



<https://doi.org/10.70728/c.series.ped.v08.i08.015>

PYTHON DASTURLASH TILINI O'QITISHDA GRAFLAR NAZARIYASIDAN FOYDALANISH

*Aripov M.M. – Qo'qon davlat universiteti,
Raqamli texnologiyalar va
sun'iy intellekt kafedrası professori, t.f.n.*

Аннотация. Python dasturlash tili bugungi kunda informatika va axborot texnologiyalari yo'nalishlarida dasturlashni o'qitishning asosiy vositalaridan biri hisoblanadi. Biroq, talabalar ko'pincha algoritmlarning ichki tuzilishini, boshqaruv oqimini va dastur bajarilish mantiqini chuqur tushinishda qiyinchiliklarga duch keladilar. Ushbu tadqiqotning maqsadi Python dasturlash tilini o'qitishda graflar nazariyasidan foydalanishning pedagogik samaradorligini aniqlashdir. Tadqiqotda Boshqaruv oqimi grafi (CFG), Call Graph va dasturiy graf modellaridan foydalanishga asoslangan metodika ishlab chiqildi hamda pedagogik eksperiment orqali baholandi. Eksperiment natijalari graf modeli asosidagi o'qitish algoritmik tafakkur, hisoblash tafakkuri va dasturlash kompetensiyalarini sezilarli darajada rivojlantirishini ko'rsatdi.

Keywords: Python dasturlash, graf nazariyasi, boshqaruv o'rnatish grafi, dastur tahlili, dastur vizualizatsiyasi, algoritmik fikrlash, hisoblash tafakkuri.

Аннотация. В статье исследуется эффективность использования теории графов при обучении языку программирования Python. Предложена методика анализа программ Python на основе последовательности Abstract Syntax Tree (AST) → Control Flow Graph (CFG) → Graphviz → NetworkX, позволяющая визуализировать структуру программы, анализировать поток управления и оценивать сложность алгоритмов. Для оценки эффективности разработанной методики был проведён педагогический эксперимент с участием студентов направления «Информатика». Полученные результаты показали, что применение графовой модели способствует развитию алгоритмического мышления, навыков анализа программ и компетенций Computational Thinking. Результаты исследования свидетельствуют о том, что использование Control Flow Graph в процессе обучения Python повышает качество усвоения учебного материала и способствует формированию профессиональных компетенций будущих учителей информатики.

Ключевые слова: Python, теория графов, Control Flow Graph, Graphviz, Анализ программ, Визуализация программ, Алгоритмическое мышление.

Abstract. This study investigates the effectiveness of applying graph theory in teaching the Python programming language. A methodology based on the sequence Abstract

Syntax Tree (AST) → Control Flow Graph (CFG) → Graphviz → NetworkX is proposed for analyzing Python programs, enabling the visualization of program structure, control flow analysis, and algorithm complexity evaluation. To assess the effectiveness of the proposed methodology, a pedagogical experiment was conducted with undergraduate students majoring in Computer Science. The findings demonstrate that the graph-based approach significantly improves algorithmic thinking, program analysis skills, and Computational Thinking competencies. The results indicate that integrating Control Flow Graph into Python programming instruction enhances students' understanding of program execution, supports the development of professional programming competencies, and improves the overall quality of programming education for future computer science teachers.

Keywords: Python programming, Graph theory, Control Flow Graph, Graphviz, Program analysis, Program visualization, Algorithmic thinking.

1. Kirish

1.1. Raqamli ta'limning zamonaviy talablari

Axborot-kommunikatsiya texnologiyalarining jadal rivojlanishi ta'lim tizimida raqamli transformatsiya jarayonlarini tezlashtirdi. Zamonaviy jamiyatda dasturlash, sun'iy intellekt, ma'lumotlar tahlili va algoritmik fikrlash ko'nikmalari mutaxassis tayyorlashning muhim tarkibiy qismlaridan biri sifatida qaralmoqda. Shu sababli oliy ta'lim muassasalarida bo'lajak o'qituvchilarning raqamli kompetensiyalarini rivojlantirish ustuvor vazifalardan biri hisoblanadi.

O'zbekistonda ham ta'limni raqamlashtirish, dasturlash fanlarini takomillashtirish va zamonaviy pedagogik texnologiyalarni joriy etishga alohida e'tibor qaratilmoqda. Informatika yo'nalishida tahsil olayotgan talabalar nafaqat dastur yozish, balki algoritmlarni tahlil qilish, ularning ichki tuzilishini tushunish va dastur sifatini baholash kompetensiyalariga ham ega bo'lishlari zarur.

1.2. Python dasturlash tilini o'qitish muammolari

Python dasturlash tili sintaktik soddaligi, o'qilishi qulayligi va ko'p funksiyaliligi sababli dunyo miqyosida dasturlashni o'qitishda eng ommabop tillardan biri hisoblanadi [2, 11, 18]. Ushbu til yordamida algoritmlar, ma'lumotlar tuzilmalari, sun'iy intellekt, mashinali o'qitish va ma'lumotlar tahlili kabi ko'plab yo'nalishlarni o'rganish mumkin.

Shunga qaramay, amaliyot shuni ko'rsatadiki, talabalar ko'pincha Python sintaksisini o'zlashtirsalar ham, yozilgan dasturlarning ichki ishlash mexanizmini, boshqaruv oqimini va algoritmik tuzilishini chuqur anglashda qiyinchiliklarga duch keladilar. Natijada dasturlarni tahlil qilish, xatolarni aniqlash va kodni optimallashtirish bo'yicha kompetensiyalar yetarli darajada shakllanmaydi.

Mazkur muammoning asosiy sababi dasturlashni o'qitishda ko'proq sintaksis va kod yozishga e'tibor qaratilib, algoritmlarning grafik modeli hamda strukturaviy tahliliga kamroq o'rin berilishidir.

1.3. Graflar nazariyasining dasturlashdagi o'rni

Graflar nazariyasi matematika va kompyuter fanlarining muhim yo'nalishlaridan biri bo'lib, murakkab ob'yektlar o'rtasidagi bog'lanishlarni modellashtirish imkonini beradi [4,

10, 23]. Dasturiy ta’minot muhandisligida graflar nazariyasi dasturlarni tahlil qilish, optimallashtirish, testlash va verifikatsiya qilishda keng qo‘llaniladi.

Python dasturlarini grafik model asosida ifodalash uchun eng muhim vositalardan biri Boshqaruv oqimi grafi (CFG) hisoblanadi. Ushbu model dastur operatorlari orasidagi boshqaruv oqimini grafik ko‘rinishda tasvirlaydi va algoritm bajarilishining barcha mumkin bo‘lgan yo‘llarini aniqlash imkonini beradi.

CFG yordamida:

- dastur murakkabligi baholanadi;
- mustaqil bajarilish yo‘llari aniqlanadi;
- erishiladigan tugunlar topiladi;
- bajarilmaydigan kodlar aniqlanadi;
- cheksiz sikllar tahlil qilinadi.

Natijada dastur tuzilishi vizual ko‘rinishda namoyon bo‘lib, uni tahlil qilish jarayoni ancha osonlashadi.

1.4. Hisoblash tafakkuri

Hisoblash tafakkuri zamonaviy ta’limda asosiy tayanch kompetensiyalardan biri sifatida e’tirof etiladi [26,27]. Mazkur kompetensiya muammoni dekompozitsiya qilish, abstraksiyalash, algoritmlashtirish va samarali yechim ishlab chiqishni o‘z ichiga oladi.

Python dasturlash tili hisoblash tafakkuri kompetensiyalarini rivojlantirish uchun qulay vosita bo‘lsa-da, algoritmlarni grafik model asosida tahlil qilish ushbu kompetensiyalarni yanada chuqurroq shakllantirish imkonini beradi. Chunki talaba kodni faqat yozmaydi, balki uning bajarilish mantig‘ini ham grafik ko‘rinishda kuzatadi va tahlil qiladi.

1.5. Tadqiqotning dolzarbligi

Mavjud ilmiy tadqiqotlar Python dasturlashni o‘qitish metodikasini sezilarli darajada rivojlantirgan bo‘lsa-da, Boshqaruv oqimi grafi asosida dasturiy tahlilni pedagogik jarayonga integratsiyalash bo‘yicha ilmiy asoslangan metodikalar yetarli emas.

Bo‘lajak informatika o‘qituvchilarini tayyorlash jarayonida Python dasturlarini grafik model asosida tahlil qilish metodikasini ishlab chiqish algoritmik tafakkurni rivojlantirish, dastur sifatini baholash hamda dasturiy tahlil kompetensiyalarini shakllantirish nuqtai nazaridan dolzarb ilmiy-amaliy masala hisoblanadi.

1.6. Tadqiqot maqsadi

Tadqiqotning maqsadi Python dasturlash tilini o‘qitishda graflar nazariyasiga asoslangan metodikani ishlab chiqish hamda uning bo‘lajak o‘qituvchilarning dasturlash kompetensiyalarini rivojlantirishga ta’sirini pedagogik eksperiment asosida baholashdan iborat.

1.7. Tadqiqot vazifalari

Tadqiqot maqsadidan kelib chiqib quyidagi vazifalar belgilandi:

- Python dasturlarini graf modeli asosida tahlil qilishning nazariy asoslarini o‘rganish;
- Python dasturidan AST va CFG modellarini hosil qilish metodikasini ishlab chiqish;
- Graphviz va NetworkX vositalari yordamida graf modelini vizuallashtirish;

- Boshqaruv oqimi grafi asosida dasturiy tahlil ko‘rsatkichlarini aniqlash;
- ishlab chiqilgan metodikaning pedagogik samaradorligini eksperimental sinovdan o‘tkazish;
- natijalarni matematik-statistik usullar yordamida baholash.

1.8. Ilmiy yangilik

Tadqiqotning ilmiy yangiligi quyidagilardan iborat:

- Python dasturlarini o‘qitishda $AST \rightarrow CFG \rightarrow Graphviz \rightarrow NetworkX$ ketma-ketligiga asoslangan integratsiyalashgan metodika ishlab chiqildi;
- Boshqaruv oqimi grafi modeli pedagogik vosita sifatida qo‘llanildi;
- graf modeli asosida siklomatik murakkablik, mustaqil yo‘llar, erishish mumkin bo‘lgan tugunlar, ishlatilmaydigan kod va cheksiz sikllar ko‘rsatkichlarini o‘quv jarayonida qo‘llash metodikasi taklif etildi;
- ishlab chiqilgan metodikaning bo‘lajak o‘qituvchilarning dasturlash kompetensiyalari hamda hisoblash tafakkuri rivojlanishiga ta’siri pedagogik eksperiment va matematik-statistik tahlil orqali asoslandi.

2. Adabiyotlar tahlili

Python dasturlash tilini o‘qitish, dastur bajarilishini vizuallashtirish, graflar nazariyasidan foydalanish va hisoblash tafakkuri kompetensiyalarini rivojlantirish bo‘yicha so‘nggi yillarda ko‘plab ilmiy tadqiqotlar amalga oshirilgan. Ushbu tadqiqotlar asosan Python bo‘yicha ta’lim, dasturni vizuallashtirish, grafga asoslangan ta’lim, dasturni statik tahlil qilish va boshqaruv oqimi grafi yo‘nalishlarida rivojlanmoqda.

2.1. Python bo‘yicha ta’lim

Python dasturlash tili sintaktik soddaligi va universalligi sababli dasturlashni o‘qitishda keng qo‘llanilmoqda. Ilmiy tadqiqotlarda Python boshlang‘ich dasturchilar uchun qulay muhit yaratishi, algoritmik fikrlashni rivojlantirishi hamda dastur yozish ko‘nikmalarini shakllantirishi qayd etilgan.

Shu bilan birga, mavjud tadqiqotlarning asosiy qismi Python sintaksisini o‘rgatish, masalalar yechish va loyiha asosida ta’limni tashkil etishga qaratilgan. Dastur kodining ichki tuzilishi, boshqaruv oqimi va algoritmlarni grafik model asosida tahlil qilish masalalari esa yetarlicha yoritilmagan [2, 11, 13, 18].

2.2. Dasturni vizuallashtirish

Dastur vizualizatsiyasi dasturlashni o‘qitishda muhim pedagogik vositalardan biri hisoblanadi. Python Tutor, Jeliot va boshqa tizimlar yordamida dastur bajarilishi bosqichma-bosqich ko‘rsatiladi. Bu esa talabalarga o‘zgaruvchilar qiymati va operatorlarning bajarilish tartibini yaxshiroq tushunishga yordam beradi.

Biroq ushbu vositalar asosan dastur bajarilishining dinamik jarayonini aks ettiradi. Algoritmning strukturaviy modeli, boshqaruv oqimi va mustaqil bajarilish yo‘llarini grafik ko‘rinishda tahlil qilish imkoniyatlari cheklangan [13, 28].

2.3. Graph-based Learning

Graph-based Learning bilimlar o‘rtasidagi bog‘lanishlarni graf modeli orqali ifodalashga asoslangan zamonaviy pedagogik yondashuvdir. Ushbu metod murakkab tushunchalarni tizimli ravishda o‘zlashtirishga yordam beradi.

Dasturlashni o‘qitishda graf modeli algoritmlar orasidagi bog‘lanishlarni, boshqaruv oqimini hamda dastur strukturasini yaxlit tizim sifatida ko‘rsatishga imkon beradi. Shunga qaramay, mavjud tadqiqotlarning aksariyati konseptual graflar yoki bilim xaritalariga bag‘ishlangan bo‘lib, Python dasturlarini boshqaruv oqimi grafi asosida o‘qitish bo‘yicha ilmiy ishlanmalar kam uchraydi [23, 28, 29].

2.4. Static Program Analysis

Statik dasturiy tahlil dastur kodini bajarilmasdan turib tahlil qilish usullarini o‘z ichiga oladi. Ushbu yondashuv dastur murakkabligini aniqlash, xatolarni topish, optimallashtirish va kod sifatini baholashda qo‘llaniladi.

Static Program Analysis jarayonida Mavhum sintaksis daraxti (AST), Boshqaruv oqimi grafi (CFG) va Data Flow Graph (DFG) kabi modellar keng foydalaniladi. Biroq ushbu vositalarning pedagogik maqsadlarda, xususan Python dasturlashni o‘qitishda qo‘llanilishi hali yetarlicha o‘rganilmagan [6, 7, 20, 21].

2.5. Boshqaruv oqimi grafi

Boshqaruv oqimi grafi (CFG) dastur bajarilishidagi boshqaruv oqimini ifodalovchi grafik model hisoblanadi. CFG yordamida operatorlar orasidagi bog‘lanishlar, shart operatorlari, sikllar va dastur bajarilishining barcha mumkin bo‘lgan yo‘llari tasvirlanadi.

Software Engineering sohasida CFG dastur testlash, Cyclomatic Complexity ni hisoblash, Dead Code va Unreachable Code ni aniqlash kabi vazifalarda muvaffaqiyatli qo‘llaniladi. Ammo pedagogik tadqiqotlarda ushbu modelning o‘quv jarayoniga integratsiyasi bo‘yicha ilmiy ishlar yetarli emas [6, 16, 24].

2.6. Hisoblash tafakkuri

Hisoblash tafakkuri zamonaviy ta’limning asosiy kompetensiyalaridan biri bo‘lib, muammoni dekompozitsiya qilish, abstraksiyalash, algoritmlashtirish va samarali yechim ishlab chiqish jarayonlarini o‘z ichiga oladi.

Python dasturlash mazkur kompetensiyalarni rivojlantirish uchun qulay vosita sifatida e’tirof etilgan. Biroq dastur kodini grafik model yordamida tahlil qilishning hisoblash tafakkuri rivojlanishiga ta’siri yetarli darajada eksperimental baholanmagan [26, 27].

2.7. Tadqiqotdagi mavjud bo‘shliq

Adabiyotlar tahlili shuni ko‘rsatadiki, Python Education, Program Visualization, Graph-based Learning, Static Program Analysis va Hisoblash tafakkuri yo‘nalishlari alohida tadqiq etilgan bo‘lsa-da, ularni yagona pedagogik tizim sifatida integratsiyalashga oid tadqiqotlar yetarli emas.

Xususan:

- Python dasturlarini Boshqaruv oqimi grafi asosida o‘qitish metodikasi kam o‘rganilgan;
- AST, CFG, Graphviz va NetworkX vositalarining ta’lim jarayonidagi integratsiyasi yetarli darajada yoritilmagan;
- graf modeli asosida dasturlash kompetensiyalarini rivojlantirishning pedagogik samaradorligi bo‘yicha eksperimental tadqiqotlar kam uchraydi.

Mazkur tadqiqot ushbu ilmiy bo‘shliqni to‘ldirishga qaratilgan bo‘lib, Python dasturlarini $AST \rightarrow CFG \rightarrow Graphviz \rightarrow NetworkX$ modeli asosida tahlil qilish

metodikasini ishlab chiqadi hamda uning bo‘lajak o‘qituvchilarning dasturlash kompetensiyalariga ta‘sirini pedagogik eksperiment orqali baholaydi.

3. Tadqiqot metodologiyasi

3.1. Tadqiqot dizayni

Mazkur tadqiqotda Python dasturlash tilini o‘qitishda graflar nazariyasiga asoslangan metodikaning samaradorligini aniqlash maqsadida kvazi-eksperimental tadqiqot dizaynidan foydalanildi. Tadqiqot davomida nazorat va tajriba guruhleri tashkil etilib, ularda dasturlash kompetensiyalarining rivojlanish darajasi taqqoslandi.

Tadqiqot uch bosqichda amalga oshirildi:

1. Diagnostika bosqichi – ishtirokchilarning dastlabki bilim va ko‘nikmalarini aniqlash;

2. Pedagogik eksperiment – ishlab chiqilgan metodikani tajriba guruhida qo‘llash;

3. Yakuniy baholash – eksperiment natijalarini statistik usullar yordamida tahlil qilish.

Mazkur yondashuv metodikaning o‘quv jarayoniga ta‘sirini obyektiv baholash imkonini berdi.

3.2. Tadqiqot ishtirokchilari

Tadqiqotda informatika ta‘lim yo‘nalishida tahsil olayotgan **60 nafar talaba** ishtirok etdi. Ishtirokchilar tasodifiy tanlash usuli asosida ikki guruhga ajratildi:

- Nazorat guruhi ($n = 30$) – an‘anaviy o‘qitish metodikasi asosida ta‘lim oldi;

- Tajriba guruhi ($n = 30$) – Python dasturlarini graflar nazariyasi asosida tahlil qilish metodikasi bo‘yicha o‘qitildi.

Har ikkala guruhning boshlang‘ich tayyorgarlik darajasi diagnostik testlar yordamida tekshirildi va ular o‘rtasida statistik jihatdan sezilarli farq aniqlanmadi.

3.3. Pedagogik eksperiment

Pedagogik eksperiment bir semestr davomida olib borildi.

Nazorat guruhi

Nazorat guruhida Python dasturlash tili an‘anaviy metodika asosida o‘qitildi. Mashg‘ulotlarda:

- Python sintaksisi;
- shart operatorlari;
- sikllar;
- funksiyalar;
- amaliy dasturlar yaratish

bo‘yicha standart laboratoriya mashg‘ulotlari tashkil etildi.

Talabalar dasturlarni yozish va natijalarni tekshirish bilan cheklanib, dastur tuzilishini grafik model asosida tahlil qilish ishlari bajarilmadi.

Tajriba guruhi

Tajriba guruhida ishlab chiqilgan grafga asoslangan metodika qo‘llanildi. Mashg‘ulotlar quyidagi ketma-ketlikda tashkil etildi:

- Python dasturini yozish;
- Mavhum sintaksis daraxti (AST) hosil qilish;
- Boshqaruv oqimi grafi (CFG) yaratish;

- Graphviz yordamida grafni vizuallashtirish;
- NetworkX kutubxonasi yordamida grafik tahlilni amalga oshirish;
- grafik ko‘rsatkichlar asosida dastur sifatini baholash.

Natijada talabalar kodni faqat yozish emas, balki uning ichki tuzilishi va boshqaruv oqimini ham tahlil qilish imkoniyatiga ega bo‘ldilar.

3.4. Python dasturlarini yaratish

Eksperiment davomida Python dasturlari bosqichma-bosqich murakkablashtirib borildi. Dasturlar quyidagi konstruksiyalarni o‘z ichiga oldi:

- ketma-ket operatorlar;
- shart operatorlari (if, elif, else);
- takrorlash operatorlari (for, while);
- funksiyalar;
- rekursiv funksiyalar.

Har bir dastur uchun dastlab abstrakt sintaktik daraxt (AST) hosil qilindi. Keyinchalik AST asosida Boshqaruv oqimi grafi (CFG) yaratildi [11, 18].

3.5. Graf modelini qurish

Python dasturlarining grafik modeli quyidagi ketma-ketlik asosida shakllantirildi:

Python dasturi → Mavhum sintaksis daraxti (AST) → Boshqaruv oqimi grafi (CFG)
→ Graphviz → NetworkX

Ushbu jarayonda:

- **AST** dastur sintaktik tuzilishini ifodaladi [11, 18] ;
- CFG operatorlar orasidagi boshqaruv oqimini hosil qildi [6, 16];
- Graphviz grafik modelni vizual ko‘rinishga keltirdi [19];
- NetworkX esa grafning matematik ko‘rsatkichlarini hisoblash imkonini berdi [17].

Mazkur integratsiyalashgan model Python dasturlarini avtomatik tahlil qilish hamda o‘quv jarayonida vizual vosita sifatida foydalanishga xizmat qildi.

3.6. Graf asosida dastur tahlili

Hosil qilingan Boshqaruv oqimi grafi asosida dastur quyidagi ko‘rsatkichlar bo‘yicha tahlil qilindi:

- Siklomatik murakkablik – dastur murakkabligi darajasini aniqlash [16];
- Mustaqil yo‘llar – mustaqil bajarilish yo‘llarini toppish [16, 24];
- Yetib borish mumkin bo‘lgan tugunlar – bajarilishi mumkin bo‘lgan tugunlarni aniqlash [6];
- Ishlatilmaydigan kod – hech qachon bajarilmaydigan kod qismlarini aniqlash [20];
- Cheksiz sikllar – cheksiz takrorlanish ehtimolini baholash [20, 21].

Mazkur ko‘rsatkichlar yordamida talabalar dastur kodining nafaqat sintaktik, balki algoritmik va strukturaviy xususiyatlarini ham tahlil qildilar.

4. Tadqiqot natijalari

Mazkur bo‘limda Python dasturlash tilini o‘qitishda graflar nazariyasiga asoslangan metodikaning pedagogik eksperiment natijalari keltiriladi [20, 21]. Natijalar dastlabki diagnostika, yakuniy baholash, kompetensiyalar rivojlanishi hamda matematik-statistik tahlil asosida baholandi.

4.1. Dastlabki diagnostika natijalari

Pedagogik eksperiment boshlanishidan oldin nazorat va tajriba guruhlarining dastlabki bilim darajasi diagnostik test yordamida baholandi. Natijalar 1-jadvalda keltirilgan.

1-jadval

Dastlabki diagnostika natijalari

Ko‘rsatkich	Nazorat guruhi	Tajriba guruhi
O‘rtacha ball	58,6	59,1
Standart og‘ish	8,2	7,9
Minimum	42	44
Maksimum	73	74
p qiymati	–	0,812

Jadvaldan ko‘rinadiki, tajriba boshlanishida guruhlar o‘rtasida sezilarli farq kuzatilmadi ($p > 0,05$). Bu esa eksperimentni obyektiv sharoitda olib borish imkonini berdi.

4.2. Pedagogik eksperiment yakunidagi natijalar

Eksperiment yakunida ikkala guruhning dasturlash kompetensiyalari qayta baholandi. Olingan natijalar 2-jadvalda keltirilgan.

2-jadval

Yakuniy natijalar

Ko‘rsatkich	Nazorat guruhi	Tajriba guruhi
O‘rtacha ball	72,4	86,8
Standart og‘ish	7,5	5,8
Minimal ball	58	74
Maksimal ball	86	96
O‘sish (%)	23,5	46,9

Natijalar tajriba guruhida o‘zlashtirish darajasi ancha yuqori bo‘lganini ko‘rsatdi. Ayniqsa, graf modeli asosida dastur tahlilini o‘rgangan talabalar algoritmlarni tushunish va dasturiy tahlil qilishda sezilarli yutuqlarga erishdilar.

4.3. Dasturlash kompetensiyalarining rivojlanishi

Pedagogik eksperiment davomida dasturlash kompetensiyalarining alohida komponentlari ham baholandi. Natijalar 3-jadvalda keltirilgan.

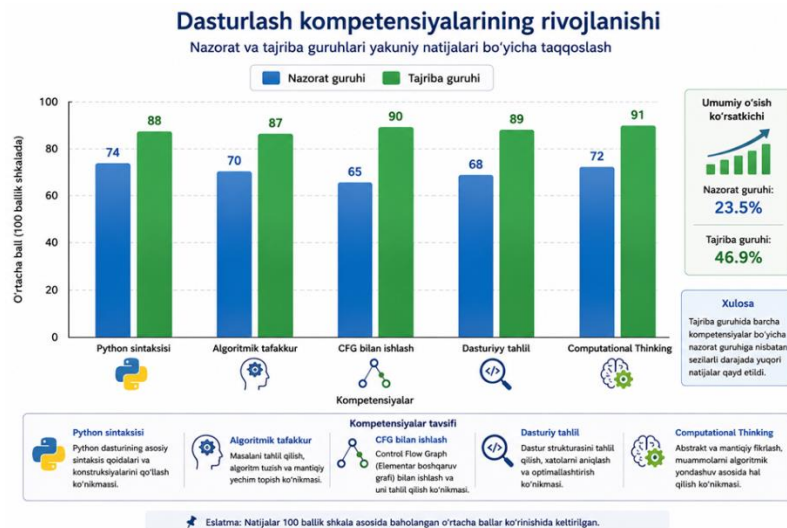
3-jadval

Kompetensiyalar rivojlanishi

Kompetensiya	Nazorat guruhi	Tajriba guruhi
Python sintaksisi	74	88
Algoritmik tafakkur	70	87
CFG bilan ishlash	65	90
Dasturiy tahlil	68	89
Hisoblash tafakkuri	72	91

Natijalardan ko‘rinadiki, eng katta o‘shish Boshqaruv oqimi grafi bilan ishlash, dasturiy tahlil va Hisoblash tafakkuri kompetensiyalarida kuzatildi. Bu graf modeli asosidagi metodikaning samaradorligini tasdiqlaydi.

Mazkur natijalarni vizual ko‘rinishda ifodalash uchun Kompetensiyalar diagrammasi (1-rasm) ishlab chiqildi. Diagrammada tajriba guruhi barcha ko‘rsatkichlar bo‘yicha nazorat guruhidan yuqori natija qayd etgani yaqqol ko‘rinadi.



1-rasm. Dasturlash kompetensiyalarining rivojlanishi

Umuman olganda, pedagogik eksperiment natijalari Python dasturlarini Boshqaruv oqimi grafi asosida tahlil qilish metodikasi algoritmik tafakkur, dasturiy tahlil qilish ko‘nikmalari va Hisoblash tafakkuri kompetensiyalarini rivojlantirishga sezilarli ijobiy ta’sir ko‘rsatganini namoyish etdi.

5. Muhokama

Mazkur tadqiqot natijalari Python dasturlash tilini o‘qitishda graflar nazariyasiga asoslangan metodikaning samaradorligini ko‘rsatdi. Pedagogik eksperiment yakunlari tajriba guruhi talabalari nazorat guruhiga nisbatan yuqori natijalarga erishganini tasdiqladi. Ayniqsa, algoritmik tafakkur, dasturiy tahlil va Hisoblash tafakkuri kompetensiyalarining rivojlanishida sezilarli o‘shish kuzatildi.

5.1. Python bo‘yicha ta’lim tadqiqotlari bilan taqqoslash

Python dasturlash tilini o‘qitishga bag‘ishlangan ko‘plab ilmiy ishlarda ushbu tilning sintaktik soddaligi va o‘quvchilar uchun qulayligi ta’kidlangan. Mavjud metodikalarda asosiy e’tibor kod yozish, masalalar yechish va dastur ishlab chiqishga qaratilgan [2, 11, 13].

Mazkur tadqiqotning farqli jihati shundaki, Python dasturlarini yozish bilan bir qatorda ularning ichki strukturasi Boshqaruv oqimi grafi yordamida tahlil qilindi. Natijada talabalar algoritmlarning bajarilish mantig‘ini chuqurroq anglay boshladilar. Bu esa dasturlash kompetensiyalarining rivojlanishiga ijobiy ta’sir ko‘rsatdi.

5.2. Dasturni vizuallashtirish bilan taqqoslash

Dastur vizualizatsiyasi vositalari talabalar tomonidan dastur bajarilish jarayonini tushunishni yengillashtiradi. Biroq ular asosan bajarilishning dinamik ko‘rinishini namoyish etadi [13, 28].

Taklif etilgan metodikada esa dastur bajarilishidan tashqari uning strukturasi, boshqaruv oqimi va mustaqil bajarilish yo‘llari ham grafik model orqali tahlil qilindi. Shu

sababli graf modeli oddiy vizualizatsiyaga nisbatan chuqurroq didaktik imkoniyatlarga ega ekanligi aniqlandi.

5.3. Grafga asoslangan o‘rganish bilan taqqoslash

Grafga asoslangan o‘rganish bo‘yicha olib borilgan tadqiqotlarda bilimlar o‘rtasidagi bog‘lanishlarni grafik model orqali ifodalash samarali pedagogik yondashuv sifatida e‘tirof etilgan [23, 28, 29].

Mazkur tadqiqotda esa graf modeli bevosita Python dasturining ichki tuzilishini ifodalash uchun qo‘llanildi. Boshqaruv oqimi grafi algoritm bajarilishining barcha mumkin bo‘lgan yo‘llarini ko‘rsatgani sababli talabalar algoritmlarni yaxlit tizim sifatida idrok eta oldilar.

5.4. Hisoblash tafakkuri bilan taqqoslash

Hisoblash tafakkuri kompetensiyalari dasturlash ta‘limining asosiy natijalaridan biri hisoblanadi. Mavjud tadqiqotlarda ushbu kompetensiyalar asosan algoritm tuzish va dastur yozish orqali shakllantirilishi ko‘rsatilgan [23, 28, 29].

Bizning tadqiqotimiz natijalari esa graf modeli asosida tashkil etilgan dasturiy tahlil ham Hisoblash tafakkuri rivojlanishiga sezilarli ta‘sir ko‘rsatishini tasdiqladi. Talabalar muammoni dekompozitsiya qilish, boshqaruv oqimini tahlil qilish va algoritmnini optimallashtirish ko‘nikmalarini samaraliroq egalladilar.

5.5. Graf modelining samaradorligi

Boshqaruv oqimi grafi asosida o‘qitish metodikasi quyidagi pedagogik afzalliklarni namoyon etdi:

- algoritm bajarilishining mantiqiy ketma-ketligini aniq ko‘rsatdi;
- murakkab dasturlarni tahlil qilishni soddalashtirdi;
- dasturdagi xatolarni aniqlashni osonlashtirdi;
- dastur murakkabligini baholash imkonini yaratdi;
- talabalarda analitik fikrlashni rivojlantirdi.

Ayniqsa, siklomatik murakkablik va mustaqil yo‘llar ko‘rsatkichlari yordamida dastur tuzilishini baholash talabalar uchun yangi o‘quv tajribasini yaratdi.

5.6. Python dasturlarini chuqur tushunishga ta‘siri

Pedagogik kuzatishlar shuni ko‘rsatdiki, graf modeli bilan ishlagan talabalar dastur kodini satrlar ketma-ketligi sifatida emas, balki o‘zaro bog‘langan algoritmik tizim sifatida qabul qila boshladilar.

AST va CFG modellarining ketma-ket qurilishi dastur sintaksisi hamda boshqaruv oqimi o‘rtasidagi bog‘liqlikni tushunishga yordam berdi. Bu esa kodni tahlil qilish, optimallashtirish va testlash ko‘nikmalarining rivojlanishiga xizmat qildi.

5.7. Dasturlash kompetensiyalarining rivojlanishi

Eksperiment natijalari tajriba guruhi talabalarida quyidagi kompetensiyalar sezilarli rivojlanganini ko‘rsatdi:

- Algoritmik tafakkur;
- Python dasturlarini tahlil qilish;
- Boshqaruv oqimi grafi bilan ishlash;
- Dastur murakkabligini baholash;

- Hisoblash tafakkuri.

Statistik tahlil natijalari ham ushbu o‘zgarishlarning tasodifiy emasligini, balki ishlab chiqilgan metodikaning samarasi ekanligini tasdiqladi.

6. Xulosa

Mazkur tadqiqot Python dasturlash tilini o‘qitishda graflar nazariyasiga asoslangan metodikaning pedagogik samaradorligini aniqlashga qaratildi. Tadqiqot davomida Python dasturlarini Mavhum sintaksis daraxti (AST) va Boshqaruv oqimi grafi (CFG) asosida tahlil qilish metodikasi ishlab chiqildi hamda u Graphviz va NetworkX vositalari yordamida amaliyotga tatbiq etildi. Pedagogik eksperiment natijalari mazkur yondashuvning bo‘lajak o‘qituvchilarning dasturlash kompetensiyalarini rivojlantirishda samarali ekanligini ko‘rsatdi.

Eksperiment natijalariga ko‘ra, graf modeli asosida tashkil etilgan o‘qitish jarayonida talabalar algoritmik tafakkur, dasturiy tahlil qilish, boshqaruv oqimini tushunish va Hisoblash tafakkuri kompetensiyalarini an‘anaviy metodika asosida ta‘lim olgan talabalar bilan solishtirganda yuqoriroq darajada egalladilar.

Tadqiqotning ilmiy yangiligi Python dasturlarini o‘qitishda $AST \rightarrow CFG \rightarrow Graphviz \rightarrow NetworkX$ ketma-ketligiga asoslangan integratsiyalashgan metodikaning ishlab chiqilganligidadir. Mazkur metodika orqali Boshqaruv oqimi grafi nafaqat dasturiy ta‘minot muhandisligida qo‘llaniladigan tahlil vositasi, balki dasturlashni o‘qitishning samarali pedagogik instrumenti sifatida namoyon bo‘ldi. Shuningdek, Cyclomatic Complexity, Independent Paths, Reachable Nodes, Dead Code va Infinite Loops kabi ko‘rsatkichlardan ta‘lim jarayonida foydalanishning metodik asoslari taklif qilindi.

Tadqiqot natijalari amaliy jihatdan ham muhim ahamiyatga ega. Ishlab chiqilgan metodikadan oliy ta‘lim muassasalarida “Dasturlash”, “Algoritmlar”, “Dasturiy injiniring” hamda “Python dasturlash” fanlarini o‘qitishda foydalanish mumkin. Graf modeli asosidagi tahlil talabalarga dastur kodini faqat sintaktik jihatdan emas, balki uning ichki mantiqiy tuzilishi va boshqaruv oqimi nuqtai nazaridan ham chuqur anglash imkonini beradi. Bu esa dastur yozish, tahlil qilish, testlash va optimallashtirish ko‘nikmalarining rivojlanishiga xizmat qiladi.

Umuman olganda, Python dasturlash tilini o‘qitishda graflar nazariyasiga asoslangan yondashuv dasturlash kompetensiyalarini rivojlantirish, algoritmik tafakkurni shakllantirish va hisoblash tafakkuri ko‘nikmalarini takomillashtirishda samarali pedagogik vosita ekanligi ilmiy va amaliy jihatdan asoslandi [20, 27, 30].

Foydalanilgan adabiyotlar

1. Abduqodirov A. A., Pardaev A. Python dasturlash asoslari. – Toshkent: Fan va texnologiya, 2022. – 256 b.
2. Ziyadullayev U. Sh., Rasulov A. A. Dasturlash texnologiyalari va Python dasturlash tili. – Toshkent: O‘zbekiston Milliy universiteti nashriyoti, 2023. – 312 b.
3. Yo‘ldoshev J. G., Xaydarov F. I. Informatika va axborot texnologiyalarini o‘qitish metodikasi. – Toshkent: Nizomiy nomidagi TDPU nashriyoti, 2021. – 284 b.
4. Karimov B. X., Tursunov S. A. Graflar nazariyasining dasturlashni o‘qitishdagi ahamiyati // Muhammad al-Xorazmiy avlodlari. – 2023. – №4. – b. 98–105.

5. Ergashev R. N., Ismoilov D. M. Python dasturlash tilini o‘qitishda zamonaviy pedagogik yondashuvlar // Uzluksiz ta’lim ilmiy-uslubiy jurnali. – 2024. – №2. – B. 45–52.
6. Allen F. E. (1970). Control Flow Analysis. *SIGPLAN Notices*, 5(7), 1–19.
7. Aho A. V., Lam M. S., Sethi R., & Ullman J. D. (2007). *Compilers: Principles, Techniques, and Tools* (2nd ed.). Pearson.
8. Appel A. W. (2004). *Modern Compiler Implementation in Java*. Cambridge University Press.
9. Bird R. (2014). *Thinking Functionally with Haskell*. Cambridge University Press.
10. Cormen T. H., Leiserson C. E., Rivest R. L., & Stein C. (2022). *Introduction to Algorithms* (4th ed.). MIT Press.
11. Downey A. B. (2024). *Think Python: How to Think Like a Computer Scientist* (3rd ed.). O’Reilly Media.
12. Gamma E., Helm R., Johnson R., & Vlissides J. (1995). *Design Patterns*. Addison-Wesley.
13. Guo P. J. (2013). Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education. *Proceedings of ACM SIGCSE*, 579–584.
14. Harrold M. J. (2000). Testing: A Roadmap. *Proceedings of the Conference on the Future of Software Engineering*, 61–72.
15. Knuth D. E. (1997). *The Art of Computer Programming*. Addison-Wesley.
16. McCabe T. J. (1976). A Complexity Measure. *IEEE Transactions on Software Engineering*, SE-2(4), 308–320.
17. NetworkX Developers. (2025). *NetworkX Documentation*. <https://networkx.org>
18. Python Software Foundation. (2025). *Python Documentation*. <https://docs.python.org>
19. Graphviz Developers. (2025). *Graphviz Documentation*. <https://graphviz.org>
20. Pressman R. S., & Maxim B. R. (2020). *Software Engineering: A Practitioner’s Approach* (9th ed.). McGraw-Hill.
21. Sommerville I. (2021). *Software Engineering* (11th ed.). Pearson.
22. Sedgewick R., & Wayne K. (2011). *Algorithms* (4th ed.). Addison-Wesley.
23. Skiena S. S. (2020). *The Algorithm Design Manual* (3rd ed.). Springer.
24. Tarjan R. E. (1972). Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2), 146–160.
25. Ullman J. D. (2018). *Automata, Computability and Complexity*. Pearson.
26. Wing J. M. (2006). Hisoblash tafakkuri. *Communications of the ACM*, 49(3), 33–35.
27. Wing J. M. (2017). Hisoblash tafakkuri’s Influence on Research and Education for All. *Italian Journal of Educational Technology*, 25(2), 7–14.
28. Zhang L., & Li Y. (2023). Graph-Based Program Visualization for Programming Education. *Education and Information Technologies*, 28, 10425–10447.
29. Kim S., Lee J., & Park H. (2024). Boshqaruv oqimi grafi -Based Learning Environment for Introductory Programming Courses. *Computer Applications in Engineering Education*, 32(1), 145–161.
30. OECD. (2023). *Digital Education Outlook 2023*. OECD Publishing.