ЛЕКЦИЯ 1. ВВЕДЕНИЕ В ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ — КРАТКАЯ ВЫЖИМКА

• Курс для 1‑го курса, идет два семестра: сейчас — зачёт, затем экзамен. Материалы и сдача работ — через электронный университет; презентации и задания выкладываются оперативно.

• Зачёт «автоматом» — если все лабораторные выполнены и сданы в срок: первая часть — до первой практики (октябрь–ноябрь), вторая — в срок +1 неделя. Иначе — обычная сдача без «автомата».

• Практики планируются дистанционно; посещаемость лекций не критична, но на практиках ведётся отметка. При полном отсутствии на практиках преподаватель может задать уточняющие вопросы при зачёте.

• Базовый язык в этом году — Python: основы программирования, алгоритмы, структуры данных, работа с API и базами данных. Далее в программе — Java и другие технологии.

• Основные направления ИТ: разработка ПО (веб, мобильная, десктоп, игры), искусственный интеллект и машинное обучение, кибербезопасность, администрирование и DevOps, облачные технологии.

• Веб: клиент‑серверная модель, HTTP/HTTPS, HTML/CSS/JavaScript, фреймворки React/Vue/Angular. Мобайл: Kotlin (Android), Swift (iOS), также Flutter/React Native.

• DevOps: автоматизация, контейнеризация и оркестрация (Docker, Kubernetes), базовые навыки работы с серверами, CI/CD. Облака: практическое использование провайдеров наподобие Yandex Cloud/VK Cloud и др.

• Git — обязательный инструмент: репозиторий, коммиты, ветки (main/develop/feature/hotfix), pull/push/merge, разрешение конфликтов, gitignore, откаты (log/checkout/reset).

• Главный навык разработчика — уметь искать информацию и отлаживать ошибки. Самообучение и регулярная практика — норма.

Курс «Введение в информационные технологии» читается на первом курсе и охватывает два семестра. В текущем семестре итогом является зачёт, во втором — экзамен. Все организационные взаимодействия (презентации, задания, методички, дедлайны и объявление об условиях «автомата») происходят через электронный университет; материалы, как правило, появляются в день занятия или на следующее утро. Практические занятия планируются в дистанционном формате: это позволяет работать из любой точки при наличии устойчивого интернета и базовой среды разработки. Посещение лекций носит рекомендательный характер, тогда как практики предполагают отметку присутствия. Если студент полностью игнорирует практики, преподаватель может задать уточняющие вопросы при зачёте; однако в целом сдать дисциплину реально при соблюдении сроков и требований к работам.

Зачёт можно получить «автоматом», если своевременно выполнить и сдать лабораторные: первую часть — к первой практической встрече (ориентировочно октябрь–ноябрь), вторую — в плюс одну неделю к соответствующей практике. При пропуске дедлайнов остаётся стандартная процедура сдачи без «автомата» — это не критично, но потребует дополнительных усилий. Работы загружаются в электронный университет, там же преподаватель оставляет комментарии. Лекции и практики разрешено конспектировать; записи и скриншоты не обязательны, так как официальные материалы доступны в системе.

Практическая часть курса в этом году сосредоточена на языке Python. Это удобная точка входа: синтаксис прост, экосистема богата, а охват тем — от алгоритмов и структур данных до работы с файлами, сетями, базами данных и API. Освоив один язык, гораздо легче переходить на другие: во втором курсе ожидается акцент на Java, при этом фундаментальные концепции (условия, циклы, функции/методы, типы данных, коллекции, основы ООП, обработка ошибок, модули и пакеты) во многом универсальны. Важный практический навык — чтение ошибок и их отладка: умение быстро понять, где и почему программа работает не так, и как это исправить.

Широкий контекст курса — обзор основных направлений ИТ. К ним относится разработка программного обеспечения: веб‑приложения, мобильные приложения, десктопные программы, игры. Веб базируется на клиент‑серверной модели и протоколах HTTP/HTTPS. На стороне клиента — HTML/CSS/JavaScript, на уровне архитектуры — фреймворки вроде React, Vue и Angular, помогающие строить полноценные интерфейсы, работать с состоянием и маршрутизацией, интегрироваться с API и базами данных. В рамках курса веб‑часть будет затронута через практические задачи, но без перегруза теорией JavaScript — приоритетом остаётся Python как учебный мост к более сложным темам.

Мобильная разработка включает нативные приложения под Android и iOS и кроссплатформенные решения. Для Android используется Kotlin, для iOS — Swift. Из кроссплатформенных подходов популярны Flutter и React Native: они позволяют писать единую кодовую базу с генерацией сборок под обе платформы, хотя в ряде случаев нативные возможности и производительность лучше достигаются нативным кодом. В дальнейшем, при желании, студент может выбрать специализацию и углубиться в одно из направлений.

Десктопная разработка остаётся востребованной для внутренних корпоративных инструментов и утилит. Здесь подойдёт Python (например, с использованием библиотек для GUI), а также C/C++ и кроссплатформенные фреймворки вроде Qt или Electron. Игровая разработка чаще связана с движками Unity, Unreal Engine и Godot; в них значимую роль играют языки семейства C (C#/C++), а также знание графических пайплайнов, физических моделей и игровых механик. В рамках текущего курса игровые движки рассматриваются обзорно, как часть общей картины.

Отдельный крупный блок — искусственный интеллект и машинное обучение. Здесь нет «магии»: за успехом стоят алгоритмы, данные и тщательная настройка гиперпараметров. Важные темы — обучение моделей, обработка естественного языка (в том числе языковые модели), компьютерное зрение, работа с «большими данными». Подробное погружение в ML в этой программе запланировано на старших курсах; однако уже сейчас стоит сформировать математическую и программную базу (Python, библиотеки, работа с данными), которая позволит уверенно войти в дисциплину позже.

Кибербезопасность — ещё одно критически важное направление: защита данных и приватности, анализ уязвимостей, безопасная разработка, практики аудита и мониторинга. Востребованность специалистов высока, а инструменты и методы постоянно обновляются, что делает самообразование и отслеживание практик отрасли обязательным.

Современная разработка практически невозможна без DevOps‑культуры: автоматизация сборок и развёртываний (CI/CD), контейнеризация и оркестрация, наблюдаемость, инфраструктура как код. Контейнеры (Docker) упрощают упаковку приложений и зависимостей; оркестраторы (Kubernetes) повышают отказоустойчивость, масштабируемость и управляемость. Даже если вы основной разработчик, базовая грамотность в DevOps помогает понимать, как приложение живёт в продакшне, как логируются ошибки и измеряются метрики, как организованы тестирование и релизы.

С DevOps тесно связаны облачные технологии. Коммерческие провайдеры предоставляют виртуальные машины, базы данных, объектные хранилища и управляемые сервисы. Модель потребления — от «оплаты по факту использования» до фиксированных тарифов. В учебных и корпоративных сценариях важно учитывать требования к местоположению данных и юридические ограничения. Задача разработчика — уметь выбирать между «собственной инфраструктурой» и облачными сервисами с точки зрения стоимости, гибкости и надёжности.

Git — обязательный инструмент любого программиста. Это распределённая система контроля версий, которая хранит историю изменений кода и упрощает совместную работу. Вокабуляр минимально необходимый: репозиторий (локальный и удалённый), коммит (фиксация изменений с коротким осмысленным сообщением), ветка (parallel‑линии разработки — main, develop, feature, hotfix), слияние (merge), отправка (push) и получение (pull/fetch) изменений. Хороший рабочий минимум команд для старта: git init, git status, git add, git commit -m, git remote add origin, git push, git pull. В реальных проектах приходится разрешать конфликты слияния (редактирование пересекающихся участков кода), использовать .gitignore для исключения временных и тяжёлых файлов, а также откатываться на предыдущие версии (git log/checkout/reset). Публичные и приватные удалённые репозитории размещают на платформах вроде GitHub, GitLab и Bitbucket; они служат не только для синхронизации команды, но и как резервная копия.

В основе профессионального роста — систематическая практика, умение искать актуальную документацию, читать ошибки и лог‑трейсы, анализировать первопричины и формулировать гипотезы, проверяя их в коротких итерациях. В этом смысле данный курс — стартовая площадка: освоив Python и базовые инструменты, студент сможет осознанно выбрать дальнейшую специализацию (веб, мобайл, десктоп, ML/AI, кибербезопасность, DevOps/облака) и выстроить траекторию развития.

Организационные выводы просты: работайте регулярно, соблюдайте дедлайны, задавайте вопросы по делу, не бойтесь ошибок и используйте Git с первого же учебного проекта. Это позволяет двигаться вперёд предсказуемо и без потерь, а также облегчает проверку и обратную связь со стороны преподавателя.