLVAQDPAEV TTVEGAGPRH  
MVFHPNEQYA YCVNELNSSV DVWELKDPHG NIECVQTLDM MPENFSSTRW AADIHITPDG  
RHLYACDRTA SLITVFSVSE DGSVLSKEGF QPTETQPRGF NVDHSGKYLI AAGQKSHHIS  
VYEIVGEQGL LHEKGRYAVG QGPMWVVNA H

//

**Listing 9.18** bir nechta yozuvlarga ega SwissProt faylidan ma'lumotlarni qanday olish mumkinligini ko'rsatadi:

---

**Listing 9.18: SwissProt faylidan ma'lumotlarni olish**

---

1 Bio import SwissProt 2 dan

ochiq("../samples/spfile.txt") fh sifatida: yozuvlar =

SwissProt.parse(fh) 3 yozuvlarda qayd qilish uchun:

4

5 print('Yozuv nomi: %s' % record.entry\_name)

**Biopython 207 ga kirish**

```

6      print('Aksessiya(lar): %s' % ','.join(record.accessions)) print('Kalit so'zlar:
7      %s' % ','.join(record.keywords)) print('Tartib: %s ' % rekord.sequence)
8

```

---

**Listing 9.19** SwissProt moduli tomonidan tahlil qilingan yozuvlardagi barcha atributlarni ko'rsatadi:

**Listing 9.19: SwissProt yozuvining atributlari**

1 Bio import SwissProt 2 dan

```

ochiq('..../samples/spfile.txt') fh sifatida: rekord =
    keyingi(SwissProt.parse(fh)) 3
4        att uchun dir(yozuv): agar
5            bo'lmasa att.startswith('__'):
6                chop etish (att, getattr (yozi, att))

```

---

## 9.4 XULOSA

Eng ko'p ishlataladigan Biopython xususiyatlari ushbu bobda yoritilgan. Bu erda keltirilgan kod namunalari va III bo'limdagi to'liq dasturlardan so'ng sizga Biopythonidan qanday foydalanish haqida tushuncha beradi. Shuningdek, siz Python o'rnatilgan yordamidan qanday foydalanishni o'rganishingiz kerak, chunki onlayn hujjatlar bosma nashrlarga qaraganda ko'proq yangilanadi. Biopiton rivojlanishi tez sur'atlar bilan sodir bo'ladi. Shu qadar tezki, men ustida ishlayotganimda bu bob bir necha marta qayta yozildi. Biopython ishlanmalaridan xabardor bo'lishning eng yaxshi usuli Biopython ishlab chiqish pochta ro'yxatiga obuna bo'lish va kodlar omboridan RSS tasmasi olishdir.

## 9.5 QO'SHIMCHA RESURSLAR

- Chang J., Chapman B., Fridberg I., Hamelryck T., de Hoon M., Cock P. va Antão, T. Biopython qo'llanmasi va pishirish kitobi. <http://www.biopython.org/DIST/docs/tutorial/Tutorial.html> yoki <http://www.biopython.org/DIST/docs/tutorial/Tutorial.pdf>.
  
- Hamelryck T. va Manderick B., PDB fayl tahlilchisi va tuzilma sinfi Pythonda eslatib o'tilgan. Bioinformatika. 2003 yil 22-novabr;19(17):2308–10. <https://www.ncbi.nlm.nih.gov/pubmed/14630660>
  
- Sohm, F., Cheklov modulidan foydalanish bo'yicha ovqat kitobi uslubidagi qo'llanma. <http://biopython.org/DIST/docs/cookbook/Restriction.html>
  
- Wu CH, Apweiler R., Bairoch A., Natale DA, Barker WC, Boeckmann B., Ferro S., Gasteiger E., Huang H., Lopez R., Magrane M., Martin MJ, Mazumder R., O. "Donovan C., Redaschi N. va Suzek B. (2006). Universal Protein Resursi (UniProt): Protein ma'lumotlarining kengayib borayotgan olami. Nuklein kislotalar tadqiqoti 34: D187-D191.

208 Bioinformatika uchun Python

- Magrane M. va Apweiler R. (2002). Swiss-Prot va TrEMBLda axborotni tashkil etish va standartlashtirish. Data Science Journal 1(1): 13–18. [http://datascience.codata.org/articles/10.2481/dsj.1.13/galley/168/ download/](http://datascience.codata.org/articles/10.2481/dsj.1.13/galley/168/)
- Benson Dennis A., Karsch-Mizrachi Ilene, Lipman David J., Ostell Jeyms va Wheeler David L.. GenBank. Nuklein kislotalari Res. 2008 yil yanvar; 36 (Ma'lumotlar bazasi muammosi): D25–D30. <http://dx.doi.org/10.1093/nar/gkm929>
- Larkin MA, Blackshields G., Brown NP, Chenna R., McGettigan PA, McWilliam H., Valentin F., Wallace IM, Wilm A., Lopez R., Tompson JD, Gibson TJ, Higgins DG. Clustal W va Clustal X 2.0 versiyasi. Bioin formati. 2007 yil 1-noyabr;23(21):2947-8. Epub 2007 yil 10 sentyabr.
- Vikipediya mualliflari. Cheklov fermenti. Vikipediya, Erkin Ensiklopediya dia. 2009 yil 13 fevral, UTC 16:44. [http://en.wikipedia.org/wiki/Restriction\\_enzyme](http://en.wikipedia.org/wiki/Restriction_enzyme).
- Sequence uchun EFetch va boshqa molekulyar biologiya ma'lumotlar bazalari. <https://www.ncbi.nlm.nih.gov/books/NBK25499/#chapter4.EFetch>
- Cock P. NCBI Entrez ElInfo (& Biopython) bilan aqlli fokuslar. <https://news.open-bio.org/2009/06/21/ncbi-einfo-biopython/>

## 9.6 O'Z-O'ZINI BAHOLASH

---

1. Biopitonda alifbo nima? Kamida to'rttasini nomlang.
2. Seq va SeqRecord obyektlarini tavsiflang.
3. Seq obyekti satrga nisbatan qanday afzallikkarni beradi?
4. Seq ob'ekti ba'zi qatorli operatsiyalarni ta'minlaydi. Nega?
5. MutableSeq obyekti nima?
6. Align va ClustalW modullari o'rtaqidagi bog'liqlik qanday?
7. SeqIO modulining usullarini ayting.
8. Nima uchun 9.3 kodida 7-qator oxiriga yaqin vergul qo'yilgan?
9. SeqUtils da topilgan beshta funktsiyani ayting.
10. Sequencing moduli yordamida qanday turdag'i ketma-ketlik fayllarini o'qish mumkin?
11. NCBI veb-serveridan ma'lumotlarni olish uchun qaysi moduldan foydalanasiz?
12. PDB faylidagi barcha tartiblangan atomlarni sanash dasturini tuzing. PDB fayli buyruq satrida dasturga o'tkazilishi kerak: program.py file.pdb.



**Taylor & Francis**  
Taylor & Francis Group  
<http://taylorandfrancis.com>



## **Kengaytirilgan mavzular**



**Taylor & Francis**  
Taylor & Francis Group  
<http://taylorandfrancis.com>

# Veb-ilovalar

---

## MAZMUNI

<b>10.1 Internetda Pythonga kirish. ....</b>	<b>Pythonda 213</b>	<b>10.2</b>
CGI. ....	214	10.2.1 CGI uchueb-
serverni sozlash. ....	215	10.2.2 Serverni skriptimiz
bilan sinab ko'rish .....	215	10.2.3 CGI dasturiga ma'lumotlarni
yuborish. ....	216	Proteinning aniq
zaryadini hisoblash uchun veb-dastur (CGI)		
versiyasi). ....	219	
10.3 WSGI. ....	221	
10.3.1 Shisha: WSGI uchun Python Web Framework .....	222	10.3.2
Shishani o'rnatish .....	223	10.3.3
Minimal shishani qo'llash. ....	223	10.3.4 Shisha
komponentlari. ....	224	
Marshrutlar. ....		
O'zgaruvchan qismalgarda ega 224 URL.		
225 Maylumotlarni olish: so'yrov .....		
225 Shablonlar. ....		
226 Statik fayllar .....		
228 10.3.5 Proteinning (shisha) sof zaryadini hisoblash uchun veb-dastur		
Versiya). ....	229	
10.3.6 Apache'da WSGI dasturini o'rnatish. ....	232	10.4
Python-ga asoslangan dinamik veb-saytlar yaratish uchun muqobil variantlar. ....	232	10.5
Skript xavfsizligi haqida ba'zi so'zlar. ....	232	10.6
Python dasturlarini qayerda joylashtirish. ....	234	
10.7 Qoshimcha manbalar .....	235	
10.8 O'z-o'zini baholash. ....	236	

## 10.1 WEBDA PYTHONGA KIRISH

Biz mahalliy dasturlarni qanday ishga tushirishni ko'rdik. Ushbu bobda ularni Internetga qanday o'tkazish ko'rsatilgan.

Dasturni Internetda mavjud qilishning asosiy afzalligi shundaki, u ko'proq foydalanuvchilarga dastur nusxasini o'rnatmasdan va Python o'rnatilishini talab qilmasdan kirishi mumkin. Ba'zan dastur oxirgi foydalanuvchining qattiq diskiga o'rnatib bo'lmaydigan ulkan ma'lumotlar bazalari kabi talab qilinadigan resurslarga kiradi.

## 214 Bioinformatika uchun Python

Veb-ilovalarni yaratish uchun sizga HTML, CSS, JS, veb-serverni boshqarish va boshqalar kabi Python-dan boshqa vositalar kerak bo'ladi. Bu mavzular ushbu kitob doirasidan tashqarida, shuning uchun agar siz ilgari hech qachon veb-sahifa yaratmagan bo'lsangiz, ularni o'qib chiqishingizni maslahat beraman. HTML asoslarini bilish alohida ahamiyatga ega, chunki aksariyat IT laboratoriyalarida veb-serverlarni sozlash va texnik xizmat ko'rsatish bilan shug'ullanadigan xodimlar mavjud, ammo HTML dizayni ular siz uchun kamdan-kam hollarda qiladigan narsadir. HTML haqida ko'proq ma'lumot olish uchun "Qo'shimcha manbalar" bo'limiga qarang. Veb-serverga kelsak, Python ishlab chiqarish va sinovdan o'tkazish uchun foydali bo'lgan, ammo ishlab chiqarishda foydalanish uchun emas. Bunday holda, siz Apache yoki Nginx kabi mustaqil server dasturidan foydalanishingiz kerak; Apache hozirgacha eng mashhur bo'lganligi sababli, bu kitob uni qanday sozlashni o'z ichiga oladi. Veb-server sizning muassasangiz tomonidan taqdim etilishi odatiy holdir, lekin IT bo'limi virtual mashinani taqdim etishi ham keng tarqalgan bo'lib, u erda siz faqat o'zingizning kompyuteringiz o'rniغا barcha kerakli dasturlarni o'rnatishingiz kerak.

ilova.

Python-ni veb-serverda CGI (Common Gateway Interface), mod\_python va WSGI (Web Server Gateway Interface) da ishlatalishning bir necha usullari mavjud. CGI veb-sahifada dinamik tarkibni ishga tushirishning eng qadimgi usuli hisoblanadi. Birinchi veb-serverlar 1993 yilda CGI protokoli aniqlangunga qadar faqat statik HTMLni ko'rsatgan. U hozirda ham qo'llanimoqda va ba'zi hosting kompaniyalari interaktiv veb-serverlar yaratishning yagona varianti sifatida CGI-ni taklif qilishadi. Afzallik sifatida, uni sozlash eng oson va qo'shimcha dasturiy ta'minotni o'rnatmasdan deyarli barcha veb-serverlarda mavjud. Bu mohiyatan istalgan tilda yozilgan dasturni veb-server bilan ularash protokolidir. mod\_python, xususan, Python-ni veb-server bilan birlashtiradigan Apache modulidan iborat. Ushbu yondashuvning afzalligi skriptni tez bajarishdir, chunki Python tarjimonini veb-server bilan yuklangan. WSGI, o'z navbatida, "veb-dasturlar bilan bog'lanish uchun veb-serverlar va dastur serverlari uchun spetsifikatsiya" dir. Bu spetsifikatsiya bo'lganligi sababli, bir nechta ilovalar mavjud. WSGI ning asosiy afzalligi shundaki, siz WSGI ilovasini yaratganingizdan so'ng uni har qanday WSGI-mos keladigan server1 (yoki hatto Python tomonidan taqdim etilgan veb-server yordamida) joylashtirishingiz mumkin. Mod\_python-da bo'lgani kabi, WSGI-ga asoslangan dasturlarning ishlash tezligi CGI-ga qaraganda yaxshiroq, chunki har bir so'rovda Python tarjimonini ishga tushirish uchun hech qanday qo'shimcha xarajatlar yo'q.

## PYTHONDA 10.2 CGI

---

Ushbu bo'lim uchun sizda allaqachon ishlayotgan Apache veb-serveringiz bor deb o'ylayman. Agar shunday bo'limasa, uni Debian/Ubuntu-ga asoslangan Linux distributivlariga o'rnatish mumkin:

```
$ sudo apt-get install apache2
```

Shu bilan bir qatorda siz istalgan veb-xosting rejasini yollashingiz mumkin; ularning ko'pchiligidagi Apache mavjud va CGI oldindan o'rnatilgan. Veb-xosting haqida qo'shimcha ma'lumot olish uchun 234-sahifaga qarang.

<sup>1</sup>WSGI veb-serverlarida taqqoslash uchun ushbu maqolani ko'ring: <https://www.digitalocean.com/community/tutorials/python-asosidagi-veb-ilovalar-uchun-veb-serverlarni-taqqoslash-qilish>

### 10.2.1 CGI uchun veb-serverni sozlash

Server konfiguratsiya faylida2 skriptlar CGI orqali bajarilishi mumkinligi, qaysi kataloglarda va qanday nomlanishi haqida spetsifikatsiyalar bo'lishi kerak.

Agar skriptlar /var/www/apache2-default/cgi-bin manzilida joylashgan bo'lsa, biz serverning konfiguratsiya fayliga quyidagi qatorlarni kiritishimiz kerak.

```
<Katalog /var/www/apache2-default/cgi-bin>
```

```
    Variantlar +ExecCGI
```

```
</Katalog>
```

Bajariladigan skriptlar ekanligini ko'rsatish uchun konfiguratsiya fayliga quyidagi qatorni qo'shing .py fayl kengaytmasiga ega bo'lganlar

```
AddHandler cgi-script .py
```

Agar konfiguratsiya faylida allaqachon ro'yxatdan o'tgan fayl kengaytmalari bilan chiziq mavjud bo'lsa, siz faqat unga .py qo'shish kerak.

```
AddHandler cgi-script .cgi .pl .py
```

Nihoyat, biz ScriptAlias o'zgaruvchisini sozlashimiz kerak. Bu shunday yo'lni talab qiladi foydalanuvchi URL va skriptlar saqlanadigan yo'lni ko'radi.

```
ScriptAlias /cgi-bin/ /var/www/apache2-default/cgi-bin/
```

Bularning barchasi server konfiguratsiya faylida mavjud. Qilish kerak bo'lgan yagona narsa - skriptning Apache foydalanuvchi ruxsatlariga ega ekanligiga ishonch hosil qilish. Server terminalidan quyidagilarni kiriting:

```
chmod a+x MyScript.py
```

Agar sizda faqat FTP ruxsati bo'lsa, ruxsatlarni o'rnatish uchun FTP mijozidan foydalaning.

### 10.2.2 Serverni Skriptimiz yordamida sinab ko'rish

Server CGI dasturlarini bajarishga tayyorligini tasdiqlash uchun quyidagi koddan foydalanish mumkin:

**Listing 10.1: firstcgi.py: Birinchi CGI skripti**

---

```
1#!/usr/bin/env python 2
print("Content-Type: text/html\n") 3
print("<html><head><title>Sinov sahifasi</title></head><body >") 4 ta chop
etish("<h1>SALOM DUNYO!</h1>") 5 ta chop etish("</body></html>")
```

---

<sup>2</sup> Apache veb-serverida ko'p hollarda konfiguratsiya fayli [httpd.conf](#) yoki [apache2.conf](#) hisoblanadi. va u /etc/apache2 katalogida joylashgan. Bu har bir o'rnatishda o'zgarishi mumkin.

**Kod tushuntirishi:** Birinchi qatorda Python interpreter joylashuvi ko'rsatilgan. Odatda, bu qator ixtiyoriyidir va u faqat Python tarjimoniga qo'ng'iroq qilmasdan skriptni to'g'ridan-to'g'ri ishga tushirishni xohlaganimizda qo'shiladi. CGI dasturlari uchun bu qator majburiydir.<sup>3</sup> Ikkinchchi qator veb-server uchun HTML sahifa yuborilishini bilishi uchun muhimdir. Biz Content-Type/html qatorini, so'ngra ikkita karetani qaytarishimiz kerak. Garchi 2-qatorda faqat bitta yashirin karetani qaytarish (`\n`) mavjud bo'lisa-da, ikkinchisi chop etish buyrug'i bilan qo'shiladi.

Dasturning qolgan qismi biz shu paytgacha bajargan boshqalarga o'xshaydi, farqi shundaki, biz HTML kodni brauzer tomonidan o'qilishi uchun chop qilamiz.

Agar biz ushbu dasturni veb-serverga yuklasak va keyin brauzerimiz yordamida sahifaga kirsak, biz ko'radian natijalar 10.2.2-rasmga o'xshash bo'ladi . Agar hamma narsa yaxshi bo'lisa, biz fayning mazmunini emas, balki uning serverda bajarilishi mahsulotini ko'ramiz. Ushbu mahsulot (HTML sahifasi) veb-brauzer tomonidan ko'rsatiladi ( 10.1-rasmga qarang).



10.1-rasm Bizning birinchi CGI.

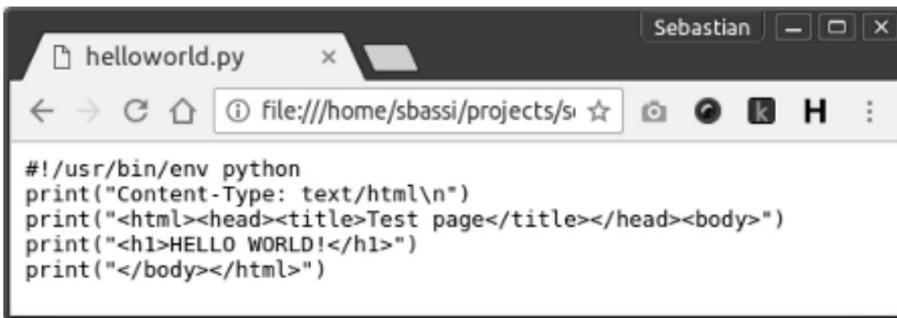
Sahifalarimizni sinab ko'rish uchun ularni to'g'ridan-to'g'ri qattiq diskimizdan ochmasdan, veb-server tomonidan qayta ishlanishi kerakligini hisobga oling. Bunday holda biz veb-brauzer tomonidan ko'rsatilgan sahifa o'rniga 10.2-rasmda ko'rgan narsaga ega bo'lamiz .

### CGI dasturiga ma'lumotlarni yuborish

Oldingi dastur unchalik foydali emas, bu shunchaki statik sahifa bo'llib, u foydalanuvchidan hech qanday parametrlarni qabul qilmaydi. Keling, ushbu ma'lumotlardan foydalanadigan Python dasturiga ma'lumotlarni yuboradigan minimalist HTML shakli misolini ko'rib chiqaylik.

Birinchi qadam shaklni loyihalashdir. Bunday holda biz oddiy shakl yaratamiz bitta maydon bilan va u greeting.html sifatida saqlanadi:

<sup>3</sup> Agar siz Python tarjimoniga yo'lni bilmasangiz, tizim administratoridan skriptingizni o'rnatishni so'rang. Boshqa variant, agar sizda server buyruq qatoriga kirish imkoningiz bo'lisa, bu python-ni ishga tushirishdir.



```
#!/usr/bin/env python
print("Content-Type: text/html\n")
print("<html><head><title>Test page</title></head><body>")
print("<h1>HELLO WORLD!</h1>")
print("</body></html>")
```

10.2-rasm CGI veb-server o'rniغا mahalliy diskdan kirish.

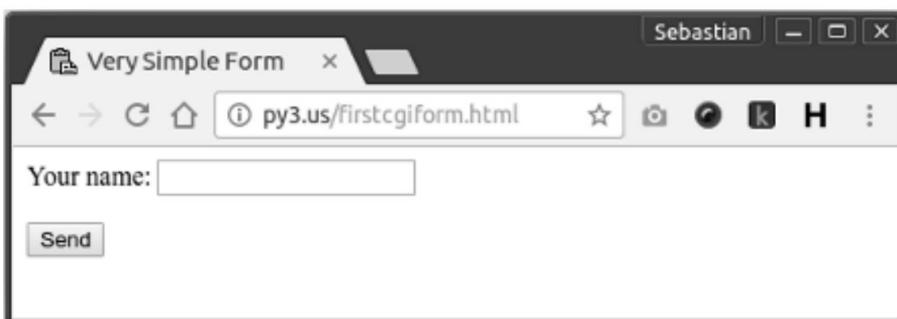
**Listing 10.2: greeting.html: CGI dasturiga ma'lumotlarni yuborish uchun HTML interfeysi**

---

```
1 <html><head><title>Juda oddiy shakl</title></head> 2 <body>
3 <form action='cgi-bin/greeting.py' method='post'> 4 Ismingiz:
< kiritish turi='text' name='username'> <p> 5 <input type='submit'
value='Send'> 6 </form></body></html>
```

---

Kod tushuntirildi: Ushbu kichik shaklda ikkita muhim xususiyatni ta'kidlash kerak. 3-qatorda ma'lumotlarni qayta ishlash uchun mo'ljallangan dastur joylashgan joy ko'rsatilgan (cgi-bin/greeting.py). 4-qatorda foydalanuvchi to'ldirishi kerak bo'lgan maydon mavjud ("matn" turi), tegishli o'zgaruvchi (foydalanuvchi nomi). Ushbu o'zgaruvchining nomi muhim ahamiyatga ega, chunki foydalanuvchi tomonidan kiritilgan ma'lumotlar ushbu nomga bog'lanadi. Shakl 10.2.2-rasmdagiga o'xshaydi .



10.3-rasm salomlashish.html: Juda oddiy shakl.

Keling, shakl va undan yuborilgan ma'lumotlarni qabul qiladigan kodni qanday yozishni ko'rib chiqaylik u "tezda" veb-sahifani yaratadi.

**Listing 10.3: greeting.py: greeting.html da shaklni qayta ishlovchi CGI dasturi.**

```
1#!/usr/bin/env python 2
import cgi 3 print("Content-
Type: text/html\n") 4 form = cgi.FieldStorage()
5 name = form.getvalue("foydalanuvchi
nomi", "NN" )[10] 6 print("<html><head><title>CGI
skripti</title></head>") 7 print("<body><h2>Salom {0}</h2></
body>".format(name))
```

Kod tushuntirildi: 4-qatorda biz cgi.FieldStorage sinfidan misol (shakl) yaratamiz. Bu sinf forma tomonidan yuborilgan qiymatlarni oladi va ularni lug'atga o'xshash tarzda foydalanishga imkon beradi. Keyingi qatorda (5) biz forma tomonidan yuborilgan ma'lumotlarga kiramiz, shuningdek, chop etish funksiyasiga o'tadigan belgilar sonini qisqartiramiz; bu potentsial xavfsizlik muammosini yumshatish uchun amalga oshiriladi.<sup>4</sup> Getvalue usuli zaruriy argument sifatida biz kontentiga kirishni istagan maydon nomini oladi. Ikkinci argument ixtiyoriyidir va agar kerakli maydon bo'sh bo'lsa, qaysi qiymat qaytarilishini ko'rsatadi. E'tibor bering, bu lug'at olish funksiyasiga o'xshaydi. 6-qatordan boshlab dastur o'zgaruvchining mazmunidan foydalanib HTML kodini chop etadi. Bu brauzerda ko'rsatiladigan kod.

Xulosa qilib aytganda, biz ismni kiritish va "Yuborish" tugmasini bosish uchun [10.2 Listingdagi veb-shakldan foydalandik](#). Bu ma'lumotlarni yuboradi. Keyin u cgi.FieldStorage klassi tufayli dastur tomonidan o'qlidi va veb-sahifa yaratish uchun dasturda ishlataladigan o'zgaruvchi nomi sifatida havola qilinadi. 10.5-rasmdagi chiqishga qarang.



10.4-rasm greeting.html ni qayta ishlovchi CGI dasturining chiqishi.

<sup>4</sup>Veb-saytlarni himoya qilish bo'yicha qo'shimcha ma'lumot olish uchun 232-betga qarang.

### 10.2.3 Proteinning aniq zaryadini hisoblash uchun veb-dastur (CGI versiyasi)

Listing 4.14 kodidan foydalanib , biz uni veb-sahifadan foydalanishga osongina moslashtira olamiz. Birinchi qadam sifatida biz foydalanuvchi ma'lumotlarni kiritishi mumkin bo'lgan shaklni loyihalashimiz kerak. Bu taklif qilingan shakl:

**Listing 10.4: protcharge.html: CGI dasturiga ma'lumotlarni yuborish uchun HTML interfeysi**

---

```

1 <!DOCTYPE html>
2 <html lang="en"> 3
<head><meta charset="utf-8">
4 <title>Protein zaryadi kalkulyatori</title> 5 <link href="css/
bootstrap.min.css" rel="stylesheet"> 6 </head> 7 <body style="background-
color:#e7f5f5;"> 8 <div class="container"><h2>Protein zaryadi kalkulyatori</
h2> 9 <form action='/cgi-bin/protcharge.py' method='post'> 10 <div
class="row">

11   <div class="col-sm-8">
12     <div class="form-group"> <label
13       for="aaseq">Aminokislolar ketma-ketligini kiriting:</label> <textarea
14       name="aaseq" rows="5" cols="40"></textarea> </div> </div> </div>
15
16
17
18   <div class="satr">
19     <div class="col-sm-8">
20       <div class="form-group"> <label
21         for="prop">Ularning nisbatlarini ko'rishni xohlaysizmi?
22           zaryadlangan aminokislota?</label>
23         <div class="radio">
24           <label>
25             <input type="radio" name="prop" value="y">Ha
26           </label> </
27         div>
28         <div class="radio">
29           <label>
30             <input type="radio" name="prop" value="n">Yo'q </label>
31           </div> <label for="title">Ish nomi (ixtiyoriy):</label> <input
32           type="text" size="30" name="title" value=""> <br>
33
34
35
36   <button type="submit" class="btn btn-primary">Yuborish

```

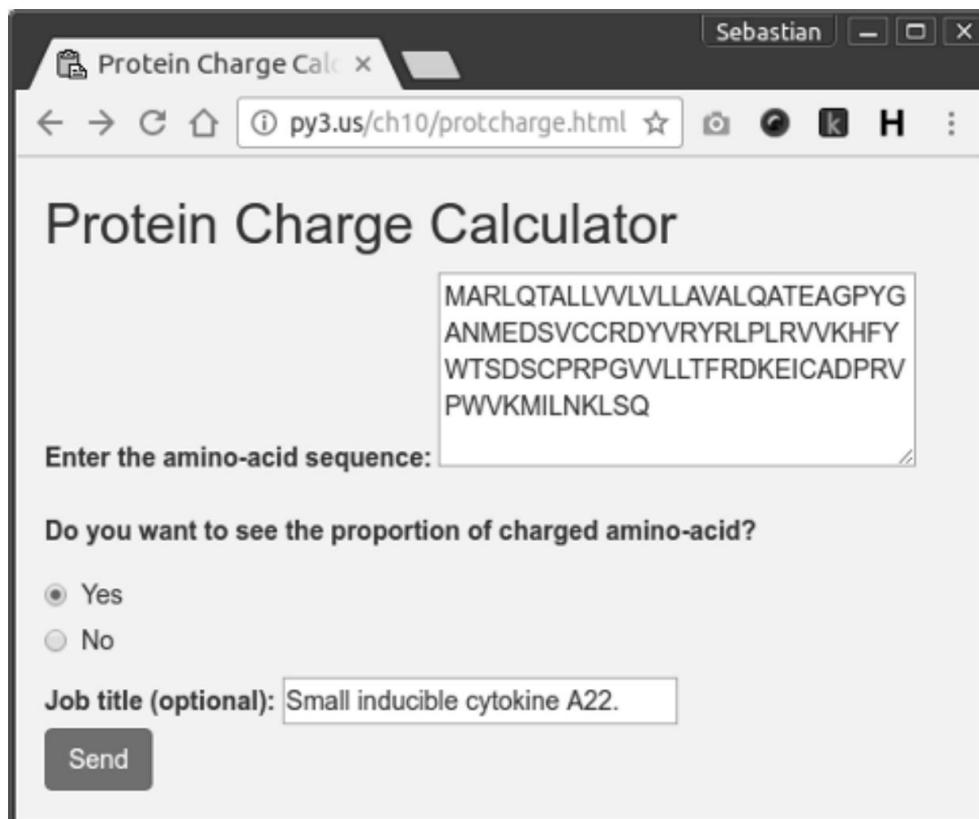
## 220 Bioinformatika uchun Python

```

37      </button>
38      </div> </
39      div> </div>
40      41 </form>
42 </div> 43 </body>
44 </html>

```

**10.5-rasmda 10.4 Listingdagi shakl veb-brauzerda qanday ko'rsatilishi ko'rsatilgan.**



10.5-rasm protcharge.html formasi topshirishga tayyor.

Quyida formadan foydalanilganda chaqiriladigan kod (protcharge.py) keltirilgan:

**Listing 10.5: protcharge.py: Proteinning sof zaryadini va zaryadlangan aminokislotalarning nisbatini hisoblash uchun orqa kod**

```

1#!/usr/bin/env python 2
import cgi, cgitb

```

```

3
4 def chargeandprop(aa_seq):
5     protseq = aa_seq.upper()
6     zaryad = -0,002 cp = 0 aa_charge
7     = {'C':-045,'D':-0999,'E':-0998,'H'
8     ':091,'K':1,'R':1,'Y':-001} protseqdagi aa uchun: zaryad +==
9     aa_charge.get(aa, 0) agar aa aa_chargededa: cp += 1
10    prop = float(cp)/len(aa_seq)*100 qaytish (zaryad, tayanch)
11
12
13
14
15
16
17 cgitb.enable() 18
print('Content-Type: text/html\n') 19 form =
cgi.FieldStorage() 20 seq = form.getvalue('aaseq',
'QWERTYYTREWQRTYEYTRQWE') 21 prop = form.getvalue ('prop', 'n') 22
jobtitle = form.getvalue('title','No title') 23 to'lov, propvalue =
chargeandprop(seq) 24 print('<html><body>Ish nomi:{0} <br/>
'.format(jobtitle)) 25 print('Sizning ketma-ketligingiz:<br/>{0}<br/>
'.format(seq)) 26 print('Sof to'ylov: {0}<br />'.format(zaryad)) 27 if prop == 'y':
print('Zaryadlangan AA nisbati: {0:.2f}<br />' 28 .format(propvalue)) 29 30
print('</ body></html>')

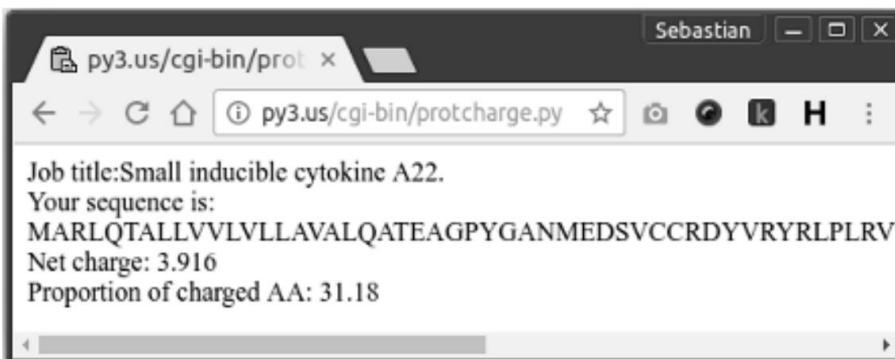
```

### **10.2.3-rasmda 10.5 ro'yxatidagi kod exe kesilganidan keyin hosil bo'lgan HTML sahifa ko'rsatilgan .**

Kod tushuntirishi: Zaryad va zaryadlangan aminokislota ulushini hisoblash uchun kod 4-satrdan boshlanadigan funksiyada joylashgan. 19-qatorda biz cgi.FieldStorage sinfining namunasini (shaklini) yaratamiz. Shakl ob'ekti shakl tomonidan yuborilgan qiymatlarni olish va ularni moda kabi lug'atda mavjud qilish uchun javobgardir. 20-qatordan 22-qatorgacha biz foydalanuvchi tomonidan kiritilgan qiymatlarni olamiz. 24-qatorda "aniq zaryad" va "zaryadlangan aminokislotalarning nisbati" baholanadi. 25-qator oxirigacha brauzerga yuboriladigan HTML-ni yaratadi.

## 10.3 WSGI

**WSGI dan oldin Python-da veb-dasturlash uchun juda ko'p mos kelmaydigan tanlovlardan mavjud edi. Ulardan ba'zilari veb-ramkalar, ya'ni dinamik veb-saytlarni ishlab chiqish uchun dasturlar to'plami edi. Ushbu ramkalarning ba'zilari bilan bog'liq muammo shundaki**



### Shakl 10.6 Net to'lov CGI natijasi.

Ularning har biri turlicha ishlagan va ularning aksariyati veb-serverga bog'langan bo'lib, veb-server/ilova juftligini tanlashni cheklab qo'ygan.

WSGI ushbu bo'shlqnini to'ldirish uchun yaratilgan va u "veb-serverlar va veb-ilovalar yoki ramkalar o'rtaсидаги oddiy va universal interfeys" sifatida belgilanadi. Ko'pgina komponentlar (yoki o'rta dastur) endi WSGI bilan mos keladi, shuning uchun dasturchi WSGI bilan bevosita shug'ullanishi shart emas. Ilova o'rta dasturiy ta'minot bilan ishlagandan so'ng, uni har qanday WSGI-ga mos serverda joylashtirish mumkin. WSGI endi standartlashtirildi va PEP-3333 da tavsiflanganidek Python tilining bir qismidir: <https://www.python.org/dev/peps/pep-3333/>. Shu sabablarga ko'ra, WSGI Python-da veb-ishlab chiqish uchun tavsiya etilgan tanlovdir.

#### 10.3.1 Shisha: WSGI uchun Python veb-ramka

Bottle Python uchun mikro web-ramkadir. U bitta fayl sifatida tarqatiladi va hech qanday bog'liliklarga ega emas, ya'ni ishlashi uchun faqat Python kerak. Shisha 4 ta asosiy komponentdan iborat:

- Marshrutlash: URL manzilni Python funksiyasiga tarjima qilish (yoki xaritalash) usuli. Shunday qilib, foydalanuvchi har safar URL so'raganida (o'zgaruvchan qismlarga ega bo'lishi mumkin), ma'lum bir funktiya bajariladi.
- Andozalar: o'rnatilgan shablon mexanizmi va uchta uchinchi tomon shablonlari (mako, jinja2 va gepard) uchun qo'llab-quvvatlash.
- Utilitalar: Shakl maylumotlari, yuklangan fayllar, cookie-fayllarga kirish uchun lug'yatga oyxshash obyekt. sarlavhalar va boshqa metama'lumotlar.
- Server: Ishlab chiqish serveri va tashqi HTTP serverini qo'llab-quvvatlash, in har qanday WSGI mos HTTP serverini o'z ichiga oladi.

Bottle uchun boshqa alternativalar mavjud. Eng mashhuri Flask ( qo'shimcha variantlar uchun ushbu bobning oxiridagi 233-betdagি [10.1-jadvalga](#) qarang), ammo bu erda

hozirda u Python 3-ga mos kelmaydigan shablon dvigatelini qo'llab-quvvatlaydi, shuning uchun Bottle hozirda eng yaxshi tanlovdir.

### 10.3.2 Shishani o'rnatish

Shishani <https://bottlepy.org> veb-sahifasidan olish mumkin , lekin ko'pgina tashqi paketlar kabi, u virtual muhitda pip install bilan o'rnatilishi mumkin. Quyidagi parcha virtual muhitni qanday yaratishni (bottleproj deb ataladi), uni qanday faollashtirishni va Bottleni bottleproj virtualiga qanday o'rnatishni ko'rsatadi.

muhit:

```
$ virtualenv bottleproj '/usr'  
asosiy prefiksidan foydalanish /  
home/sb/bottleproj/bin/python3 da bajariladigan yangi python. Shuningdek, /  
home/sb/bottleproj/bin/python da bajariladigan fayl yaratish. Setuptools, pip,  
wheel o'rnatish... bajarildi. $. bottleproj/bin/activate (bottleproj) $ pip install  
bottle Shishani yig'ish Shishani yuklab olish-0.12.13.tar.gz (70kB) (...)
```

Muvaffaqiyatli qurilgan shisha

To'plangan paketlarni o'rnatish: shisha

Muvaffaqiyatli o'rnatilgan shisha-0.12.13

Anaconda tarqatish uchun ekvivalent buyruq:

```
$ conda create -n bottleproj shisha
```

Shisha faylda bo'lganligi sababli, muqobil o'rnatish usuli faylni <https://raw.githubusercontent.com/bottlepy/bottle/master/bottle.py> dan yuklab olishdir. va uni skriptingiz joylashgan katalogga nusxalang.

### 10.3.3 Minimal shishadan foydalanish

Mana Bottledagi oddiy "Salom dunyo" ilovasi:

**Listing 10.6: hellobottle.py: Shishadagi salom dunyo**

---

```
Shishani import qilish marshrutidan 1,  
yugurish 2  
3 @marshrut('/')  
4 def indeks():  
5     qaytish '<h2>Salom dunyo!</h2>'  
6  
7 ishga tushirish (host = 'localhost', port = 8000)
```

Birinchi qatorda biz Shishanining ikkita komponentini import qilamiz (marshrut va ishga tushirish). 3-qatorda biz keyingi satrda boshlanadigan funktsiyaga marshrut yoki yo'lni tayinlaymiz. Bu foydalanuvchi ushbu sahifani olish uchun domenden keyin kiritishi kerak bo'lgan URL (veb-manzil). Agar foydalanuvchi o'z brauzerida ushbu yo'lni (bu holda, ildiz darajasini) yozsa, funktsiya indeksi ishga tushadi. Bu funktsiya (4-qatorda) shunchaki “`<h2>Salom Dunyo!</h2>`”ni qaytaradi. 7-qator serverni ishga tushiradi.

Mana bu dasturning terminalga chiqishi:

```
(bottleproj) $ python helloworldbottle.py Bottle v0.13-dev
server ishga tushmoqda (WSGIRefServer() yordamida)...
http://localhost:8000/ da tinglash Chiqish uchun
Ctrl-C tugmalarini bosing.
```



10.7-rasm Brauzerda ko'rinish turganidek, Shishada yaratilgan Salom Dunyo dasturi.

#### 10.3.4 Shishanining komponentlari

##### Marshrutlar

@marshrut dekoratoridan foydalanib, biz har bir sahifa uchun URL qanday ko'rinishini aniqlaymiz. Quyidagi parcha 2 sahifani, hech qanday yo'lsiz ildiz sahifani va URL manzilida /about yo'li bo'yigan haqida sahifani ko'rsatadi:

```
@route('/')
def index():
    "Yuqori daraja yoki indeks sahifasi" ni qaytarish

@marshrut('/haqida')
```

```
def about():
    "Haqida" sahifasini qaytarish
```

O'zgaruvchan qismlarga ega URL

Ba'zi saytlarda URLning bir qismi veb-sahifani yaratish uchun serverga uzatiladigan bir yoki bir nechta o'zgaruvchilardan iborat; masalan, <https://stackoverflow.com/questions/6224052> da, 6224052 raqami bo'lgan qism o'zgaruvchan qismdir. Bu raqam dasturga uzatiladi va ma'lumotlar bazasida maqola mazmunini qidirish uchun kalit sifatida ishlataladi.

Quyidagi kod sobit va o'zgaruvchan qismga ega URL manzilini ko'rsatadi. Ruxsat etilgan qism /salomlash/, o'zgaruvchan qism esa nom deb ataladi. O'z o'rniда bo'lgan har qanday satr parametr sifatida bog'langan funksiyaga (`shows_greeting`) uzatiladi.

```
@route('/greets/<name>') def
show_greeting(nom): "Salom
{0}"ni qaytaring.format(nom)
```

Agar siz `http://127.0.0.1:5000/greets/Adele` URL manzilini bossangiz , sahifani ko'rasiz Salom Adele matni bilan.

#### Ma'lumot olish: so'rov

so'rov ba'zi foydali xususiyatlarga ega lug'atga o'xshash ob'ektdir. U cookie-fayllarni, shaklda yuborilgan qiymatlarni, HTTP sarlavhalarini, fayllarni va boshqalarni saqlaydi. Keling, ba'zi foydali xususiyatlarni ko'rib chiqaylik:

- `request.form`: Veb-shakldagi barcha o'zgaruvchilar ushbu lug'atga o'xshash ob'ektda mavjud. Agar forma maydonida foydalanuvchi nomi deb nomlangan ma'lumotlar mavjud bo'lsa, maydon qiymatiga kirishning yo'lli `request.forms.get('foydalanuvchi nomi')`.
- `so'rov.metod`: Sahifani so'rashda ishlataladigan HTTP usuli. Brauzer veb-sahifani olganida, u "GET" so'rov turini yuboradi. Shakl yuborilayotganligi sababli URL manzili bosilsa, bu "POST" so'rovidir. Boshqa turdag'i so'yrovlari ham mavjud ("QOÝYISH", "PATCH" va "DELETE"), lekin ular bu yerda koýrib chiqilmaydi.<sup>5</sup>
- `request.args`: URLda yuborilgan parametrлarga kirish uchun. URL manzilida ?key=value shakli mavjud bo'lгanda foydalaniladi. Shuningdek, u lug'atga o'xshash ob'ektdir. Agar sizda <http://example.com/position?lat=37.51&long=115.71> kabi URL manzili boylgan saytingiz boylsa , qiymatlari mos ravishda 37,51 va 115,71 bo'lgan ikkita lat va long kalitlari mavjud. Latni olish uchun siz `request.args['lat']` yoki foydalanishingiz mumkin

---

<sup>5</sup>So'rov usullari haqida qo'shimcha ma'lumot olish uchun [http://www.w3schools.com/tags/ref\\_httpmethods.asp](http://www.w3schools.com/tags/ref_httpmethods.asp) saytiga qarang.

`request.args.get('lat')`. Bu turdagি URL manzillar tavsija etilmaydi va foydalanuvchilarga qulay URL manzillar norma hisoblanadi, bu holda <http://example.com/position/37.51/115.716> bo'ylishi mumkin . .

- `request.files`: Fayl yuklanganda u dasturga sifatida uzatiladi  
`request.files['fayl nomi']`.

## Shablonlar

Oldingi misollarda bizning usullarimiz satrlarni qaytarish edi (oddiy matnli satr yoki HTML satri). Foydalanuvchiga yuboriladigan HTML faylini yaratishning afzal usuli shablon va o'zgaruvchan ma'lumotlarga ega bo'lish va ikkala komponent bilan yakuniy (yoki ko'rsatilgan) HTMLni yaratishdir. Buning uchun Bottle tomonidan taqdim etilgan shablon usulidan foydalaning. Shablon usulining umumiy shakli: andoza(shablon\_nomi, \*\*lug'at).

Shablon odatda yakuniy qiymatlar uchun joy ushlagich sifatida o'zgaruvchilarga ega bo'lgan HTML faylidir. Quyidagi matn bitta o'zgaruvchiga ega shablondir:

**Listing 10.7: index.tpl: O'zgaruvchilar bilan shisha uchun shablon**

---

```
1 <html lang="en"> 2
<tana> 3
    <h1>Salom {{ name }}!</h1>
4 </body> 5 <
html>
```

---

Agar bu fayl index.tpl deb nomlangan bo'lsa va u papka ko'rinishlarida saqlangan bo'lsa, u bo'lishi mumkin ushbu kod bilan berilgan:

**Listing 10.8: indextemplate.py: O'zgaruvchilar bilan shablon uchun shisha kodi**

---

```
1 shisha import marshrutidan, ishga tushirish, shablon 2
3 @marshrut('/greets/<foydalanuvchi
nomi>') 4 def show_greeting(foydalanuvchi
nomi): 5 shablonni qaytarish ('indeks', **{'name':foydalanuvchi nomi})
6 7 ishga tushirish (host = 'localhost', port = 8000)
```

---

Shablonlarda oqimni boshqarish buyruqlari ham bo'lishi mumkin, shuning uchun siz shablonning qaysi qismi ko'rsatilishini boshqarishingiz mumkin. Misol uchun, quyidagi shablonni ko'rib chiqing (index2.tpl):

**Listing 10.9: index2.tpl: O'zgaruvchilar va oqim nazorati bilan shisha uchun shablon**


---

```

1 <html lang="en"> 2
<body> 3 %if
name[0].isalpha(): <h1>Salom
<h1>Foydalanamega bilan maslighat bilan
boshlanmasligi kerak</h1> 7 %end 8 <
6       body> 9 </html>

```

---

Ushbu shablon 3, 5 va 7-qatorlarda Python-ga o'xshash kodga ega. U Python-ga o'xshaydi, lekin oddiy Python sintaksisining bir qismi bo'limgan %end (7-qator) ga ega. Buning sababi, Python-da kod bloklari chekinish bilan belgilangan va shablonlarda chekinish hisobga olinmaydi, shuning uchun %end belgisi mavjud bo'lishi kerak. Ushbu shablondan foydalanish uchun [10.8 ro'yxitidagi](#) 5-qatorni index2.tpl fayliga ishora qilib o'yzgartiring.

Yangi [Listing \(10.10\)](#) indextemplate2.py deb ataladi:

**Listing 10.10: index2.py: o'zgaruvchilar bilan shablon uchun shisha kodi**


---

```

1 shisha import marshrutidan, ishga tushirish, shablon
2
3 @marshrut('/greets/<foydalanuvchi
nomi>') 4 def show_greeting(foydalanuvchi
nomi): 5 shablonni qaytarish('index2', **{'name':foydalanuvchi nomi})
6
7 ishga tushirish (host = 'localhost', port = 8000)

```

---

Ushbu ro'yxat 3-qatorda o'zgaruvchi nomidagi birinchi belgini tekshiradigan index2.tpl shablonidan ([10.9 ro'yxi](#)) foydalanadi; agar bu rost bo'lsa, u birinchi shablondagi kabi bir xil xabarni chop etadi, agar bo'limasa, shablonning 6-qatorida ko'rishingiz mumkin bo'lgan xabarni chop etadi.

Shuni yodda tutingki, biz qarorni shablonda qilish o'rniga kodda (if bandi) qabul qilish orqali bir xil natijaga erishishimiz mumkin, quyidagi kod va shablonga qarang:

**Listing 10.11: indextemplate3.py: Shablonlar o'rniga kodda mantiq bilan shisha kodi**


---

```

1 shisha import marshrutidan, ishga tushirish, shablon
2
3 @marshrut('/greets/<foydalanuvchi
nomi>') 4 def show_greeting(foydalanuvchi nomi):

```

## 228 Bioinformatika uchun Python

```

5     if username[0].isalpha(): msg =
6         'Salom {0}!'.format(foydalanuvchi nomi)
7     boshqa:
8         msg = "Foydalanuvchi nomingiz raqam bilan boshlanmasligi kerak"
9     shablonini qaytarish('index3', **{'msg':msg})
10
11 ishga tushirish (host = 'localhost', port = 8000)

```

---

**Listing 10.11 (index3.tpl) uchun shablon :**

**Listing 10.12: index3.tpl: indextemplate3.py uchun shablon**

---

```

1 <html lang="en"> 2
<body>{msg}</body>
3     html>
4

```

---

**Listing 10.11** (fayl indextemplate3.py) da biz 6-qatordagi foydalanuvchi nomining birinchi harfini tekshiramiz, shuning uchun biz mantiqni shablondan uzoqlashtiramiz, natijada shablonni o'qish osonroq bo'ladi. Ikkala [Listing 10.10](#) va [10.11](#) ham bir xil natijani berganligi sababli , ikkala strategiya ham ekvivalentga o'xshaydi. Ular emas. Shablonlar mantiqni qo'llab-quvvatlaydi, lekin HTMLda emas (bu erda odatda Python tilini bilmaydigan veb-dizayner uni tahrir qiladi) emas, balki kodingizda (uni disk raskadrovska qilish uchun yaxshiroq vositalar mavjud) murakkab mantiqqa ega bo'lish yaxshiroqdir. Bunday holda, [10.11 ro'yxati](#) 10.10dan ko'ra afzalroqdir.

Bu har qanday shablonda mantiqdan foydalanishdan qochish kerak degani emas. Ba'zida bu juda mantiqiy bo'ladi, masalan, bu vaziyatda:

```

<ul>
Elementlardagi element
uchun %: <li>{{item}}</li>
% end </ul>

```

Bundan chiqish, siz taxmin qilgan shablonlarda mantiqdan foydalanishdir bu saytni saqlashni qiyinlashtirmaydi.

### Statik fayllar

Ba'zi fayllar statik tarzda xizmat ko'rsatadi, ya'ni ular backend jarayoni tomonidan tezda yaratilmaydi. Eng keng tarqalgan holatlar CSS, JavaScript va tasvir fayllari. Statik faylni shablonni hech qanday o'zgaruvchilarsiz qaytarish orqali qaytarishingiz mumkin, ammo Bottle bu fayllar bilan ishlash uchun static\_file usuliga ega. static\_file

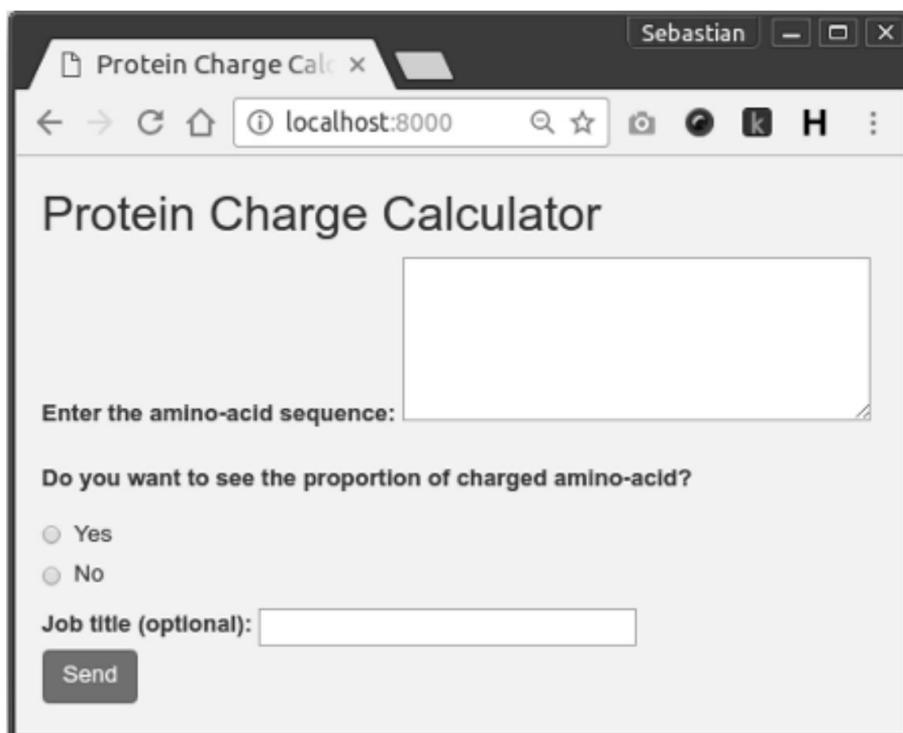
bu holda zarur boylgan qoysimcha funksiyalarni tayminlaydi.<sup>7</sup> Siz fayl nomi va ushbu fayl joylashgan yoylini kiritishingiz kerak:

```
@route('/static/rss.xml') def
rss_static(): static_file('rss.xml',
    root='static/')ni qaytaring
```

Yo'l o'zgaruvchan qismlar bilan o'tishi mumkin, o'zgaruvchi qismini < va > orasiga qo'yish orqali:

```
@route('/static/js/<filename>') def
js_static(fayl nomi):
    statik_faylni qaytaring (fayl nomi, root ='static/js/')
```

#### 10.3.5 Proteinning aniq zaryadini hisoblash uchun veb-dastur (shisha versiyasi)



10.8-rasm Proteinning sof zaryadini hisoblash uchun veb-lova uchun shakl.

Mana, a sof to'lovini hisoblash uchun veb-dasturning Shisha versiyasi

---

<sup>7</sup><https://bottlepy.org/docs/dev/tutorial.html#tutorial-static-files>ga qarang Ushbu usul bo'yicha ko'proq ma'lumot olish uchun.

## 230 Bioinformatika uchun Python

oqsil. Bizga veb-shakl uchun HTML shablon kerak. Bu holda biz protchargeformcgi.html (219-bet) dagi kabi HTML fayldan 9-qatordagi modifikatsiya bilan foydalanishimiz mumkin. “Forma elementi”dagi “harakat atributi” yangi URL manziliga ishora qilishi kerak. Endi u shunday o'qiydi:

```
<forma harakati='/protcharge' method='post'>
```

To'liq fayl protchargeformbottle.html deb nomlanadi va kitob omborini <https://github.com/Serulab/Py4Bio/tree/master/code> manzilida topish mumkin .

Shakl ishlataliganda va foydalanuvchi SEND tugmasini bosganda, brauzer /protcharge URL manziliga POST so'rovini yuboradi. Bu quyidagi kodni bajaradi:

**Listing 10.13: protchargebottle.py: Bottle yordamida oqsilning sof zaryadini hisoblash uchun dasturning orqa tomoni**

---

```
1 shisha import marshrutidan, ishga tushirish, statik_fayl, ko'rish, post, so'rov 2

3 def zaryadlash va prop (aa_seq):
4     """ Proteinning aniq zaryadini va zaryadlangan AA nisbatini hisoblaydi
5
6     protseq = aa_seq.upper()
7     zaryad = -0,002 cp = 0
8     aa_charge =
9     {'C':-045,'D':-0999,'E':-0998,'H':091, 'K ':1,'R':1,'Y':-001} protseqdagi
10        aa uchun: zaryad += aa_charge.get(aa, 0) agar aa
11    aa_charged: cp += 1 tayanch = float(cp)/ len(aa_seq)*100 qaytish
12        (zaryad, tayanch)
13
14
15
16
17
18 @route('/')
19 def index(): return
20     static_file('protchargeformbottle.html', root='views/')
21
22 @route('/css/<filename>')
23 def
24 css_static(fayl nomi): 24 qaytish
25     static_file(fayl nomi, root='css/')
26
27 @post('/protcharge')
28 def
29     protcharge():
30         seq = request.forms.get('aaseq', 'QWERTYYTREWQRTYEYTRQWE')
```

```

30     prop = request.forms.get('prop','n') title =
31     request.forms.get('title', 'Sarlavha yo'q') to'lov, propvalue =
32     chargeandprop(seq) return {'seq': seq,' prop': tayanch, 'title':
33     sarlavha,
34             'zaryad': dumaloq (zaryad, 3), 'propvalue': propvalue}
35
36 ishga tushirish (host = 'localhost', port = 8000)

```

---

**10.13 ro'yxatida** ( protchargebottle.py fayli) 4 ta funktsiya mavjud. Haqiqiy sof to'lovni hisoblash bilan shug'ullanadigan bitta funktsiya (zaryad va prop) va uchta turli URL-manzillarni xaritalash imkonini beradi. Indeks funksiyasi foydalanuvchi bosh sahifaga kirganda va HTMLni forma (protchargeformbottle.html fayli) bilan qaytarganda ishga tushiriladi, buni **10.8-rasmda ko'rish mumkin**. Css\_static funktsiyasi shakl va natija sahifasini to'g'ri ko'rsatish uchun zarur bo'lgan CSS-ni qaytaradi. Protcharge funktsiyasi url domeni/protcharge post so'rovini olganida bajariladi va bu foydalanuvchi protchargeformbottle.html faylidagi shakldagi "YUBORAT" tugmasini bosganida yuboriladi. Bu funksiya natija sahifasini ko'rsatish uchun zarur bo'lgan barcha qiymatlarga ega lug'atni qaytaradi. Bu holatda qo'llaniladigan shablon 27-qatordagi ko'rinish dekoratorida ko'rsatilganidek, natija.html faylidir. Ushbu dekorator ishlashi uchun ushbu fayl katalog ko'rinishida bo'lishi kerakligini unutmang.

**Listing 10.14:** result.html: Protcharge usuli natijasini ko'rsatish uchun shablon

---

```

1 <!DOCTYPE html>
2 <html lang="en"> 3
<head><meta charset="utf-8">
4 <title>Protein zaryadi kalkulyatori: Natija</title> 5 <link href="css/
bootstrap.min.css" rel="stylesheet"> 6 </head> 7 <body style="background-
color:#e7f5f5" ;"> 8 <div class="container"><h2>Natija</h2> 9 <p>Ish
nomi: {{title}}</p> 10 <p>Sizning ketma-ketligingiz: {{seq}}</p> 11 <p>Sof
to'yllov: {{zaryad}}</p> 12 % agar prop == 'y': 13 <p>To'ylangan AA ulushi:
{{propvalue}}<p> 14% oxiri 15 </div> 16 </body> 17 </html>

```

---

### 10.3.6 Apache da WSGI dasturini o'rnatish

Apache veb-serverida WSGI-ni ishga tushirishning bir necha yo'li mavjud. Ushbu kitobda biz Python WSGI interfeysi qo'llab-quvvatlaydigan Python ilovalarini joylashtirish uchun yaratilgan mod\_wsgi, Apache modulidan foydalanamiz.

Modulni loyiha veb-saytidan yuklab olish mumkin<sup>8</sup> yoki operatsion tizim paketi menejeri.<sup>9</sup>

Mod\_wsgi o'rnatilgandan so'ng, quyidagi qatorni qo'shish orqali apache.conf faylini o'zgartirishingiz kerak:

**WSGIScriptAlias veb yo'li path\_in\_server**

bu erda webpath foydalanuvchi tomonidan ko'rildigan yo'l va path\_in\_server bu katalogdagi barcha so'rovlarni qabul qiladigan faylga yo'ldir. Masalan,

**WSGIScriptAlias / /var/www/sitepath/htdocs/test.wsgi**

Bu shuni anglatadiki, veb-serverning asosiy katalogidagi istalgan sahifaga ishora qiluvchi har bir so'rov /var/www/sitepath/htdocs/- test.wsgi-da joylashgan skript tomonidan bajariladi.

## 10.4 PYTHON ASOSLI DINAMIK VEB-SAYTLARNI YASALASH UCHUN MUqqobil Variantlar

---

Shu nuqtaga qadar taqdim etilgan echimlar kichik va o'rta o'lchamdagи saytlarni boshidan qurish uchun etarlicha foydalidir. Ammo agar sizning veb-sayingiz ma'lumotlar bazasini qo'llab-quvvatlash, foydalanuvchi va sessiyalarni boshqarish, ma'muriy interfeys, internatsionalizatsiya, keshlash va boshqalar kabi ilg'or xususiyatlardan foydalansa, ushbu xususiyatlarning aksariyati allaqachon qamrab olingen to'liq xususiyatli veb-ramkadan foydalanish yaxshi bo'ladi. Ushbu turdagи ilovalar ushbu kitob doirasidan tashqarida bo'lganligi sababli, men **10.1-jadvaldagи eng muhim ramkalarni jamlagan jadvalni ko'rsataman**. Jadval abstraktsiya darajasida taxminan tartiblangan. Birinchi yozuvlar kamroq funksiyalarga ega tizimlar va yuqori darajadagi ramkaga qaraganda bir xil natijaga erishish uchu

Hech bir ramka "Python rasmiy veb-ramka" maqomini olmagan, shuning uchun foydalanish va ishlab chiquvchilarining tarqalishi mavjud, ammo Django hozirgacha eng mashhur Python veb-ramka hisoblanadi. Agar siz eng ko'p ishlatalidigan va qo'llab-quvvatlanadigan veb-ramkani o'rganmoqchi bo'lsangiz, Django bu yo'ldir.

## 10.5 SCKRIPT XAVFSIZLIGI HAQIDA BA'ZI SO'ZLAR

---

Agar skriptlaringiz ishonchli muhitlarda (ya'ni Internetda emas) ishlayotgan bo'lsa, siz ushbu bo'limni o'tkazib yuborishingiz va 234-betdagи keyingi bo'limga o'tishingiz mumkin.

Veb-ilovalarni loyihalashda yodda tutish kerak bo'lgan narsa bu foydalanuvchi

<sup>8</sup><http://code.google.com/p/modwsgi/>

<sup>9</sup>Debian-ga asoslangan tizimlarda libapache2-mod-wsgi deb ataladi.

### 10.1-JADVAL Veb-ishlab chiqish uchun ramkalar

Ism	URL	Tavsif
Kolba	<a href="http://tornadoeb.org">tornadoeb.org</a>	Oddiy veb-ramka juda o'xshash Shisha
Tornado	<a href="http://tornadoeb.org">tornadoeb.org</a>	Uzoq so'rovlari va veb-rozetskalar uchun ishlatalidigan veb-ramka va asinxron tarmoq ish kutubxonasi
Plone	<a href="http://plone.net">plone.net</a>	Python-ga asoslangan sozlanadigan kontentni boshqarish tizimidan foydalanishga tayyor
Django	<a href="http://djangoproject.com">djangoproject.com</a>	Tez rivojlanishni rag'batlantiradigan yuqori darajadagi Python veb-ramkasi AJAX va ko'p ma'lumotlar bazasini qo'llab-quvvatlaydigan to'liq stekli yechim.
TurboGears	<a href="http://turbogears.org">turbogears.org</a>	ma'lumotlar bazasiga asoslangan veb-iлоvalarni jadal rivojlantirish uchun bepul ochiq manbali to'liq stekli ramka.
Web2py	<a href="http://www.web2py.com">www.web2py.com</a>	Tez, kengaytiriladigan, xavfsiz va portativ ma'lumotlar bazasiga asoslangan veb-iлоvalarni jadal rivojlantirish uchun bepul ochiq manbali to'liq stekli ramka.

ma'lumotlarni kutilmagan formatda kiritishi mumkin (va bo'ladi). Shaki Internetda hamma uchun ochiq bo'lsa, bu tahdidni e'tiborsiz qoldirmaslik kerak.

Onlayn shaklni qanday to'ldirishni bilmaydigan va eng yaxshi deb hisoblagan narsani sinab ko'radian odamlar bo'ladi. Saytingizni har qanday ekspluatatsiya qilinadigan zaiflikni qidirib sinab ko'radian tajovuzkorlar bo'ladi.

Skriptlaringizdan noto'g'ri foydalanishni oldini olish uchun ishlatalishi mumkin bo'lgan birinchi to'siq bu shaklni tekshirish uchun JavaScript (JS) dan foydalanishdir. Ushbu kitobning maqsadi JSni o'rgatish emas, shuning uchun "Qo'shimcha manbalar" bo'limida havolalar mavjud.

JS oxirgi foydalanuvchi bilan bog'liq muammolarni oldini olish uchun ishlatalishi mumkin, ammo bu sizning serveringizga hujum qilishga qaror qilgan har bir kishi uchun to'sqinlik qiluvchi vosita sifatida foydasiz. Agar kimdir sizning skriptingiz bilan o'zaro aloqada bo'lishni xohlasa, ular buni veb-brauzerdan foydalanmasdan, diqqat bilan yaratilgan JS kodingizni butunlay chetlab o'tishlari mumkin. Shuning uchun barcha ma'lumotlarni tekshirish "server tomonida" ham amalga oshirilishi kerak.

E'tibor qilish kerak bo'lgan yana bir muhim nuqta - skript ma'lumotlar bazasi mexanizmiga kirishi. Buzg'unchi istalmagan natijalarga erishish uchun SQL buyruqlarini kiritishi mumkin (masalan, foydalanuvchi nomlari va parollar kabi nozik ma'lumotlarga ega jadvalning to'liq tarkibini ro'yxatga olish10). Ushbu turdagи hujum "SQL in'ektsiyasi" deb ataladi va u "Python va ma'lumotlar bazalari" bo'limida yoritiladi ([12-bob](#)).

Har qanday kirishni qanday tozalash kerakligi haqida hech qanday qoida yo'q, lekin u

---

10 Ma'lumotlar bazasida oddiy matnda parollarni saqlamasligingiz kerak. Eng yaxshi amaliyot PBKDF2 yoki bcrypt kabi xesh funksiyasidan foydalanib, parol xeshini saqlashdir. Xeshdan tashqari, bir oz "tuz" qo'shishingiz kerak, ya'ni tajovuzkorni oldindan xeshlangan kalitlardan foydalanib, moslikni topish uchun tasodify qator qo'shishingiz kerak. MD5, SHA1, SHA512 va boshqalar kabi sinovdan o'tmagan maxsus algoritmlardan yoki tezkor kriptografik xesh funksiyalaridan foydalanmang.

muayyan ilovaga bog'liq. Quyida ilovangiz xavfsizligini loyihalashda e'tiborga olish kerak bo'lgan ba'zi bir sxemalar mavjud.

1. Ma'lumotlarning ilovaga kirishi mumkinligini aniqlang. Shubhasiz, kirishning eng aniq nuqtasi ma'lumotlarni kiritish uchun o'rnatgan shakklardir. Ammo agar sizning skriptlaringiz RSS tasmasi kabi tashqi manbalarni o'qisa, URL manzillari, serverda saqlangan fayllar va boshqa veb-saytlar kabi boshqa kirish nuqtalarini e'tiborsiz qoldirmasligingiz.
2. Ilovangiz o'zaro aloqada bo'lgan dastur tomonidan ishlataladigan qochish belgilariga e'tibor bering. Ular har doim filtrlanishi kerak. Agar dasturingiz Unix qobig'iga kirsa, ";" ni filrlang. belgi (nuqtali vergul), chunki u ixtiyoriy buyruqlar berish uchun ishlatalishi mumkin. Bu sizning tizimingiz foydalanadigan qobiq turiga bog'liq. Tomosha qilish kerak bo'lgan belgilar: ;, &&, ||, \ va ".
3. Satrlaringiz faqat kerakli belgilarga ega ekanligiga ishonch hosil qilish uchun to'g'ri qabul qilingan belgilarni ("oq ro'yxat") tuzishni o'ylab ko'ring.
4. Veb-server dasturining ishlash imtiyozlari imkon qadar past bo'lishi kerak. Ko'pgina Unix tizimlari veb-server jarayoni uchun adhoc foydalanuvchidan foydalanadi. Bu "Eng kam imtiyozlar printsipi" deb ataladi. Dasturga o'z ishini bajarish uchun zarur bo'lgan eng kichik imtiyozlar beriladi. Bu veb-server jarayoni tajovuzkor tomonidan o'g'irlangan bo'lsa, tizimga nisbatan amalga oshirilishi mumkin bo'lgan suiiste'mollikni cheklaydi.

## 10.6 PYTHON DASTURLARINI QAYERDA HOZLASH MUMKIN

---

Agar siz skriptlaringizni mahalliy serveringizda qoniqarli sinovdan o'tkazgan bo'lsangiz, butun dunyo ulardan bahramand bo'lishi uchun ularni Internetga joylashtirish vaqt keldi. Odatta siz ishlayotgan muassasada skriptlaringizni saqlashingiz mumkin bo'lgan veb-server mavjud bo'lib, buning uchun birinchi qadam IT bo'limidan yordam so'rash bo'ladi. Agar qoniqarli javob olmagan bo'lsangiz, muammoni o'zingiz hal qilish haqida o'ylishingiz kerak bo'ladi. Bu juda qiyin emas. Minglab veb hosting kompaniyalari mavjud. Python-ni aniq qo'llab-quvvatlaydigan birini qidiring.

Veb-xosting bizneslari tomonidan taklif qilinadigan turli xil rejalar orasida, agar sizning skriptingiz juda oddiy bo'lsa va dasturlar yoki qo'shimcha modullarni o'rnatishni o'z ichiga olmasa, "birgalikda" reja turini tanlang. Agar skriptingiz Biopythonda bo'lgani kabi serverda o'rnatilмаган dasturlarni ishga tushirsa, uni o'rnatishni so'rashingiz mumkin. Xizmat bilan shartnoma tuzishdan oldin modullarni talab bo'yicha o'rnatadimi yoki yo'qligini so'rang. Yana bir muammo veb-ramkalar bilan bog'liq. Ba'zilar hosting shartnomasida ruxsat etilmagan uzoq davom etadigan jarayon sifatida ishlaydi.

Xosting serverida o'rnatilgan Python versiyasi sizning skriptlaringizga mos kelishiga ishonch hosil qiling. Serverlar uchun ishlataladigan operatsion tizimlar har bir dasturiy ta'minotning so'nngi emas, balki "barqaror" versiyasidan foydalanishini hisobga olsak, bu kichik mavzu e

Agar veb-xosting xizmati dasturni o'rnatishga ruxsat bermasa, siz kompyuterga ildiz kirish huquqiga ega bo'lgan maxsus hosting yechimini ko'rib chiqishingiz kerak bo'ladi.

o'rnatishingiz mumkin bo'lgan narsalarga nisbatan cheklovlari yo'q. Virtualizatsiya texnologiyalari tufayli, maxsus virtual hosting rejasini arzonroq narxda (shuningdek, VPS Virtual Private Server sifatida ham tanilgan) shartnomaga qilish mumkin.

Buning sababi shundaki, kompyuter turli foydalanuvchilar o'rtasida taqsimlanadi, lekin u umumiy hosting rejasini turidan farq qiladi, chunki har bir foydalanuvchi serverga to'liq kirish huquqiga ega. Juda talabchan ilovalar uchun bu eng yaxshi yechim bo'lmashligi mumkin va siz maxsus xostingdan (virtual emas) foydalanishga murojaat qilishingiz kerak bo'lishi mumkin.

Serverlarga alternativa Google App Engine hisoblanadi. Ushbu tizim sizga Google ilovalarini quvvatlaydigan bir xil kengaytiriladigan tizimlarda veb-i-lovalarni yaratish imkonini beradi. Google-ga Apache veb-server konfiguratsiya fayllari, ishga tushirish skriptlari, ma'lumotlar bazalari, server monitoringi va dasturiy ta'minotni yangilash bilan shug'ullanishiga ruxsat beriladi. Ushbu dvigatel uchun mo'ljallangan ilovalar Python dastur ming tilidan foydalangan holda amalga oshiriladi. App Engine Python ish vaqtini muhitga Python tarjimonining maxsus versiyasini o'z ichiga oladi. "Google App Engine" haqida qo'yshimcha ma'lumot olish uchun <https://cloud.google.com saytiga qarang>.

Amazon Web Services (AWS) shuningdek, Lambda deb nomlangan "serversiz" yechimga ega. AWS Lambda bilan siz faqat kodingizga e'tibor qaratasiz va Lambda xotira, protsessor, tarmoq va boshqa resurslar balansini taklif qiluvchi kompyuterlar parkini boshqaradi. Salbiy tomoni shundaki, siz har bir dastur bajarilishi uchun to'laysiz va dasturlarni allaqachon yozganingizdek ishga tushira olmaysiz; uni ushbu maxsus muhitga moslashtirish uchun ba'zi o'zgarishlar kiritishingiz kerak. U bir nechta tillarda ishlaydi, ammo Python kiritilgan. Qo'shimcha ma'lumot uchun <https://aws.amazon.com/lambda/> saytiga qarang. va <http://docs.aws.amazon.com/lambda/>

## 10.7 QO'SHIMCHA RESURSLAR

---

- W3Schools: JavaScript shaklini tekshirish.  
[https://www.w3schools.com/js/js\\_validation.asp](https://www.w3schools.com/js/js_validation.asp)
- Ma'lumotlarni tekshirish.  
[https://www.owasp.org/index.php/Data\\_Validation](https://www.owasp.org/index.php/Data_Validation)
- JavaScript-Coder.com: JavaScript shaklini tekshirish: Tez va oson!  
<http://www.javascript-coder.com/html-form/javascript-form-validation. phtml>
- HTML ma'lumotnomasi: HTML bo'yicha bepul qo'llanma.  
<http://htmlreference.io>
- Bootstrap: ishlab chiqish uchun eng mashhur HTML, CSS va JS ramka Internetdagi sezgir, mobil birinchi loyihalari. <http://getbootstrap.com>
- PureCSS: Har birida foydalanishingiz mumkin bo'lgan kichik, sezgir CSS modullari to'plami veb-loyiha.  
<http://purecss.io>

## 236 Bioinformatika uchun Python

- **BottlePlate:** python 3.3+ veb-ilovakasi yoki API uchun shisha shabloni serverlar.  
<https://github.com/Rolinh/bottleplate>
- **Shisha + uWSGI:** oddiy veb-ilova konfiguratsiyasi va qiziqarli yashirin xususiyatlar.  
<https://goo.gl/X8Up6S>
- **Dekanter:** Misol ko'rinishi bilan shishaga asoslangan katalog tuzilmasini yaratadi va boshqaruvchi.  
<http://gengo.github.io/decanter/>
- **Veb ishlab chiqishni o'rganing.**  
<https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django>
- **Python tilida veb-dasturlash.** <http://wiki.python.org/moin/WebProgramming>
- **wuzz:** HTTP tekshiruvi uchun interaktiv buyruq qatori vositasi. <https://github.com/asciimoo/wuzz>

### 10.8 O'Z-O'ZINI BAHOLASH

---

1. CGI nima?
2. WSGI nima? Nima uchun bu veb-dasturlash uchun tavsiya etilgan tanlov?
3. Bottle yoki boshqa har qanday "veb-ramka"dan foydalanishning mantiqiy asosi nimada?
4. Shablon tili nima?
5. Statik fayl nima va nima uchun unga boshqacha xizmat qilish kerak?
6. Python cheklangan veb-serverni o'z ichiga oladi. Nima uchun bunday veb-serverdan foydalanasiz?  
Agar Apache kabi bepul to'liq xususiyatlari veb-serverlar mavjud bo'lsa?
7. Web-serverni ishga tushirishda hisobga olinadigan xavfsizlik masalalarini nomlang Internetda.
8. Nima uchun mijoz tomonidan ma'lumotlarni tekshirish server tomonidan ma'lumotlarni tekshirish sifatida foydali emas?
9. Umumiyligi ajratilgan va virtual ajratilgan xosting o'rtaсидаги farq nima? Umumiyligi reja bo'yicha maxsus xostingdan qachon foydalanasiz?

# XML

---

## MAZMUNI

---

11.1 XML ga kirish .....	.....	.....
XML nima? .....	237 .....	.....
XML 10 nuqtada .....	.....	238
11.2 XML hujjatining tuzilishi .....	.....	241
Prolog .....	.....	242
Tana .....	.....	243 11.3 XML
hujjatidagi ma'lumotlarga kirish usullari .....	246	
cElementTree .....	.....	246 11.3.1 SAX:
cElementTree Iterparse .....	.....	246 Go'zal
sho'rva .....	.....	248 11.4
Xulosa .....	.....	251 11.5 Qo'shimcha
manbalar .....	.....	252 11.6 O'z-o'zini
baholash .....	.....	252

---

## 11.1 XML TILIGA KIRISH

---

### XML nima?

Axborot texnologiyalarining barcha sohalarida keng tarqalgan muammo ma'lumotlarni saqlash va almashishdir. Har bir ilovada yaratilgan ma'lumotlarni saqlashning o'ziga xos usuli bor, bu ko'pincha muammo tug'diradi, ayniqsa bizda ma'lumotlarni yaratgan dastur bo'lmasa.

Masalan, Applied Biosystems tomonidan ishlab chiqarilgan Sanger DNK sekvenserlari ma'lumotlarni .ab1 kengaytmali fayllarda saqlaydi. Agar biz bunday faylda saqlangan ma'lumotlarga kirishni istasak, uning ichki tuzilishini bilishimiz kerak. Bunday holda, format yaratuvchisi faylning spetsifikatsiyasini chiqardi1 va bu fayllardan ma'lumotlarimizni ajratib olish uchun kod yozish oson bo'lmasa ham mumkin bo'ladi. Odatda bizda bunday omad yo'q va ma'lumotlar fayli formatlarini yomon hujjatlashtirilgan yoki umuman hujjatlashtirilmagan holda topish juda keng tarqalgan. Ko'p hollarda ushbu fayllarni ochmoqchi bo'lganlar aralash natijalar bilan "teskari muhandislik" ga murojaat qilishlari kerak edi. Ushbu turdag'i muammolarni oldini olish va ilovalar o'rtaida ko'proq suyuqlik almashinuvini amalga oshirish uchun

---

<sup>1</sup>ABI fayllari uchun fayl formati spetsifikatsiyasi [http://www6.appliedbiosystems.com/support/software\\_community/ABIF\\_File\\_Format.pdf](http://www6.appliedbiosystems.com/support/software_community/ABIF_File_Format.pdf) manzilida mavjud .

Turli ishlab chiqaruvchilar W3C2 XML nomi bilan mashhur bo'lgan kengaytiriladigan belgilash tilini ishlab chiqdi.

XML - bu ma'lumotlarni taqdim etish usuli. XML yordamida deyarli har qanday turni ifodalash mumkin. Konfiguratsiya fayllari, ma'lumotlar bazalari, veb-sahifalar, elektron jadvallar va hatto chizmalar XMLda taqdim etilishi va saqlanishi mumkin.

Ba'zi maxsus ilovalar uchun ma'lum turdag'i ma'lumotlarni taqdim etish uchun tayyorlangan XML ning quyi to'plamlari mavjud. Shunday qilib, matematik formulalar MathML deb nomlangan XML lahjasida, SVGda 3 vektor grafiklarida, 4 da kimyoviy formulalarda va CML.5 da saqlanishi mumkin. Barcha asosiy bioinformatika ma'lumotlar bazalarida o'yz ma'lumotlari XML formatida mavjud. Bu shuni anglatadiki, XML o'qishni o'rganish orqali biz eng xilma-xil kelib chiqadigan ko'plab fayllarga kirishimiz mumkin.

Ushbu turdag'i faylni qanday qayta ishlash haqida batafsil ma'lumotga kirishdan oldin ushbu W3C-ni ko'rib chiqing Katta rasmni ko'rsatadigan "10 nuqtada XML"6 hujjati :

## 10 nuqtada XML

- 1. XML ma'lumotlarni tizimlashtirish uchun mo'ljallangan:** Strukturaviy ma'lumotlar elektron jadvallar, manzillar kitoblari, konfiguratsiya parametrlari, moliyaviy operatsiyalar va texnik chizmalar kabi narsalarni o'z ichiga oladi. XML - bu sizning ma'lumotlaringizni tuzilishga imkon beradigan matn formatlarini loyihalash uchun qoidalar to'plamidir (siz ularni ko'rsatmalar yoki konvensiyalar deb ham o'ylishingiz mumkin). XML dasturlash tili emas va undan foydalanish yoki uni o'rganish uchun dasturchi bo'lish shart emas. XML kompyuterga ma'lumotlarni yaratish, ma'lumotlarni o'qish va ma'lumotlar strukturasining bir ma'noli bo'lishini ta'minlashni osonlashtiradi. XML til dizaynidagi keng tarqalgan xatolardan qochadi: u kengaytirilishi mumkin, platformadan mustaqil va xalqarolashtirish va mahalliylashtirishni qo'llab-quvvatlaydi. XML to'liq Unicode bilan mos keladi.
- 2. XML biroz HTMLga o'xshaydi:** HTML kabi, XML ham teglardan ('<' va '>' qavs ichida joylashgan so'zlar) va atributlardan (name="value" shaklidagi) foydalanadi. HTML har bir teg va atribut nimani anglatishini va ko'pincha ular orasidagi matn brauzerda qanday ko'rinishini aniqlasa-da, XML teglardan faqat ma'lumotlar qismlarini chegaralash uchun foydalanadi va ma'lumotlarning talqinini butunlay uni o'qiydigan dasturga qoldiradi. Boshqacha qilib aytganda, agar siz XML faylida "<p>" ni ko'rsangiz, uni paragraf deb o'ylamang. Kontekstga qarab, u narx, parametr, shaxs, p ... bo'lishi mumkin (va bu "p" bilan so'z bo'lishi kerakligini kim aytadi?).
- 3. XML - bu matn, lekin o'qish uchun mo'ljallanmagan:** Elektron jadvallar, manzillar kitoblari va boshqa tuzilgan ma'lumotlarni ishlab chiqaruvchi dasturlar ko'pincha ushbu ma'lumotlarni diskda saqlay

---

<sup>2</sup>World Wide Web Consortium, qisqartirilgan W3C, xalqaro konsortsium  
World Wide Web uchun standartlar ishlab

<sup>3</sup>chiqaradi. <http://www.w3.org/Math>

<sup>4</sup><http://www.w3.org/Graphics/SVG>

<sup>5</sup><http://www.xml-cml.org>

<sup>6</sup>[Http://www.w3.org/XML/1999/XML-in-10-points](http://www.w3.org/XML/1999/XML-in-10-points) dan olingan . “@[1999] tomonidan tasdiqlangan Butunjahon Internet konsorsiumi, (Massachusetts texnologiya instituti, Informatika va matematika bo'yicha Yevropa tadqiqot konsorsiumi, Keio universiteti). Barcha huquqlar himoyalangan.”

ikkilik yoki matn formati. Matn formatining afzalliklaridan biri shundaki, u odamlarga, agar kerak bo'lsa, ma'lumotlarni uni ishlab chiqqan dastursiz ko'rish imkonini beradi; bir chimdim ichida siz sevimli matn muharriri bilan matn formatini o'qishingiz mumkin. Matn formatlari, shuningdek, ishlab chiquvchilarga ilovalarni osonroq disk raskadrovska qilish imkonini beradi. HTML kabi, XML fayllar ham odamlar o'qishi kerak bo'lmagan matnli fayllardir, lekin kerak bo'lganda mumkin. HTML bilan solishtirganda, XML fayllari uchun qoidalar kamroq o'zgarishlarga imkon beradi. Unutilgan teg yoki tirnoqsiz atribut XML faylini yaroqsiz holga keltiradi, HTMLda esa bunday amaliyotga ko'pincha ruxsat beriladi. Rasmiy XML spetsifikatsiyasi ilovalarga buzilgan XML fayli yaratuvchisini taxmin qilishga urinishni taqiqlaydi; agar fayl buzilgan bo'lsa, dastur shu erda to'xtab, xato h

4. **XML dizayni bo'yicha batafsil:** XML matn formati bo'lgani uchun va u ma'lumotlarni chegaralash uchun teglardan foydalanadi, XML fayllari deyarli har doim taqqoslanadigan ikkilik formatlardan kattaroqdir. Bu XML dizaynerlarining ongli qarori edi. Matn formatining afzalliklari aniq (3-bandga qarang) va kamchiliklar odatda boshqa darajada qoplanishi mumkin. Disk maydoni avvalgiga qaraganda arzonroq va zip va gzip kabi siqish dasturlari fayllarni juda yaxshi va juda tez siqishi mumkin. Bundan tashqari, modem protokollari va HTTP/1.1, Internetning asosiy protokoli kabi aloqa protokollari ma'lumotlarni tezda siqib chiqarishi mumkin, bu esa o'tkazish qobiliyatini ikkilik format kabi samarali saqlashi mumkin.
5. **XML – texnologiyalar oilasi:** XML 1.0 – “teglar” va “atributlar” nima ekanligini belgilaydigan spetsifikatsiya. XML 1.0 dan tashqari, "XML oilasi" - bu muhim va tez-tez talab qilinadigan vazifalarni bajarish uchun foydali xizmatlarni taklif qiluvchi modullar to'plami. XLink XML fayliga giperhavolalar qo'shishning standart usulini tavsiflaydi. XPointer - bu XML hujjatining qismlariga ishora qilish uchun ishlab chiqilayotgan sintaksis. XPointer biroz URL manziliga o'xshaydi, lekin u Internetdagi hujjatlarga ishora qilish o'rniga, XML faylidagi ma'lumotlar bo'laklariga is CSS, uslublar jadvali tili, HTML uchun bo'lgani kabi XML uchun ham qo'llaniladi. XSL uslublar jadvallarini ifodalash uchun ilg'or tildir. U teglar va atributlarni qayta tartibga solish, qo'shish va o'chirish uchun ishlatiladigan o'zgartirish tili bo'lgan XSLT-ga asoslangan. DOM - bu XML (va HTML) fayllarni dasturlash tilidan manipulyatsiya qilish uchun funksiya chaqiruvlarining standart to'plami. XML sxemalari 1 va 2 ishlab chiquvchilarga o'zlarining XML-ga asoslangan formatlarining tuzilmalari. Yana bir nechta modul va vositalar mavjud yoki ishlab chiqilmoqda. W3C texnik hisobotlar sahifasini kuzatib boring.
6. **XML yangi, lekin unchalik yangi emas:** XMLni ishlab chiqish 1996 yilda boshlangan va u 1998 yil fevral oyidan beri W3C tavsiysi bo'lib kelgan, bu sizni bu yetarlicha etuk texnologiya ekanligiga shubha qilishingiz mumkin. Aslida, texnologiya unchalik yangi emas. XMLdan oldin 80-yillarning boshlarida ishlab chiqilgan SGML mavjud edi, 1986 yildan beri ISO standarti va yirik hujjatlashtirish loyihibarida keng qo'shilishni rivojanishi 1990 yilda boshlangan. XML dizaynerlari HTML bilan ishlash tajribasidan kelib chiqqan holda SGML ning eng yaxshi qismlarini olishgan va

**SGML** dan kam bo'Imagan kuchliroq va ancha muntazamroq va ishlatish uchun oddiy narsalarni ishlab chiqardi. Biroq, ba'zi evolyutsiyalarni inqiloblardan ajratish qiyin ... Va shuni aytish kerakki, **SGML** asosan texnik hujjatlar uchun va boshqa turdag'i ma'lumotlar uchun kamroq qo'llaniladi, **XML** bilan bu butunlay teskari.

7. **XML HTMLni XHTML ga olib boradi:** Hujjat formati bo'lgan muhim XML ilovasi mavjud: W3C ning XHTML, HTML ning vorisi. XHTML HTML bilan bir xil elementlarning ko'piga ega. XML qoidalariiga mos kelish uchun sintaksis biroz o'zgartirildi. "XML-asosli" format sintaksisni XML-dan meros qilib oladi va uni ma'lum usullar bilan cheklaydi (masalan, XHTML "<p>" ga ruxsat beradi, lekin "<r>" ga emas); shuningdek, bu sintaksisga ma'nio qo'shadi (XHTML "<p>" "narx", "odam" yoki boshqa narsalarni emas, balki "paragraf" degan ma'noni anglatadi).
8. **XML modulli:** XML boshqa formatlarni birlashtirish va qayta ishlatish orqali yangi hujjat formatini aniqlash imkonini beradi. Mustaqil ravishda ishlab chiqilgan ikkita format bir xil nomdagi elementlar yoki atributlarga ega bo'lishi mumkinligi sababli, bu formatlarni birlashtirishda ehtiyyot bo'lish kerak ("<p>" mat uchun "paragraf" yoki "shaxs" degan ma'noni anglatadimi?). Formatlarni birlashtirishda nom chalkashliklarini bartaraf qilish uchun XML nom maydoni mexanizmini taqdim etadi. XSL va RDF nomlar bo'shliqlaridan foydalanadigan XML-ga asoslangan formatlarning yaxshi namunalari. XML sxemasi XML hujjat tuzilmalarini aniqlash darajasida modullik uchun ushbu qo'llab-quvvatlashni aks ettirish uchun mo'ljallangan, bu ikkita sxemani birlashtirilgan hujjat tuzilmasini qamrab oluvchi uchinchisini.
9. **XML RDF va Semantik Internet uchun asosdir:** W3C Resurs tavsifi ramkasi (RDF) musiqa iyo ro'yxatlari, fotosuratlar to'plamlari va bibliografiyalar kabi manba tavsifi va metama'lumotlar ilovalarini qo'llab-quvvatlaydigan XML matn formatidir. Masalan, RDF shaxsiy kontaktlar ro'yxatidagi ma'lumotlardan foydalangan holda veb-fotoalbumdagi odamlarni aniqlash imkonini berishi mumkin; keyin sizning pochta mijozingiz o'sha odamlarga ularning fotosuratları Internetda ekanligi haqida xabarni avtomatik ravishda boshlashi mumkin. Xuddi HTML integratsiyalangan hujjatlar, tasvirlar, menu tizimlari va asl Internetni ishga tushirish uchun ilovalarni yaratgani kabi, RDF ham Internetni biroz ko'proq Semantik Internetga aylantirish uchun yanada ko'proq integratsiyalash uchun vositalarni taqdim etadi. Odamlar o'z muloqotlarida foydalanadigan so'zlarning ma'nolari to'g'risida kelishib olishlari kerak bo'lgani kabi, kompyuterlar ham samarali muloqot qilish uchun atamalarning ma'nolarini kelishish mexanizmlariga muhtoj. Ma'lum bir sohada (masalan, xarid qilish yoki ishlab chiqarishda) atamalarning noto'g' RDF, ontologiyalar va kompyuterlar odamlarga ishlashda yordam berishi uchun ma'noni ifodalash - bularning barchasi Semantik veb-faoliyatning mavzulari.
10. **XML litsenziyasiz, platformadan mustaqil va yaxshi qo'llab-quvvatlanadi:** XMLni loyiha uchun asos sifatida tanlab, siz keng va o'yisib borayotgan vositalar hamjamiyatiga kirish huquqiga ega bo'ylasiz (ulardan biri sizga kerak bo'ylgan narsani qila oladi).

texnologiya sohasida tajribaga ega. XML-ni tanlash ma'lumotlar bazalari uchun SQL-ni tanlashga o'xshaydi: siz hali ham o'z ma'lumotlar bazasini va uni boshqaradigan o'z dastur va protsedralaringizni yaratishingiz kerak, ammo ko'plab vositalar mavjud va sizga yordam beradigan ko'plab odamlar mavjud. XML litsenziyasiz bo'lgani uchun siz hech kimga hech narsa to'lamasdan uning atrofida o'z dasturiy ta'minotiningizni yaratishingiz mumkin. Katta va o'sib borayotgan yordam sizning bitta sotuvchiga bog'lanmaganligingizni anglatadi. XML har doim ham eng yaxshi yechim emas, lekin har doim e'tiborga olish kerak.

## 11.2 XML HUJJATINING TUZILISHI

---

XML hujjatining ichki tuzilishi tafsilotlarini bilishimiz shart emas.

Buning sababi shundaki, Python ushbu turdag'i fayllarga kirish uchun o'z vositalariga ega. Python dasturchilari ushbu vositalarni yaratish uchun XMLning ichki qismlari bilan shug'ullanishlari kerak edi; ammo Python tomonidan taqdim etilgan vositalardan yaxshiroq foydalanish uchun XML fayllari tuzilishi haqida ozgina tasavvurga ega bo'lish kerak deb o'layman.

Keling, XML hujjatining namunasini ko'rib chiqaylik, bu holda UniProt yozuvi:<sup>7</sup>

Listing 11.1: Q9JJE1.xml: UniProt yozuvi XML formatida

---

```
<?xml version="1.0" encoding="UTF-8"?> <uniprot
xmlns="http://uniprot.org/uniprot" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
uniprot.org/uniprot http://www.uniprot.org/support/docs/uniprot.xsd">
<entry dataset="TrEMBL" created="2000-10-01" version="35">

    <accession>Q9JJE1</accession>
    <organism key="1"> <name
        type="scientific">Musculus musculus</name> <lineage>
        <taxon>Eukariota</taxon> <taxon>Metazoa</taxon> <
        taxon>Chordata</taxon> <taxon>Craniata</taxon>
        <taxon>Vertebrata</taxon> <taxon>Euteleostomi</taxon>
        <taxon>Sutemizuvchilar</taxon> <taxon>Eutheria</taxon>
        <taxon> Euarchontoglires</taxon> <taxon>Glires</taxon>
        <taxon>Rodentia</taxon> <taxon>Sciurognathi</taxon>
```

---

<sup>7</sup>Ushbu yozuv sahifaga mos ravishda o'zgartirildi. Ushbu faylni Github repozitoriyida topish mumkin namunalar katalogi ostida uniprotrecord.xml nomi bilan.

```

<taxon>Muroidea</taxon>
<taxon>Muridae</taxon>
<taxon>Murinae</taxon>
<taxon>Mus</taxon> </lineage>

</organism> <dbReference
type="UniGene" id=" Mm.248907"
kalit="5"/> <tartibning uzunligi="393" nazorat summasi="E0C0CC2E1F189B8A">

MPKKKPTPIQLNPAPDGSANVGTSSAETNLEALQKKLEELDEQQRKRL
EAFLTQKQKVGEKDDDFEKISELGAGNGGVFKVSHKPSGLVMARKLIH
LEIKPAIRNQIIRELQLVHECNSPYIVGFYGAFYSDGEISICMEHMDGGS
LDQVLKKAGRIPQILGKVSIAVIKGLTYLREKHKIMHRDVKPSNILVNS
RGEIKLCDFGVSGQLIDSMANSFVGTRSYMSPERLQGTHYSVQSDIWSMG
LSLVEMAVGRYPIPPDAKELELLFGCHVEGDAETPPPRTPGGPLSSY
GMDSRPPMAIFELLDYIVNEPPPKLPSGVFSLEFQDFVNKCLIKNPAERA
DLKQLMVHAFIKRSDAEEVDFAFWLCSTIGLNQPSTPTHAASI
</sequence> </entry> </uniprot>

```

Keng konturlarda XML hujjatining tuzilishi oddiy. U asosan muqaddima, tana va epilogdan iborat.8

## Prolog

Prolog - bu XML ma'lumotlarining boshlanishini belgilovchi va tahlil qiluvchiga muhim ma'lumotlarni beruvchi ixtiyoriy bo'lim. Muqaddima faqat bitta satrga ega bo'lishi mumkin, masalan,

```
<?xml version="1.0" encoding="UTF-8"?>
```

Yoki bir nechta qatorlar:

```
<?xml version="1.0"?> <!
DOCTYPE BlastOutput PUBLIC "-//NCBI//NCBI BlastOutput/EN" " http://www.ncbi.nlm.nih.gov/dtd/NCBI\_BlastOutput.dtd" <!-- XMLSPY bilan tahrirlangan
(http://www.xmlspy.com) Andy --> tomonidan
```

Birinchi qator - XML deklaratsiyasi, bu erda XML versiyasi va belgi kod belgilanadi. Belgilar kodi haqidagi ma'lumot faqat hujjat UTF-8 yoki UTF-16 da kodlangan bo'lsa, ixtiyoriyidir.

Ikkinci qator DOCTYPE deklaratsiyasi bo'lib, uning maqsadi XML hujjatini hujjat turi ta'rifi (DTD) bilan bog'lashdir. Ushbu DTD fayli mavjud

<sup>8</sup>Epilog kamdan-kam qo'llaniladi, shuning uchun prolog va asosiy XML faylining eng muhim qismlari hisoblanadi.

XML faylining muayyan tuzilishi haqida ma'lumot: unda qaysi teglar va atributlarga ruxsat berilganligi, shuningdek ularni qaerdan topish mumkinligi aytildi. Ba'zi hollarda DTD ma'lumotnomasi o'rniga XML sxemasi deb ataladigan DTD uchun muqobil usulga havolalar mavjud bo'lib, u bir xil funktsiyaga xizmat qiladi, lekin yaxshi ishlashi va XMLga asoslangan sintaksisi bilan. DTD yoki XML sxema faylining tuzilishi ushbu kitob doirasidan tashqarida; ammo, Internetda bir nechta, juda to'liq, havolalar mavjud (ushbu bobning oxiridagi Qo'shimcha manbalarga qarang).

Uchinchi qator, bu holda, sharhdir. Bu Pythonda # ga teng. U "<lyy>" bilan boshlanib, ">" bilan tugaydi va XML hujjatining bosh qismida ham, boshida ham bo'lishi mumkin. Bu HTML-da qo'llaniladigan bir xil turdag'i sharh va u bir nechta satrlarni qamrab olishi mumkin.

## Tana

Tana - bu elementlar yashaydigan joy, XML fayllarining haqiqiy qahramonlari. Element - bu boshlang'ich tegning boshidan oxirigacha bo'lgan ma'lumot, shu jumladan ularning orasidagi barcha narsalar.

Bu erda XML korpusida topilishi mumkin bo'lgan elementning namunasi hujjat:

```
<taxon>Eukariota</taxon>
```

Bu yerda <taxon> - boshlang'ich teg, </taxon> - yakuniy teg va mazmuni (Eukaryota) - bu ikki teg o'rtaida joylashgan.

Elementlar bo'sh ko'rinishi mumkin. Yozish to'g'ri, masalan:

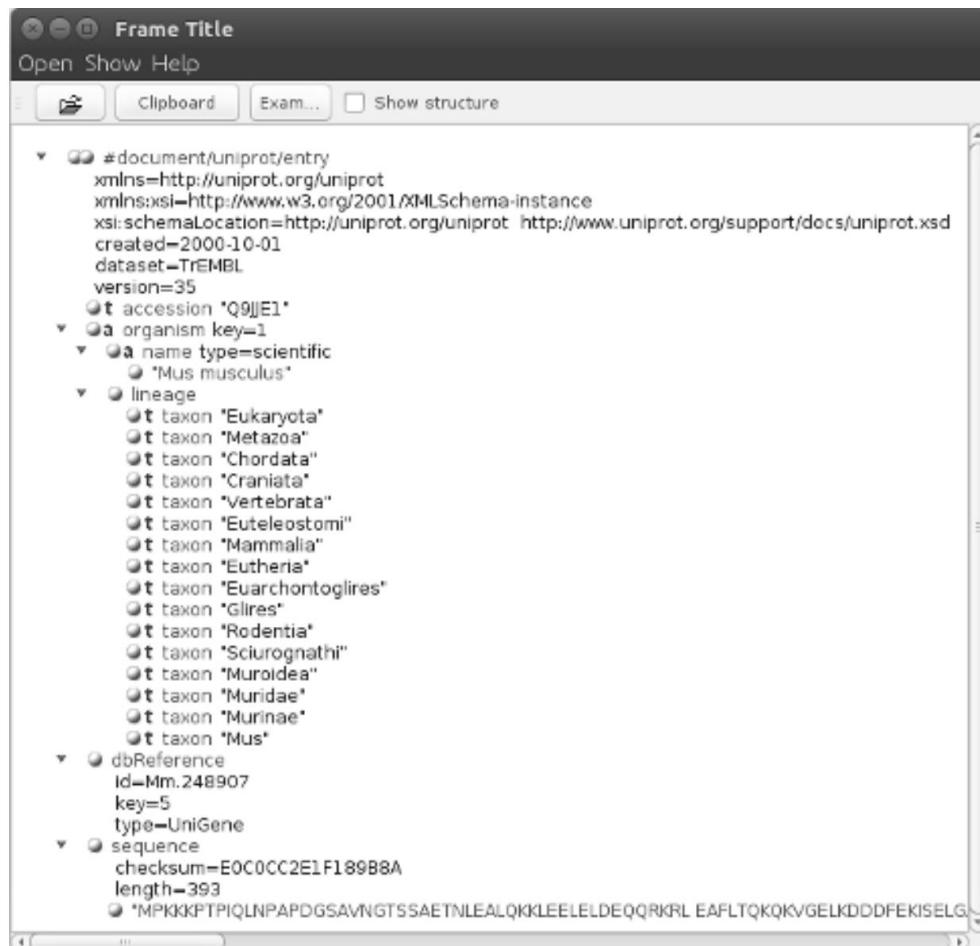
```
<accession></accession>
```

Garchi bu holda "qo'shilish" elementida hech narsa bo'lmasligi unchalik ma'noga ega bo'lmasa-da (UniProt yozuvi har doim bir nechta ulanishlarga ega bo'lishi kerak), boshqa turdag'i ma'lumotlar uchun elementning mazmuni bo'lishi mumkin. Ixtiyoriy.

Bo'sh elementlarni ifodalashning qisqartirilgan usuli mavjud bo'lib, "bo'sh element yorlig'i" deb ataladi, bu element nomidan keyin to'g'ridan-to'g'ri chiziq (/) dan iborat bo'lib, barchasi burchakli qavslar bilan o'ralgan, masalan:

```
<qo'shilish/>
```

Elementlar bir-birining ichiga "joylashtirilgan" bo'lishi mumkin. [11.1 ro'yxatida](#) biz "takson" elementi "nasab" ichida qanday joylashtirilganligini ko'rishimiz mumkin. Bu ierarxik tuzilma haqida tasavvur beradi: boshqalarga bo'ysunadigan elementlar mavjud. Biz "takson" ni ko'ramiz. "organizm" elementi bo'lgan "nasab" elementidir. Odatda bu turdag'i tuzilma daraxt bilan taqqoslanadi. Birinchi element "Hujjat elementi" deb nomlanadi (bu holda "uniprot"), undan qolgan barcha narsalar, ya'ni uning "bolalari" osilgan. Ushbu daraxtning grafik tasvirini olish uchun XML Viewer,9 kabi dasturdan ([11.2-rasmga qarang](#)) yoki <http://codebeautify.org/xmlviewer> kabi veb-saytdan foydalanish mumkin. Bu 11.2-rasmda ko'rsatilganidek, ma'lumotlar va hujjat tuzilishini ko'rsatadi .



11.1-rasm XML ko'rurvchisi skrinshoti: Daraxt ko'rurvchisi Q9JJE1.xml hujjatining tuzilishini ko'rsatadi ([11.1 ro'yxat](#) ).

Ba'zi elementlarda "atributlar", ya'ni ele haqida qo'shimcha ma'lumotlar mavjud  
ment. Atributga ega elementning umumiyl sintaksisi

```
<element attributeName="value">
```

Listing 11.1 misolda davom etsak , biz atributlarga ega bo'lgan boshqa elementlarga  
duch kelamiz, masalan,

```
<nom turi="ilmiy">
```

Bu holda "nom" deb nomlangan element "tur" atributiga ega bo'lib, unda a

Shakl 11.2 Codebeautify XML ko'ruchisi, Q9JJE1.xml hujjatidagi tuzilma va ma'lumotlarni ko'rsatadi ([11.1 ro'yxati](#)).

"ilmiy" qiymati. Bundan tashqari, u "ketma-ketlik" elementidagi kabi bir nechta atributga ega bo'lishi mumkin:

```
<sequence length="393" checksum="E0C0CC2E1F189B8A">
```

Bu erda atributlar "uzunlik" va "nazorat summasi", ularning qiymatlari "393" va Mos ravishda "E0C0CC2E1F189B8A".

Ushbu bosqichda bizda XML faylidagi ma'lumotlar haqida tasavvurga ega bo'lган elementlar mavjud. Q9JJE1.xml faylining yozuvidan ([11.1 ro'yxat](#)) shuni aytishimiz mumkinki, "ketma-ketlik" elementida 393 bp uzunlikdagi nukleotidlar ketma-ketligi, ma'lum imzo va UniProt bazasidan "Q9JJE1" identifikatori mavjud. Bularning barchasi ma'lumotlar tuzilishini oldindan bilmasdan va maxsus dasturdan foydalanmasdan.

Tanish mumkin bo'lgan elementni topishingiz mumkinligini bilish uchun .ab1 faylini ochishga harakat qiling.

XML-fayllarning tuzilishi haqida umumiylar ma'lumotga ega bo'lisingizga qaramay, siz ushbu formatda ushbu kitob doirasidan tashqariga chiqadigan boshqa xususiyatlarni topasiz. Agar siz XML haqida ko'proq bilmogchi bo'lsangiz, bobning oxiridagi resurslar ro'yxatiga qarang.

Quyidagi bo'limga Python yordamida XML hujjatlarining mazmuniga qanday kirish mumkinligi ko'rsatilgan.

## 11.3 XML HUJJATDAGI MA'LUMOTLARGA KIRISH USULLARI

---

Siz foydalanadigan dasturlash tilidan qat'i nazar, XML faylidagi ma'lumotlarga kirish uchun ikkita strategiyadan foydalanishingiz mumkin.

Bir tomondan, siz faylni to'liq o'qib chiqishingiz, elementlar o'rtasidagi munosabatlarni tahlil qilishingiz va daraxt tipidagi tuzilmani yaratishingiz mumkin, buning yordamida dastur ma'lumotlar bo'yicha harakatlana oladi. Bu Document Object Model (DOM) deb ataladi va XML hujjatlarini tahlil qilishda W3C tomonidan tavsiya etilgan usuldir.

Yana bir imkoniyat shundaki, dastur daraxt turi tasvirini yaratish zaruratsiz elementning boshlanishi va tugashi kabi hodisalarni aniqlaydi va hisobot beradi. Agar daraxt tasviri kerak bo'lsa, bu vazifa dasturchiga qoldiriladi. Bu XML uchun Simple API (SAX) tomonidan qo'llaniladigan usul.

Odatda bu turdagи tahlilchilar "hodisaga asoslangan tahlilchilar" deb ataladi. Ushbu bobda biz voqeaga asoslangan tahlil qiluvchi misol sifatida cElementTree-dan Iтерparse-ni ko'ramiz.

Ba'zi hollarda DOM dan foydalanish qulay, boshqa hollarda esa SAX afzalroqdir. DOM odatda butun daraxtni keyinchalik o'tish uchun xotirada saqlashni nazarda tutadi. Bu katta hajmdagi hujjatlarni tahlil qilish vaqtida muammo tug'dirishi mumkin, ayniqsa siz faqat bitta element qiymatining mavjudligini aniqlamoqchi bo'lsangiz. Bunday hollarda SAX eng samarali tahlil qiluvchi hisoblanadi. Shunga qaramay, ko'pgina ilovalar daraxt ichidagi barcha elementlarda ishlashni talab qiladi, buning uchun biz DOMga murojaat qilishimiz kerak. Dasturchi nuqtai nazaridan, DOM interfeysidan foydalanish SAX-ga qaraganda osonroq, chunki u hodisaga asoslangan dasturlashni talab qilmaydi.

### cElementTree

cElementTree - SAX tahlilchisi bo'lib, u tez va kamroq xotiradan foydalangan holda tahlil qilish uchun optimallashtirilgan. cElementTree ning yana bir afzalligi Iтерparse deb nomlangan funksiyadir. Ushbu funktsiya bizga keyingi bo'limda tushuntiriladigan voqeaga asoslangan tahlilchidan foydalanishni ta'minlaydi.

#### 11.3.1 SAX: cElementTree Iтерparse

cElementTree Iтерparse SAX emas, lekin u bu yerga kiritilgan, chunki boshqa tahlilchilardan farqli o'laroq, u voqealarga asoslangan.

Iтерparse shakldagi kortejlar bo'yicha takrorlanadigan oqimni qaytaradi (**hodisa, element**). Bu elementlarni takrorlash va ularni tezda qayta ishlash uchun ishlataladi.

Protein ketma-ketligini va uning atributlarini oling:

```
>>> hodisa uchun xml.etree.cElementTree ni cET
>>> sifatida import qiling, cET.Iтерparse('uniprotrecord.xml', events=('start',
 'end')) da element: agar event=='end' va elem.tagdagi "ketma-ketlik":  
  
    print('Tartib: {0}'.format(elem.text)) print('Nazorat
summasi: {0}'.format(elem.attrib["nazorat summasi"])) print('Uzunlik:
{0}'.format (elem.attrib["uzunlik"]))
```

```
elem.clear()
```

**Ketma-ketlik:**

```
MPKKKPTIQLNPAPDGSAVNGTSSAETNLEALQKKLEEELDEQQRKRL
EAFLTQKQKVGELKDDDFEKISELGAGNGGVFKVSHKPSGLVMARKLIH
LEIKPAIRNQIIRELQVLHECNSPYIVGFYGAFYSDGEISICMEHMDGGS
LDQVLKKAGRIPQILGKVSIAVIKGLTYLREKHKIMHRDVKPSNILVNS
RGEIKLCDFGVSGQLIDSMANSFVGTRSYMSPERLQGTHYSVQSDIWSMG
LSLVEMAVGYPPIPPPDAKELELLFGCHVEGDAAETPPRPRTPGGPLSSY
GMDSRPPMAIFELLDYIVNEPPPKLPSGVFSLEFQDFVNKCLIKNPAERA
DLKQLMVHAFIKRSDAEEVDFAGWLCSTIGLNQPSTPTHAASI
```

Tekshirish summasi: E0C0CC2E1F189B8A

Uzunligi: 393

iterparse kortejni qaytaradi. Kortejning birinchi elementi hodisadir va ikkita qiymatdan biri bo'lishi mumkin: "start" yoki "end". Agar biz qabul qilgan hodisa "start" bo'lsa, bu biz element nomi va uning atributlariga kirishimiz mumkinligini anglatadi, lekin uning matni shart emas. Biz "tugash" ni olganimizda, biz ushbu elementning barcha komponentlarini qayta ishlaganimizga amin bo'lishimiz mumkin. Shu sababli oldindi kod nafaqat tanlangan elementga yetib kelganimizni, balki "end" hodisasini ham topganimizni tekshirdi.<sup>10</sup> Agar tahlilchi faqat "end" ni qaytarsa, bu tekshiruvga ehtiyoj qolmaydi:

```
>>> hodisa uchun, cET.iterparse('uniprotrecord.xml') da element: elem.tagdagi 'ketma-
ketlik':
    print('Tartib: {0}'.format(elem.text)) print('Nazorat summasi:
{0}'.format(elem.attrib["nazorat summasi"])) print('Uzunlik: {0}'.format
(elem.attrib["uzunlik"])) elem.clear()
```

**Ketma-ket:**

```
MPKKKPTIQLNPAPDGSAVNGTSSAETNLEALQKKLEEELDEQQRKRL
EAFLTQKQKVGELKDDDFEKISELGAGNGGVFKVSHKPSGLVMARKLIH
LEIKPAIRNQIIRELQVLHECNSPYIVGFYGAFYSDGEISICMEHMDGGS
LDQVLKKAGRIPQILGKVSIAVIKGLTYLREKHKIMHRDVKPSNILVNS
RGEIKLCDFGVSGQLIDSMANSFVGTRSYMSPERLQGTHYSVQSDIWSMG
LSLVEMAVGYPPIPPPDAKELELLFGCHVEGDAAETPPRPRTPGGPLSSY
GMDSRPPMAIFELLDYIVNEPPPKLPSGVFSLEFQDFVNKCLIKNPAERA
DLKQLMVHAFIKRSDAEEVDFAGWLCSTIGLNQPSTPTHAASI
```

---

<sup>10</sup>Joriy amalga oshirishda tahlilchi 16 Kb bo'laklarni o'qiydi, shuning uchun bu holda butun ketma-ketlikni "start" elementidan o'qish mumkin edi. Barcha elementlarni olganingizga ishonch hosil qilish uchun uni "tugash" elementidan keyin o'qishingiz kerak.

Tekshirish summasi: E0C0CC2E1F189B8A

Uzunligi: 393

Toza usulga kelsak, u ishlataligandan keyin tugunni "tozalash" uchun ishlataladi, chunki ElementTree kabi klassik SAX tahlilchisidan farqli o'laroq, iterparse to'liq daraxtni yaratadi. Ushbu kod bilan bog'liq muammo shundaki, asosiy element barcha (hozir bo'sh) bolalar bilan qoladi va u xotiradan foydalanadi. Ushbu oddiy misolda bu xatti-harakatlar muammoli emas, lekin katta hajmdagi fayllarni qayta ishlashda bo'lishi mumkin. Ideal, uni tozalash uchun ota-onha tuguniga kirishdir.

Buning bir usuli - birinchi o'zgaruvchiga havolani saqlash; Buning uchun biz yaratamiz iterator va undan birinchi elementni oling, uni "ildiz" deb nomlang:

```
>>> allelements = iterparse('uniprotrecord.xml', events=('start','<='
    'oxiri'))
>>> allelements = iter (allelements) >>> hodisa,
root = keyingi (allelements)
```

Endi biz uni avvalgidek qayta ishlaymiz, faqat bu safar biz asosiy elementni o'chirib tashlashimiz mumkin:

```
>>> hodisa uchun, allelementlarda element: if
    event=='end' va 'sequence' element.tag: print(elem.text)
        root.clear()
```

## BeautifulSoup

BeautifulSoup11 - bu XML va HTML fayllarini tahlil qilish uchun ishlataladigan tashqi modul. Uning Python o'rnatilgan parserlaridan asosiy afzalligi shundaki, u noto'g'ri tuzilgan (buzilgan) HTML fayllarini tahlil qila oladi.

Ushbu modul fonda tahlil qilish ishini bajaradigan boshqa modulni chaqiradi. Bunday holda biz lxml dan foydalanamiz; boshqalar ham bor, lekin men undan foydalanaman, chunki u Python tilida XML va HTMLni qayta ishlash uchun eng boy va ishlatalish uchun qulay kutubxonasi.

Bu tashqi modul bo'lgani uchun uni o'rnatishingiz kerak:

```
$ pip o'rnatning beautifulsoup4
```

yoki agar siz Anaconda dan foydalansangiz:

```
$ conda beautifulsoup4 o'rnatish
```

O'rnatilgandan so'ng, birinchi qadam uni chaqirish orqali BeautifulSoup ob'ektini yaratishdir ikki parametrali, fayl ob'ekti va tahlil qiluvchi sinf. Mana umumiy shakl:

---

<sup>11</sup><https://www.crummy.com/software/BeautifulSoup> saytida mavjud .

**BeautifulSoup(FILE\_OBJECT yoki STRING, PARSER)**

Keling, buni amalda ko'rib chiqaylik:

```
>>> bs4 dan BeautifulSoup ni bs sifatida import qiling
>>> sho'rva = bs(open('uniprotrecord.xml'), 'lxml')
```

Agar xml fayli mahalliy fayl emas, balki Internet-resurs bo'lsa, siz foydalanishingiz mumkin so'rovlari kutubxonasi.<sup>12</sup> Agar shunday bo'lsa, avval so'rovlarni o'rnatish va import qiling:

```
(py4bio) $ pip o'rnatish so'rovlari
So'rovlarni yig'ish
Yuklab olish soyo'rovlar-2.13.0-py2.py3-none-any.whl (584kB) 100% |
*****| 593kB 968kB/s
```

Yig'ilgan paketlarni o'rnatish: soyo'rovlar Muvaffaqiyatli  
o'rnatilgan soyo'rovlar-2.13.0 (py4bio) \$ python Python  
3.5.2 (standart, 2016-yil 17-noyabr, 17:05:23)

[GCC 5.4.0 20160609] Linuxda Qo'shimcha

ma'lumot uchun "yordam", "mualliflik huquqi", "kreditlar" yoki "litsenziya" ni kriting. >>> import  
so'rovlari

**Import qilingandan so'ng, kontentni olish uchun URL va kontentni olish uchun .get() dan foydalaning:**

```
>>> url = 'https://s3.amazonaws.com/py4bio/uniprotrecord.xml' >>> talab =
requests.get(url) >>> c = req.content
```

**XML faylining mazmuni (c) BeautifulSoup uchun parametr sifatida ishlatalishi mumkin:**

```
>>> bs4 dan BeautifulSoupni bs sifatida import qiling
>>> sho'rva = bs(c, 'lxml')
```

Endi bizda sho'rva deb ataladigan BeautifulSoup ob'ekti bor. Elementga kirish uchun element nomidan ushbu obyektning xususiyati sifatida foydalaning. Ketma-ketlikni olish uchun foydalaning sho'rva.ketma-ket:

```
>>> soup.sequence
<sequence checksum="E0C0CC2E1F189B8A" length="393">
MPKKKPTPIQLNPAPDGSAVNGTSSAETNLEALQKKLEELELDEQQRKRL
EAFLTQKQKVGELKDDDFEKISELGAGNGGVFKVSHKPSGLVMARKLIH
LEIKPAIRNQIIRELQVLHECNSPYIVGFYGAFYSDGEISICMEHMDGGS
LDQVLKKAGRIPQEQLGKVSIAVIKGLTYLREKHKIMHRDVKPNSILVNS
```

<sup>12</sup>Internetdan fayllarni olish uchun o'rnatilgan kutubxonalar mavjud (masalan, urllib2), lekin so'rovlari kamroq, murakkab va har qanday o'rnatilgan kutubxonaga qaraganda ko'proq xususiyatlarga ega.

## 250 Bioinformatika uchun Python

```
RGEIKLCDFGVSGQLIDSMANSFVGTRSYMSPERLQGTHYSVQSDIWSMG  
LSLVEAVGRYPPIPPDAKELELLFGCHVEGDAAETPPRPTPGGPLSSY  
GMDSRPPMAIFELLDYIVNEPPPVLPSGVFSLEFQDFVNKCLIKNPAERA  
DLKQLMVHAFIKRSDAEEVDFAGWLCSTIGLNQPSTPTHAASI  
</sequence>
```

Agar siz ushbu elementning mazmunini xohlasangiz, qatordan foydalaning:

```
>>> soup.sequence.string  
'\nMPKKPPTPIQLNPAPDGSAVNGTSSAETNLEALQKKLEELEDEQQRKRL\nEAFLTQKQKV<=  
GELKDDDFEKISELGAGNGGVFKVSHPSGLVMARKLIH\nLEIKPAIRNQIIRELQLVHECNS<=  
PYIVGFYGAFYSDGEISICMEHMDGGS\nLDQVLKKGRIPEQILGKVSIAVIKGLTYLREKHKI<=  
MHRDVKPSNILVNS\nRGEIKLCDFGVSGQLIDSMANSFVGTRSYMSPERLQGTHYSVQSDIWSM<=  
G\nLSLVEAVGRYPPIPPDAKELELLFGCHVEGDAAETPPRPTPGGPLSSY\nGMDSRPPMAI<=  
FELLDYIVNEPPPVLPSGVFSLEFQDFVNKCLIKNPAERA\nDLKQLMVHAFIKRSDAEEVDFAG<=  
WLCSTIGLNQPSTPTHAASI\n'
```

Ushbu elementning "nazarat summasi" va "uzunlik" xususiyatlarini olish uchun .get() dan foydalaning:

```
>>> soup.sequence.get('checksum')  
'E0C0CC2E1F189B8A'  
>>> soup.sequence.get('uzunlik')  
'393'
```

Agar elementda bir nechta bolalar bo'lsa, siz uni takrorlashingiz mumkin. Masalan, "nasab" elementi "takson" tipidagi bir nechta elementlarga ega.

```
>>> soup.lineage.childrendagi takson uchun: if  
taxon.string != '\n': print(taxon.string)
```

Eukariota  
Metazoa  
Chordata  
Kraniata  
Umurtqali hayvonlar  
Euteleostomi  
Sutemizuvchilar  
Evteriya  
Euarchontoglires  
Yaltiroq  
Kemiruvchilar  
Sciurognathi  
Muroidea  
Muridae

Murinae

Mus

Mana bir xil ma'lumotlarni olish usuli (ketma-ketlik, nazorat summasi va uzunlik) biz cElementTree tahlilchisi bilan bir xil formatda olish mumkin:

```
>>> chop etish ('Tartib: {0}'.format(sho'rva.sequence.string))
Sequence:
MPKKKPTPIQLNPAPDGSAVNGTSSAETNLEALQKKLEELELDEQQRKRL
EAFLTQKQKVGEKLDDDFEKISELGAGNGGVFKVSHKPSGLVMARKLIH
LEIKPAIRNQIIRELQVLHECNSPYIVGFYGAFYSDGEISICMEHMDGGS
LDQVLKKAGRIPQEQLGKVSIAVIKGLTYLREKHKIMHRDVKPSNILVNS
RGEIKLCDFVGSQLIDSMANSFVGTRSYMSPERLQGTHYSVQSDIWSMG
LSLVEMAVGRYPIPPDAKELELLFGCHVEGDAETPPRPRTPGGPLSSY
GMDSRPPMAIFELLDYIVNEPPPQLPSGVFSLEFQDFVNKCLIKNPAERA
DLKQLMVHAFIKRSDAEEVD FAGWL CSTIGLNQPSTPTHAASI >>>
print('Tekshiruv summasi: {0}'.format(soup.sequence.get('checksum')))
Tekshirish summasi: E0C0CC2E1F189B8A
>>> print('Uzunlik: {0}'.format(soup.sequence.get('uzunlik')))
Uzunligi: 393
```

## 11.4 XULOSA

---

XML kengaytiriladigan belgilash tilini anglatadi va ma'lumotlarni saqlash va almashishning standart usulini yoqish uchun yaratilgan. XML ning afzalliklaridan biri shundaki, u turli xil dasturlash tillari tomonidan qo'llab-quvvatlanadi, ular orasida Python ham bor. XML hujjatlari prolog, korpus va epilogdan iborat. Prologda ushbu hujjatning versiyasi, kodlanishi va tuzilishi haqida ma'lumotlar mavjud. Tana ierarxik yoki ajratilgan elementlarga bo'lingan hujjatning barcha ma'lumotlarini o'z ichiga oladi. Har bir element o'z matni bilan tegdan iborat. Ixtiyoriy ravishda, element atributlarga ega bo'lishi mumkin. Bundan tashqari, "bo'sh elementlar" deb ataladigan umuman matnsiz elementlar mavjud.

Amaldagi dasturlash tilidan tashqari, ushbu turdag'i fayllarga kirishda ikkita asosiy strategiya qo'llaniladi. Bir tomonidan, u barcha elementlar o'rtaqidagi munosabatlarni tahlil qilishi va tegishli daraxtni qurishi mumkin. Bu butun fayl strukturasi xotirada bo'lishini nazarda tutadi va Hujjat Ob'ekt Modeli (DOM) deb ataladi. Boshqa variant esa faylni qayta tiklash va har bir alohida elementda takrorlanib sayohat qilishimiz mumkin bo'lgan voqealarni yaratishdir. Har bir tadbirda biz ma'lumotlarimizni qayta ishlashimiz mumkin. Ular "voqealarga asoslangan tahlilchilar" deb ataladi va eng mashhuri XML uchun Simple API (SAX).

Ushbu bobda biz voqealarga asoslangan tahlil qiluvchi misol sifatida taqdim etdik va cElementTree tomonidan taqdim etilgan Iterparse-dan foydalanishni ko'rdik. DOM-dan foydalanish ko'pincha osonroq, chunki u hodisalarni boshqarishni o'z ichiga olmaydi; ammo, ba'zi hollarda, ayniqsa, katta fayllar uchun voqealarga asoslangan tahlilchidan foydalanish qulayroqdir. Muq

## 252 Bioinformatika uchun Python

Buzilgan HTMLni tahlil qilish uchun ham ishlatalishi mumkin bo'lgan tashqi modul - BeautifulSoup. Bu modul boshqa tahlilchiga (odatda lxml) tayanadi, lekin ulardan foydalanish oson.

### 11.5 QO'SHIMCHA RESURSLAR

---

- Extensible Markup Language (XML). W3C tavsiyalariga havolalar, pro tavsiyalar va ishchi loyihalarni ishlab chiqdi.  
<http://www.w3.org/XML>
- Chiroqli sho'rva hujjatlari. <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
- Veb qirqish ustaxonasi. so'rovlар va Go'zal sho'rva foydalanish, eng bilan Yaqinda Go'zal sho'rva 4 doc.  
<https://gist.github.com/bradmontgomery/1872970>
- Mark Pilgrim. Pythona sho'ng'ish 3. **12-bob**; XML ishlov berish. <http://www.diveintopython3.net/xml.html>
- Python va XML: Kirish.  
[http://www.boddie.org.uk/python/XML\\_intro.html](http://www.boddie.org.uk/python/XML_intro.html)
- DTD bo'yicha manbalar.  
<http://www.w3schools.com/dtd/>, <http://www.xmlfiles.com/dtd>, va <http://www.w3.org/TR/REC-xml/#dt-doctype>.
- XML sxemasidagi manbalar:  
[https://www.w3schools.com/xml/schema\\_intro.asp](https://www.w3schools.com/xml/schema_intro.asp).

### 11.6 O'Z-O'ZINI BAHOLASH

---

1. OpenOffice formatining RSS tasmasi va Google Earth geografik koordinatalari bilan qanday umumiyligi bor?
2. Ma'lumotlarni saqlash va ma'lumotlar inter uchun XML foydalanish afzalliklari nima o'zgartirish?
3. Qachon XML dan foydalanmaysiz?
4. Nima uchun XML tahlilchisi noto'g'ri tuzilgan XML hujjatini o'qimasligi kerak?
5. Teg, element, atribut, qiymat, DTD va atamalarini farqlang Sxema.
6. Misol XML faylida ([Listing 11.1](#)) bitta bo'sh element tegi mavjud. Qaysi bittami?

7. XML fayl pro ning SAX va DOM modellari o'rtasidagi farq nima  
to'xtatish?
8. Mavjud operativ xotiraga yaqinlashib yoki kattaroq hajmga ega bo'lgan XML faylni tahlil  
qilishingiz kerak bo'lsa, qanday tahlil qilish tavsiya etiladi?
9. ceElementTree.iterparse da boshlanish va tugatish hodisasi turlari mavjud. Odatiy bo'lib, u  
faqat yakuniy hodisani qaytaradi. Boshlanish tadbirda ma'lumotdan qachon foydalanasiz?
10. XML BLAST chiqishidagi barcha hit nomlarini tahlil qilish uchun ikkita dastur tuzing. Bir  
dastur Python XML vositalaridan foydalanishi kerak, ikkinchisi esa kirish faylini matn  
sifatida o'qishi kerak.



**Taylor & Francis**  
Taylor & Francis Group  
<http://taylorandfrancis.com>

# Python va ma'lumotlar bazalari

---

## MAZMUNI

12.1 Ma'lumotlar bazalariga kirish .....	256	Ma'lumotlar bazasi nima? .....	256
Ma'lumotlar bazasi turlari .....	256		
12.1.1 Ma'lumotlar bazasini boshqarish: RDBMS .....	257	12.1.2 Relyatsion ma'lumotlar bazasi komponentlari .....	258
tushuncha: Birlamchi kalit .....	259	Asosiy	
Ma'lumotlar bazasi ma'lumotlar turlari .....	259	12.1.3	
12.2 Ma'yilumotlar bazasiga ulanish .....	261	12.3 MySQL	
ma'lumotlar bazasini yaratish .....	262	12.3.1 Jadvallar	
yaratish .....	263	12.3.2 Jadvalni	
yuklash .....	264	12.4 Oldinga	
rejalashtirish .....	266	12.4.1 PythonU:	
Namuna ma'yilumotlar bazasi .....	266	Baholar	
jadvali .....	266	Kurslar	
jadvali .....	268	12.5 TANLASH:	
Ma'yilumotlar bazasiga so'rov o'ytkazish .....	269	Oddiy	
so'rov .....	269	Ikki so'rovni	
birlashtirish .....	269	Bir nechta jadvallarni	
so'rash .....	270	12.5.1 So'rovni	
yaratish .....	271	12.5.2 Ma'lumotlar bazasini	
yangilash .....	273	12.5.3 Ma'lumotlar bazasidan	
yozuvni o'chirish .....	273	12.6 Python'dan ma'lumotlar bazasiga	
kirish .....	274	12.6.1 PyMySQL	
moduli .....	274	12.6.2 Ulanishni	
o'rnatish .....	274	12.6.3 Python'dan so'rovni	
bajarish .....	275	12.7	
SQLite .....	276	12.8 NoSQL	
ma'yilumotlar bazalari: MongoDB .....	278	12.8.1 MongoDB	
dan PyMongo bilan foydalanish .....	278	12.9 Qo'shimcha	
manbalar .....	282	12.10 O'z-o'zini	
baholash .....	284		

## 12.1 MA'LUMOT BAZALARIGA KIRISH

---

Oddiy bioinformatika loyihasida ko'rib chiqiladigan ma'lumotlar miqdori bizni Python bilan birlashtirilgan ma'lumotlar tuzilmalaridan ko'ra ko'p qirraliroq narsadan foydalanishga majbur qiladi. Ro'yxatlar, kortejlar va lug'atlar juda moslashuvchan, ammo ular haqiqiy dunyo ma'lumotlari bilan bog'liq barcha murakkabliklarni modellashtirish uchun mos emas. Ba'zan doimiy ma'lumotlar omboriga ega bo'lish kerak (kompyuter tilida bu ma'lumotlarning doimiyligi deb ataladi), chunki ma'lumotlar tuzilmalari faqat dastur ishlayotgan paytda mavjud bo'ladi. Pikle yordamida ma'lumotlarni faylga yozish mumkin bo'lsa-da, bu shu maqsadda ishlab chiqilgan ma'lumotlar bazasi mexanizmida

Ma'lumotlar bazasi nima?

Ma'lumotlar bazasi - tegishli ma'lumotlarning tartiblangan to'plami. Umuman olganda, ular real vaziyatlarni modellashtirish uchun tuzilgan: insonning video to'plami, universitet talabalari, firma inventarları va boshqalar. Ma'lumotlar bazasida dasturimiz foydalanuvchilari uchun tegishli ma'lumotlar saqlanadi. Universitet talabalarini modellashtirishda biz ism va familiyani, kirish yili va o'rganilgan fanlarni hisobga olishimiz kerak; Biz talabaning soch rangi yoki bo'yiga ahamiyat bermasdig. Ma'lumotlar bazasini loyihalash tabiiy jarayonni modellashtirishga o'xshaydi. Birinchi qadam tegishli o'zgaruvch

Ma'lumotlar bazalarining afzalligi shundaki, ular ma'lumotlarni saqlashdan tashqari, qidiruv vositalarini taqdim etadi. Ushbu qidiruvlarning ba'zilari darhol javob beradi, masalan, "qancha talaba bor?" Boshqalar esa, javob berish uchun turli ma'lumot manbalarini birlashtirishni o'z ichiga olgan murakkabroqdir, masalan, "2017 yil birinchi kurs talabasi o'rtacha qancha turli mavzularni olgan?" Biologik ma'lumotlar bazasida odatiy savol bo'lishi mumkin: "Og'irligi 134 kDa dan kam bo'lgan oqsillar qanday kristallangan?" Qizig'i shundaki, so'ralishi mumkin bo'lgan barcha savollarni oldindan bilishning hojati yo'q; lekin eng keng tarqalgan savollar haqida fikrga ega bo'lish dizayn jarayoniga yordam beradi.

Har holda, ma'lumotlar bazasiga ega bo'lishning afzalligi shundaki, biz qidiruv mexanizmini dasturlashtirmasdan turib, biz ushbu savollarni berishimiz va bu javoblarni olishimiz m. Bu katta hajmdagi ma'lumotlarni tezda qayta ishlash uchun optimallashtirilgan ma'lumotlar bazasi dvigatelining ishi. Python-dan foydalanib, biz ichki jarayonlar haqida qayg'urmasdan, ma'lumotlar bazasi mexanizmi bilan aloqa o'rnatamiz va uning javoblarini qayta ishlaymiz. Bu biz ma'lumotlar bazasi faoliyatidan butunlay voz kechishimiz kerak degani emas, chunki biz ichki ma'lumotlarni qanchalik ko'p tushunsak, shuncha y

Ma'lumotlar bazasi turlari

Hamma ma'lumotlar bazalari bir xil emas. Ma'lumotlar bazasi tuzilishini ham, ma'lumotlarning o'zaro bog'liqligini ham tavsiflash uchun turli xil nazariy modellar mavjud. Eng mashhur modellardan ba'zilari quyidagilardir: ierarxik, tarmoq, relational, ob'ekt munosabatlari va hujjatga asoslangan (yoki NoSQL). Turli xil modellar orasidan tanlash bioinformatika tadqiqotchilaridan ko'ra IT mutaxassislari uchun ko'proq ishdir. Ushbu bobda biz ko'p vaqtimizni relyatsion model bilan o'tkazamiz, chunki uning moslashuvchanligi

mavjud bo'lgan ko'plab ilovalar va (nima uchun emas?) uning mashhurligi. NoSQL ma'lumotlar bazalarining umumiy ko'rinishi ham mavjud.

Relyatsion ma'lumotlar bazasi - bu umumiy atributlardan foydalangan holda ma'lumotlarni guruhlaydigan ma'lumotlar bazasi. Natijada tashkil etilgan ma'lumotlar to'plamini mantiqiy tarzda qayta ishlash mumkin.

Masalan, shahardagi barcha ko'chmas mulk operatsiyalarini o'z ichiga olgan ma'lumotlar to'plamini bitim sodir bo'lgan yil bo'yicha guruhlash mumkin; yoki bitimning sotish narxi bo'yicha guruhlanishi mumkin; yoki uni xaridorning familyasi bo'yicha guruhlash mumkin; va hokazo.

Bunday guruhlash relyatsion modeldan foydalanadi. Shuning uchun bunday ma'lumotlar bazasi "relyatsion ma'lumotlar bazasi" deb ataladi. Relyatsion ma'lumotlar bazasini boshqarish uchun siz SQL (Tuzilgan so'rovlar tili) deb nomlangan maxsus kompyuter tilidan foydalanishingiz kerak. Bu dasturchiga ma'lumotlar bazasidan ma'lumotlarni yaratish, so'rash, yangilash va o'chirish imkonini beradi. SQL ANSI standarti bo'lisa-da, bir nechta mos kelmaydigan ilovalar mavjud. Ular har xil bo'lisa ham, barcha versiyalar bir xil nashr etilgan standartga asoslanganligi sababli, bilimlaringizni bir SQL lahjasidan boshqasiga o'tkazish qiyin emas.

Relyatsion ma'lumotlar bazalari va so'rovlar tillarining turli xil ilovalari orasida ushbu kitob ulardan ikkitasiga qaratilgan: MySQL va SQLite. MySQL (u "My Ess Cue Ell" deb talaffuz qilinadi) 10 milliondan ortiq o'rnatish bilan veb-ivalovalarda ishlatiladigan eng mashhur ma'lumotlar bazasi. Kichik va o'rta veb-saytlarning aksariyati MySQL-dan foydalanadi. Ko'pgina tizim ma'murlari MySQL-dan juda talabchan ilovalar uchun foydalanishni o'yamasalar-da, ko'plab yuqori trafikli saytlar undan to'liq foydalanmoqda. Bunga misol sifatida YouTube.com ni keltirish mumkin. MySQL-ga asoslangan boshqa mashhur saytlar - Vikipediya, Flickr, Facebook va Slashdot.org. 1 SQLite-ning maqsadi ancha torroq: u kichik o'rnatilgan tizimlar, ham apparat, ham dasturiy ta'minot uchun yaratilgan. Firefox brauzeri macOS va boshqalar kabi SQLite'dan ichki foydalanadi.

Ushbu ko'p qirralilik uning kichik hajmi (taxminan 250 KB), tashqi bog'liqliklarning yo'qligi va ma'lumotlar bazasini bitta faylda saqlashi bilan bog'liq. Kichik o'lchamdag'i va soddalikning bu afzalliklari xususiyatlarning etishmasligi bilan qoplanadi, ammo o'ziga xos joy uchun bu muammo emas.

Ikkala holatda ham asoslar o'xshash va ushbu bobda tushuntirilgan tushunchalar barcha relyatsion ma'lumotlar bazalariga tegishli. Xususiyat ma'lumotlar bazasiga xos bo'lisa, bu ko'rsatiladi.

### 12.1.1 Ma'lumotlar bazasini boshqarish: RDBMS

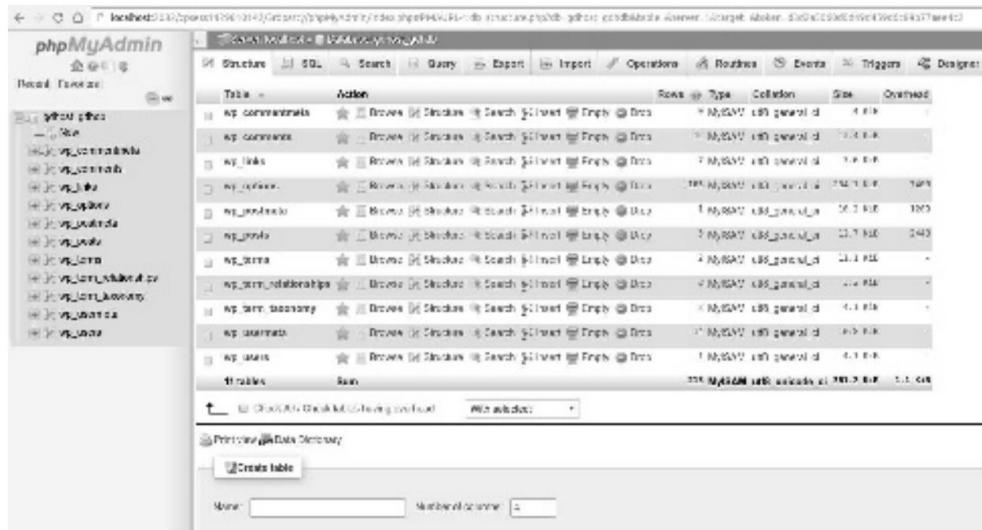
RDBMS Relational Database Management System degan ma'noni anglatadi. Bu ma'lumotlar bazasi mexanizmi, foydalanuvchi va ilovalar o'rtasida interfeys vazifasini bajarish uchun mo'ljallangan dasturiy ta'minot. Yuqorida aytib o'tilgan MySQL va SQLite RDBMS.2 ga misoldir

MySQL holatida RDBMS ikki komponentga bo'linadi: server va mijoz. Server - bu ma'lumotlar bazasi mexanizmi bilan bog'liq qiyin ishlarni bajaradigan dastur; u bizning shaxsiy kompyuterimizda yoki masofadan turib kirish mumkin bo'lgan serverda ishlashi mumkin. Mijoz - bu bizga serverga interfeys beradigan dastur.

<sup>1</sup>To'g'ri, ular tovar uskunasida ishlaydigan standart o'rnatishlar emas, balki markali uskunada ishlaydigan yuqori darajada optimallashtirilgan o'rnatishlar.

<sup>2</sup>Boshqa taniqli RDBMSlar Oracle, DB2 va PostgreSQLdir.

MySQL o'z mijozini ([mysql](#)) taqdim etadi, bu buyruq qatori dasturidir, ammo ba'zi alternativalar mavjud. Ommabop mijoz PhpMyAdmin 3 bo'lib, u veb-serverni ishga tushirishni talab qiladi, lekin oxirgi foydalanuvchiga MySQL serverining veb-ga asoslangan yaxshi interfeysi taqdim etadi ([12.1-rasmga qarang](#)). MySQL Workbench<sup>4</sup> (Oracle'dan bepul va ko'p platformali rasmiy versiyasi), SQLyog<sup>5</sup>, va Navicat<sup>6</sup> kabi bir xil funktsiyaga ega ish stoli mijozlari ham mavjud .



Shakl 12.1 PhpMyAdmin skrinshoti: MySQL ma'lumotlar bazasini boshqarish uchun foydalanish uchun qulay HTML interfeysi.

Boshqa tomonidan, SQLite sizning dasturlaringizga kiritish uchun kutubxona sifatida yoki mustaqil bajariladigan fayl sifatida mavjud. Python SQLite bilan interfeysga kirish uchun o'rnatilgan modulga (`sqlite3`) ega va agar Python SQLite bilan tuzilgan bo'lsa (eng ehtimol variant) "qutidan tashqarida" ishlaydi. U `pysqlite2.dbapi2` moduli bilan tashqi bajariladigan faylga ham bog'lanishi mumkin.

### 12.1.2 Relyatsion ma'lumotlar bazasi komponentlari

Biz tushunishimiz kerak bo'lgan birinchi ma'lumotlar bazalari tushunchasi - bu ob'ektlar. Rasmiy ravishda ob'ekt saqlanishi kerak bo'lgan har bir muhim element sifatida belgilanadi. Biz ob'ekt turi va ob'ektning paydo bo'lishini farqlashimiz kerak. Ma'muriyat ma'lumotlar bazasida Talabalar ob'ekt turi bo'lib, har bir talaba bu ob'ektning bir hodisasiadir.

Har bir ob'ektning o'ziga xos xususiyatlari bor. Atributlar ob'ekt bilan bog'langan ma'lumotlardir. Keling, kollej ma'muriyatining ma'lumotlar bazasiga qaytaylik

<sup>3</sup><http://www.phpmyadmin.net>

<sup>4</sup><https://dev.mysql.com/downloads/workbench/>

<sup>5</sup><http://webyog.com/en/>

<sup>6</sup><http://www.navicat.com>

**sxema bo'yicha.** ism, familiya, qo'shilgan sana va OutstandingBalance talabalar tashkilotining atributlaridir .

O'z navbatida, har bir ob'ekt o'ziga xos xususiyatlarga ega. Atributlar ob'ekt bilan bog'langan ma'lumotlardir. Kollej ma'muriyati ma'lumotlar bazasini yarataylik, unda Ism, Familiya, Qo'shilgan sana va Balans Students tashkilotining atributlari hisoblanadi.

Relyatsion ma'lumotlar bazasidagi ma'lumotlar alohida emas, balki nomidan ko'rinish turibdiki, ular munosabatlar bilan ifodalanadi. Munosabat kalitni yoki tugmalar guruhini qatorlar guruhi bilan taqqoslaydi. Har bir kalit ushbu hodisa bilan bog'liq atributlar guruhiga tegishli bo'lgan bir ob'ektning hodisasiga mos keladi. Bu munosabatlar jismoniy jihatdan qanday saqlanishidan qat'i nazar, jadval sifatida ko'rsatiladi. Ma'lumotlar bazasida bir nechta jadvallar bo'lishi mumkin. Universitet ma'muriyati ma'lumotlar bazasi misolini davom ettiradigan bo'lsak, bizda talabalar to'g'risidagi ma'lumotlar va boshqa professorlar to'g'risidagi ma'lumotlar jadvaliga ega bo'lishimiz mumkin, chu-

**12.1-jadvalda** biz o'quvchilar munosabatining misolini ko'rishimiz mumkin.

#### 12.1-JADVAL Python universiteti talabalari

Ism	Familiya Qo'shilgan sanasi
Garri	Wilkinson, 2006-02-10 ý
Jonatan Xant	2004-02-16 ý
Garri	Xyuz 2005-03-20 ý
Kayla	Allen 2001-03-15 Ha
Virjiniya Gonsales, 2003-04-02 ý	

#### Asosiy tushuncha: birlamchi kalit

Har bir jadval ma'lumotlar qatorini aniqlash vositasiga ega bo'lishi kerak; u noyob identifikator vazifasini bajaradigan atribut yoki atributlar guruhiga ega bo'lishi kerak. Bu atribut asosiy kalit deb ataladi. Agar bitta atribut asosiy kalit sifatida ishlatalmasa, kompozit kalit yaratish uchun bir vaqtning o'zida bir nechtasini olish mumkin. 12.1-jadvalga qaysak , Name atributidan birlamchi kalit sifatida foydalanish mumkin emasligini ko'rishimiz mumkin, chunki bir xil atributga ega bo'lgan bir nechta ob'ekt bor (Jo Kempbell va Jo Doe bir xil nomga ega). Ushbu muammoning yechimlaridan biri ism va familiyani kompozit kalit sifatida ishlatishdir; lekin bu eng yaxshi yechim bo'lmaydi, chunki boshqa Joe Doe kabi ushbu alohida kompozit kalitni baham ko'radigan ob'ektning bir nechta holatlariga ega bo'lish hali ham mumkin. Shu sababli, odatda biz jadvalga asosiy kalitga ega bo'lish uchun ma'lumotlarga bog'liq bo'lish or'niga ID maydonini - noyob identifikatorni qo'shamiz. Ko'pgina ma'lumotlar bazalarida ma'lumotlarni kiritishda bunday asosiy kalitni avtomatik ravishda yaratish mexanizmlari mavjud. Keling, **12.1-jadvalning** asosiy kalit sifatida ishlatalishi mumkin bo'lgan yangi atributli versiyasini ko'rib chiqaylik:

**12.2-JADVAL Birlamchi kalitli jadval**

ID nomi	Familiya Qo'shilgan sanasi
1 Garri	Wilkinson, 2006-02-10 ý
2 Jonatan Xant	2004-02-16 ý
3 Garri	Xyuz
4 Kayla	Allen
5 Virjiniya Gonsales	2001-03-15 Ha
	2003-04-02 ý

**12.1.3 Ma'lumotlar bazasi ma'lumotlar turlari**

Dasturlash tillarida bo'lgani kabi, ma'lumotlar bazalarining ham o'ziga xos ma'lumotlar turlari mavjud. Masalan, Pythonda bizda int, float va string mavjud (**boshqalar qatorida**); ma'lumotlar bazalari tinyint, smallint, mediumint, int, bigint, float, char, varchar, text va boshqalar kabi o'ziga xos ma'lumotlar turlariga ega . Nega shunchalik ko'p ma'lumotlar turlari (masalan, butun sonlar uchun besh xil ma'lumotlar turi) borligiga hayron bo'lishingiz mumkin. Asosiy sabab shundaki, juda ko'p variantlar bilan mavjud resurslardan maksimal darajada foydalanish mumkin. Agar bizga o'quvchilarning yoshini saqlamoqchi bo'lgan maydon kerak bo'lsa, biz bunga tinyint **tipidagi** maydon yordamida erishishimiz mumkin , chunki u -128 va 127 (bir baytda saqlanishi mumkin) orasidagi qiymatlar oralig'ini qo'llab-quvvatlaydi. Albatta, biz uni ý2147483648 dan 2147483647 gacha (ya'ni 4 bayt) oralig'ini qo'llab-quvvatlaydigan int tipidagi maydonda ham saqlashimiz mumkin; lekin bu xotirani behuda sarflash bo'ladi, chunki tizim keraksiz joyni zaxiralashi kerak. Baytlar sonidagi farq tufayli int sifatida saqlangan raqam tinyint sifatida saqlanganidan 4 barobar ko'p operativ xotira va disk maydonini egallaydi. Bir va to'rt bayt o'rtasidagi farq ahamiyatsiz bo'lib tuyulishi mumkin va eslatib o'tishga arzimaydi, lekin keyin uni sizda mavjud bo'lgan ma'lumotlar yozuvlari soniga ko'paytiring; ma'lumotlar to'plami etarlicha katta bo'lsa, disk maydoni va kirish vaqtini muammo bo'lishi mumkin. S

**12.3-jadvalda** MySQL-dagi asosiy ma'lumotlar turlarining xarakteristikalari jamlangan. E'tibor bering, ba'zi bir kichik xususiyatlar ishlataladigan MySQL versiyasiga qarab farq qilishi mumkin, shuning uchun o'zingizning maxsus versiyangiz uchun hujjatlar bilan tanishib chiqish tavsiya etiladi.<sup>8</sup> SQLite holatida faqat 5 turdag'i ma'lumotlar mavjud: INTEGER, REAL, TEXT, BLOB va NULL. Biroq, SQLite toifasiz ekanligini va har qanday ma'lumotni istalgan ustunga kiritish mumkinligini tushunish kerak. Shu sababli, SQLite "turga yaqinlik" g'oyasiga ega: u ma'lumotlar turlarini ta

7Vaziyat uchun qanday ma'lumotlar turlari mos kelishini taxmin qilish kichik muammo emas. World of Warcraft onlayn ko'p o'yinchi o'yinida ba'zi o'yinchilar pul saqlanadigan o'zgaruvchi chegarasiga, imzolangan 32 bitli tamsayiga yetganlarida, ko'proq oltin ololmasligini aniqladilar. Ariane 5 raketasidagi dasturiy ta'minot 64 bitli real 16 bitli imzolangan butun songa aylantirilganda ancha jiddiyroq edi. Bu esa 370 million dollarga tushadigan butun parvozni yo'q qilish bilan yakunlangan muammolar kaskadiga olib keldi.

8MySQL to'liq onlayn ma'lumotnomaga ega. MySQL 5.7 uchun ma'lumotlar turi hujjatlari <https://dev.mysql.com/doc/refman/5.7/en/data-types.html> saytida mavjud .

9"Tipga yaqinlik" g'oyasi haqida qo'shimcha ma'lumot olish uchun men "SQLite 3-versiyasidagi ma'lumotlar turlari" bo'limini tavsiya qilaman (<http://www.sqlite.org/datatype3.html>) SQLite on

## 12.3-JADVAL Eng ko'p ishlataladigan MySQL ma'lumotlar turlari

Ma'lumotlar turi	Izoh
TINYINT	$\pm 127$ (0-255 UNSIG.)
KICHIK	$\pm 32767$ (0-65535 UNSIG.)
ORTA	$\pm 8388607$ (0-16777215 UNSIG.) $\pm 2147483647$
INT	(0-4294967295 UNSIG.)
BIGINT	$\pm 9223372036854775807$ (0-18446744073709551615 UNSIG.)
FLOAT	Suzuvchi kasrli kichik raqam.
DOUBLE	Suzuvchi kasrli katta raqam.
DATETIME	'1000-01-01 00:00:00' dan '9999-12-31 23:59:59' gacha DATE
'1000-01-01' dan '9999-12-31' gacha CHAR(n)	n ta belgidan iborat (255 tagacha) sobit boylim.
VARCHAR(n)	n ta belgidan iborat boylgan oyzgaruvchilar boylimi (255 tagacha).
TEXT	Maksimal uzunligi 65535 belgidan iborat boylgan satr.
BLOB	TEXT ning ikkilik qatorli versiyasi.
MEDIUMTEXT	Maksimal uzunligi 16777215 belgidan iborat qator.
MEDIUMBLOB	Ikkilik qator MEDIUMTEXT ga ekvivalent.
LONGTEXT	Maksimal uzunligi 4294967295 belgidan iborat qator.
LONGBLOB	Ikkilik qator LONGTEXT ga ekvivalent.
ENUM	String qiymati ruxsat etilgan qiymatlar royyxatidan olingan.

## 12.2 MA'LUMOTLAR BAZAGA ULANISH

MySQL ma'lumotlar bazasi serveriga ulanish uchun sizga haqiqiy foydalanuvchi kerak, foydalanuvchini sozlash uchun esa ma'lumotlar bazasiga ulanish kerak. Ushbu catch-22 standart hisob ma'lumotlari (foydalanuvchi: "root" va parolsiz) bilan serverga kirish orqali hal qilinadi. Buyruqlar qatoridan, agar server bitta kompyuterda bo'lsa, ushbu buyruq bilan kirish mumkin:

```
$ mysql -u root -p
Parolni kriting: MySQL
monitoriga xush kelibsiz. Buyruqlar bilan tugaydi; yoki \g.
MySQL ulanish identifikatoringiz: 234787469
Server versiyasi: 5.5.51-38.2 Percona Server (GPL), 38.2 reliz
```

Mualliflik huquqi (c) 2000, 2016, Oracle. Barcha huquqlar himoyalangan.

Oracle Oracle korporatsiyasi va/yoki uning filiallarining royyxatdan oytgan savdo belgisidir. Boshqa nomlar ularning tegishli savdo belgilari bo'lishi mumkin egalari.

'yordam;' yoki yordam uchun '\h'. Joriy inp=<ni tozalash uchun '\c' kriting ut bayonoti.

mysql>

Bundan buyon MySQL serveri bilan o'zaro aloqa php MyAdmin front-end yordamida ko'rsatiladi.

### 12.3 MYSQL MA'LUMOTLAR BAZASINI YARATISH

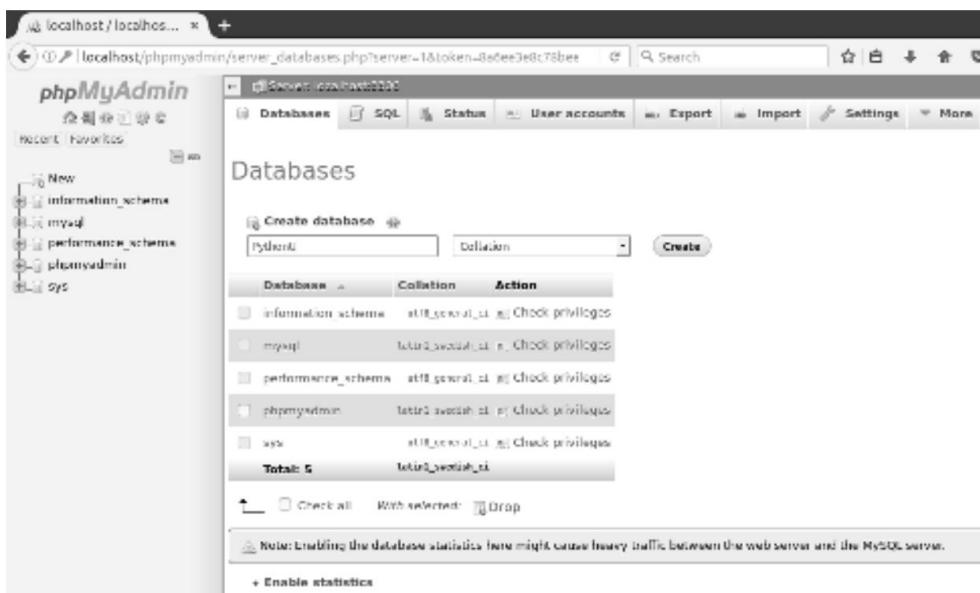
Ma'lumotlar bazasi bilan ishlashdan oldin biz uni yaratishimiz kerak. Agar siz ilgari yaratilgan ma'lumotlar bazasiga kirishni rejalashtirmoqchi bo'lsangiz, bu bosqichni o'tkazib yuborishingiz mumkin. Ammo ertami-kechmi o'z ma'lumotlar bazasini yaratishingiz kerak bo'ladi. Ma'lumotlar bazasini yaratish oddiy vazifa bo'lib, siz ishlov bermoqchi bo'lgan ma'lumotlarni tushunishga va yanada samarali so'rovlarni yaratishga yordam beradi.

Ma'lumotlar bazasini yaratish har bir ma'lumotlar bazasi uchun bir marta bajariladigan narsa bo'lgani uchun, bu vazifani dastur bilan avtomatlashtirishga ko'p ehtiyoj yo'q. Ushbu qadam odatda qo'lda amalga oshiriladi. Mening tavsiyam ma'lumotlar bazasini loyihalash uchun grafik vositadan foydalanishdir. phpMyAdmin yoki Navicat ishni bajaradi.

MySQL konsolidan ma'lumotlar bazasini yaratish uchun:

mysql> PythonU ma'lumotlar bazasini  
yaratish; So'rov OK, 1 qatorga ta'sir qildi (0,01 sek)

Bu PythonU MySQL ma'lumotlar bazasini yaratadi. PhpMyAdmin-dan ma'lumotlar bazasini yaratish uchun chap paneldag'i "Yangi" tugmasini bosing va "Yangi ma'lumotlar bazasini yaratish" bo'limidagi ma'lumotlar bazasining taklif qilingan nomi bilan ariza maydonini to'ldiring ( 12.2-rasmga qarang).



12.2-rasm phpMyAdmin yordamida yangi ma'lumotlar bazasini yaratish.

### 12.3.1 Jadvallarni yaratish

Yangi yaratilgan ma'lumotlar bazasiga ega bo'lgach, keyingi qadam ma'lumotlar saqlanadigan jadvallarni yaratishdir. Ushbu turdag'i dasturiy ta'minotdan foydalangan holda jadvallarni yaratish ushbu kitobda eslatib o'tadigan muammo emas, shuning uchun biz GUI vositasi bilan ishslash tartibiga emas, balki jadval tuzilishiga ko'proq e'tibor qaratamiz.

Shuni yodda tutishimiz kerakki, jadval ma'lumotlar orasidagi munosabatni ifodalaydi; bir ob'ekt uchun jadval yaratish va keyin uni boshqa ob'ektning ma'lumotlari bilan to'ldirish mantiqiy emas. "Python universiteti" misolida davom etar ekanmiz, talabalar jadvalida talabalarga tegishli qanday ma'lumotlarni saqlashimiz kerakligi haqida o'ylashimiz mumkin.

**Yuqorida ko'rganimizdek, Talabalar jadvalida biz quyidagi maydonlarni tayinladik: ID, Ism, Familiya, Qo'shilgan sana va OutstandingBalance.**

Ma'lumotlar bazasini loyihalash uchun "yaxshi amaliyotlar" mavjud. Bu makondagi har bir vaziyat uchun samarali dizaynga erishish uchun zarur bilimlarni etkazish, albatta, oson emas; Har holda, yaxshi ma'lumotlar bazasi dizayni - bu amaliyot bilan o'rganadigan narsa.

Keling, bu holda har bir maydonni qanday aniqlashimizni ko'rib chiqaylik:

**ID:** Har bir ro'yixatdan oytgan shaxs uchun noyob identifikator. Python universitetida bir nechta talaba bo'lishi kutilganligi sababli, imzosiz INT ma'lumotlar turi ishlataladi (4294967295 gacha). IDda manfiy raqamlardan foydalanish shart emas, shuning uchun bu maydon imzosiz sifatida belgilanishi kerak.

**Nom:** Ismning o'chami 255 ta belgidan kam bo'lgan o'zgaruvchan bo'lgani uchun VAR CHAR ishlataladi. Belgillardagi nomlarning maksimal o'chami, mening oddiy mezonlarimga ko'ra, 150 ni tashkil qiladi.

**Familiya:** Bu maydon avvalgi maydon bilan bir xil mezonlar bilan o'rnatilgan. Faqat farq familiya uchun maksimal o'chamda; bu 200 belgiga o'rnatiladi.

**Qo'shilgan sana:** Bu erda tanlov juda ko'p emas. Oddiy DATE maydoni buni eng yaxshi qiladi.

**OutstandingBalance:** Bu maydon talaba o'qish uchun to'lovnini to'liq to'lagan yoki to'lamaganligini ko'rsatadi. Faqat ikkita mumkin bo'lgan qiymat (to'langan yoki to'lanmagan) bo'lgani uchun BOOLEAN ma'lumotlar turi tanlanadi. Ushbu ma'lumotlar turi 0 yoki 1 ni saqlaydi. Ushbu qiymatga ma'no berish dasturchiga bog'liq, ammo matematik belgilarda 0 FALSE va 1 TRUE ma'nosini bildiradi, shuning uchun odatda bu konvensiya qo'llaniladi.

Oxirgi tanlov jadval turi (InnoDB yoki MyISAM). Bunday holda, ko'pchilik foydalanish uchun mos bo'lgan standart variantni (MyISAM) qoldirish OK.

Iltimos, ikkala jadval turi bo'yicha qisqacha muhokama qilish uchun 265-betdag'i Murakkab maslahat: MyISAM va InnoDB ga qarang.

Agar siz jadvalni qo'lda yaratmoqchi bo'lsangiz, avval siz ma'lumotlar bazasini tanlashingiz kerak

foydalanish:

mysql> PythonU dan foydalaning;

Ma'lumotlar bazasi o'zgartirildi

Va ma'lumotlar bazasi tanlangandan so'ng, ushbu buyruqlarni MySQL-ga kiriting prompt (shuningdek, GitHub kitob omborida db/studentstbl.sql sifatida mavjud):



12.3-rasm phpMyAdmin yordamida yangi jadval yaratish.

#### "Talabalar" JADVAL TUZISH (

```
'ID' INT UNSIGNED NO NULL AUTO_INCREMENT,
'Ism' VARCHAR(150) NULL EMAS,
'Familiya' VARCHAR(200) NULL EMAS,
"Qo'shilgan sana" sanasi NULL EMAS,
'OtstandingBalance' BULEAN NULL EMAS,
PRIMARY KEY ('ID') ) ENGINE = MyISAM;
```

Jadvalni loyihalash uchun GUI-dan foydalanishni tavsiya qilganim ajablanarli emas!

Maslahat: Shablon sifatida boshqa ma'lumotlar bazasidan foydalanib ma'lumotlar bazasini yaratish.

Har bir jadvaldagi har bir maydonni qo'lda belgilash o'rninga, boshqa ma'lumotlar bazasidan "MySQL dump" ni import qilishingiz va bir qadamda ma'lumotlar bazasini yaratishingiz mumkin. Ikki xil turdag'i dump fayllar mavjud: "Faqat tuzilma" va "tuzilma va ma'lumotlar" dump fayllari. Ikkala fayl ham xuddi shu tarzda ma'lumotlar bazasiga import qilinadi:

```
$ mysql -p database_name < dbname.sql
```

Dump faylini qayerdan olasiz? Siz dump faylini boshqa ma'lumotlar bazasining zaxira nusxasidan yoki ma'lumotlar bazasini talab qiladigan dasturning o'rnatish fayllaridan olishingiz mumkin.

#### 12.3.2 Jadvalni yuklash

Jadvalni yaratganimizdan so'ng, unga ma'lumotlarni yuklash vaqtি keldi. Ushbu operatsiyani har qanday MySQL front-enddan, qatorga yoki to'plamda bajarish mumkin. Boshida yuklash uchun bir nechta ma'lumotlar borligi va ma'lumotlarni qo'lda yuklash intuitiv bo'lgani uchun, keling, ommaviy rejimda ma'lumotlarni qanday yuklashni ko'rib chiqamiz.

Ma'lumotlarni yuklashning eng keng tarqalgan usuli csv fayllaridan foydalanishdir. Bunday fayl edi

5.3-bo'limda ko'rib chiqilgan (90-bet). [12.2-jadvalda](#) ko'rsatilgan ma'lumotlarni yuklash uchun biz quyidagi formatdagi csv faylini (dbdata.csv) tayyorlashimiz mumkin:

- 1, Garri, Uilkinson, 2016-02-10
- 2, Jonathan, Hunt, 2014-02-16
- 3, Garri, Xyuz, 2015-03-20
- 4, Kayla, Allen, 2016-03-15, 1
- 5, Virjiniya, Gonsales, 2003-04-02

Csv faylini MySQL ma'lumotlar bazasiga yuklash uchun MySQL so'rovida LOAD DATA INFILE buyrug'i 10 dan foydalaning:

```
mysql> MA'LUMOTLAR LOCAL INFILE 'dbdata.csv' JADVALGA YUKLASH Talabalar
      ',' BILAN TUG'ILGAN MAYDLAR;
```

Agar siz buni o'zingiz qilmaslikni istasangiz, <https://sqlizer.io> saytida CSV faylini MySQL jadvaliga aylantiradigan veb-xizmat ham mavjud . 5000 tagacha majlumotlar qatori va shaxsiy foydalanish uchun faylni SQL ga aylantirish bepul. Mening maslahatim - buni o'zingiz qilishga harakat qiling, bu unchalik qiyin emas.

**INSERT** iboralari yordamida muqobil usul:

```
INSERT INTO 'Talabalar' ('ID', 'Ism', 'Familiya', 'Sana qo'shilgan',
'Ajoyib Balans') QIYMATLAR
(1, 'Garri', 'Uilkinson', '2016-02-10', 0),
(2, 'Jonatan', 'Hunt', '2014-02-16', 0),
(3, 'Garri', 'Hyuz', '2015-03-20', 0),
(4, 'Kayla', 'Allen', '15-03-2016', 1),
(5, 'Virjiniya', 'Gonsales', '2017-04-02', 0);
```

Ma'lumotlar bazaga yuklangandan so'ng, jadval [12.4-rasmdagi kabi ko'rindi](#).

Kengaytirilgan maslahat: MyISAM va InnoDB

MySQL jadvallarida ichki ma'lumotlar tuzilmalari uchun bir nechta formatlar mavjud. Eng ko'p ishlataladigan formatlar InnoDB va MyISAM. MyISAM sukut bo'yicha ishlataladi va yuqori o'qish tezligi (SELECT operatsiyalari yordamida) bilan tavsiflanadi va u kamroq disk maydonini ishlataadi. Yozishda u InnoDB ga qaraganda sekinroq, chunki ma'lumotlar yozib olinayotganda jadval tugaguniga qadar bir lahzalik bloklanadi, shuning uchun boshqa barcha operatsiyalar tugashini kutishi kerak. Bu cheklov InnoDB jadvalida mavjud emas. Ushbu formatning asosiy afzalligi shundaki, u xavfsiz operatsiyalarni amalg

<sup>10</sup>Ushbu buyruq haqida to'qliq ma'lumot olish uchun [https://dev.manzilidagi MySQL onlayn](https://dev.manzilidagi.MySQL.onlayn) qo'llanmasiga qarang . [mysql.com/doc/refman/5.7/en/load-data.html](http://mysql.com/doc/refman/5.7/en/load-data.html).

ID	Name	LastName	DateJoined	OutstandingBalance
1	Harry	Wilkinson	2016-02-10	0
2	Jonathan	Hunt	2014-02-16	0
3	Harry	Hughes	2015-03-20	0
4	Kayla	Allen	2016-03-15	1
5	Virginia	Gonzalez	2017-04-02	0

## 12.4-rasm Talabalar jadvalining ko'rinishi.

halokatni tiklash. Shunday qilib, InnoDB intensiv yangilangan jadvallar va nozik ma'lumotlarni saqlash uchun tavsiya etiladi. Xulosa qilib aytadigan bo'lsak, siz bir xil ma'lumotlar bazasida turli xil jadval turlaridan foydalanishingiz mumkinligi sababli, har bir jadvalda eng ko'p bajariladigan operatsiyaga muvofiq jadval turini tanlang.

---

## 12.4 REJAJLASH

Ma'lumotlar bazasini yaratish biroz rejalashtirishni talab qiladi. Bu, ayniqsa, katta hajmdagi ma'lumotlar mavjud bo'lganda to'g'ri keladi va biz so'rovlarimizga javob berish uchun ketadigan vaqtни optimallashtirishni xohlaymiz. Yomon dizayn ma'lumotlar bazasini yaroqsiz holga keltirishi mumkin. Ushbu bo'limda men ma'lumotlar bazasi dizaynining asoslarini ko'rsatish uchun namunaviy ma'lumotlar bazasini taqdim etaman. Relyatsion ma'lumotlar bazalarini loyihalash haqida yaxshiroq tasavvurga ega bo'lish uchun siz "Qo'shimcha resurslar" bo'limida ma'lumotlar bazasini normallashtirish haqida o'qishingiz mumkin , ammo bu bo'lim sizga ushbu mavzu haqida qisqacha ma'lumot berishi kerak.

### 12.4.1 PythonU: Ma'lumotlar bazasi namunasi

Talabalar ma'lumotlari va olingan mavzularni saqlash uchun xayoliy Python universiteti (ma'lumotlar bazasi PythonU deb ataladi) talabalar ma'lumotlar bazasi misolini saqlaymiz. Talabalar ma'lumotlarini saqlash uchun bizda talabalar jadvali mavjud. Biz har bir fan bo'yicha baholarni saqlash uchun jadval tuzishimiz kerak ("Baholar" jadvali). Dasturlashning boshqa ko'plab jihatlarida bo'lgani kabi, buni amalga oshirishning bir nechta usullari mavjud. Nima uchun tavsiya etilgan usul mavjudligini yaxshiroq tushunish uchun biz ba'zi optimal bo'limagan usullarni ko'rsatishdan boshlaymiz.

### Baholar jadvali

Jadvalda biz har bir talaba uchun Baholarni saqlamoqchimiz: o'rganilgan fanlar, baho va har bir kurs o'tilgan sana.

Ikki kurs uchun tavsiya etilgan Baholar jadvali 12.5-rasmida tasvirlangan .

Ushbu dizayn (sxema) bir nechta kamchiliklarga ega. Jadvaldag'i birinchi dizayn xatosi uning

StudentID	Python_101	Python_101-TERM	Mathematics_for_CS	Mathematics_for_CS-TERM
1	7	2016/1		8 2016/1
2	6	2015/1		9 2015/2
3	5	2016/1		7 2016/1
4	9	2016/1		8 2016/2

12.5-rasm Qasddan noto'g'ri "Baholar" jadvali.

moslashuvchanlik. Agar biz yangi kurs qo'shmoqchi bo'lsak, jadvalni o'zgartirishimiz kerak. Bu yaxshi dasturlash amaliyoti deb hisoblanmaydi; allaqachon to'ldirilgan jadvalning tuzilishini o'zgartirish qimmat operatsiya bo'lib, iloji bori Ushbu dizayndan kelib chiqadigan boshqa muammo, diagrammada ko'rinish turibdiki, kursni bir necha marta olgan talabaning bahosini saqlash uchun joy yo'q. Buni qanday hal qilamiz? Yana aqlii dizayn bilan. Deyarli optimal yechimni 12.6-rasmda ko'rish mumkin .

StudentID	Course	Grade	Term
1	Python 101	7	2016/1
1	Mathematics for CS	8	2016/1
2	Python 101	6	2015/1
2	Mathematics for CS	9	2015/2
3	Python 101	5	2016/1
3	Mathematics for CS	7	2016/2
4	Python 101	9	2016/1
4	Mathematics for CS	8	2016/2

12.6-rasm Yaxshiroq "Baholar" jadvali.

Birinchi muammo, yangi fanlarni kiritish uchun jadvalni qayta loyihalash zarurati, biz kurs nomini yangi maydon sifatida kiritish orqali hal qilamiz: Kurs. Bu maydon TEXT yoki VARCHAR turida bo'lishi mumkin. Talaba kursni bir necha marta o'qiganini kuzatib borish muammosini biz Term maydoni bilan hal qildik. Bu avvalgisidan qat'iy yaxshiroq dizayn bo'lsa-da, u optimal emas. Ko'rinish turibdiki, har bir talaba uchun har bir fan nomini saqlash keraksiz resurslarni isrof qilishdir. Ushbu bo'shlinqni saqlash usuli - Kurs maydonida ENUM ma'lumotlar turidan foydalanish; Shunday qilib, biz katta hajmdagi jojni tejashimiz mumkin, chunki MySQL ushbu turdag'i har bir kirish uchun bir yoki ikki baytdan foydalanadi. Jadval avvalgidek ([12.6-rasm](#)) saqlanib qoladi va faqat Kurs maydonini belgilash usulini o'zgartiradi va aytib o'tilganidek, disk maydonini tejaydi.

Bu eng yaxshi yo'lmi? Kurs maydoni bilan ENUM dan foydalanish muammosi shundaki, biz yangi mavzu qo'shmoqchi bo'lsak, jadval tuzilishini o'zgartirishimiz kerak. ENUMga yangi variantni qo'shish uchun ushbu modifikatsiya yangi ustun qo'shish kabi qimmatga tushmaydi, lekin kontseptual jihatdan ta'rifni o'zgartirish yaxshi fi

## 268 Bioinformatika uchun Python

yangi turdag'i ma'lumotlarni joylashtirish uchun yangi jadval. Bunday hollarda biz "qidiruv jadvallari" ga murojaat qilamiz.

## Kurslar jadvali

CourseID	Course_Name
1	Python 101
2	Mathematics for CS

12.7-rasm Kurslar jadvali: Qidiruv jadvali.

Qidiruv jadvali boshqa jadvalda joylashgan ustunning mazmuni sifatida foydalaniladigan qiymatlarni saqlash uchun foydalaniladigan mos yozuvlar jadvalidir. Python universiteti misolida davom etsak, biz mavzular uchun qidiruv jadvalini tuzishimiz mumkin ([12.7-rasmga qarang](#)).

Ushbu Kurslar jadvali kurs identifikatorini (CourseID) va boshqa kurs nomini (Kurs\_nomi) saqlash uchun maydonni o'z ichiga oladi. Ushbu sxema ishlashi uchun biz maydonni o'zgartirishimiz kerak Kurs jadvali Baholar; ENUM maydoni o'rniiga biz endi INT maydonidan foydalanamiz ([12.8-rasmga qarang](#)).

StudentID	Course	Grade	Term
1	1	7	2016/1
1	2	8	2016/1
2	1	6	2015/1
2	2	9	2015/2
3	1	5	2016/1
3	2	7	2016/1
4	1	9	2016/1
4	2	8	2016/2

Shakl 12.8 O'zgartirilgan "Baholar" jadvali.

CourseID ma'lumotlari endi Baholar jadvalidagi Kurs maydoniga mos keladi. Bitta qidiruvdan foydalanib, biz identifikatorni tegishli kurs nomi bilan bog'lashimiz mumkin. Shu tariqa biz Kurs maydoni uchun ENUM dan foydalanganda bo'lganidek, Talabalar jadvalida bir xil bo'sh joyni tejayimiz, buning qo'shimcha afzalligi shundaki, biz Kurslar jadvaliga bitta element qo'shish orqali fanlar ro'yxatini kengaytirishimiz mumkin.

---

Maslahat: Izlash jadvaliga nisbatan ENUM maydon turi

ENUM o'rniiga qidirish jadvalidan foydalanish qanchalik qulay ekanligini ko'rdik