Notebook For Learning Python Programming Language

Module 2

Control Flow and Data Structures

Control Flow: Control flow refers to the order in which statements and blocks of code are executed. Python provides several control flow constructs, such as if statements, for loops, and while loops, that allow you to write code that makes decisions based on certain conditions or to repeat a block of code for a specified number of times.

Data Structures: Data structures are used to store and organize data in an efficient manner. Python provides several built-in data structures, such as lists, tuples, dictionaries, and sets, that allow you to store, retrieve, and manipulate data.

- Lists are ordered sequences of elements that can be of different data types and can be modified.
- Tuples are ordered, immutable sequences of elements that can be of different data types.
- Dictionaries are unordered collections of key-value pairs.
- Sets are unordered collections of unique elements.

Conditional statements and Loops

Conditional statements and loops are control structures in Python that allow you to execute different blocks of code based on certain conditions or to repeat a block of code for a specified number of times or until a certain condition is met.

Conditional statements: Conditional statements allow you to execute different code blocks based on the truthiness of an expression. In Python, you can use the `if`, `elif` (else-if), and `else` keywords to implement conditional statements.

In [ ]:
```
age = 25
if age >= 18:
    print("You are an adult.")
else:
    print("You are a minor.")
```

Loops: Loops allow you to repeat a block of code for a specified number of times or until a certain condition is met. In Python, you can use `for` loops to iterate over a sequence of items and `while` loops to repeat a block of code until a condition is met.

In [ ]:
```
# Using a for loop to print the numbers from 0 to 9
for i in range(10):
    print(i)
```

```
# Using a while loop to print the numbers from 0 to 9
i = 0
while i < 10:
    print(i)
    i += 1
```

In [ ]:
```
# While loop example
counter = 0
while counter < 10:
    print(counter)
    counter += 1

# For loop example
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    print(num)

for i, num in enumerate(numbers):
    print(i, num)
```

In the above code, the `while` loop runs as long as the value of `counter` is less than 10. At each iteration, the value of `counter` is incremented by 1, and the new value is printed. The `for` loop, on the other hand, iterates over a list of numbers and prints the value of each element. When used with the `enumerate` function, the `for` loop also prints the index of each element.

Lists, tuples, and dictionaries

Lists, Tuples, and Dictionaries are built-in data structures in Python that allow you to store, retrieve, and manipulate data in an efficient manner.

Lists: Lists are ordered sequences of elements that can be of different data types and can be modified. Lists are defined using square brackets `[]`.

In [ ]:
```
# Example of a list
numbers = [1, 2, 3, 4, 5]
print(numbers) # Output: [1, 2, 3, 4, 5]

# Adding an element to a list
numbers.append(6)
print(numbers) # Output: [1, 2, 3, 4, 5, 6]

# Removing an element from a list
numbers.remove(3)
print(numbers) # Output: [1, 2, 4, 5, 6]
```

Tuples: Tuples are ordered, immutable sequences of elements that can be of different data types. Tuples are defined using parentheses `()`.

In [ ]:
```
# Example of a tuple
person = ("John Doe", 35)
print(person) # Output: ('John Doe', 35)

# Trying to modify a tuple
# person[0] = "Jane Doe" # This will raise an error because tuples are immutable.
```

Dictionaries: Dictionaries are unordered collections of key-value pairs. Dictionaries are defined using curly braces `{}`.

In [ ]:
```python
# Example of a dictionary
person = {"name": "John Doe", "age": 35}
print(person) # Output: {'name': 'John Doe', 'age': 35}

# Accessing a value in a dictionary
print(person["name"]) # Output: John Doe

# Adding a key-value pair to a dictionary
person["city"] = "New York"
print(person) # Output: {'name': 'John Doe', 'age': 35, 'city': 'New York'}

# Removing a key-value pair from a dictionary
del person["age"]
print(person) # Output: {'name': 'John Doe', 'city': 'New York'}
```

Functions and Modules

In Python, functions are blocks of code that can be called and executed multiple times within your program. Functions allow you to encapsulate logic and reuse it throughout your code, making your code more readable and maintainable. Functions are defined using the `def` keyword, followed by the function name and a list of parameters in parentheses.

Modules are collections of functions, classes, and variables that can be imported into other Python scripts to use. Modules allow you to organize your code into separate files and reuse the code across multiple projects.

Here's an example of how to create and use functions in Python:

In [ ]:
```python
# Function definition
def say_hello(name):
    """This function prints a greeting"""
    print("Hello, {}!".format(name))

# Function call
say_hello("John")

# Function with a return value
def add(a, b):
    """This function adds two numbers and returns the result"""
    return a + b

result = add(3, 5)
print(result)

# Importing a module
import math

print(math.pi)
```

In the above code, we defined two functions: `say_hello` and `add`. The first function takes a single argument, `name`, and prints a greeting. The second function takes two arguments, `a` and `b`, and returns the sum of the two numbers. To import the `math` module, we use the `import` keyword, which gives us access to a number of mathematical functions and constants, such as `pi`.

Exercises

Here are a few exercise scripts for practicing conditional statements and loops in Python:

1. Write a program that prints the numbers from 1 to 10 using a for loop.
2. Write a program that prints the numbers from 1 to 10 using a while loop.
3. Write a program that prints the even numbers from 1 to 10 using a for loop.
4. Write a program that prints the odd numbers from 1 to 10 using a while loop.
5. Write a program that takes a number from the user and checks if it is positive, negative, or zero using an if-elif-else statement.
6. Write a program that takes two numbers from the user and prints the largest of the two using an if-else statement.
7. Write a program that takes a list of numbers from the user and prints the sum of all the numbers using a for loop.
8. Write a function to remove duplicates from a list while preserving the order.
9. Write a function to find the second largest number in a list.
10. Write a function to count the frequency of words in a string using a dictionary.
11. Write a function to find the common elements in two dictionaries.
12. Write a function to find the top N items in a dictionary based on their values.

In [ ]: