Chapter 1 - Review of Basic Concepts

1. Syntax of a Python Code

Python is an interpreted, high-level, dynamically typed programming language. It uses indentation instead of curly brackets for code blocks.

Example:

```python
# Basic Python syntax
name = "John Doe"  # Variable assignment
age = 25           # Integer variable

if age >= 18:
    print(f"{name} is an adult.")
else:
    print(f"{name} is a minor.")
```

2. Lists, Tuples, Sets, Dictionaries

Lists:

A **list** is an ordered, mutable collection of elements.

```python
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")
print(fruits)  # ['apple', 'banana', 'cherry', 'orange']
```

Tuples:

A **tuple** is an immutable, ordered collection of elements.

```python
tuple_example = (1, 2, 3, "Python")
print(tuple_example[1])  # 2
```

Sets:

A **set** is an unordered collection of unique elements.

```
unique_numbers = {1, 2, 3, 3, 2, 1}
print(unique_numbers)  # {1, 2, 3}
```

Dictionaries:

A **dictionary** stores key-value pairs.

```
student = {"name": "Alice", "age": 22, "grade": "A"}
print(student["name"])  # Alice
```

3. Functions

Functions allow code reusability.

```
def greet(name):
    return f"Hello, {name}!"

print(greet("John"))  # Hello, John!
```

4. If-Elif-Else Statements

Conditional statements control the flow of execution.

```
x = 10
if x > 10:
    print("Greater than 10")
elif x == 10:
    print("Equal to 10")
else:
    print("Less than 10")
```

5. For / While Loops

For Loop:

```
for i in range(5):
    print(i)  # 0, 1, 2, 3, 4
```

While Loop:

```
count = 0
while count < 5:
    print(count)
    count += 1
```

6. Modules

A module is a file containing Python definitions and statements.

```
# math module example
import math
print(math.sqrt(25))  # 5.0
```

7. Package Managers

PyPI:

The **Python Package Index (PyPI)** hosts third-party Python packages.

Pip:

pip is Python's package manager.

```
pip install requests
```

Conda:

A package manager for Python and other languages.

```
conda install numpy
```

8. Regular Expressions

Regular expressions (regex) help in pattern matching.

```
import re
pattern = r"\d+"
result = re.findall(pattern, "User123 has 456 points")
print(result)  # ['123', '456']
```

9. Django Installation and Setup

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites.

Installing Django

Ensure you have Python installed (version 3.x recommended). Then install Django using pip:

```
pip install django
```

Verify the installation:

```
django-admin --version
```

Creating a Django Project

Run the following command to start a new project:

```
django-admin startproject myproject
```

Navigate into the project directory:

```
cd myproject
```

Run the development server:

```
python manage.py runserver
```

This will start the Django server, and you can access the default Django welcome page at `http://127.0.0.1:8000/`.

Creating a Django App

Inside your project directory, create a new app:

```
python manage.py startapp myapp
```

Register the app in `settings.py` by adding `'myapp'` to the `INSTALLED_APPS` list.

Defining a Model

In `myapp/models.py`, define a simple model:

```
from django.db import models

class Item(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
```

Run migrations:

```
python manage.py makemigrations
python manage.py migrate
```

Creating a View

In `myapp/views.py`, create a simple view:

```
from django.http import HttpResponse

def home(request):
    return HttpResponse("Hello, Django!")
```

Configuring URLs

In `myapp/urls.py`, define a URL pattern:

```
from django.urls import path
from .views import home

urlpatterns = [
    path('', home, name='home'),
]
```

Include this in the project's main `urls.py`:

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapp.urls')),
]
```

Running the Server

Start the Django server again:

```
python manage.py runserver
```

Now visit `http://127.0.0.1:8000/` in your browser to see your first Django view in action.

Exercises

1. Create a list of numbers from 1 to 10 and print only even numbers.
2. Write a function that takes a name as input and returns "Hello, !".
3. Write a program that asks for a user's age and prints if they are a minor or an adult.
4. Write a while loop that prints numbers from 5 to 0.
5. Use the `math` module to compute the square root of a number entered by the user.
6. Use a regular expression to extract all email addresses from a given text.
7. Install Django and create a simple project with an app that returns "Welcome to Django!" on the homepage.