Дәріс 2. Деректерді талдауға арналған Python негіздері.

Қарқынды Python курсы

Синтаксис негіздері

Деректер түрлері (жолдар, тізімдер, сөздіктер, жиындар)

Ағынды басқару, функциялар, ерекшеліктер

ОББ (Объектіге-бағытталған бағдарламалау), генераторлар, zip, args/kwargs

Аннотация түрлері

Бүгінгі таңда аннотация жасауға ұшырайтын 21-ғасырдың негізгі құндылықтарының бірі – деректер. Оларды тиімді жинау, өңдеу және түсіндіру мүмкіндігі заманауи ғылым мен экономика үшін өмірлік маңызды салалардың біріне айналды. Осы орайда, Python деректерді талдау және ғылыми зерттеу саласында жұмыс істейтін әмбебап құрал ретінде ерекше орын алады. Python-ның жүйелі синтаксисі, кең көлемді функционалдылығы және кітапхана экожүйесі оны деректер ғылымында үстемдік ететін тілге айналдырды.

Қарқынды Python курсы

Python - қазіргі деректер ғылымында ең көп таралған бағдарламалау тілдерінің бірі. Оның қарапайым синтаксисі, көп функциялы кітапханалары және ашық бастапқы экожүйесі Python-ды деректерді талдау, машиналық оқыту және жасанды интеллект салаларындағы негізгі құралға айналдырды.

Intensive Python курсының осы дәрісінің мақсаты келесі тақырыптарды қамтиды:

1. Python тіліне қатысты негізгі құрылымдарды, синтаксистік ережелерді, деректердің түрлерін және басқаруды.
2. NumPy, pandas және Matplotlib кітапханаларының ғылыми қолданулары.

Intensive Python заманауи деректер ғылымына ең лайықты бағдарламалау тілдерінің бірі. Синтаксистің қарапайымдығы, үлкен кітапханалары және ашық код экожүйесінің кеңдігі Python-ды деректерді талдау, машина оқытуы және жасанды интеллект секілді салалардың негізі болып табылатын құралға айналдырады. Осылайша, Intensive Python – бұл қысқа мерзімді, бірақ кешенді оқу бағдарламасы, ол сізге Python-ды жүйелі әрі ғылыми тәсілмен меңгеруге және оны практикаға негізделген түрде қолдануға көмектеседі.

Python 1990 жылдары жасалғанмен, соңғы он жыл ішінде ол ғылым, индустрия және т.б. үшін негізгі талдау тіліне айналды. Бұл неғұрлым қарапайым құрылым, кең көлемді функционалдылық және мықты кітапхана экожүйесімен ғана емес, тұрақты негізде мүмкін болды. Python программалау тілі ғана емес, жалпы зерттеулер мен талдаулар үшін әмбебап

ғылыми құрал ретінде жариялайды. Python тілін зерттеу барысында зерттеуші тұрақты программалау дағдыларымен қатар, алгоритмдік және логикалық ойлау дағдыларын да алады. Программалау барысында қолданушы жүйелік түрде мәселенің құрылымын, деректер арасындағы қарым-қатынасты талдап, оны шешудің ең тиімді әдісін іздейді. Бұл процесс ғылыми және зерттеу ойлауын дамытады, нақты деректер бойынша негізделген қорытындылар жасау қабілетін арттырады. Сондай-ақ, Python зерттеушілерге модельдерді көзбен салуға, деректерді өңдеуге, есептеулер жүргізуге және қорытынды жасауға мүмкіндік береді.

Осылайша, ғылыми есептеулер әртүрлі модельдер немесе қосымшаларды, статистикалық модельдерден бастап машина оқытуға дейінгі және болжамды талдауды қолдайтын Python қолдануды білдіреді. Бұл тәсіл дәстүрлі әдісті толықтырумен қатар, деректерге негізделген ғылым сынды жаңа ғылыми бағыттарды жүзеге асыруға мүмкіндік береді. Сонымен қатар, Python-ның әмбебаптығы тілдің биоинформатикадан бастап эконометрикаға, әлеуметтік немесе басқа ғылымдарға дейінгі әртүрлі салаларда қолданылуы мүмкін екендігін білдіреді.

Синтаксис негіздері

Python синтаксисі. Бұл Python тілінің грамматикалық және құрылымдық негізі, яғни бағдарламаның қалай жазылатынын және түсіндірілетінін анықтайтын ережелер жиынтығы.

Синтаксис бағдарламашы мен компьютер арасындағы "ортақ тіл" сияқты: ол кодтың дұрыс жазылуын және орындалуын қамтамасыз етеді. Python-ның ең көрнекті ерекшелігі - оның оқуға оңай және интуитивті синтаксисі. Python тілін зерттеу барысында зерттеуші тұрақты программалау дағдыларымен қатар, алгоритмдік және логикалық ойлау дағдыларын да алады. Программалау барысында қолданушы, жүйелік түрде мәселенің құрылымын, деректер арасындағы қарым-қатынасты талдап, оны шешудің ең тиімді әдісін іздейді. Бұл процесс ғылыми және зерттеу ойлауын дамытады, нақты деректер бойынша негізделген қорытындылар жасау қабілетін арттырады. Сондай-ақ, Python зерттеушілерге модельдерді көзбен салуға, деректерді өңдеуге, есептеулер жүргізуге және қорытынды жасауға мүмкіндік береді.

Осылайша, ғылыми есептеулер әртүрлі модельдер немесе қолданбаларды, статистикалық модельдерден бастап ғылыми және процестік бағдарланған болжамды талдауды қамтиды. Дәстүрлі әдіс ашық болғанымен, мұндай жүзеге асыру басқару мен деректерге негізделген ғылым сынды көптеген ғылыми бағыттарды жүзеге асыруға да

мүмкіндік береді. Осы тілдің жобаларындағы жасаушысы Гидо ван Россумның принциптерінің бірі "айқындық және қарапайымдылық" болып табылды. Бұл Python тілінің табиғи тіл заңына мүмкін болатын құрылымда жазылуы мүмкін екенін білдіреді. Python тілінен өзгеше, басқа тілдер жолдар арасын бөлу үшін бос жерлер мен шегіністерді талап етеді; Python қажетті жақшалар немесе тіркестер орнына осы механизмді пайдаланады. Бұл әдіс кодының құрылымын жеңілдетеді және логикалық блоктарды бір-бірінен визуалды түрде ажыратуға көмектеседі.

Python тілінде код жазуды және жазуды үйрену алдында, айнымалылар мен мәндердің қалай істейтінін түсінуіңіз қажет. Кез келген бағдарлама деректермен жұмыс істейді: ол оны файлдан немесе кіріспеден сақтайды, оны алады, өңдейді және өңделген форматта сол деректерді береді(нәтижені шығарады).

Бұл жағдайда айнымалы ұғымы орталық болып табылады. Айнымалы – деректер контейнері, ал осы деректермен операциялар оның ішінде орындалады. Айнымалы жарияланатын қағида келесідей: айнымалы жарияланған кезде деректерді тағайындау деректер түрі көрсетілмей автоматты түрде жасалады:

Мысалы:

```
a = 10          # бүтін сан (int)
b = 5.7         # бөлшек сан (float)
name = "Data" # жол(сөз) (str)
```

Келесі атаулар қағидалары сақталуы тиіс: Программалау, логикалық, арифметикалық және басқарушы операцияларға мүмкіндік беретін операторлар.

Python синтаксистің қарапайымдығы операторларды табиғи тілге ұқсас жазуға мүмкіндік береді. Олардың оқылуын және түсініктігін жеңілдетеді.

Айнымалылар - деректерді сақтау және өңдеу үшін пайдаланылатын контейнерлер. Python тілінде айнымалыны жариялау кезінде деректер түрін көрсету қажет емес, аудармашы оны автоматты түрде анықтайды.

Айнымалыларды дұрыс атау – кодтың оқылуын жақсартады және олардың қатесіз шығуына кепіл болады. Python тілінде атау ережелері келесі түрде жүреді:

- Айнымалы әріптен немесе (_) таңбасынан басталады;
- Тек әріптер сандар және (_) таңбасы пайдаланылады;
- Регистрді дұрыс қадағылап, жазу керек (Age және age екі түрлі айнымалы);
- Кілт сөздер(for, if, while, etc.) атаулар ретінде қолданыла алмайды.

Негізгі операторлар

Python тіліндегі операторлар - айнымалылар мен мәндер бойынша әртүрлі әрекеттерді орындауға арналған таңбалар немесе кілт сөздер болып табылады. Олар бағдарламалауда логикалық, арифметикалық және басқару амалдарын орындауға мүмкіндік береді. Python синтаксисінің қарапайымдылығы табиғи тілге ұқсас операторларды жазуды жеңілдетеді, сондықтан оларды оқу және түсіну өте оңай.

1. Арифметикалық операторлар(Arithmetic operators)

Арифметикалық операторлар математикалық есептеулер жүргізуге қолданылады. Олар сандық деректермен (int, float) жұмыс істейді.

Оператор	Атауы	Мысалы	Нәтиже
+	Қосу	6+4	10
-	Азайту	8-2	6
*	Көбейту	7*8	56
/	Бөлу(нәтиже float)	24/6	4
//	Бүтін бөлу	8//3	2
%	Қалдық бөлу	10%3	1
**	Дәреже	2**3	8

Python код:

```
x = 10 y = 3
print(x + y)   # 13
print(x - y)   # 7
print(x * y)   # 30
print(x / y)   # 3.333...
print(x // y)  # 3 — бүтін бөлу
print(x % y)   # 1 — қалдық
print(x ** y)  # 1000 — 10^3
```

2. Салыстыру операторлары (Comparison Operators)

Бұл операторлар екі мәнді салыстыру үшін қолданылады және логикалық нәтиже (True немесе False) қайтарады. Олар шартты өрнектерде жиі қолданылады.

Оператор	Атауы	Мысалы
==	Тең	2==2
!=	Тең емес	3!=1
>	Үлкен	9>4

<	Кіші	2<8
>=	Үлкен немесе тең	5>=5
<=	Кіші немесе тең	3<=4

Python код:

```
a = 7 b = 10
print(a == b)  # False
print(a != b)  # True
print(a < b)   # True
print(a >= b)  # False
```

3. Логикалық операторлар (Logical Operators)

Логикалық операторлар бірнеше шарттарды біріктіру немесе инверсия жасау үшін қолданылады. Олар көбінесе булев (bool) мәндермен (True, False) жұмыс істейді.

Оператор	Атауы	Түсіндірме
and	Логикалық және	Екі шарт та True болса, нәтиже True
or	Логикалық немесе	Кем дегенде бір шарт True болса, нәтиже True
not	Терістеу(жоқ)	Мәнді керісінше айналдырады

AND:

```
age = 30
has_job = True
if age >= 18 and has_job:
    print("Жұмысқа жарамды және жұмысы бар адам")
```

OR:

```
if age < 18 or age > 65:
    print("Жұмысқа жарамсыз жаста")
```

NOT:

```
is_student = False
if not is_student:
    print("Бұл адам студент емес")
```

Блоктар және шегіністер (Indentation)

Python тілінің тағы бір маңызды және ерекше синтаксистік ерекшелігі - блоктар және шегіністер ережесі болып табылады. Басқа көптеген тілдерде – C++, Java, JavaScript блоктар {} жақшалар арқылы жазылады. Python-да кодтың құрылымы шегініс деңгейімен көрсетіледі. Бұл кодты оқыту және құрылымдау үшін жеңілдетеді.

Блок – белгілі бір логикалық бірлікті бейнелейтін код тобы. Әдетте, ол кейбір бақылау құрылымдарына қатысты қолданылады, мысалы, if, for, while, def, class.

Python интерпретаторы кодтың қай жолы қай блокқа тиесілі екенін шегініс көлеміне қарай анықтайды.

Шегініс бос орындар(spaces) немесе табуляция(tab) арқылы жасалады, алайда оларды араластыру мүмкін емес, себебі одан қате пайда болады(IndentationError).

Мысалы:

```
age = 25
if age < 18:  < if блогы
шегініс >  print("Кәмелетке толмаған")
elif age < 65: < elif блогы
шегініс >   print("Жұмысқа жарамды жаста")
else:   < else блогы
шегініс >   print("Зейнет жасында")
```

Комментарийлер мен құжаттандыру

Түсініктемелер (комментарийлер) - бағдарламаның орындалуына әсер етпейтін, бірақ оның құрылымы мен логикасын түсіндіруге арналған мәтін элементтері болып табылады. Түсініктемелер бағдарламашыға да, басқа зерттеушілерге де кодтың не істейтінін, қандай параметрлер қолданылатынын және қандай нәтижелер күтілетінін түсінуге мүмкіндік береді.

Python тілінде функциялар мен сыныптарға арнайы құжаттама жолдарын (docstring) қосуға болады. Бұл жолдар бағдарламаның логикасын сипаттайды және оны автоматты түрде құжаттауға мүмкіндік береді. Docstring үш қос тырнақшаның арасына жазылады және функция немесе класс анықталғаннан кейін бірден орналастырылады.

Комментарий:

```
# Бұл айнымалы пайдаланушының жасын сақтайды
age = 25
print(age)  # Айнымалыны экранға шығару
```

Құжаттама:

```
def add_numbers(a: int, b: int) -> int:
    """
    Екі бүтін санды қосу функциясы.
    Параметрлер:
        a (int): Бірінші сан
        b (int): Екінші сан
    Қайтарады:
        int: Екі санның қосындысы
    """
    return a + b
```

Кіріс және шығыс операторлары

Python бағдарламаларының негізгі әрекеттерінің бірі-пайдаланушымен өзара әрекеттесу. Ол екі бағытта жүреді:

Енгізу(Input) - пайдаланушыдан деректерді қабылдау;

Шығару(Output) -нәтижені немесе хабарламаны экранда көрсету.

print() – Python тіліндегі ақпаратты экранға шығару үшін ең жиі пайдаланылатын функция.

```
print("Сәлем, Әлем!")
```

input () функциясы пайдаланушыдан деректерді алуға мүмкіндік береді. Бұл функция енгізілген мәнді жол (str) ретінде қайтарады.

```
name = input("Атыңызды енгізіңіз: ")
print("Сәлем,", name)
```

Деректер түрлері (жолдар, тізімдер, сөздіктер, жиындар)

Деректер түрлері Python-ның негізгі ерекшеліктерінің бірі болып табылады және олар деректер құрылымдарында қарапайымдылық пен икемділікті ұсынады. Сонымен қатар, деректер түрлері программа логикасының негізі болып табылады және ақпаратты сақтау, өңдеу және талдау мүмкіндігін жасайды. Деректердің ғылыми негізде құрылымы арқасында, кейінгі талдаудың дұрыстығы мен тиімділігі нақты түсініледі, бұл ұғымдар әсіресе маңызды.

Python тілінде деректердің екі негізгі тобы бар:

Қарапайым (primitive) типтер - бүтін (int), бөлшек (float), логикалық (bool), жолдық (str) типтері;

Құрама (composite) типтер – тізімдер (list), кортеждер (tuple), сөздіктер (dict), жиындар (set).

Деректер ғылымында көбіне төрт түр қолданылады және олар әсіресе маңызды: жолдар (str), тізімдер (list), сөздіктер (dict) және жиындар (set).

1. Жолдар - бұл мәтіндік деректерді сақтау үшін қолданылатын түрі. Python тілінде жолдар тырнақшаға алынған таңбалар тізбегі ретінде жазылады.

2. Тізімдер - Python тілінде ең көп қолданылатын және икемді деректер құрылымдарының бірі болып табылады. Бұл бір контейнерде бірнеше заттарды (әртүрлі типтегі) сақтауға мүмкіндік береді.

3. Сөздіктер - ақпаратты "кілт-мән" жұптары түрінде сақтайтын құрылым. Бұл ақпаратқа жылдам және мағыналы қол жеткізуге мүмкіндік береді.

4. Жиындар - ретсіз және қайталанбайтын элементтердің жиынтығы. Бұл математикалық жиын ұғымына ұқсас.

Ағынды басқару және функциялар

Python бағдарламалау тілінде басқару ағыны (control flow), функциялар (functions) мен ерекшеліктер (exceptions) - бағдарламаның логикалық құрылымын, модульдік ұйымдастырылуын және сенімділігін қамтамасыз ететін негізгі ұғымдар болып табылады. Бұл элементтер кез-келген күрделі жүйенің негізін құрайды, өйткені олар бағдарламаға шешім қабылдауға, логиканы қайталауға және ақауларға төзімділікке мүмкіндік береді.

Python жоғары деңгейлі, түсіндірілетін тіл болғандықтан, оның грамматикасы табиғи тілге жақын және оқуға оңай. Сондықтан ағынды басқару элементтері мен функцияларын құру процесінде код логикасы айқын көрінеді,

1. Ағынды Басқару

Басқару ағыны - бағдарламаны орындаудың тәртібін анықтайтын механизм. Әдетте, код жолдары жоғарғы жақтан төмен қарай орындалады. Кейбір жағдайларда, кодтың орындалуы өзгереді: шешім немесе циклге жеткенде. Python программалау тіліндегі басқару ағыны операторларын үш негізгі топқа бөлуге болады:

- Шартты мәлімдемелер
- Циклдік құрылымдар (ілмектер)
- Блоктау және басқару операторлары (үзіліс, жалғастыру, өту)

- Шартты операторлар - белгілі бір логикалық жағдайға байланысты бағдарламаның әртүрлі тармақтарын орындауға мүмкіндік береді. Python тілінде шартты мәлімдемелер if, elif және else кілт сөздерін пайдаланып жасалады.

```
t = 18
```

```
if t > 25:
    print("Ауа райы ыстық.")
elif t >= 15:
    print("Ауа райы жылы.")
else:
    print("Ауа райы салқын.")
```

- Циклдік құрылымдар - қайталанатын әрекеттерді автоматтандырудың негізгі құралы болып табылады. Python тілінде ең жиі қолданылатын цикл түрлері for және while болып табылады.

For циклі белгілі бір тізімдегі элементтерді бірінен соң бірін қайталап өту үшін пайдаланылады:

```
students = ["Айгерім", "Нұрлан", "Ермек"]
for name in students:
    print("Сәлем,", name)
```

While циклі берілген шарт ақиқат болғанша орындалады:

```
counter = 0
while counter < 3:
    print("Count:", counter)
    counter += 1
```

2. Функциялар (Functions)

Функциялар – бағдарламалау тілдеріндегі маңызды құрылымдардың бірі болып табылады. Олар бағдарламаның логикалық құрылымын шағын, мағыналы және тәуелсіз бөліктерге бөлуге мүмкіндік береді. Функциялар арқылы код қайталанбайды, оқу оңайырақ және тест тапсыру процесі жеңілдетіледі.

Python бағдарламалау тілінде функциялар қайталану және реттілікті автоматтандыру, деректерді өңдеу логикасын бөлек сақтау, модульдік бағдарламалау принциптерін жүзеге асырудың негізгі құралдары болып табылады.

Бағдарламалау тұрғысынан функция дегеніміз - белгілі бір тапсырманы орындайтын тәуелсіз код блогы.

Ал математикалық мағынада алатын болсақ функция: "енгізу → өңдеу → шығару" типін бейнелеу(mapping) болып табылады.

Мысалы:

```
def add(a, b)
    return a + b
```

Функцияны анықтау құрылымы:

```
def функция_атауы(параметрлер):
    """Докстринг (құжаттандыру мәтіні)"""
    денесі (код блогы)
    return нәтиже
```

Функцияның маңызын бөліп және ашып жазатын болсақ:

1. Функциялар бағдарламаның тиімділігін арттырады.:
2. Модульдік етеді-әр бөлік өз жұмысын орындайды;
3. Қайта пайдалануға мүмкіндік береді.
4. Тестілеуді жеңілдетеді (әр функцияны жеке тексеруге болады);
5. Кеңейту үшін қолайлы (жаңа логиканы қосу оңай);
6. Жадты үнемдейді, себебі код қайталанбайды.

Деректер ғылымында функциялар деректер модельдерін тазарту, түрлендіру, визуализациялау және құру процестерін автоматтандыру үшін кеңінен қолданылады. Мысалы, бір функция бірнеше көздерден деректерді жинай алады, ал басқа функция оларды талдап, статистикалық есеп шығарады.

ОББ (Объектіге-бағытталған бағдарламалау), генераторлар, zip, args/kwargs

Объектіге-бағытталған бағдарламалау (Object - Oriented Programming) - бұл объектілер мен сыныптарды қолдана отырып бағдарлама құру тәсілі.

Python – толыққанды ОВВ тілдерінің бірі.

- Класс (класс) - бұл объектінің моделі, яғни оның қасиеттері мен әрекеттерін сипаттайтын құрылым.
- Объект (Объект) - бұл класс негізінде жасалған нақты мысал.
- Атрибуттар - бұл объектілік деректер (айнымалылар).
- Әдістер - бұл объект орындайтын әрекеттер (функциялар).

Мысалы:

```
class Student:
    def __init__(self, name, age):  # Конструктор
        self.name = name
        self.age = age
```

```
        def greet(self):
            print(f"Сәлем, менің атым {self.name}, жасым {self.age}")


    s = Student("Айгүл", 20)
s.greet()
```

Бұл кодтың негізгі принциптерін ашатын болсақ:

Инкапсуляция — деректерді жасыру;

Мұрагерлік (inheritance) — бір кластың қасиеттерін екіншісіне беру;

Полиморфизм — бір әдістің әр түрлі әрекет ету қабілеті.

Генераторлар

Генераторлар - бұл деректерді түгел бәрін емес, ретімен (кезегімен) шығаратын функциялар.

Олар жадты үнемдей отырып, үлкен көлемдегі деректермен тиімдірек жұмыс істеуге мүмкіндік береді.

Мысалы:

```
def count_up_to(n):
    for i in range(1, n + 1):
        yield i  # yield — генератор құру операторы

for num in count_up_to(5):
    print(num)
```

zip() функциясы

zip () — бірнеше тізімдерді (list, tuple) жұптап біріктіретін кірістірілген функция.

Мысалы:

```
names = ["Айгүл", "Нұрбек", "Ержан"]
ages = [20, 22, 19]

for name, age in zip(names, ages):
    print(name, age)
```

Нәтиже:

Айгүл 20

Нұрбек 22

Ержан 19

args/kwargs

Python функцияларындағы аргументтер санын икемді басқаруға мүмкіндік береді:

Args - позициялық аргументтер жиынтығы (кортеж түрінде);

Kwargs - номиналды аргументтер жиынтығы (сөздік түрінде).

Мысалы:

```
def show_data(*args, **kwargs):
    print("args:", args)
    print("kwargs:", kwargs)

show_data(1, 2, 3, name="Айгерім", age=25)
```

Нәтиже:

```
args: (1, 2, 3)
kwargs: {'name': 'Айгерім', 'age': 25}
```

Аннотация түрлері

Аннотациялар - python тіліндегі айнымалылардың күтілетін деректер түрін, функция параметрлерін және олардың қайтару мәндерін көрсетуге арналған құрал болып табылады.

Олар кодтың оқылуын жақсартады, қателіктердің алдын алуға көмектеседі және автоматты тексеру жүйелеріне (мысалы, mypy) кодтың дұрыстығын тексеруге мүмкіндік береді.

Мысалы:

```
name: str = "Алия"       # жол (string)
age: int = 21            # бүтін сан (integer)
height: float = 1.68     # қалқымалы сан (float)
is_student: bool = True  # логикалық мән (boolean)
```

Қорытындылайтын болсақ Python тілі деректерді талдау саласындағы ең кең таралған және тиімді құралдардың бірі болып табылады. Оның қарапайым синтаксисі, бай кітапхана экожүйесі (NumPy, Pandas, Matplotlib) және икемді құрылымы зерттеушілер мен талдаушыларға үлкен көлемдегі деректермен тиімді жұмыс істеуге мүмкіндік береді.

Python негіздерін меңгеру - синтаксис, мәліметтер типтері, ағындарды басқару, функциялар, Объектіге бағытталған бағдарламалау принциптері - деректерді талдау процесінің негізі болып табылады. Сонымен қатар, аннотациялар, генераторлар және args/kwargs сияқты құралдар бағдарламаның құрылымын анық және сенімді етуге көмектеседі.

Яғни, Python-бұл жай ғана бағдарламалау тілі емес, ол деректер арқылы ойлау және талдау мәдениетін қалыптастыратын әмбебап құрал.

Әдебиеттер тізімі:

1. Fractal Data Science Professional Certificate (в т.ч. курс “Python for Data Science”) — https://www.coursera.org/professional-certificates/fractal-data-science Coursera
2. Python For Data Analytics: A Systematic Literature Review of Tools, Techniques, and Applications - https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5051680
3. McKinney, W. (2021). Python for Data Analysis**.** In Python Data Analytics. Springer, Singapore. DOI: https://doi.org/10.1007/978-981-16-8615-3_7