ӘОЖ 373.5.016:004.42

PYTHON ТІЛІНДЕ ЕСЕП ШЕШУ ҮДЕРІСІ АРҚЫЛЫ ОҚУШЫЛАРДЫҢ СЫНИ ЖӘНЕ АЛГОРИТМДІК ОЙЛАУЫН ДАМЫТУ

Қутлимуратов Нурмахасан Қанатұлы
nurmakanat7@gmail.com
Академик Е.А. Бөкетов атындағы Қарағанды ұлттық зерттеу университеті,Қарағанды қ., Қазақстан Республикасы
Ғылыми жетекшісі, техника ғылымдарының магистрі, оқытушы – Сайлаубаев С.Ш.

Ақпараттық-коммуникациялық технологиялардың қоғамның барлық салаларына терең бойлауы білім беру парадигмасындағы басты құндылықтарды қайта қарауды талап етіп отыр. Егер жиырмасыншы ғасырдың педагогикалық жүйесі оқушының жадында нақты фактілер мен ережелерді сақтауға бағытталса, қазіргі постиндустриялық кезеңде адамның танымдық қабілеті ақпаратты игерумен емес, оны құрылымдау, талдау және белгісіздік жағдайында шешім қабылдау икемділігімен өлшенеді. Бұл контексте информатика пәні шеңберінде бағдарламалау тілдерін, соның ішінде семантикалық құрылымы табиғи тілге барынша жақын Python тілін оқыту тек кәсіби-техникалық машықтарды қалыптастыру аясынан әлдеқашан шығып кеткен. Бағдарламалау үдерісі бүгінде оқушының зияткерлік әлеуетін ашатын, оның сыни және алгоритмдік ойлауын бір мезгілде дамытатын іргелі когнитивтік тренажер ретінде қарастырылады. Мектеп қабырғасындағы оқушы үшін компьютер коды тек машинаға берілетін нұсқаулық емес, ол – адамның өз ойын формализациялау, логикалық тізбектерді құру және күрделі мәселелерді шешуге бағытталған интеллектуалды акт.

Осы мәселенің ғылыми-теориялық табиғатына үңілетін болсақ, алдымен алгоритмдік ойлау (Computational Thinking) феноменінің мәнін ашып алу қажет. Бұл ұғымды 2006 жылы ғылыми айналымға қайта енгізіп, оның тұжырымдамалық негізін қалаған Карнеги-Меллон университетінің профессоры Жаннет Уинг болды. Оның іргелі пайымдауынша, алгоритмдік ойлау информатиканың тар шеңберіне ғана тиесілі ерекшелік емес, ол оқу, жазу және математикалық есептеумен қатар әрбір адам меңгеруге тиіс базалық сауаттылық түрі. Уинг бұл ұғымды абстракция деңгейлерін таңдай білу, үлкен жүйелерді басқару үшін оларды кішігірім құрамдас бөліктерге ыдырату, мәліметтер массивінен заңдылықтар іздеу және ең бастысы – адам мен машинаның ақпарат өңдеудегі мүмкіндіктері мен шектеулерін үйлестіре білу қабілеті деп түсіндіреді. Сыни ойлау мен алгоритмдік ойлаудың өзара ұштасу нүктесі дәл осы жерде айқын көрінеді. Сыни ойлау мәселенің неліктен туындағанын, оның себеп-салдарын талдап, ақпараттың дұрыстығына күмәнмен қарауға үйретсе, алгоритмдік ойлау сол мәселені шешудің нақты, қадамдық және оңтайландырылған үлгісін ұсынады. Яғни, сыни ойлау – бағалау құралы болса, алгоритмдік ойлау – жасампаздық пен орындалу құралы. Python тілінде есеп шығару оқушыдан осы екі қабілеттің үздіксіз синхронизациясын талап етеді: оқушы есептің шартын сыни тұрғыдан сараптайды, маңызды емес айнымалыларды алып тастайды және компьютер түсінетіндей дәлдікпен логикалық құрылым жасайды [1, 33-35, б.].

Бағдарламалаудың бала психикасына және когнитивтік дамуына тигізетін әсерін зерттеуде Массачусетс технологиялық институтының көрнекті ғалымы Сеймур Папперттің еңбектеріне сүйенбей өту мүмкін емес. Паперт Жан Пиаженің когнитивтік даму теориясын негізге ала отырып, білім берудегі конструктивизм бағытын жаңа деңгейге көтерді. Ол өзінің зерттеулерінде бағдарламалауды жай ғана пән емес, баланың ойлау процестерін сыртқа шығарып, оны өзіне көрсетуге мүмкіндік беретін «айна» ретінде сипаттады. Папперттің теориясы бойынша, оқушы Python ортасында код жазған кезде, ол дайын ақпаратты пассивті тұтынушыдан белсенді «микроәлемді» құрушыға айналады. Бұл микроәлемде оқушы математикалық немесе физикалық заңдылықтарды жаттап алмайды, ол сол заңдылықтардың қалай жұмыс істейтінін код арқылы өз қолымен модельдейді. Мұндай тәжірибе оқушының Пиаже айтқан «нақты операциялар» сатысынан «формальды-логикалық операциялар» сатысына өтуін күрт жылдамдатады. Абстрактілі ұғымдар (мысалы, цикл, функция, рекурсия) компьютер экранында нақты нәтижеге айналғанда, оқушының санасында жаңа нейрондық байланыстар беки түседі. Ол өзінің жіберген қателігін экраннан көріп, оның себебін түсінген сәтте метатанымдық (өз ойы туралы ойлау) процестер іске қосылады.

Сонымен қатар, Python тіліндегі есеп шығару үдерісін Лев Выготскийдің әлеуметтік-мәдени теориясы, дәлірек айтсақ, «жақын арадағы даму аймағы» (Zone of Proximal Development) ұғымы

тұрғысынан қарастыру өте маңызды. Дәстүрлі педагогикада балаға қиын тапсырманы орындау үшін үлкендердің немесе біліктірек құрдастарының көмегі (скаффолдинг) қажет екені айтылады. Алайда, бағдарламалау ортасында бұл рөлді белгілі бір деңгейде Python интерпретаторының өзі атқарады. Оқушы қате код жазғанда, бағдарлама оны сөкпейді, бағасын төмендетпейді, тек қатенің түрін (мысалы, SyntaxError немесе TypeError) және оның қай жолда орналасқанын объективті түрде көрсетеді. Бұл эмоционалдық тұрғыдан бейтарап кері байланыс оқушының аффективті сүзгісін (affective filter) төмендетіп, қателік жасаудан қорқу сезімін жояды. Қателік – сәтсіздік емес, зерттеу мен жөндеудің (debugging) бастапқы нүктесі ретінде қабылдана бастайды. Выготскийдің теориясына сәйкес, сыртқы ортамен (бұл жағдайда бағдарламалау интерфейсімен) болған мұндай диалог біртіндеп оқушының ішкі психологиялық құрылымына еніп, оның дербес сыни талдау жасау қабілетіне айналады. Оқушы компьютермен интерактивті қарым-қатынас жасай отырып, мәселені өз бетінше шешуге қабілетті тұлға болып қалыптасады.

Неліктен басқа тілдер емес, дәл Python осындай когнитивтік трансформацияның басты құралы ретінде таңдалып отыр? Бұл сұрақтың жауабын австралиялық психолог Джон Свеллердің «Когнитивтік жүктеме теориясынан» (Cognitive Load Theory) табуға болады. Кез келген жаңа ақпаратты игеру кезінде адамның жұмыс жады шектеулі ресурсқа ие. Егер бағдарламалау тілінің синтаксисі тым күрделі болса (мысалы, C++ тіліндегі жадты қолмен басқару немесе міндетті түрдегі типтеу сияқты), оқушының миындағы когнитивтік қуаттың басым бөлігі есептің мағынасын шешуге емес, жақшалардың дұрыс қойылуын немесе нүкте-үтірдің ұмытылмауын қадағалауға, яғни «бөгде жүктемеге» (extraneous cognitive load) жұмсалады. Ал Python тілі семантикалық тұрғыдан адамның табиғи ойлау логикасына барынша бейімделген. Оның оқылуға жеңіл (readability) құрылымы, қатаң шегіністері (indentation) және ағылшын тілінің қарапайым сөздерін (if, for, in, while) оператор ретінде қолдануы когнитивтік кедергілерді жояды. Оқушы техникалық бөлшектерге алаңдамай, барлық интеллектуалды ресурсын тікелей мәселенің алгоритмдік шешімін табуға, яғни «орынды жүктемеге» (germane cognitive load) бағыттайды. Бұл еркіндік оқушыға бір мәселенің бірнеше балама шешімдерін іздеуге, олардың тиімділігін салыстыруға және ең оңтайлысын таңдауға мүмкіндік береді, бұл өз кезегінде ғылыми зерттеушілік ойлаудың көрінісі [3, 257-285 б].

Осылайша, бағдарламалау сабағындағы Python ортасы жай ғана код жазуға арналған экран емес, ол баланың танымдық процестері шыңдалатын, теориялық білім практикалық дағдыға ұласатын интеллектуалды шеберхана болып табылады. Мәселені түсіну, оны бөлшектеу, заңдылықтарды табу және оны шешудің логикалық жобасын жасау – бұл қадамдар оқушының санасында жаңа когнитивтік архитектураны қалыптастырады.

Алгоритмдік ойлаудың ішкі құрылымын педагогикалық тұрғыдан талдайтын болсақ, оның теориялық негіздері практикалық деңгейде төрт ірі когнитивтік компонентке ыдырайды, ал олардың әрқайсысы Python тілінің нақты синтаксистік құралдары арқылы жүзеге асады. Оқушының танымдық эволюциясы осы төрт сатыдан кезекпен немесе қатар өту арқылы қалыптасады. Бұл процестің ең алғашқы әрі ең күрделі сатысы – декомпозиция, яғни ауқымды және көп белгісізі бар мәселені логикалық тұрғыдан оқшауланған, шешілуі жеңіл кішігірім бөлшектерге бөлу өнері. Дәстүрлі оқыту жүйесінде оқушы үлкен тапсырманы көргенде көбінесе когнитивтік тоқырауға (ступорға) ұшырайды, өйткені оның жұмыс жады бірден бүкіл мәселені қамти алмайды. Бағдарламалау сабағында бұл психологиялық тосқауыл Python тілінің функциялар (def) және модульдік архитектурасы арқылы сәтті еңсеріледі. Оқушыға, мәселен, мәтін ішіндегі ең жиі кездесетін сөздерді тауып, олардың жиілігін графикте көрсететін бағдарлама жазу тапсырылды делік. Сыни ойлауға дағдыланған оқушы бұл есепті тұтастай жазуға тырыспайды, ол мәселені қадамдарға бөлшектейді: мәтінді оқу функциясы, тыныс белгілерінен тазарту функциясы, сөздерді санау функциясы және нәтижені визуалдау функциясы. Әрбір жазылған функция – бұл оқушының миындағы белгілі бір шешілген микро-мәселе. Функцияны бір рет дұрыс жазып, оны тексергеннен кейін, оқушы оны жай ғана «қара жәшік» (black box) ретінде келесі қадамдарда қолдана береді, оның ішкі механикасына қайта бас қатырмайды. Бұл үдеріс оқушыны инженериядағы абстракциялық деңгейлерді басқаруға, жобаны жүйелі түрде жоспарлауға және күрделілікті басқаруға (complexity management) үйретеді. Декомпозиция арқылы оқушы кез келген өмірлік немесе ғылыми проблеманың шешілмейтін түйін емес, тек рет-ретімен орындалуы тиіс шағын тапсырмалар жиынтығы екенін түсінеді.

Осы үдеріспен органикалық түрде сабақтасатын келесі танымдық акт – абстракциялау. Егер декомпозиция мәселені бөлшектесе, абстракция сол бөлшектердің ішінен тек ең маңыздыларын іріктеп алып, есептің логикасына тікелей әсер етпейтін «шуды» (noise) немесе екінші кезектегі

детальдарды елемеуді көздейді. Психологиялық тұрғыдан абстракциялау – адам миының ең жоғары қызметі, өйткені ол нақтылықтан алшақтап, құбылыстың идеалды моделін жасауды талап етеді. Python тілі жоғары деңгейлі (high-level) бағдарламалау құралы ретінде оқушыға абстракцияның бай экожүйесін ұсынады. Айнымалыларды (variables) атаудан бастап, деректер құрылымдарын (мәселен, list, dictionary, tuple, set) таңдауға дейінгі әрбір қадам – таза аналитикалық шешім. Мысалы, мектеп журналындағы оқушылардың бағаларын бағдарламада сақтау қажет болғанда, оқушы оны жай ғана тізім (list) арқылы емес, әрбір оқушының аты-жөнін оның бағаларымен байланыстыратын сөздік (dictionary) арқылы модельдеудің әлдеқайда тиімді екенін түсінеді. Бұл жерде оқушы физикалық нысанның (қағаз журналдың) цифрлық абстракциясын жасайды. Python-дағы объективті-бағдарланған бағдарламалау (ООР) элементтерін, яғни кластар (class) мен объектілерді (object) өткен кезде оқушының абстрактілі ойлауы шыңына жетеді. Ол нақты нысандардың қасиеттері (атрибуттары) мен іс-әрекеттерін (әдістерін) жалпылап, олардың әмбебап қаңқасын құрастырады. Осындай тәжірибе арқылы оқушы айналасындағы әлемді жекелеген заттардың хаосы емес, белгілі бір қасиеттер мен функцияларға ие абстрактілі модельдердің жиынтығы ретінде қабылдауға дағдыланады.

Үшінші маңызды элемент – үлгіні тану (Pattern Recognition), ол ақпарат массивінің ішінен ұқсастықтарды, қайталануларды және ішкі заңдылықтарды анықтауға бағытталған. Адамның табиғи интеллектуалдық қабілеттерінің бірі заңдылықтарды іздеу болса, бағдарламалау бұл процесті саналы түрде басқаруға және оны автоматтандыруға үйретеді. Сыни тұрғыдан ойлайтын оқушы біртектес әрекеттерді қайта-қайта орындаудың (копипаст жасаудың) логикалық тиімсіздігін тез аңғарады. Python тіліндегі for және while циклдары, сондай-ақ итераторлар мен генераторлар оқушыға осы қайталануларды бір ғана қысқа құрылымға сыйдыруға мүмкіндік береді. Мысалы, берілген мыңдаған санның ішінен тек жай сандарды бөліп алу алгоритмін жазу кезінде оқушы әрбір санды жеке тексермейді, ол математикалық бөліну заңдылығын анықтап, оны цикл ішіндегі динамикалық шартқа айналдырады. Үлгіні тану қабілеті тек кодты қысқарту үшін ғана емес, сонымен қатар үлкен деректерді (Big Data) талдау кезінде трендтерді болжау үшін аса қажет. Python тілінде жұмыс істейтін оқушы алгоритмдік заңдылықтарды көре білу арқылы математикалық прогрессиялардың, физикалық тербелістердің немесе биологиялық популяциялар өсімінің ішкі табиғатын тереңірек ұғынады. Оның бойында «интеллектуалдық үнемділік» ұстанымы қалыптасады: барынша аз күш жұмсап, барынша ауқымды деректерді өңдеуге ұмтылу.

Алгоритмдік ойлаудың төртінші компоненті – мәселені шешудің қадамдық нұсқаулығын, яғни алгоритмді жобалау. Бұл кезең алдыңғы үш кезеңнің (декомпозиция, абстракция, үлгіні тану) синтезі болып табылады. Оқушы Python тілінің синтаксисі арқылы өз ойын қатаң логикалық тізбекке салады. Бағдарламалау тілінің қаталдығы сол – ол ешқандай екіұштылықты, түсініксіздікті немесе логикалық секірістерді кешірмейді. Адамдар өзара сөйлескенде контекст арқылы бірін-бірі жарты сөзден түсінсе, компьютерге әрбір тривиальды қадамды дәл түсіндіру қажет. Бұл оқушыны өз ойын барынша анық, құрылымды және аргументті түрде жеткізуге үйретеді. Алгоритмді жобалаудың ең жоғарғы психологиялық көрінісі Python-дағы шартты операторлар (if, elif, else) мен бульдік логиканы (Boolean logic – and, or, not) қолдану кезінде байқалады. Есеп шығару барысында оқушы «Егер пайдаланушы санның орнына әріп терсе не болады?», «Егер тізім бос болып қалса, бағдарлама қалай әрекет етуі тиіс?» деген сияқты балама (альтернативті) сценарийлерді алдын ала болжауға мәжбүр. Мұндай сыни көзқарас оқушыны қарапайым «сызықтық ойлаудан» (linear thinking) күрделі «тармақталған ойлауға» (branching thinking) көшіреді. Ол кез келген шешімнің бірнеше салдары болуы мүмкін екенін, әрбір жағдайдың өзіндік алғышарттары бар екенін түсінеді. Нәтижесінде, алгоритм құру процесі оқушының бойында себеп-салдарлық байланыстарды көре білу, жағдайды жан-жақты сараптау және кез келген тосын жағдайға алдын ала дайын болу (өмірлік resilience) қабілеттерін шыңдайды.

Осы төрт компоненттің Python ортасындағы үздіксіз айналымы оқушының танымдық белсенділігін мүлдем басқа сапалық деңгейге шығарады. Есеп шығару барысында оқушы тек ақпараттық объектілермен жұмыс істеп қана қоймайды, ол өз іс-әрекетінің архитекторына айналады. Компьютер экранындағы жүздеген жолдан тұратын жұмыс істеп тұрған код – бұл жай ғана мәтін емес, бұл оқушының декомпозицияланған, абстракцияланған, заңдылықтары табылған және қатаң логикалық қадамдарға негізделген ойлау процесінің материалдық көрінісі. Бұл көрініс оқушыға өзінің интеллектуалдық мүмкіндіктеріне деген зор сенімділік ұялатады және оның ішкі мотивациясын (intrinsic motivation) оятып, сыртқы баға үшін емес, мәселенің табиғатын тану үшін оқуға ынталандырады. Бұл тұрғыда бағдарламалау сабағының философиясы дәстүрлі жаттау

жүйесіне толықтай қарама-қайшы келетін, нағыз зияткер-ізденушіні қалыптастыратын заманауи білім берудің алтын діңгегіне айналатыны сөзсіз.

Оқушының бағдарламалау барысындағы интеллектуалдық триумфы, яғни жазған кодының алғаш рет жүйелі түрде жұмыс істеп, күтілген нәтижені экранға шығаруы, әдетте ұзаққа созылатын күрделі ізденіс жолының тек соңғы нүктесі ғана болып табылады. Шын мәнінде, танымдық дамудың ең үлкен секірісі бағдарлама мінсіз орындалған кезде емес, керісінше, алгоритм істен шыққанда, интерпретатор қызыл түспен қателік туралы хабарлама (traceback) берген сын сағаттарда жүзеге асады. Педагогикалық психология тұрғысынан алғанда, дәстүрлі білім беру ортасындағы «қателік» ұғымы көбінесе оқушының білімсіздігін немесе сәтсіздігін білдіретін жағымсыз индикатор ретінде қабылданады, бұл баланың бойында қорқыныш пен жалтақтықты тудырады. Алайда, Python ортасындағы қателіктермен жұмыс (debugging) үдерісі бұл деструктивті парадигманы толықтай бұзады. Бағдарламалаудағы қате – бұл айыптау үкімі емес, ол ақпараттық жүйенің нақты және объективті кері байланысы, диалогтың жаңа формасы. Оқушы қате кодымен бетпе-бет келгенде, ол амалсыздан когнитивтік диссонанс күйін кешеді: оның ішкі логикалық жорамалы мен компьютердің объективті орындалу нәтижесі бір-біріне қайшы келеді. Осы қарама-қайшылықты шешу процесі нағыз сыни талдаудың, яғни рефлексияның генераторына айналады. Оқушы «Менің ойлау тізбегімде қай буын үзілді?», «Компьютер менің нұсқауымды неліктен мен күткендей емес, дәл осылай түсінді?» деген сұрақтарды өзіне қоя отырып, өз санасындағы логикалық құрылымдарды сырттай бақылап, оларға тәуелсіз сараптама жасай бастайды.

Дәл осы дебаггинг кезеңінде оқушының бойында әлемнің ғылыми бейнесін қалыптастыратын және оның жалпы ойлау стилін түбегейлі өзгертетін іргелі дағдылар бекиді. Алгоритмдік логика бұл тұста жай ғана техникалық құрал емес, ол теориялық танымда нақты әдіснамалық қызмет атқарады. Ғылыми танымның классикалық әдіснамасы «гипотеза – эксперимент – бақылау – қорытынды» тізбегіне негізделсе, Python тілінде қатені іздеу процесі осы тізбектің кішірейтілген, бірақ өте жылдам (интерактивті) моделі болып табылады. Оқушы SyntaxError, TypeError немесе одан да күрделі логикалық қателерді (LogicError) кездестіргенде, ол әрбір айнымалының мәнін қадамдап тексеру (tracing) арқылы зерттеушіге айналады. Ол «Егер мен мына циклдің шекарасын өзгертсем, нәтиже дұрыстала ма?» деген гипотеза құрады, оны кодқа енгізіп эксперимент жасайды, интерпретатордың жауабын бақылайды және қорытынды шығарады. Бұл эвристикалық цикл оқушыны догматикалық ойлаудан арылтып, кез келген тұжырымды тек эмпирикалық дәлелдер негізінде ғана қабылдауға, яғни таза ғылыми скептицизмге баулиды. Оның үстіне, Python-дағы ерекше жағдайларды өңдеу (try-exceptблоктары) механизмдерін меңгеру оқушыға жүйенің осал тұстарын алдын ала болжауды, апаттық жағдайлардың алдын алуды және белгісіздік жағдайында жүйенің тұрақтылығын сақтап қалуды үйретеді. Бұл – қазіргі құбылмалы әлемде аса қажет болатын стресске төзімділіктің (resilience) интеллектуалдық баламасы.

Бағдарлама сәтті жұмыс істеп, қателер толық жойылғаннан кейін, оқушының танымдық эволюциясы тоқтап қалмайды, ол сыни бағалаудың (evaluation) келесі, одан да жоғары сатысына көтеріледі. Бұл саты кодты рефакторингтеу (refactoring) және оңтайландыру (optimization) деп аталады. Жұмыс істеп тұрған алгоритмді жай ғана қанағат тұтпай, оқушы өзіне «Бұл шешімді бұдан да тиімді, талғампаз (elegant) және ресурстарды үнемдейтіндей етіп қалай қайта жазуға болады?» деген сұрақ қояды. Блум таксономиясы бойынша бұл деңгей анализден асып, синтез бен бағалауға сәйкес келеді. Осы тұста оқушы алгоритмдердің уақыттық және кеңістіктік күрделілігі (Time and Space Complexity – Big O notation) туралы бастапқы базалық түсініктермен қаруланады. Ол жүздеген жолдан тұратын қайталанбалы ішкі циклдардың компьютердің процессорына және жедел жадына қаншалықты ауыр жүк түсіретінін саналы түрде есептей бастайды. Python тілінің бай кітапханалары мен кіріктірілген функциялары (built-in functions) оқушыға бір мәселенің бірнеше балама жолдарын ұсынады. Мысалы, тізім элементтерін сұрыптауды қарапайым «көпіршік әдісімен» (Bubble Sort) де, кіріктірілген sort()әдісімен де жасауға болады. Мұғалімнің бағыттауымен осы екі әдістің жылдамдығын салыстыра отырып, оқушы функционалдық сауаттылықтың жаңа қырын – ресурстарды үнемдеу және шешімнің архитектуралық эстетикасын түсінеді. Оқушы үшін жақсы бағдарлама – тек дұрыс жауап беретін бағдарлама емес, ол логикалық жағынан мөлдір, артық операциялардан тазартылған және басқа адам оқуға жеңіл (readable) интеллектуалды өнімге айналады. Бұл ұмтылыс оқушының бойында кәсіби этика мен эстетикалық талғамды қалыптастырады.

Алгоритмдік ойлаудың дамуындағы тағы бір қуатты катализатор – ұжымдық код жазу (pair programming) және өзгенің кодын оқып, оған рецензия жасау (code review) үдерісі. Бағдарламалау –

сырттай қарағанда жалғыз адамның компьютермен оңаша жұмысы болып көрінгенімен, шын мәнінде ол терең әлеуметтік-когнитивтік процесс. Оқушы сыныптасының жазған кодын талдау барысында өзінің қалыптасқан стереотиптерінен шығып, басқа адамның ойлау стиліне енуге мәжбүр болады. Бұл «когнитивтік эмпатия» деп аталатын өте күрделі психологиялық дағдыны қалыптастырады. Оқушы бір мәселені шешудің жалғыз ғана «дұрыс» жолы жоқ екенін, әрбір алгоритмнің өзіндік артықшылықтары мен кемшіліктері бар екенін іс жүзінде көреді. Біреудің шешімі жадты үнемдесе, екіншісінің шешімі процессор уақытын үнемдейді, ал үшіншісінікі адамға түсініктірек жазылған. Осы балама пікірлер мен логикалық құрылымдарды салыстыра отырып, оқушы өзгенің интеллектуалдық еңбегін объективті, аргументті түрде сынауға және сол сынды қабылдауға үйренеді. Бұл тәжірибе догматикалық ойлаудың кез келген формасын жойып, тұлғаның интеллектуалдық икемділігін (intellectual flexibility) барынша кеңейтеді. Өзгенің қатесін табу немесе өз кодының осал тұсын мойындау арқылы оқушылар арасында сау бәсекелестік пен ғылыми ынтымақтастық мәдениеті орнығады.

Жоғарыда сипатталған дебаггинг, рефакторинг және кодты өзара талдау процестерінің жиынтығы оқушыны бірте-бірте тұтынушылық психологиядан жасампаздық психологиясына өткізеді. Ол цифрлық әлемнің дайын қосымшаларын (бағдарламаларын) пайдаланушы ғана емес, сол қосымшалардың ішкі механикасын түсінетін, қажет болса оны өзгерте алатын және жаңасын жасай алатын «цифрлық агентке» (digital agent) айналады. Python ортасы оқушы үшін қауіпсіз, шексіз мүмкіндіктері бар виртуалды зертхана қызметін атқарады, мұнда құлап қалудан немесе материалдық шығынға ұшыраудан қорықпай, ең батыл интеллектуалдық эксперименттерді жүргізуге болады. Бағдарламалау процесіндегі әрбір сәтсіздік пен оны жеңу жолындағы логикалық күрес оқушының санасында жаңа когнитивтік үлгілерді (паттерндерді) бекітіп, оны кез келген ақпараттық хаос ішінен жүйе құруға қабілетті, сыни көзқарасы тұрақты тұлға ретінде шыңдай түседі. Бұл қабілеттер жиынтығы оқушының тек информатика пәніндегі үлгерімін ғана емес, оның жалпы академиялық потенциалын, логика-математикалық пайымдауын және өмірлік қиындықтарды парасаттылықпен еңсеру әлеуетін айтарлықтай жоғарылатады.

Оқушы мәселені қадамдарға бөлшектейді. Төмендегі мысалда мәтінді өңдеу есебінің бірнеше шағын функцияға бөлінуі көрсетілген:

```
#  1. Мәтінді тазалау (артық таңбаларды алып тастау)
def clean_text(text):
    symbols = ".,!?"
    for s in symbols:
        text = text.replace(s, "")
    return text.lower()
#  2. Сөздердің жиілігін есептеу (Pattern Recognition)
def get_word_freq(text):
    words = text.split()
    freq = {}
    for word in words:
        freq[word] = freq.get(word, 0) + 1
    return freq
#  Негізгі алгоритм (Декомпозиция нәтижесі)
input_text = "Python - болашақ тілі. Python өте қарапайым!"
cleaned = clean_text(input_text)
result = get_word_freq(cleaned)
print(result) # Шығыс: {'python': 2, 'болашақ': 1, 'тілі': 1, ...}
```

Бұл мысалда оқушы мәселені шешу үшін алдымен деректерді дайындау (clean_text) және оны өңдеу (get_word_freq) кезеңдерін жеке қарастырады. Бұл оның жүйелі жоспарлау және логикалық блоктармен жұмыс істеу дағдысын қалыптастырады.

Оқушының бағдарламалау ортасындағы осындай цифрлық субъектілікке (agency) ие болуы оның интеллектуалдық әлеуетін тек информатика кабинетінің қабырғасымен шектеп қоймайды, керісінше, оны жаратылыстану және гуманитарлық ғылымдардың кеңістігіне алып шығады. Алгоритмдік және сыни ойлаудың нағыз прагматикалық құндылығы олардың пәнаралық (интеграциялық) табиғатында жатыр. Қазіргі заманауи білім беру парадигмасында STEM (ғылым, технология, инженерия және математика) бағытының күшеюімен бірге, Python тілі осы пәндердің

басын біріктіретін әмбебап дәнекерге (framework) айналды. Мәселен, математика немесе физика сабақтарындағы күрделі формулалар мен құбылыстарды оқушылар көбінесе статикалық түрде, тақтадағы сандар мен ережелер жиынтығы ретінде ғана қабылдайды, бұл олардың абстрактілі ұғымдарды терең түсінуіне кедергі келтіреді. Алайда, оқушы сол математикалық теңдеуді немесе физикалық заңдылықты Python ортасында бағдарламалық кодқа айналдырған сәтте, статикалық ақпарат динамикалық модельге ұласады. Кинематикадағы дененің еркін түсу үдеуін немесе химиялық реакциялардың жылдамдығын модельдеу барысында оқушы жай ғана дайын жауапты есептеп қоймайды, ол «Егер мен бастапқы жылдамдықты немесе ауа кедергісін өзгертсем, траектория қалай құбылады?» деген эвристикалық сұрақ қоя отырып, нағыз ғылыми эксперимент жүргізеді. Бұл үдеріс абстрактілі ғылыми концепцияларды көзге көрінетін, басқаруға келетін және болжауға болатын физикалық шындыққа жақындатады. Бұрын тек қағаз бетінде өмір сүрген заңдылықтар компьютер экранында симуляцияланғанда, оқушының танымдық көкжиегі кеңейіп, ол кез келген ғылыми құбылыстың ішкі алгоритмдік табиғатын (architecture) тани бастайды.

Осы пәнаралық байланыстың ең жоғарғы аналитикалық деңгейі деректер ғылымымен (Data Science) жұмыс істеу кезінде көрініс табады. Бүгінгі ақпараттық қоғамда мәліметтердің ауқымды көлемі (Big Data) адамзаттың басты ресурсына айналғаны белгілі. Оқушыларға Python тілінің бай экожүйесін, соның ішінде деректерді өңдеуге және визуалдауға арналған кітапханаларды (мысалы, pandas, matplotlib немесе seaborn) меңгерту арқылы біз олардың бойында дәлелдерге негізделген пайымдау (evidence-based reasoning) мәдениетін қалыптастырамыз. Оқушыларға нақты шынайы мәліметтерді – мысалы, аймақтағы экологиялық өзгерістерді, демографиялық статистиканы немесе жаһандық жылыну динамикасын – талдау тапсырылғанда, олар құрғақ сандар массивінің артындағы әлеуметтік немесе табиғи тенденцияларды көруге дағдыланады. Мыңдаған жолдан тұратын кестені код арқылы тазартып (data cleaning), оны гистограмма немесе графиктік үлгіге айналдырған оқушы, ақпаратты жай ғана қабылдаушыдан оны интерпретациялаушы (сарапшы) рөліне ауысады. Сыни ойлау бұл жерде ақпараттың манипуляциясына берілмей, тек объективті статистикалық корреляцияларға сүйеніп шешім шығару қабілеті ретінде көрінеді. Оқушы «Бұл график нені көрсетеді?», «Ауытқулар (outliers) қандай факторлардың әсерінен туындады?», «Бұл деректер негізінде қандай болжам жасауға болады?» деген сұрақтар арқылы таза ғылыми-зерттеушілік рефлексияны басынан кешіреді. Бұл – мектеп қабырғасында қалыптасатын ең жоғары функционалдық сауаттылық, себебі ол оқушыны өмірлік жағдаяттарда интуицияға емес, нақты деректер мен логикаға сүйенуге тәрбиелейді.

Оқушының осындай аналитикалық және алгоритмдік дағдыларын бір арнаға тоғыстыратын кешенді дидактикалық тәсіл – жобалық оқыту (Project-Based Learning). Жобалық жұмыс бағдарламалауды жекелеген операторлар мен синтаксистік ережелерді жаттау деңгейінен шығарып, оны жүйелі инженерия деңгейіне көтереді. Оқушыға белгілі бір қолданбалы мәселені шешуге бағытталған жоба (мысалы, мектеп асханасындағы кезекті оңтайландыратын электронды қосымша, физикалық есептер калькуляторы немесе шағын викториналық ойын) берілгенде, ол алгоритмдік ойлаудың барлық сатыларын (декомпозиция, абстракция, үлгіні тану, дебаггинг) толықтай өз бетінше басқаруға мәжбүр болады. Жобаны жоспарлау кезеңінде оқушы уақыт пен ресурстарды бағалайды, интерфейс (UX/UI) дизайнын ойластырады және логикалық архитектураны құрады. Бұл үдеріс оның «жүйелі ойлауын» (systems thinking) шыңдайды, өйткені жобадағы кез келген шағын өзгеріс тұтас жүйенің жұмысына әсер етеді. Оқушы бағдарламалаудың тек математикалық логика емес, сонымен бірге тұтынушының (пайдаланушының) психологиясын түсінуді талап ететін гуманитарлық өлшемі бар екенін аңғарады. «Бұл бағдарлама пайдаланушы үшін қаншалықты ыңғайлы?», «Ол алға қойған мәселені толық шеше ала ма?» деген сұрақтар оқушының бойында әлеуметтік жауапкершілік пен эмпатияны оятады. Жобалық оқыту барысында оқушылар топпен жұмыс істеуге (collaboration), міндеттерді бөлісуге және өзгенің жазған кодымен интеграция жасауға үйренеді, бұл заманауи кәсіби ортада (Agile/Scrum форматтарында) талап етілетін ең басты «жұмсақ дағдылардың» (soft skills) қайнар көзі болып табылады.

Осы тұста алгоритмдік ойлау мен лингвистикалық қабілеттердің арасындағы нейрокогнитивтік байланысты да ерекше атап өту қажет. Python тілі табиғи ағылшын тілінің грамматикасы мен математикалық логиканың түйіскен нүктесінде орналасқандықтан, ол оқушының ауызша және жазбаша сөйлеу мәдениетіндегі дәлдік пен құрылымдылықты (precision of thought) арттырады. Бағдарламалау тілінің қатаң ережелері оқушыны екіұштылықтан (ambiguity) аулақ болуға, әрбір терминді өз орнында дәл қолдануға және күрделі ойды мейлінше қысқа, түсінікті формада жеткізуге баулиды. Код жазу барысындағы аргументация мен шартты операторларды құру логикасы оқушының күнделікті өмірдегі дебаттық немесе эссе жазу дағдыларына тікелей

трансферттеледі (көшеді). Себебі, компьютерге нұсқау беру үшін алдымен өз ойыңды мінсіз ретке келтіруің керек. Психолингвистикалық зерттеулер көрсеткендей, бағдарламалауды терең меңгерген оқушылар мәтінді оқығанда оның негізгі идеясын тез бөліп алады (абстракция) және мәтіндегі логикалық қателер мен манипуляцияларды (дебаггинг) оңай таниды. Яғни, бағдарламалауды оқыту арқылы біз шын мәнінде оқушының жалпы ақпараттық иммунитетін күшейтеміз.

Сонымен қатар, Python тілін меңгерудің қазіргі жасанды интеллект (AI) және машиналық оқыту (Machine Learning) құралдары қарқынды дамып жатқан тарихи кезеңдегі маңызы өте терең. Генеративті нейрожелілер (LLM) қарапайым бағдарламалық кодтарды адамнан да жылдам жаза алатын деңгейге жеткен бүгінгі таңда, кейбір сарапшылар «бағдарламалауды үйренудің қажеті бар ма?» деген скептикалық сұрақ қояды. Алайда, ғылыми педагогикалық тұрғыдан алғанда, жасанды интеллект дәуірінде алгоритмдік ойлаудың құндылығы төмендемейді, керісінше, экспоненциалды түрде артады. Өйткені машина кодты генерациялағанымен, сол кодтың архитектурасын құру, оған дұрыс логикалық тапсырма (prompt) беру және ең бастысы – машина шығарған нәтиженің қауіпсіздігін, этикалық дұрыстығын және тиімділігін сыни тұрғыдан бағалау (validation) тек адамның, нақтырақ айтқанда, алгоритмдік ойлауы дамыған адамның ғана қолынан келеді. Python тілінің логикасын іштей түсінбейтін адам жасанды интеллект үшін тек пассивті тұтынушы болып қала береді және технологиялық «қара жәшіктің» (black box) алдында дәрменсіздік танытады. Ал бағдарламалаудың іргетасын өз қолымен қалаған, декомпозиция мен рефакторингтің азабын сезінген оқушы алгоритмдердің табиғатын, шектеулері мен қауіптерін терең түсінеді. Ол AI құралдарын сиқыр немесе абсолютті ақиқат ретінде емес, өзінің интеллектуалдық кеңейтілімі (cognitive extension) ретінде қабылдайды. Осылайша, мектеп қабырғасында алгоритмдік ойлауды қалыптастыру – бұл оқушыны болашақтағы автоматтандыру толқынынан (технологиялық жұмыссыздықтан) қорғайтын ең сенімді интеллектуалдық қалқан болып табылады. Адамның машинадан басты артықшылығы оның код жазу жылдамдығында емес, сол кодтың мағынасын, мақсатын және салдарын философиялық әрі сыни тұрғыдан бағамдай алуында екені даусыз. Бұл – білім беру жүйесінің технологиялық детерминизмге беретін ең мықты антропологиялық жауабы.

Технологиялық детерминизмге қарсы тұра алатын осындай зияткерлік қалқанды мектеп қабырғасында қалыптастыру, өз кезегінде, педагогикалық процестің басты фигурасы – мұғалімнің рөлін және жалпы оқыту әдіснамасын түбегейлі қайта қарауды талап етеді. Дәстүрлі парадигмада мұғалім ақиқаттың жалғыз иесі әрі дайын білімді трансляциялаушы ретінде қабылданса, бағдарламалау сабағында, әсіресе сыни ойлауды дамыту көзделген ортада, бұл модель толықтай күйрейді. Python тілін оқыту барысында мұғалім «ақпарат беруші» (information provider) функциясынан бас тартып, «когнитивтік тәлімгер» немесе «зияткерлік фасилитатор» (cognitive scaffolder) деңгейіне өтуі шарт. Оқушы алгоритм құру кезінде тығырыққа тірелгенде немесе оның коды қате бергенде, мұғалімнің міндеті – дайын шешімді көрсету немесе қатені өз қолымен түзеп беру емес. Керісінше, бұл сәт Сократтық диалогты іске қосатын ең құнды педагогикалық мүмкіндікке айналуы тиіс. «Бұл айнымалыға қандай мән бердің?», «Осы циклдің тоқтау шарты қандай?», «Егер біз деректерді тізімге емес, сөздікке салсақ, бағдарламаның жылдамдығы қалай өзгереді?» деген сияқты ашық әрі бағыттаушы сұрақтар оқушының метакогнитивтік рефлексиясын оятады. Мұғалімнің мұндай ұстанымы оқушыға мәселенің шешімін сырттан күтпей, оны өз ішінен, өз логикасынан іздеуге үйретеді. Бұл үдеріс оқушының бойында академиялық дербестікті (academic autonomy) және өзінің интеллектуалдық күшіне деген сенімділікті қалыптастырады, нәтижесінде бағдарламалау сабағы таза техникалық нұсқаулықтарды орындаудан интеллектуалдық ізденіс пен шығармашылық еркіндік алаңына айналады.

Мұқым оқу үдерісінің осылайша өзгеруі оқушылардың жетістіктерін бағалау жүйесін де трансформациялауды қажет етеді. Егер біздің түпкі мақсатымыз жай ғана Python синтаксисін жаттату емес, алгоритмдік және сыни ойлауды дамыту болса, онда оқушының білімін дәстүрлі жабық тесттермен немесе жаттанды бақылау жұмыстарымен өлшеу мүлдем тиімсіз болып табылады. Тест сұрақтары оқушының жадында сақталған статикалық ақпаратты ғана тексере алады, ал сыни ойлау – бұл динамикалық процесс. Сондықтан бағдарламалауды оқытуда критериалды және формативті бағалаудың эвристикалық әдістеріне, соның ішінде кодтық портфолиоларға, жобаны қорғауға және рефлексивтік эсселерге басымдық берілуі керек. Бағалау кезінде басты назар бағдарламаның соңғы компиляциядан өткен-өтпегеніне емес, сол нәтижеге жету жолындағы оқушының логикалық қадамдарына, оның балама шешімдерді қалай қарастырғанына және туындаған қателіктерді (багтарды) қалай талдағанына аударылуы тиіс. Педагогикалық психологияда бұл тәсіл «өнімді сәтсіздік» (productive failure) деп аталады. Оқушы мінсіз, бірақ біреуден көшіріп алған немесе жаттап алған кодты тапсырғаннан гөрі, көптеген қателіктермен

күресіп, оларды түзетудің алгоритмін түсіндіріп бере алатын, толық аяқталмаған жобаны ұсынса, оның танымдық құндылығы әлдеқайда жоғары бағалануы қажет. Оқушының өз алгоритмін аудитория алдында қорғауы, оның тиімділігін аргументтермен дәлелдеуі және басқалардың сыни сұрақтарына жауап беруі оның коммуникативтік және аналитикалық дағдыларын біртұтас жүйеде шыңдайды. Осындай көпөлшемді бағалау жүйесі ғана оқушының шынайы когнитивтік өсімін объективті түрде көрсете алады.

Алгоритмдік және сыни ойлаудың мектеп бағдарламасындағы орнын пайымдауда біз айналып өте алмайтын тағы бір өте маңызды парадигма бар – ол бағдарламалаудың этикалық және әлеуметтік өлшемі. Ұзақ уақыт бойы информатика пәні қоғамдық процестерден оқшауланған, таза математикалық және абстрактілі сала ретінде қарастырылып келді. Алайда цифрлық технологиялар адам өмірінің барлық аспектілеріне (медицина, қаржы, құқық қорғау, медиа) тікелей араласып жатқан қазіргі кезеңде код жазу ешқашан бейтарап (neutral) процесс бола алмайды. Әрбір жазылған алгоритмнің артында белгілі бір құндылықтар, таңдаулар және әлеуметтік салдарлар жатыр. Python тілінде жасанды интеллект немесе деректерді талдау негіздерін оқып жатқан оқушы тек техникалық операторларды ғана емес, сонымен бірге өзі құрастырып жатқан жүйенің қоғамға қандай әсер ететінін сыни тұрғыдан бағалауды да үйренуі тиіс. Мәселен, бет-әлпетті танитын (facial recognition) алгоритмнің немесе несие беруші автоматтандырылған жүйенің моделін жасау барысында оқушы «Бұл алгоритм қандай деректер негізінде оқытылды?», «Онда алгоритмдік біржақтылық (algorithmic bias) немесе дискриминация жоқ па?», «Пайдаланушылардың жеке құпиялылығы (privacy) қалай қорғалады?» деген іргелі философиялық және этикалық сұрақтарға жауап іздеуі керек. Сыни ойлау осы тұста технологиялық құралдардың әлеуетін ғана емес, олардың моральдық шекараларын да айқындауға көмектеседі. Бағдарламалауды оқыту барысында цифрлық этика мәселелерін интеграциялау оқушыны жай ғана білікті «кодер» (орындаушы) емес, жаһандық мәселелерге бейжай қарамайтын, өз іс-әрекетінің адамзатқа тигізер әсерін терең сезінетін саналы цифрлық азамат (digital citizen) етіп тәрбиелейді. Бұл – алгоритмдік ойлаудың ең жоғарғы, гуманистік шыңы.

Осы ауқымды зерттеуімізді тұжырымдай келе, Python тілінде есеп шығару үдерісі мектептегі білім беру жүйесінің тамырына қан жүгіртетін, оқушының зияткерлік әлеуетін тұтастай қайта құрастыратын қуатты педагогикалық механизм екенін толық сеніммен айтуға болады. Жаннет Уингтің алгоритмдік ойлау концепциясы, Сеймур Паперттің конструкционистік философиясы, Выготскийдің әлеуметтік-мәдени даму теориясы және Джон Свеллердің когнитивтік жүктеме ілімі – осылардың барлығы Python-ның қарапайым да тұңғиық синтаксисінде тоғысып, бірегей танымдық экожүйені құрайды. Бұл экожүйеде оқушы үлкенді кішіге бөлшектеуді (декомпозиция), шуылдан мағынаны іріктеуді (абстракция), хаостан заңдылық табуды (үлгіні тану) және қателіктермен диалогқа түсіп, оларды жеңуді (дебаггинг) үйренеді. Бұл дағдылардың жиынтығы оқушының ойлауын сызықтық, догматикалық қалыптан шығарып, оны икемді, аналитикалық және жасампаз деңгейге көтереді. Бағдарламалау сабағындағы әрбір сәтті оңтайландырылған алгоритм, әрбір табылған және түзетілген логикалық қате – бұл оқушының нейрондық желісінде қашалған жаңа интеллектуалдық соқпақ. Қазіргі құбылмалы, белгісіздікке толы (VUCA) әлемде адам баласына тіршілік ету үшін нақты фактілердің энциклопедиялық жиынтығы емес, сол фактілерді жылдам өңдеуге, олардың арасындағы байланысты көруге және беймәлім жағдайларда оңтайлы шешім қабылдауға мүмкіндік беретін алгоритмдік және сыни ойлау матрицасы қажет. Python тілі дәл осы матрицаны қалыптастыратын әмбебап тіл болып табылады. Біз оқушыларға бағдарламалауды үйрету арқылы олардың болашақта міндетті түрде IT маманы болуын көздемейміз, біз оларды экономиканың, ғылымның және қоғамның кез келген саласында технологиялық процестерді басқара алатын, ақпараттық тасқында адаспайтын және ең бастысы, өздігінен ойлау еркіндігін сақтап қалатын зияткер тұлға ретінде дайындаймыз. Осылайша, бағдарламалауды оқытудың дидактикалық әлеуеті мен когнитивтік тереңдігі оны жиырма бірінші ғасырдың білім беру парадигмасындағы ең стратегиялық маңызды, әрі тұлғаның интеллектуалдық егемендігін қамтамасыз ететін іргелі бағытқа айналдырады.

Қолданылған әдебиеттер тізімі:

1 Уинг Дж. М. Алгоритмдік ойлау. – Communications of the ACM, 2006, 33-35 б.

2 Паперт С. Ақыл-ой дауылы: Балалар, компьютерлер және қуатты идеялар. – Нью-Йорк: Basic Books, 1980, 230 б.

3 Свеллер Дж. Мәселені шешу кезіндегі когнитивтік жүктеме: Оқуға тигізетін әсері. – Cognitive Science, 1988, 257-285 б.

4 Выготский Л. С. Педагогикалық психология. – Мәскеу: Педагогика, 1991, 480 б.

5 Гровер С., Пи Р. Мектептегі алгоритмдік ойлау: Саланың жай-күйіне шолу. – Educational Researcher, 2013, 38-43 б.

6 Қазақстан Республикасы Оқу-ағарту министрлігі. Негізгі орта білім беру деңгейінің 7-9-сыныптарына арналған «Информатика» пәнінен жаңартылған мазмұндағы үлгілік оқу бағдарламасы. – Астана: Ы. Алтынсарин атындағы ҰБА, 2022, 45 б.

7 Ы. Алтынсарин атындағы Ұлттық білім академиясы. Қазақстан Республикасының орта білім беру ұйымдарында оқу-тәрбие процесін ұйымдастырудың ерекшеліктері туралы әдістемелік нұсқау хат. – Астана: ҰБА, 2023, 315 б.