Hila Levitan

Indentation

Python uses indentation (spaces or tabs) to define code blocks. Consistency is essential.

```
if True:
    print("This is indented correctly.")
    print("Still inside the if block.")
```

Input

Get user input using the input() method

```
user_name = input("Enter you name: ")
user_age = int(input("Enter you age: "))
```

if/elif/else

`elif`: Short for "else if," it allows you to add additional conditions in an `if` statement chain

```
age = 20
if age < 18:
    print("You are a minor.")
elif age < 65:
    print("You are an adult.")
else:
    print("You are a senior.")
```

for ... in Loop

`for` loops in Python iterate over items in a sequence (e.g., list, string)

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

for ... in Loop

we can loop through a range of numbers using the range() method

```
for number in range(6):
    print(number)

for number in range(3,9,2):
    print(number)
```

Logical Operators: not, or, and

```
age = 25
if age > 18 and age < 30:
    print("You're in your twenties.")
if not age > 60:
    print("You're not a senior yet.")
```

def (Defining Functions)

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

greet("Alice") #prints "Hello, Alice!"
```

Exceptions

Python uses `try/except` to handle errors gracefully without stopping the program.

```
try:
    user_input = int(input('Give me a number: '))
except ValueError:
    print('That is not a number!')
except:
    print("something went worng :(")
finally:
    print("this block is executed anyways")
```

Classes and `__init__` (Object-Oriented Programming)

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def print_age(self):
        print(f"Age: {self.age}")

person = Person("Alice", 30)
person.print_age()
```

Random and list

```
import random

numbers = [1, 2, 3, 4, 5]
print(random.choice(numbers))  # Randomly picks an item from the list
numbers.append(6)
print(numbers)

 #Return random integer in range [a, b], including both end points.
random_index = random.randint(0,len(numbers)-1)
```

File Reading

Note: Using `with` ensures the file is properly closed after reading.

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