PYTHON PROGRAMMING - COMPREHENSIVE NOTES

1. Introduction to Python

Python is a high-level, interpreted programming language with dynamic semantics. It supports multiple programming paradigms and is known for its easy syntax.

Example:

```
print("Hello, World!")
```

2. Variables and Data Types

Python supports various data types like int, float, str, list, tuple, dict, and set.

Example:

```
x = 10
name = "Alice"
```

3. Control Flow (if, elif, else)

Conditional statements help you make decisions in code.

Example:

```
if x > 0:
print("Positive")
elif x == 0:
print("Zero")
else:
print("Negative")
```

4. Loops (for, while)

Used for iterating over sequences or repeating a block of code.

Example:

```
for i in range(5):
print(i)
```

5. Functions

Functions are blocks of reusable code.

Example:

```
def greet(name):
return "Hello " + name

print(greet("Alice"))
```

6. Data Structures

Includes lists, tuples, dictionaries, and sets.

List Example:

```
numbers = [1, 2, 3]
print(numbers[0])
```

7. Object-Oriented Programming

Encapsulation of data and functions into classes and objects.

Example:

```
class Person:
def __init__(self, name):
self.name = name
def speak(self):
print("Hi, I am " + self.name)

p = Person("Alice")
p.speak()
```

8. File Handling

Used to read and write files.

Example:

```
with open("file.txt", "r") as file:
data = file.read()
print(data)
```

9. Exception Handling

Used to catch and handle errors.

Example:

```
try:
x = 1 / 0
except ZeroDivisionError:
```

```
print("Cannot divide by zero")
```

10. Modules and Packages

Organize and reuse code.

Example:

```
import math
print(math.sqrt(16))
```

11. Libraries (NumPy, Pandas)

Powerful libraries for data processing.

Example using pandas:

```
import pandas as pd
df = pd.DataFrame({"name": ["Alice", "Bob"], "age": [25, 30]})
print(df)
```

12. GUI Programming (Tkinter)

Used for creating graphical user interfaces.

Example:

```
from tkinter import *
window = Tk()
window.title("GUI")
window.mainloop()
```