Python Basics

Python PIP

PIP — это менеджер пакетов Python или модулей, если хотите.
Что такое пакет?

Пакет содержит все файлы, необходимые для модуля.
Модули — это библиотеки кода Python, которые вы можете включить в свой проект.

Pipenv: Python Dev Workflow

Pipenv — это инструмент, цель которого — привнести лучшее из мира пакетов.
Он автоматически создает виртуальную среду для ваших проектов и управляет ею, а также добавляет/удаляет пакеты из ваших `Pipfile` при установке/удалении пакетов. Он также генерирует файл everimportant `Pipfile.lock`, который используется для создания детерминированных сборок.
Pipenv в первую очередь предназначен для предоставления пользователям и разработчикам приложений простого способа настройки рабочей среды.
https://pipenv.pypa.io/en/latest/

Variables

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Data Types

In programming, data type is an important concept. Variables can store data of different types, and different types can do different things.
Python has the following data types built-in by default, in these categories:

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`
None Type:	`NoneType`

В языке программирования Python существует четыре типа данных коллекций:

- **[Список]** представляет собой набор, который упорядочен и может быть изменен. Позволяет дублировать участников.
- **(Кортеж)** — это упорядоченная и неизменяемая коллекция. Позволяет дублировать участников.
- **{Набор}** — это неупорядоченная, неизменяемая* и неиндексированная коллекция. Нет повторяющихся членов.
- **Словарь** представляет собой сборник упорядоченный** и изменяемый. Нет повторяющихся членов.

List Methods:

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Tuple Metods:

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Set Methods

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

Keep ONLY the Duplicates

The `intersection_update()` method will keep only the items that are present in both sets.

Keep All, But NOT the Duplicates

The `symmetric_difference_update()` method will keep only the elements that are NOT present in both sets.

Convert set to list:

```
# Python3 program to convert a
# set into a list
my_set = {'Geeks', 'for', 'geeks'}

s = list(my_set)
print(s)
```

Convert list to set:

```
# sample_list is defined list
sample_list = [1,2,3,'seeker',3,7.5]
# set() to convert list to set
sample_set  = set(sample_list)
print(sample_set) #printing set
```

Dictionary Methods

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Пространство имен

Пространство имен - это в основном система, которая гарантирует, что все имена в программе уникальны и могут использоваться без каких-либо конфликтов. Возможно, вы уже знаете, что все в Python строки, списки, функции и т.д. - это объекты.

- Локальное пространство имен: это пространство имен содержит локальные имена внутри функции. Это пространство имен создается при вызове функции и продолжается до тех пор, пока функция не вернется.
- Глобальное пространство имен: это пространство имен, которое включает имена из различных импортированных модулей, которые вы используете в проекте. Оно создается, когда модуль включен в проект, и оно существует до завершения скрипта.
- Встроенное пространство имен: это пространство имен содержит встроенные функции и встроенные имена исключений.

Python Functions

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.

Python Inheritance

Наследование позволяет нам определить класс, который наследует все методы и свойства другого класса.

Родительский класс — это наследуемый класс, также называемый базовым классом.

Дочерний класс — это класс, который наследуется от другого класса, также называемого производным классом.

Polymorphism in Python

Полиморфизм происходит от греческих слов Poly (много) и morphism (формы). Это означает, что одно и то же имя функции может использоваться для разных типов. Это делает программирование более интуитивным и простым.

Дочерний класс наследует все методы родительского класса. Однако в некоторых ситуациях метод, унаследованный от родительского класса, не совсем подходит для дочернего класса. В таких случаях вам придется повторно реализовать метод в дочернем классе.

Environment Variables

https://www.twilio.com/blog/environment-variables-python

Переменные среды предоставляют отличный способ настроить приложение Python для использования секретов и паролей, а так же, избавляя от необходимости редактировать исходный код при изменении конфигурации. Общие элементы конфигурации, которые часто передаются в приложение через переменные среды, — это сторонние ключи API, сетевые порты, серверы баз данных и любые настраиваемые параметры, которые могут потребоваться вашему приложению для правильной работы.

Как получить доступ к переменным среды из Python?

Environment variables are implemented through the os package, specifically os.environ.

To see all environment variables on your system just call it:

```
import os
print(os.environ)
```

```
import os
stage = os.environ['STAGE'].upper()
output = f"We're running in {stage}"

if stage.startswith('PROD'):
    output = "DANGER!!! - " + output
print(output)
```

Decorators:

Декораторы позволяют нам обернуть другую функцию, чтобы расширить поведение обернутой функции, не изменяя ее навсегда.

```
def decator(func):
    def inner():
        print('start decorator...')
        func()
        print('finish decorator...')
    return inner

def say():
    print("Hello world!")

say = decator(say)    # функция say поступает в качестве входного аргумента внутрь функции decator
# То есть я должен вызвать функцию и ЕЁ ЖЕ паередать внутрь другой функции в качестве аргумента!
say()                 # Тут я уже вызываю саму функцию, которую я декорирую и получаю результат!

Output:
start decorator
Hello world!
finish decorator

# Теперь в имени say() хранится ссылка не на def say(), а на def inner!(которую мы вернули из нашего
декоратора)
# Для проверки этого:
print(say)

# Результат вызываем по тому же имени функции, которую мы декорируем!
```

https://www.youtube.com/watch?v=Va-ovLxHmus

По сути - это функция, которая принимает другую функцию и возвращает функцию! И изменяет её поведение, если нужно.

Все аргументы желательно пробрасывать через *args и **kwargs - потому как мы не знаем какое кол-ство аргументов будет принимать наша декорируемая функция.

Debugging in Python

https://realpython.com/python-debugging-pdb/
https://docs.python.org/3/library/pdb.html

Модуль `pdb` определяет интерактивный отладчик исходного кода для программ Python. Он поддерживает установку (условных) точек останова и пошаговое выполнение на уровне строки исходного кода, проверку кадров стека, листинг исходного кода и оценку произвольного кода Python в контексте любого кадра стека.

Python FLASK

Flask — фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2. Относится к категории так называемых микрофреймворков — минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности.