КОММЕРЦИЯЛЫҚ ЕМЕС АКЦИОНЕРЛІК ҚОҒАМЫ
«ӘБІЛҚАС САҒЫНОВ АТЫНДАҒЫ ҚАРАҒАНДЫ ТЕХНИКАЛЫҚ УНИВЕРСИТЕТІ»

Киберқауіпсіздік және жасанды интеллект кафедрасы

Аубакирова А.Е.

«PYTHON ОБЬЕКТІГЕ БАҒЫТТАЛҒАН БАҒДАРЛАМАЛАУ»

ПӘНІ БОЙЫНША
ДӘРІСТЕРГЕ
ӘДІСТЕМЕЛІК НҰСҚАУЛАР

Қарағанды 2026

КОММЕРЦИЯЛЫҚ ЕМЕС АКЦИОНЕРЛІК ҚОҒАМЫ
«ӘБІЛҚАС САҒЫНОВ АТЫНДАҒЫ ҚАРАҒАНДЫ ТЕХНИКАЛЫҚ УНИВЕРСИТЕТІ»

Киберқауіпсіздік және жасанды интеллект кафедрасы

Аубакирова А.Е.

«PYTHON ОБЬЕКТІГЕ БАҒЫТТАЛҒАН БАҒДАРЛАМАЛАУ»

ПӘНІ БОЙЫНША
ДӘРІСТЕРГЕ
ӘДІСТЕМЕЛІК НҰСҚАУЛАР

6B06105 «Data Science» білім беру бағдарламасы

Қарағанды 2026

ДӘРІС КОНСПЕКТІЛЕРІ

Тақырып 2.
Объектіге бағытталған бағдарламалау парадигмасының ерекшелігі

Объектіге - бағытталған бағдарламалау бағдарламалық жасақтаманы әзірлеу әдістемелерінің (парадигмаларының) бірі болып табылады, бұл әдістемеге "объектілер" тұжырымдамасына негізделген тілдер кіреді – өрістерді (деректерді) және әдістерді (әрекет объектісі орындайтын) біріктіретін шартты нысандар.
Объектіге бағытталған бағдарламалау тілінің не екенін нақты анықтау өте қиын, негізгі тәсілдердің бірі-ОББ негізін қалаушы Алан Кэйдің 6 қағидасы.
1. Барлығы объект болып табылады.
2. Нысандар хабарламаларды жіберу және қабылдау арқылы өзара әрекеттеседі. Хабарлама-бұл әрекетті орындау кезінде қажет болуы мүмкін дәлелдер жиынтығымен толықтырылған әрекетті орындау туралы сұраныс.
3. Әрбір нысанда басқа нысандардан тұратын тәуелсіз жады бар.
4. Әрбір объект объектілердің жалпы қасиеттерін білдіретін класс өкілі болып табылады (мысалы, бүтін сандар немесе тізімдер).
5. Сыныпта объектінің функционалдығы бір сыныптың даналары болып табылатын барлық нысандар бірдей әрекеттерді орындай алады.
6. Сыныптар мұрагерлік иерархиясы деп аталатын ортақ тамыры бар бір ағаш тәрізді құрылымға ұйымдастырылған. Белгілі бір сыныптың даналарымен байланысты жад пен мінез-құлық иерархиялық ағаштың астында орналасқан кез келген сыныпқа автоматты түрде қол жетімді.
Объектіге бағытталған тілдердің ерекшеліктері – мұрагерлік, инкапсуляция, полиморфизм сияқты механизмдерді қолдану.
Мұрагерлік-жаңа элементтерді (әдістерді) қосу арқылы объектілердің жаңа класын құру. Кейбір объектіге бағытталған тілдер бірнеше мұрагерлікті жүзеге асыруға мүмкіндік береді, яғни бірнеше басқа сыныптардың мүмкіндіктерін бір сыныпта біріктіреді.
Инкапсуляция-бағдарламаның басқа бөліктеріне ауыртпалықсыз өзгерістер енгізуге мүмкіндік беретін іске асырудың егжей-тегжейін жасыру, Бұл бағдарламалық жасақтаманы сүйемелдеу мен өзгертуді айтарлықтай жеңілдетеді.
Полиморфизм-полиморфизмде ата-аналық кластың кейбір бөліктері (әдістері) осы ұрпаққа тән әрекеттерді жүзеге асыратын жаңаларымен ауыстырылады. Осылайша, сынып интерфейсі өзгеріссіз қалады және бірдей атаумен және параметрлер жиынтығымен әдістерді енгізу әр түрлі болады.

Python қолданбаларды жазудың өзекті тілі ретінде
Python-жоғары деңгейлі объектіге бағытталған бағдарламалау тілі. Тілдің негізгі сипаттамалары-әзірлеудің жоғарылауы және жазылған кодтың оқылуы. Python өте қарапайым синтаксиске ие.

Негізгі архитектуралық ерекшеліктер-жадыны автоматты басқару, динамикалық теру, көп ағынды есептеулерді қолдау, жоғары деңгейлі деректер құрылымдары, ерекшеліктерді өңдеу механизмі, бағдарламаларды модульдерге бөлу, модульдерді пакеттерге біріктіру.
Python динамикалық теруді қолдайды, нәтижесінде айнымалы түрі тек орындау кезінде анықталады. Python-да белгілі бір кіріктірілген түрлері бар. Әрбір мән функциялар, модульдер, класстар мен әдістерді қоса алғанда объект болып табылады. Жаңа түрлерді қосу мұрагерлікті қолдайтын класстарды жазу арқылы мүмкін болады.

Python-дағы блоктар қойынды арқылы ерекшеленеді, бұл бағдарламашыларды кодты жазудың таза стиліне үйретеді және Код жолдарының санын азайтады.
Атаулар кез-келген латын әрпінен немесе кез-келген Юникод алфавитінің әріптерінен Python 3-тен, кез-келген регистрден немесе астын сызудан басталуы мүмкін, содан кейін сандарды да көрсетуге болады, бірақ кілт сөздерді қолдануға болмайды. Python аудармашысы үш имен кеңістігіне қол жеткізе алады: жергілікті, Ғаламдық және ендірілген.

ASCII емес таңбаларды пайдалану үшін модульдің басында бастапқы кодты кодтау қажет.
Сондай-ақ, Python көптеген желілік протоколдар мен интернет форматтарын қолдайтын стандартты кітапханаға ие, операциялық жүйемен жұмыс істеуге арналған модульдер жиынтығы, мәтіндік кодтармен, күрделі математикалық өрнектермен, криптографиялық хаттамалармен, мұрағаттармен, мультимедиялық форматтармен жұмыс істеуге арналған модульдер және басқалар. Стандартты кітапханадан басқа, орнатуға болатын көптеген әртүрлі кітапханалар бар.
Мұның бәрі Python -. бүкіл әлемдегі көптеген компанияларда қолданылатын танымал, үнемі дамып келе жатқан және кең таралған тілге айналдырады, оның қарапайымдылығы мен қысқалығы қуатты әр түрлі құралдарды қолдану оны сценарий тілі ретінде өте ыңғайлы. Python-стандартты бағдарламалардың мүмкіндіктерін кеңейту үшін пайдалануға болады, әр түрлі кітапханалардың көмегімен Python - ыңғайлы басқару құралы ретінде пайдалануға болады.
Python - калькулятордан бастап Youtube, Facebook және Instagram сияқты қызметтерге дейін кез келген қиындықтағы қолданбаларды жазу тілінің тамаша таңдауына айналдырады.

Шартты Нұсқаулық синтаксисі if

Шартты Нұсқаулық if бағдарламаның сызықтық құрылымын бұзуға мүмкіндік береді. Айталық, біз осы x саны бойынша оның абсолютті мәнін (модуль) анықтағымыз келеді. Бағдарлама x айнымалысының мәнін басып шығаруы керек, егер x>0 немесе шамасы x болса. Бағдарламаның сызықтық құрылымы бұзылады: x > 0 шарттарының әділдігіне байланысты бір немесе басқа шаманы шығару керек. Питондағы бағдарламаның сәйкес бөлігі келесідей:

```
x = int(input())
if x > 0:
    print(x)
else:
    print(-x)
```

Бұл бағдарламада шартты Нұсқаулық қолданылады if (егер). If сөзінен кейін қос нүктемен аяқталатын тексерілетін шарт (x > 0) көрсетіледі. Осыдан кейін, егер шарт шын болса, орындалатын нұсқаулар блогы келеді, біздің мысалда бұл x экранының шығысы, содан кейін қос нүктемен аяқталатын else (басқаша) сөзі және тексерілетін шарт дұрыс болмаса орындалатын нұсқаулар блогы келеді, бұл жағдайда-x мәні шығарылады.
Сонымен, Питондағы шартты нұсқаулықта келесі синтаксис бар:
if шарты:

Нұсқаулық блогы 1
else:
Нұсқаулық блогы 2

Егер шарт шын болса, 1-Нұсқаулық блогы орындалады. Егер шарт жалған болса, нұсқаулар блогы 2 орындалады.
Шартты нұсқаулықта else командасы және одан кейінгі блок болмауы мүмкін. Мұндай нұсқаулық толық емес тармақталу деп аталады. Мысалы, егер x саны берілсе және біз оны абсолютті x мәніне ауыстырғымыз келсе, онда мұны келесідей жасауға болады:

```
x = int(input())
if x < 0:
    x = -x
print(x)
```

Бұл мысалда x айнымалысына-x мәні тағайындалады, бірақ x<0 болған жағдайда ғана. Бірақ print(x) нұсқаулығы тексерілетін шартқа қарамастан әрқашан орындалады.
If немесе else нұсқауларына қатысты нұсқаулар блогын бөлектеу үшін Питон тілінде шегіністер қолданылады. Бір блокқа қатысты барлық нұсқаулар шегініс шамасына тең болуы керек, яғни жолдың басында Бос орындар саны бірдей болуы керек. 4 бос орын шегінісін пайдалану ұсынылады және шегініс ретінде қойынды (табуляция) белгісін пайдалану ұсынылмайды.

Шартты нұсқаулар
Шартты нұсқаулардың ішінде кез-келген Питон тілінің нұсқауларын, соның ішінде шартты нұсқауларды қолдануға болады. Нәтижесінде біз кірістірілген тармақталуды аламыз-бір шарттың ішінде бағдарламаны орындау барысында басқа шарт пайда болады.

Бұл жағдайда кірістірілген блоктар шегініс өлшемінен үлкен болады (мысалы, 8 бос орын). Мұны нөлдік емес санмен берілген бағдарламаның мысалында көрсетейік X және Y (X,y) нүктесі координаталық жазықтықтың төрттен бірінде қайсысында екенін анықтайды:

```
x = int(input())
y = int(input())
if x > 0:
   if y > 0:              # x > 0 и y > 0
      print("Бірінші тоқсан")
   else:                  # x > 0, а y < 0
      print("Төртінші тоқсан")
else:
   if y > 0:              # x < 0, а y > 0
      print("Екінші тоқсан")
   else:                  # x < 0, и y < 0
      print("Үшінші тоқсан ")
```

Бұл мысалда біз түсініктемелерді қолдандық – аудармашы елемейтін мәтін. Питондағы түсініктемені көрсету үшін # таңбасы қолданылады. Осы таңбадан кейінгі барлық мәтін жолдың соңына дейін түсініктеме болып табылады.

Салыстыру операторлары

Кесте 1.

Салыстыру операторы	Сипаттама
>	мысалы, a > b екенін тексеріңіз
<	мысалы, a < b екенін тексеріңіз
==	мысалы, a = b екенін тексеріңіз
!=	мысалы, a тең емес b ө ға екенін тексеріңіз
>=	Көп немесе тең екенін тексереді
<=	Кіші немесе тең екенін тексереді

Питондағы салыстыру операторларын тізбектерге біріктіруге болады.
Мысалы, a == b == c или 5 < x <= 12

Мысалы 1.

```
if n <= 100:
        b = n + a
```

Мысалы 2.

```
tovar1 = 180
tovar2 = 400
if tovar1+ tovar2 > 500:
   print("500 тенге жетпейді")
else:
   print("Чек төленген")
```

Каскадты шартты нұсқаулар

if условие 1:

операторы
elif условие 2:
операторы
elif условие 3:
операторы
…..
else:
операторы

Координаталық жазықтықтың төрттен бірін анықтайтын бағдарлама мысалын операция арқылы "каскадты" тізбекті пайдаланып қайта жазуға болады if... elif... else:

```
x = int(input())
y = int(input())
if x > 0 and y > 0:
   print("Бірінші тоқсан")
elif x > 0 and y < 0:
   print("Төртінші тоқсан")
elif y > 0:
   print("Екінші тоқсан")
else:
   print("Үшінші тоқсан")
```

Бірнеше elif басқа бағдарламалау тілдерінде switch - case конструкциясынын тамаша алмастыра алады.

Логикалық операторлар

Кейде бір уақытта бір емес, бірнеше шартты тексеру қажет. Мысалы, санның жұп екенін тексеруге болады (N % 2 == 0) (n-ді 2-ге бөлудің қалдығы 0-ге тең). Егер n және m екі бүтін сандардың жұп екенін тексеру қажет болса, екі шарттың да әділдігін тексеру қажет: n % 2 == 0 және m % 2 == 0. Ол үшін оларды and (логикалық n) логикалық операторының көмегімен біріктіру керек: n % 2 == 0 and m % 2 == 0.

Питонда стандартты логикалық операторлар бар: логикалық көбейту И, логикалық қосу ИЛИ, логикалық теріске шығару НЕ.

Логикалық көбейту И екілік оператор болып табылады (яғни екі операндты оператор: сол және оң) және and формасына ие. And операторы True мәнін қайтарады, егер оның екі операндында да True мәні болса.

Логикалық қосу ИЛИ екілік оператор болып табылады және кем дегенде бір Операнд True болғанда ғана True мәнін қайтарады. Оператор логикалық ИЛИ or түрінде болады.

Логикалық емес (теріске шығару) унарлы (яғни бір операндпен) оператор болып табылады және not формасына ие, содан кейін жалғыз операнд. Егер операнд жалған болса және керісінше болса, логикалық НЕ шын мәнін қайтармайды.

Мысал.

a немесе b сандарының кем дегенде біреуі 0-ге аяқталатынын тексерейік:

```
a = int(input())
b = int(input())
```

```
if a % 10 == 0 or b % 10 == 0:
   print('YES')
else:
   print('NO')
```
a санының оң, ал b санының теріс емес екенін тексерейік:
```
if a > 0 and not (b < 0):
```

Not (B < 0) орнына (b >= 0) жазуға болады.

Шарттарға қарапайым бағдарламаларды қарастырыңыз.

Цикл операторлары

Циклдар құрылымдық бағдарламалаудың шартты операторлар сияқты маңызды бөлігі болып табылады. Циклдардың көмегімен код бөлімдерінің қайталануын ұйымдастыруға болады. Бұл қажеттілік жиі туындайды.

Мысалы, пайдаланушы сандарды дәйекті түрде енгізеді және олардың әрқайсысы жалпы сомаға қосылуы керек. Немесе экранға 1-ден 100-ге дейінгі бүтін сандардың квадраттарын шығару керек.

While цикл операторы

While-Python-дағы ең әмбебап циклдердің бірі. Оператор цикл шартының мәні болғанша цикл денесін орындайды.
```
i = 5
while i < 15:
   print(i)
   i = i + 2
```

Шығады:
```
5
7
9
11
13
```
While операторында оның денесі аяқталғаннан кейін бағдарлама цикл тақырыбына оралып, шартты қайтадан тексереді. Егер логикалық өрнек шындықты қайтарса, онда цикл денесі қайтадан орындалады. Содан кейін біз қайтадан тақырыпқа ораламыз және т.б.

Цикл өз жұмысын тақырыптағы логикалық өрнек өтірікті қайтарған кезде ғана аяқтайды, яғни циклды орындау шарты енді орындалмайды. Осыдан кейін цикл блогының астында орналасқан операторлар орындалады. Олар "циклден шығу бар" дейді.

While циклімен екі ерекше жағдай болуы мүмкін:

1. Егер циклге бірінші рет кірген кезде логикалық өрнек жалған болса, онда цикл денесі бір рет орындалмайды. Бұл жағдайды қалыпты деп санауға болады, өйткені белгілі бір жағдайларда бағдарламаның логикасы цикл денесінің өрнектерін орындаудың қажеті жоқ деп болжауы мүмкін.
2. Егер while тақырыбындағы логикалық өрнек ешқашан жалғандықты қайтармаса, бірақ әрқашан шындыққа тең болса, онда цикл ешқашан аяқталмайды, егер оның денесінде циклден мәжбүрлі шығу операторы болмаса (break - үзіліс) немесе бағдарламадан шығу функцияларына қоңырау шалу – Python жағдайында quit (), exit (). Егер цикл шексіз рет қайталанса, онда бағдарламада цикл болады. Осы уақытта ол қатып қалады және өзін-өзі аяқтай алмайды.

Мысалды қарастырайық:
```
x1 = 100
```

```
i = 0
while i < 5:
    n = int(input())
    x1 = x1 - n
    i = i + 1
print("Қалды", x1)
```

For цикл операторы

Python-да цикл for кілт сөзінен басталады, содан кейін айнымалының ерікті атауы, for циклі мәндер жиынтығында жүгіреді, әр мәнді айнымалыға орналастырады, содан кейін циклде біз осы айнымалымен әр түрлі әрекеттерді жасай аламыз. Жалпы синтаксис for...in Python да келесідей:

```
for айнымалы in мәндер жиынтығы:
    операторлар
```

Цикл орындалған кезде Python жиынтықтағы барлық мәндерді дәйекті түрде алады және олардың айнымалысын береді. Жиынтықтағы барлық мәндер қайталанғаннан кейін цикл өз жұмысын аяқтайды.

Мәндер жиынтығы ретінде, мысалы, таңбалар жиынтығын білдіретін жолды қарастыруға болады.

Мысалды көрейік:

```
message = "Hello"
for x in message:
    print(x)
```

Цикл x айнымалысын анықтайды, in операторынан кейін "Hello"жолын сақтайтын message айнымалысы қайталанатын жиын ретінде көрсетіледі. Нәтижесінде, for циклі message жолындағы барлық таңбаларды дәйекті түрде сұрыптап,оларды x айнымалысына орналастырады.

Range () функциясы

Range () функциясы сандар тізбегін құру үшін қолданылады. Егер біз range(10) тапсырсақ, ол 0-ден 9-ға дейінгі сандарды жасайды. Range () функциясының синтаксисі төменде келтірілген.

```
range(start,stop,step size)
```

- Start итерацияның басталуын білдіреді.
- Stop цикл stop-1-ге дейін қайталанатынын білдіреді. range (1,5) 1-ден 4 итерацияға дейінгі сандарды жасайды. Бұл қосымша параметр.
- size итерацияда белгілі бір сандарды өткізіп жіберу үшін қолданылады. Оны пайдалану міндетті емес. Әдепкі бойынша, қадам өлшемі 1-ге тең.

Келесі мысалдарды қарастырыңыз:

Мысал 1.

Сандарды ретімен басып шығаруға арналған бағдарлама.

```
for i in range(10):
      print(i,end = ' ')
```

Мысал 2.

Берілген Сан үшін көбейту кестесін басып шығаруға арналған бағдарлама.

```
n = int(input("Enter the number "))
for i in range(1,10):
   c = n*i
   print(n,"*",i,"=",c)
```

Бағдарламаны орындағаннан кейін біз аламыз:
Enter the number 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45

Мысал 3.
Range () ішіндегі қадам өлшемін пайдаланып жұп санды басып шығаруға арналған бағдарлама.

```
n = int(input("Enter the number "))
for i in range(2,n,2):
    print(i)
```

For циклі while циклінен әлдеқайда жылдам. Бұл нұсқаулық цикл денесін бірнеше рет орындайды.
Мысалы:
1-ден k-ге дейінгі бүтін сандардың қосындысын табыңыз

```
sum = 0
k=int(input('введите к  '))
for i in range(1, k+1):
    sum = sum + i
print(sum)
```

Итерация қадамын 2-ге қою үшін жазыңыз

```
for n in range(1, k+1, 2):
```

For циклінің басқа форматтарын қарастырыңыз:
Мысал 1.

```
for X in 10, 12, 18, 'бір', 'екі', 'үш':
print(x)
```

Бағдарламаны орындағаннан кейін біз аламыз
10
12
18
бір
екі
үш

Мысал 2.

```
s = 'Сәлем'
siemens үшін: # әрбір s таңбасы үшін
print(sim)
```

Бағдарламаны орындағаннан кейін біз аламыз
С
Ә
Л
Е

M