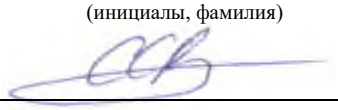
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
"САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ"

Кафедра № 32

УТВЕРЖДАЮ
Руководитель образовательной программы

к.т.н.,доц.
(должность, уч. степень, звание)

С.В. Солёный
(инициалы, фамилия)



(подпись)
«27» июня 2024 г

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

«Объектно-ориентированное программирование»
(Наименование дисциплины)

Код специальности	13.05.02
Наименование специальности	Специальные электромеханические системы
Наименование направленности	Электромеханические системы специальных устройств и изделий
Форма обучения	очная
Год приема	2024

Санкт-Петербург – 2024

Лист согласования рабочей программы дисциплины

Программу составил (а)

Старший преподаватель		Белай В.Е.
(должность, уч. степень, звание)	(подпись, дата)	(инициалы, фамилия)

Программа одобрена на заседании кафедры № 32

«26» июня 2024 г, протокол № 10

Заведующий кафедрой № 32

к.т.н.,доц.		С.В. Солёный
(уч. степень, звание)	(подпись, дата)	(инициалы, фамилия)

Заместитель директора института №3 по методической работе

ст. преподаватель		Н.В. Решетникова
(должность, уч. степень, звание)	(подпись, дата)	(инициалы, фамилия)

Аннотация

Дисциплина «Объектно-ориентированное программирование» входит в образовательную программу высшего образования – программу специалитета по специальности 13.05.02 «Специальные электромеханические системы» направленности «Электромеханические системы специальных устройств и изделий». Дисциплина реализуется кафедрой «№32».

Дисциплина не является обязательной при освоении обучающимся образовательной программы и направлена на углубленное формирование следующих компетенций:

ПК-3 «Способность участвовать в планировании, подготовке, выполнении и обработке результатов научно-исследовательских и опытно-конструкторских работ»

Содержание дисциплины охватывает круг вопросов, связанных с объектно-ориентированным программированием на высокоуровневом языке Python.

Преподавание дисциплины предусматривает следующие формы организации учебного процесса: лекции, лабораторные работы, самостоятельная работа обучающегося.

Программой дисциплины предусмотрены следующие виды контроля: текущий контроль успеваемости, промежуточная аттестация в форме зачета.

Общая трудоемкость освоения дисциплины составляет 2 зачетных единицы, 72 часа.

Язык обучения по дисциплине «русский».

1. Перечень планируемых результатов обучения по дисциплине

1.1. Цели преподавания дисциплины

Получение обучающимися необходимых знаний в области работы с высокоуровневым языком программирования Python, алгоритмов разработки программного обеспечения.

1.2. Дисциплина является факультативной дисциплиной по специальности образовательной программы высшего образования (далее – ОП ВО).

1.3. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения ОП ВО.

В результате изучения дисциплины обучающийся должен обладать следующими компетенциями или их частями. Компетенции и индикаторы их достижения приведены в таблице 1.

Таблица 1 – Перечень компетенций и индикаторов их достижения

Категория (группа) компетенции	Код и наименование компетенции	Код и наименование индикатора достижения компетенции
Профессиональные компетенции	ПК-3 Способность участвовать в планировании, подготовке, выполнении и обработке результатов научно-исследовательских и опытно-конструкторских работ	ПК-3.З.1 знает методы и средства планирования и организации опытно-конструкторских разработок и практических экспериментальных исследований; методы проведения экспериментов и наблюдений, обобщения и обработки информации, в том числе с применением технологий искусственного интеллекта ПК-3.У.1 умеет применять соответствующее программное обеспечение для оформления результатов научно-исследовательских и опытно-конструкторских работ

2. Место дисциплины в структуре ОП

Дисциплина может базироваться на знаниях, ранее приобретенных обучающимися при изучении следующих дисциплин:

– «Информатика»,
– «Высшая математика»,
– «Физика»,

Знания, полученные при изучении материала данной дисциплины, имеют как самостоятельное значение, так и могут использоваться при изучении других дисциплин:

– «Программирование микроконтроллеров».

3. Объем и трудоемкость дисциплины

Данные об общем объеме дисциплины, трудоемкости отдельных видов учебной работы по дисциплине (и распределение этой трудоемкости по семестрам) представлены в таблице 2.

Таблица 2 – Объем и трудоемкость дисциплины

Вид учебной работы	Всего	Трудоемкость по семестрам
		№6
1	2	3
Общая трудоемкость дисциплины, ЗЕ/ (час)	2/ 72	2/ 72
Из них часов практической подготовки	17	17

Аудиторные занятия, всего час.	34	34
в том числе:		
лекции (Л), (час)	17	17
практические/семинарские занятия (ПЗ), (час)		
лабораторные работы (ЛР), (час)	17	17
курсовой проект (работа) (КП, КР), (час)		
экзамен, (час)		
Самостоятельная работа, всего (час)	38	38
***Вид промежуточной аттестации*:** зачет, дифф. зачет, экзамен (Зачет, Дифф. зач, Экз.**)	Зачет	Зачет

Примечание: **кандидатский экзамен

4. Содержание дисциплины

4.1. Распределение трудоемкости дисциплины по разделам и видам занятий. Разделы, темы дисциплины и их трудоемкость приведены в таблице 3.

Таблица 3 – Разделы, темы дисциплины, их трудоемкость

Разделы, темы дисциплины	Лекции (час)	ПЗ (СЗ)	ЛР (час)	КП (час)	СРС (час)
Семестр 6					
Раздел 1. Введение в Python Тема 1.1. Интерпретатор языка Тема 1.2. Переменные, оператор присваивания, типы данных Тема 1.3. Функции ввода-вывода Тема 1.4. Арифметические операторы Тема 1.5. Условные и логические операторы Тема 1.6. Циклы, операторы break и continue Тема 1.7. Строки Тема 1.8. Списки Тема 1.9. Словари Тема 1.10. Кортежи Тема 1.11. Функции Тема 1.12. Рекурсия. Лямбда функции Тема 1.13. Тестирование программ Тема 1.14. Множества Тема 1.15. Итераторы Тема 1.16. Вложенные функции Тема 1.17. Обработка исключений Тема 1.18. Работа с файлами Тема 1.19. Работа с модулями Тема 1.20. Работа с пакетами Тема 1.21. Декораторы функций и замыкания.	8	-	8	-	18

Раздел 2. Объектно-ориентированное программирование на языке Python Тема 2.1. Концепция ООП Тема 2.2. Классы и объекты Тема 2.3. Методы классов Тема 2.4. Инициализатор и финализатор Тема 2.5. Магические методы Тема 2.6. Методы класса Тема 2.7. Инкапсуляция Тема 2.8. Паттерны ООП Тема 2.9. Свойства proprety Тема 2.10. Дескрипторы Тема 2.11. Методы сравнения. Тема 2.12. Наследование Тема 2.13. Обработка исключений для ООП Тема 2.14. Менеджеры контекстов Тема 2.15. Вложенные классы Тема 2.16. Метаклассы Тема 2.17. Пользовательские метаклассы Тема 2.18. Python Data Classes Тема 2.19. Принцип SOLID	9	-	9	-	20
Итого в семестре:	17		17		38
Итого	17	0	17	0	38

Практическая подготовка заключается в непосредственном выполнении обучающимися определенных трудовых функций, связанных с будущей профессиональной деятельностью.

4.2. Содержание разделов и тем лекционных занятий.

Содержание разделов и тем лекционных занятий приведено в таблице 4.

Таблица 4 – Содержание разделов и тем лекционного цикла

Номер раздела	Название и содержание разделов и тем лекционных занятий
1	Интерпретатор языка. Переменные, оператор присваивания, типы данных. Функции ввода-вывода. Арифметические операторы. Условные и логические операторы
1	Циклы, операторы break и continue. Строки. Списки. Словари. Кортежи
1	Функции. Рекурсия. Лямбда функции. Тестирование программ. Множества. Итераторы
1	Вложенные функции. Обработка исключений. Работа с файлами. Работа с модулями. Работа с пакетами. Декораторы функций и замыкания.
2	Концепция ООП. Классы и объекты. Методы классов. Инициализатор и финализатор. Магические методы
2	Методы класса. Инкапсуляция. Паттерны ООП. Свойства proprety. Дескрипторы
2	Методы сравнения. Наследование. Обработка исключений для ООП. Менеджеры контекстов. Вложенные классы
2	Метаклассы. Пользовательские метаклассы. Python Data Classes. Принцип SOLID

4.3. Практические (семинарские) занятия

Темы практических занятий и их трудоемкость приведены в таблице 5.

Таблица 5 – Практические занятия и их трудоемкость

№ п/п	Темы практических занятий	Формы практических занятий	Трудоемкость, (час)	Из них практической подготовки, (час)	№ раздела дисциплины
Учебным планом не предусмотрено					
Всего					

4.4. Лабораторные занятия

Темы лабораторных занятий и их трудоемкость приведены в таблице 6.

Таблица 6 – Лабораторные занятия и их трудоемкость

№ п/п	Наименование лабораторных работ	Трудоемкость, (час)	Из них практической подготовки, (час)	№ раздела дисциплины
Семестр 6				
1	Коллекции в Python	2	1	1
2	Шифр Цезаря	2	1	1
3	Построение графиков	2	1	1
4	Рассылка сообщений	2	1	1
5	Классы и экземпляры	2	1	2
6	Наследование	2	1	2
7	Множественное наследование	2	1	2
8	Обработка исключений	3	1	2
Всего		17		

4.5. Курсовое проектирование/ выполнение курсовой работы

Учебным планом не предусмотрено

4.6. Самостоятельная работа обучающихся

Виды самостоятельной работы и ее трудоемкость приведены в таблице 7.

Таблица 7 – Виды самостоятельной работы и ее трудоемкость

Вид самостоятельной работы	Всего, час	Семестр 6, час
1	2	3
Изучение теоретического материала дисциплины (ТО)	33	33
Курсовое проектирование (КП, КР)	-	-
Расчетно-графические задания (РГЗ)	-	-
Выполнение реферата (Р)	-	-
Подготовка к текущему контролю успеваемости (ТКУ)	3	3
Домашнее задание (ДЗ)	-	-
Контрольные работы заочников (КРЗ)	-	-

Подготовка к промежуточной аттестации (ПА)	2	2
Всего:	38	38

5. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине (модулю)

Учебно-методические материалы для самостоятельной работы обучающихся указаны в п.п. 7-11.

6. Перечень печатных и электронных учебных изданий

Перечень печатных и электронных учебных изданий приведен в таблице 8.

Таблица 8– Перечень печатных и электронных учебных изданий

Шифр/ URL адрес	Библиографическая ссылка	Количество экземпляров в библиотеке (кроме электронных экземпляров)
004.4 О-75	Основы программирования на языке Python : учебно-методическое пособие / А. И. Савельев [и др.] ; С.-Петерб. гос. ун-т аэрокосм. приборостроения. - Санкт-Петербург : Изд-во ГУАП, 2019. - 38 с. : табл. - Библиогр.: с. 37 (3 назв.). - Б. ц. - Текст : непосредственный.	5

7. Перечень электронных образовательных ресурсов информационно-телекоммуникационной сети «Интернет»

Перечень электронных образовательных ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины приведен в таблице 9.

Таблица 9 – Перечень электронных образовательных ресурсов информационно-телекоммуникационной сети «Интернет»

URL адрес	Наименование
https://disk.yandex.ru/i/yifM4EcZXT7nFA	Введение в программирование на Python
https://disk.yandex.ru/i/-SEuMuCGNGXm8w	Объектно-ориентированное программирование на Python

8. Перечень информационных технологий

8.1. Перечень программного обеспечения, используемого при осуществлении образовательного процесса по дисциплине.

Перечень используемого программного обеспечения представлен в таблице 10.

Таблица 10– Перечень программного обеспечения

№ п/п	Наименование
	Не предусмотрено

8.2. Перечень информационно-справочных систем,используемых при осуществлении образовательного процесса по дисциплине

Перечень используемых информационно-справочных систем представлен в таблице 11.

Таблица 11– Перечень информационно-справочных систем

№ п/п	Наименование
	Не предусмотрено

9. Материально-техническая база

Состав материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине, представлен в таблице12.

Таблица 12 – Состав материально-технической базы

№ п/п	Наименование составной части материально-технической базы	Номер аудитории (при необходимости)
1	Мультимедийная лекционная аудитория	31-04

10. Оценочные средства для проведения промежуточной аттестации

10.1. Состав оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине приведен в таблице 13.

Таблица 13 – Состав оценочных средств для проведения промежуточной аттестации

Вид промежуточной аттестации	Перечень оценочных средств
Зачёт	Перечень вопросов Тесты;

10.2. В качестве критериев оценки уровня сформированности (освоения) компетенций обучающимися применяется 5-балльная шкала оценки сформированности компетенций, которая приведена в таблице 14. В течение семестра может использоваться 100-балльная шкала модульно-рейтинговой системы Университета, правила использования которой, установлены соответствующим локальным нормативным актом ГУАП.

Таблица 14 –Критерии оценки уровня сформированности компетенций

Оценка компетенции 5-балльная шкала	Характеристика сформированных компетенций
«отлично» «зачтено»	– обучающийся глубоко и всесторонне усвоил программный материал; – уверенно, логично, последовательно и грамотно его излагает; – опираясь на знания основной и дополнительной литературы, тесно привязывает усвоенные научные положения с практической деятельностью направления; – умело обосновывает и аргументирует выдвигаемые им идеи; – делает выводы и обобщения; – свободно владеет системой специализированных понятий.
«хорошо» «зачтено»	– обучающийся твердо усвоил программный материал, грамотно и по существу излагает его, опираясь на знания основной литературы; – не допускает существенных неточностей; – увязывает усвоенные знания с практической деятельностью направления; – аргументирует научные положения; – делает выводы и обобщения; – владеет системой специализированных понятий.

Оценка компетенции 5-балльная шкала	Характеристика сформированных компетенций
«удовлетворительно» «зачтено»	– обучающийся усвоил только основной программный материал, по существу излагает его, опираясь на знания только основной литературы; – допускает несущественные ошибки и неточности; – испытывает затруднения в практическом применении знаний направления; – слабо аргументирует научные положения; – затрудняется в формулировании выводов и обобщений; – частично владеет системой специализированных понятий.
«неудовлетворительно» «не зачтено»	– обучающийся не усвоил значительной части программного материала; – допускает существенные ошибки и неточности при рассмотрении проблем в конкретном направлении; – испытывает трудности в практическом применении знаний; – не может аргументировать научные положения; – не формулирует выводов и обобщений.

10.3. Типовые контрольные задания или иные материалы.

Вопросы (задачи) для экзамена представлены в таблице 15.

Таблица 15 – Вопросы (задачи) для экзамена

№ п/п	Перечень вопросов (задач) для экзамена	Код индикатора
	Учебным планом не предусмотрено	

Вопросы (задачи) для зачета представлены в таблице 16.

Таблица 16 – Вопросы (задачи) для зачета

№ п/п	Перечень вопросов (задач) для зачета	Код индикатора
1	1. Переменной var_int присвойте значение 10, var_float - версия содержит текст уроков курса и решения с пояснениями к практическим работам значение 8.4, var_str -версия содержит текст уроков курса и решения с пояснениями к практическим работам "No". 2. Значение, хранимое в переменной var_int, увеличьте в 3.5 раза. Полученный результат свяжите с переменной var_big. 3. Измените значение, хранимое в переменной var_float, уменьшив его на единицу, результат свяжите с той же переменной. 4. Разделите var_int на var_float, а затем var_big на var_float. Результат данных выражений не привязывайте ни к каким переменным. 5. Измените значение переменной var_str на "NoNoYesYesYes". При формировании нового значения используйте операции конкатенации и повторения строки. 6. Выведите значения всех переменных.	ПК-3.З.1
2	1. Напишите программу (файл user.py), которая запрашивала бы у пользователя: - его имя (например, "What is your name?") - возраст ("How old are you?")	ПК-3.У.1

	- место жительства ("Where are you live?") После этого выводила бы три строки: "This is *имя*" "It is *возраст*" "(S)he live in *место_жительства*" Вместо *имя, возраст, место_жительства* должны быть данные, введенные пользователем. Примечание: можно писать фразы на русском языке, но если вы планируете стать профессиональным программистом, привыкайте к английскому. 2. Напишите программу (файл arithmetic.py), которая предлагала бы пользователю решить пример 4 * 100 - 54. Потом выводила бы на экран правильный ответ и ответ пользователя. Подумайте, нужно ли здесь преобразовывать строку в число. 3. Запросите у пользователя четыре числа. Отдельно сложите первые два и отдельно вторые два. Разделите первую сумму на вторую. Выведите результат на экран так, чтобы ответ содержал две цифры после запятой.	
3	1. Присвойте двум переменным любые числовые значения. 2. Используя переменные из п. 1, с помощью оператора **and** составьте два сложных логических выражения, одно из которых дает истину, другое – ложь. 3. Аналогично выполните п. 2, но уже с оператором **or**. 4. Попробуйте использовать в логических выражениях переменные строкового типа. Объясните результат. 5. Напишите программу, которая запрашивала бы у пользователя два числа и выводила бы True или False в зависимости от того, больше первое число второго или нет.	ПК-3.З.1
4	1. Напишите программу (файл TakeInput.py), которая просит пользователя что-нибудь ввести с клавиатуры. Если он вводит какие-нибудь данные, то на экране должно выводиться сообщение "ОК". Если он не вводит данные, а просто нажимает Enter, то программа ничего не выводит на экран. 2. Напишите программу (файл PosNeg.py), которая запрашивает у пользователя число. Если оно больше нуля, то в ответ на экран выводится число 1. Если введенное число не является положительным, то на экран должно выводиться - 1.	ПК-3.У.1
5	Напишите программу, которая запрашивает ввод двух значений. Если хотя бы одно из них не является числом, то должна выполняться конкатенация, т. е. соединение, строк. В остальных случаях введенные числа суммируются.	ПК-3.З.1
6	Напишите программу, которая запрашивает на ввод число. Если оно положительное, то на экран выводится цифра 1. Если число отрицательное, выводится -1. Если введенное число – это 0, то на экран выводится 0. Используйте в коде условный оператор множественного ветвления.	ПК-3.У.1
7	Используя цикл while, выведите на экран для числа 2 его	ПК-3.З.1

	степени от 0 до 20. Возведение в степень в Python обозначается как **.	
8	В программировании можно из одной функции вызывать другую. Для иллюстрации этой возможности напишите программу по следующему описанию. Основная ветка программы, не считая заголовков функций, состоит из одной строки кода. Это вызов функции test(). В ней запрашивается на ввод целое число. Если оно положительное, то вызывается функция positive(), тело которой содержит команду вывода на экран слова "Положительное". Если число отрицательное, то вызывается функция negative(), ее тело содержит выражение вывода на экран слова "Отрицательное". Понятно, что вызов test() должен следовать после определения функций. Однако имеет ли значение порядок определения самих функций? То есть должны ли определения positive() и negative() предшествовать test() или могут следовать после него? Проверьте вашу гипотезу, поменяв объявления функций местами. Попробуйте объяснить результат.	ПК-3.У.1
9	В языке Python можно внутри одной функции определять другую. Напишите программу по следующему описанию. В основной ветке программы вызывается функция cylinder(), которая вычисляет площадь цилиндра. В теле cylinder() определена функция circle(), вычисляющая площадь круга по формуле πr2. В теле cylinder() у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле 2πrh, или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции circle(). Как вы думаете, можно ли из основной ветки программы вызвать функцию, вложенную в другую функцию? Почему?	ПК-3.З.1
10	1. Напишите программу, в которой вызывается функция, запрашивающая с ввода две строки и возвращающая в программу результат их конкатенации. Выведите результат на экран. 2. Напишите функцию, которая считывает с клавиатуры числа и перемножает их до тех пор, пока не будет введен 0. Функция должна возвращать полученное произведение. Вызовите функцию и выведите на экран результат ее работы.	ПК-3.У.1
11	Напишите программу, в которой определены следующие четыре функции: 1. Функция getInput() не имеет параметров, запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку. 2. Функция testInput() имеет один параметр. В теле она проверяет, можно ли переданное ей значение преобразовать к целому числу. Если можно, возвращает	ПК-3.З.1

	логическое True. Если нельзя – False. 3. Функция strToInt() имеет один параметр. В теле преобразовывает переданное значение к целочисленному типу. Возвращает полученное число. 4. Функция printInt() имеет один параметр. Она выводит переданное значение на экран и ничего не возвращает. В основной ветке программы вызовите первую функцию. То, что она вернула, передайте во вторую функцию. Если вторая функция вернула True, то те же данные (из первой функции) передайте в третью функцию, а возвращенное третьей функцией значение – в четвертую.	
12	1. Напишите программу, которая циклично запрашивает у пользователя номера символов по таблице Unicode и выводит соответствующие им символы. Завершает работу при вводе нуля. 2. Напишите программу, которая измеряет длину введенной строки. Если строка длиннее десяти символов, то выносится предупреждение. Если короче, то к строке добавляется столько символов *, чтобы ее длина составляла десять символов, после чего новая строка должна выводиться на экран. 3. Напишите программу, которая запрашивает у пользователя шесть вещественных чисел. На экран выводит минимальное и максимальное из них, округленные до двух знаков после запятой. Выполните задание без использования встроенных функций min() и max().	ПК-3.У.1
13	1. Используя функцию randrange() получите псевдослучайное четное число в пределах от 6 до 12. Также получите число кратное пяти в пределах от 5 до 100. 2. Напишите программу, которая запрашивает у пользователя границы диапазона и какое (целое или вещественное) число он хочет получить. Выводит на экран подходящее случайное число.	ПК-3.З.1
14	1. Напишите программу, которая запрашивает с ввода восемь чисел, добавляет их в список. На экран выводит их сумму, максимальное и минимальное из них. Для нахождения суммы, максимума и минимума воспользуйтесь встроенными в Python функциями sum(), max() и min(). 2. Напишите программу, которая генерирует сто случайных вещественных чисел и заполняет ими список. Выводит получившийся список на экран по десять элементов в ряд. Далее сортирует список с помощью метода sort() и снова выводит его на экран по десять элементов в строке. Для вывода списка напишите отдельную функцию, в качестве аргумента она должна принимать список.	ПК-3.У.1
15	1. Заполните список случайными числами. Используйте в коде цикл for, функции range() и randint(). 2. Если объект range (диапазон) передать встроенной в Python функции list(), то она преобразует его к списку.	ПК-3.З.1

	Создайте таким образом список с элементами от 0 до 100 и шагом 17. 3. В заданном списке, состоящем из положительных и отрицательных чисел, посчитайте количество отрицательных элементов. Выведите результат на экран. 4. Напишите программу, которая заполняет список пятью словами, введенными с клавиатуры, измеряет длину каждого слова и добавляет полученное значение в другой список. Например, список слов – [a; b].'yes', 'no', 'maybe', 'ok', 'what'], список длин – [a; b].3, 2, 5, 2, 4]. Оба списка должны выводиться на экран.	
16	1. Вводится строка, включающая строчные и прописные буквы. Требуется вывести ту же строку в одном регистре, который зависит от того, каких букв больше. При равном количестве преобразовать в нижний регистр. Например, вводится строка "HeLLo World", она должна быть преобразована в "hello world", потому что в исходной строке малых букв больше. В коде используйте цикл for, строковые методы upper() (преобразование к верхнему регистру) и lower() (преобразование к нижнему регистру), а также методы isupper() и islower(), проверяющие регистр строки или символа. 2. Строковый метод isdigit() проверяет, состоит ли строка только из цифр. Напишите программу, которая запрашивает с ввода два целых числа и выводит их сумму. В случае некорректного ввода программа не должна завершаться с ошибкой, а должна продолжать запрашивать числа. Обработчик исключений try-except использовать нельзя.	ПК-3.У.1
17	1. Чтобы избежать изменения исходного списка, не обязательно использовать кортеж. Можно создать его копию с помощью метода списка copy() или взять срез от начала до конца [a; b].:]. Скопируйте список первым и вторым способом и убедитесь, что изменение копий никак не отражается на оригинале. 2. Заполните один кортеж десятью случайными целыми числами от 0 до 5 включительно. Также заполните второй кортеж числами от -версия содержит текст уроков курса и решения с пояснениями к практическим работам5 до 0. Для заполнения кортежей числами напишите одну функцию. Объедините два кортежа с помощью оператора +, создав тем самым третий кортеж. С помощью метода кортежа count() определите в нем количество нулей. Выведите на экран третий кортеж и количество нулей в нем.	ПК-3.З.1
18	1. Создайте словарь, связав его с переменной school, и наполните данными, которые бы отражали количество учащихся в разных классах (1а, 1б, 2б, 6а, 7в и т. п.). Внесите изменения в словарь согласно следующему: а) в одном из классов изменилось количество учащихся, б) в школе появился новый класс, с) в школе был расформирован (удален) другой класс. Вычислите общее	ПК-3.У.1

	количество учащихся в школе. 2. Создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод items(), полученный объект dict_items передайте в написанную вами функцию, которая создает и возвращает новый словарь, "обратный" исходному, т. е. ключами являются строки, а значениями – числа.	
19	Создайте файл nums.txt, содержащий несколько чисел, записанных через пробел. Напишите программу, которая подсчитывает и выводит на экран общую сумму чисел, хранящихся в этом файле.	ПК-3.З.1
20	Напишите генератор списка, который заполняет список данными, которые вводит пользователь. Другой генератор списка должен преобразовывать данные в числа. Если какой-либо элемент первого списка нельзя преобразовать в число, то во второй список этот элемент попадать не должен.	ПК-3.У.1
21	Множество как структура данных подходит для хранения списка тегов. Если пользователь вводит новый тег, он добавляется во множество. Если вводит тег, который уже есть во множестве, дублирования не произойдет. Напишите программу, в которой пользователь вводит слова. Если слово до этого не вводилось, оно добавляется ко множеству. Если слово уже было введено ранее, то пользователь получает сообщение "уже есть".	ПК-3.З.1
22	1. Напишите одну строчку кода, в котором у пользователя запрашивается строка. Если он ничего не вводит и нажимает Enter, то соответствующей переменной присваивается вами заданная строка по умолчанию. 2. Напишите программу, в которой проверяется, содержится ли введенное значение в списке. Если оно не содержится, происходит выход из программы. Проверку и выход запишите в одну строку.	ПК-3.У.1
23	1. Создайте список лямбда- выражений. Одно складывает аргументы, другое – вычитает, третье – умножает, четвертое – делит. Запросите у пользователя два числа и в цикле "прогоните" их через все lambda-функции списка. 2. Дан список строк. Отсортируйте его сначала по последним буквам слов, потом по длине слов.	ПК-3.З.1
24	Напишите программу по следующему описанию. Есть класс "Воин". От него создаются два экземпляра юнита. Каждому устанавливается здоровье в 100 очков. В случайном порядке они бьют друг друга. Тот, кто бьет, здоровья не теряет. У того, кого бьют, оно уменьшается на 20 очков от одного удара. После каждого удара надо выводить сообщение, какой юнит атаковал, и сколько у противника осталось здоровья. Как только у кого-то заканчивается ресурс здоровья, программа завершается сообщением о том, кто одержал победу.	ПК-3.У.1
25	Помимо конструктора объектов в языках программирования есть обратный ему метод – деструктор. Он вызывается, когда объект не создается, а уничтожается.	ПК-3.З.1

	В языке программирования Python объект уничтожается, когда исчезают все связанные с ним переменные или им присваивается другое значение, в результате чего связь со старым объектом теряется. Удалить переменную можно с помощью команды языка del. В классах Python функцию деструктора выполняет метод __del__(). Напишите программу по следующему описанию: 1. Есть класс Person, конструктор которого принимает три параметра (не учитывая self) – имя, фамилию и квалификацию специалиста. Квалификация имеет значение заданное по умолчанию, равное единице. 2. У класса Person есть метод, который возвращает строку, включающую в себя всю информацию о сотруднике. 3. Класс Person содержит деструктор, который выводит на экран фразу "До свидания, …" (вместо троеточия должны выводиться имя и фамилия объекта). 4. В основной ветке программы создайте три объекта класса Person. Посмотрите информацию о сотрудниках и увольте самое слабое звено. 5. В конце программы добавьте функцию input(), чтобы скрипт не завершился сам, пока не будет нажат Enter. Иначе вы сразу увидите, как удаляются все объекты при завершении работы программы. В Python деструктор используется редко, так как интерпретатор и без него хорошо убирает "мусор".	
26	Разработайте программу по следующему описанию. В некой игре-стратегии есть солдаты и герои. У всех есть свойство, содержащее уникальный номер объекта, и свойство, в котором хранится принадлежность команде. У солдат есть метод "иду за героем", который в качестве аргумента принимает объект типа "герой". У героев есть метод увеличения собственного уровня. В основной ветке программы создается по одному герою для каждой команды. В цикле генерируются объекты-солдаты. Их принадлежность команде определяется случайно. Солдаты разных команд добавляются в разные списки. Измеряется длина списков солдат противоборствующих команд и выводится на экран. У героя, принадлежащего команде с более длинным списком, увеличивается уровень. Отправьте одного из солдат первого героя следовать за ним. Выведите на экран идентификационные номера этих двух юнитов.	ПК-3.У.1
27	Разработайте класс с "полной инкапсуляцией", доступ к атрибутам которого и изменение данных реализуются через вызовы методов. В объектно-ориентированном программировании принято имена методов для извлечения данных начинать со слова get (взять), а имена методов, в которых свойствам присваиваются значения, – со слова set (установить). Например, getField, setField.	ПК-3.З.1

28	class WinDoor:
	  def __init__(self, x, y):
	  self.square = x * y
	
	class Room:
	  def __init__(self, x, y, z):
	  self.square = 2 * z * (x + y)
	  self.wd = []
	  def addWD(self, w, h):
	  self.wd.append(WinDoor(w, h))
	  def workSurface(self):
	  new_square = self.square
	  for i in self.wd:
	  new_square -= i.square
	  return new_square
	r1 = Room(6, 3, 2.7)
	print(r1.square) # выведет 48.6
	r1.addWD(1, 1)
	r1.addWD(1, 1)
	r1.addWD(1, 2)
	print(r1.workSurface()) # выведет 44.6
	
	Приведенная выше программа имеет ряд недочетов и недоработок. Требуется исправить и доработать, согласно следующему плану. При вычислении оклеиваемой поверхности мы не "портим" поле self.square. В нем так и остается полная площадь стен. Ведь она может понадобиться, если состав списка wd изменится, и придется заново вычислять оклеиваемую площадь. Однако в классе не предусмотрено сохранение длин сторон, хотя они тоже могут понадобиться. Например, если потребуется изменить одну из величин у уже существующего объекта. Площадь же помещения всегда можно вычислить, если хранить исходные параметры. Поэтому сохранять саму площадь в поле не обязательно. Исправьте код так, чтобы у объектов Room были только четыре поля – width, lenght, height и wd. Площади (полная и оклеиваемая) должны вычислять лишь при необходимости путем вызова методов.
	Программа вычисляет площадь под оклейку, но ничего не говорит о том, сколько потребуется рулонов обоев. Добавьте метод, который принимает в качестве аргументов длину и ширину одного рулона, а возвращает количество необходимых, исходя из оклеиваемой площади.
	Разработайте интерфейс программы. Пусть она запрашивает у пользователя данные и выдает ему площадь оклеиваемой поверхности и количество необходимых рулонов.
		ПК-3.У.1

29	Напишите класс Snow по следующему описанию. В конструкторе класса инициируется поле, содержащее количество снежинок, выраженное целым числом. Класс включает методы перегрузки арифметических	ПК-3.З.1

	операторов: __add__() – сложение,__sub__() – вычитание, __mul__() – умножение, __truediv__() – деление. В классе код этих методов должен выполнять увеличение или уменьшение количества снежинок на число n или в n раз. Метод __truediv__() перегружает обычное (/), а не целочисленное (//) деление. Однако, пусть в методе происходит округление значения до целого числа. Класс включает метод makeSnow(), который принимает сам объект и число снежинок в ряду, а возвращает строку вида "*****\n. Функция repr() n*****\n. Функция repr() n*****…", где количество снежинок между 'a), поэтому вызывать его надо \n. Функция repr() n'a), поэтому вызывать его надо равно переданному аргументу, а количество рядов вычисляется, исходя из общего количества снежинок. Вызов объекта класса Snow в нотации функции с одним аргументом, должен приводить к перезаписи значения поля, в котором хранится количество снежинок, на переданное в качестве аргумента значение.	

30	from math import pi
	class Cylinder:
	  @staticmethod
	  def make_area(d, h):
	  circle = pi * d ** 2 / 4
	  side = pi * d * h
	  return round(circle*2 + side, 2)
	  def __init__(self, diameter, high):
	  self.dia = diameter
	  self.h = high
	  self.area = self.make_area(diameter, high)
	a = Cylinder(1, 2)
	print(a.area)
	print(a.make_area(2, 2))
	
	Приведенный код - плохой. Мы можем менять значения полей dia и h объекта за пределами класса простым присваиванием (например, a.dia = 10). При этом площадь никак не будет пересчитываться. Также мы можем назначить новое значение для площади, как простым присваиванием, так и вызовом функции make_area() с последующим присваиванием. Например, a.area = a.make_area(2, 3). При этом не меняются высота и диаметр.
	Защитите код от возможных логических ошибок следующим образом:
	• Свойствам dia и h объекта по прежнему можно выполнять присваивание за пределами класса. Однако при этом "за кулисами" происходит пересчет площади, т. е. изменение
	значения area.
	• Свойству area нельзя присваивать за пределами класса. Можно только получать его значение.
	Подсказка: вспомните про метод __setattr__(plustilino)),
		ПК-3.У.1

	упомянутый в уроке про инкапсуляцию.	
31	Напишите класс-итератор, объекты которого генерируют случайные числа в количестве и в диапазоне, которые передаются в конструктор.	ПК-3.З.1
32	Напишите класс-генератор, объекты которого генерируют случайные числа в количестве и в диапазоне, которые передаются в конструктор.	ПК-3.У.1

Перечень тем для курсового проектирования/выполнения курсовой работы представлены в таблице 17.

Таблица 17 – Перечень тем для курсового проектирования/выполнения курсовой работы

№ п/п	Примерный перечень тем для курсового проектирования/выполнения курсовой работы
	Учебным планом не предусмотрено

Вопросы для проведения промежуточной аттестации в виде тестирования представлены в таблице 18.

Таблица 18 – Примерный перечень вопросов для тестов

№ п/п	Примерный перечень вопросов для тестов	Код индикатора
1	Какие типы данных есть в Python в соответствии с их наименованиями? 1) int 2) real 3) float 4) bool 5) complex 6) string 7) str	ПК-3.З.1

2	В каком случае будет выполнено тело if?
	1) if (True):
	body
	2) if (0):
	body
	3) c = 10
	if (c>5):
	body
	4) a = 0
	if (a):
	body
		ПК-3.З.1

3	Какая конструкция соответствует бесконечному циклу и будет постоянно выполнять тело цикла while?
	1) while True:
	body
	2) c = 0
	while (c<10):
	body
	3) c1 = 0
	c2 = 5
	while ((c1<2) and (c2<4)):
	body
		ПК-3.У.1

	4) c = 0 while (c<10): body c+=1	
4	Какой цикл for выполнит 10 итераций цикла? В телах цикла не предусмотрены конструкции выхода из тела. 1) for i in range(10) 2) for j in range(1,10) 3) for x in range(-5,5) 4) for a in range(0,20,2)	ПК-3.У.1
5	Какой оператор позволяет выйти из тела цикла? 1) exit 2) out 3) break	ПК-3.З.1
6	Какое ключевое слово используется в роли «затычки», устанавливаемой внутри какого либо тела? 1) plug 2) pass 3) wait 4) nobody	ПК-3.З.1
7	Какие значения инициализированы в функции def func(x=0,y=0) локально, если её вызов осуществился как func(5)? 1) x = 0, y = 0 2) x = 5, y = 5 3) x= 5, y =0 4) x = 0, y = 5	ПК-3.З.1
8	Как обратиться к значению равному 29 в словаре d = {'Белай' : 29, 'Сергеев' : 45, Комаров' : 35}? 1) d['Белай'] 2) d[0] 3) d[29] 4) d{'Белай'} 5) d('Белай')	ПК-3.У.1
9	К строкам применяется метод isalpha(). Метод какой строки вернёт значение True? 1) s='Белай Василий Евгеньевич' s.isalpha() 2) n="130302" n.isalpha() 3) name="Вячеслав" name.isalpha()	ПК-3.У.1
10	Какое выражение вернёт значение True? 1) True and True or False 2) False and False or True 3) True or True and False	ПК-3.З.1
11	Как обратиться к переменной со значением 25 внутри списка l = [[2,1,3],[412,512,23],[[65,27],[25,76]]]? 1) l[2] 2) l[2][2] 3) l[2][2][1] 4) l[2][2][0]	ПК-3.З.1
12	Имеется кортеж:	ПК-3.У.1

	t = (1, 3.14, True, "Python", [2,1,3], {'one' : 1, 'two' : 2, 'three' : 3}). Внутри кортежа хранятся переменные разных типов данных под индексами: t[0] – int, t[1] – float, t[2] – bool, t[3] – str, t[4] – list, t[5] – dict. Переменные под какими индексами можно изменить внутри кортежа, не выполняя операции по его преобразованию в другие коллекции? 1) 0 2) 1 3) 2 4) 3 5) 4 6) 5	

Примечание:
Задание 1 типа с выбором одного верного ответа из четырех предложенных и обоснованием выбора:
Полное совпадение с верным ответом – 1 балл.
Неверный ответ или его отсутствие – 0 баллов.

Задание 2 типа с выбором нескольких вариантов ответа из предложенных и развернутым обоснованием выбора:
Полное совпадение с верным ответом 1 балл.
Отсутствие минимум одного правильно ответа или полное отсутствует ответа – 0 баллов.

Задание 3 типа на установление соответствия:
Полное совпадение с верным ответом - 1 балл.
Неверное сопоставление ответов или отсутствие ответа – 0 баллов.

Задание 4 типа на установление последовательности:
Полное правильное совпадение очередности ответов - 1 баллом
Нарушение правильного порядка ответов или отсутствие ответа – 0 баллов.

Задание 5 типа с развернутым ответом:
Правильный ответ за задание оценивается - 3 балла.
Если допущена одна ошибка \ неточность \ ответ правильный, но не полный - 1 балл.
Если допущено более 1 ошибки \ ответ неправильный \ ответ отсутствует – 0 баллов.

Перечень тем контрольных работ по дисциплине обучающихся заочной формы обучения, представлены в таблице 19.

Таблица 19 – Перечень контрольных работ

№ п/п	Перечень контрольных работ
	Не предусмотрено

10.4. Методические материалы, определяющие процедуры оценивания индикаторов, характеризующих этапы формирования компетенций, содержатся в локальных нормативных актах ГУАП, регламентирующих порядок и процедуру проведения текущего контроля успеваемости и промежуточной аттестации обучающихся ГУАП.

11. Методические указания для обучающихся по освоению дисциплины

11.1. Методические указания для обучающихся по освоению лекционного материала .

Основное назначение лекционного материала – логически стройное, системное, глубокое и ясное изложение учебного материала. Назначение современной лекции в рамках дисциплины не в том, чтобы получить всю информацию по теме, а в освоении фундаментальных проблем дисциплины, методов научного познания, новейших достижений научной мысли. В учебном процессе лекция выполняет методологическую, организационную и информационную функции. Лекция раскрывает понятийный аппарат конкретной области знания, её проблемы, дает цельное представление о дисциплине, показывает взаимосвязь с другими дисциплинами.

Планируемые результаты при освоении обучающимися лекционного материала:

– получение современных, целостных, взаимосвязанных знаний, уровень которых определяется целевой установкой к каждой конкретной теме;

– получение опыта творческой работы совместно с преподавателем;

– развитие профессионально-деловых качеств, любви к предмету и самостоятельного творческого мышления.

– появление необходимого интереса, необходимого для самостоятельной работы;

– получение знаний о современном уровне развития науки и техники и о прогнозе их развития на ближайшие годы;

– научиться методически обрабатывать материал (выделять главные мысли и положения, приходить к конкретным выводам, повторять их в различных формулировках);

– получение точного понимания всех необходимых терминов и понятий.

Лекционный материал может сопровождаться демонстрацией слайдов и использованием раздаточного материала при проведении коротких дискуссий об особенностях применения отдельных тематик по дисциплине.

Структура предоставления лекционного материала:

– Лекционный материал представляется преподавателям устно, а также публикуется в сервисе «Личный кабинет».

11.2. Методические указания для обучающихся по выполнению лабораторных работ

В ходе выполнения лабораторных работ обучающийся должен углубить и закрепить знания, практические навыки, овладеть современной методикой и техникой эксперимента в соответствии с квалификационной характеристикой обучающегося. Выполнение лабораторных работ состоит из экспериментально-практической, расчетно-аналитической частей и контрольных мероприятий.

Выполнение лабораторных работ обучающимся является неотъемлемой частью изучения дисциплины, определяемой учебным планом, и относится к средствам, обеспечивающим решение следующих основных задач обучающегося:

– приобретение навыков исследования процессов, явлений и объектов, изучаемых в рамках данной дисциплины;

– закрепление, развитие и детализация теоретических знаний, полученных на лекциях;

– получение новой информации по изучаемой дисциплине;

– приобретение навыков самостоятельной работы с лабораторным оборудованием и приборами.

Задание и требования к проведению лабораторных работ

Лабораторные работы следует выполнять в ходе прохождения курса «Промышленная робототехника», внимательно разбирая представленный методический материал преподавателем в установленные в «Личном кабинете ГУАП» сроки.

Структура и форма отчета о лабораторной работе

Структура и форма отчёта выполнения лабораторной работы приведена на сайте университета.

https://guap.ru/standart/doc

Требования к оформлению отчета о лабораторной работе

В отчёте представляется спроектированные согласно заданию отдельные единицы ИПС, так и сама ИПС, а также её математическая модель, поясняющая работу проектируемой ИПС.

11.3. Методические указания для обучающихся по прохождению самостоятельной работы

В ходе выполнения самостоятельной работы, обучающийся выполняет работу по заданию и при методическом руководстве преподавателя, но без его непосредственного участия.

В процессе выполнения самостоятельной работы, у обучающегося формируется целесообразное планирование рабочего времени, которое позволяет им развивать умения и навыки в усвоении и систематизации приобретаемых знаний, обеспечивает высокий уровень успеваемости в период обучения, помогает получить навыки повышения профессионального уровня.

Методическими материалами, направляющими самостоятельную работу обучающихся являются:

– учебно-методический материал по дисциплине;

11.4. Методические указания для обучающихся по прохождению текущего контроля успеваемости.

Текущий контроль успеваемости предусматривает контроль качества знаний обучающихся, осуществляемого в течение семестра с целью оценивания хода освоения дисциплины.

Формы текущего контроля и основные требования:

устный опрос. Данная форма может осуществляться преподавателем на каждом занятии или периодически, может иметь различную продолжительность. Цель устного опроса – проверка усвоения обучающимся основных терминов, понятий и принципов взаимодействия. Устный опрос может относиться к материалу темы, рассматриваемой на данном занятии, а также к материалам предыдущих лекций. Вопросы могут задаваться устно или в виде системы карточек, по списку каждому студенту или всем в формате «мозгового штурма». Количество максимальных баллов и продолжительность времени для ответов определяется непосредственно преподавателем. По усмотрению преподавателя устный опрос может быть заменен тестированием.

тестирование. Тестирование в качестве текущего контроля успеваемости не является обязательной формой работы и предлагается обучающимся по усмотрению преподавателя. Цель тестирования – мониторинг уровня усвоения теоретического материала, а также качества самостоятельной работы, выявление неуспевающих студентов.

Тестирование может проводиться периодически (один или два раза в семестр), а может – на каждом занятии, на усмотрение преподавателя. Текущее тестирование может быть организовано на дистанционной платформе LMS. Тестируемые темы заранее озвучиваются обучающимся или обозначаются в начале курса преподавателем.

11.5. Методические указания для обучающихся по прохождению промежуточной аттестации.

Промежуточная аттестация обучающихся предусматривает оценивание промежуточных и окончательных результатов обучения по дисциплине. Она включает в себя:

– зачет – это форма оценки знаний, полученных обучающимся в ходе изучения учебной дисциплины в целом или промежуточная (по окончании семестра) оценка знаний обучающимся по отдельным разделам дисциплины с аттестационной оценкой «зачтено» или «не зачтено».

Лист внесения изменений в рабочую программу дисциплины

Дата внесения изменений и дополнений. Подпись внесшего изменения	Содержание изменений и дополнений	Дата и № протокола заседания кафедры	Подпись зав. кафедрой