Types - MAD 102 Week 2 Notes

Hia Al Saleh

September 11th, 2024

Contents

1 Getting Input

- Information required for the program to operate comes as input.
- Use `input()` function to capture input as text (string).
- Example:

```
name = input("Enter your name")
print("Hello", name)
```

1.1 Input Details

- The input string can be assigned to a variable.
- The argument in `input()` represents prompt text displayed to the user.
- Input always returns a string, so numerical input needs conversion.

1.2 Converting Input Types

Explicit Conversion:

```
age = int(input("Enter your Age"))
print(type(age))  # Data Type: <class 'int'>
```

Implicit Conversion:

```
num1 = 1
num2 = 3.4
sum = num1 + num2  # Data Type: <class 'float'>
```

2 Outputting Information

- Use the `print()` function to display results in the console.
- Multiple arguments in `print()` are separated by commas and displayed with spaces in between.
- `print()` ends with a newline character by default.
- You can use `end=""` to avoid moving to the next line.

Example:

```
print("Hello", end='')
print(" World!")  # Output: Hello World!
```

3 Strings

- Strings are a sequence of characters enclosed in single or double quotes.
- Strings are immutable (cannot be changed once created).
- Access individual characters using indexing:

```
name = 'Luke'
print(name[1])  # Output: u
```

- Positive indices start from 0 (left to right); negative indices start from -1 (right to left).
- Strings can be concatenated using the `+` operator.

3.1 Formatted Strings

- Use f-strings with `f" "` to format strings with placeholders {}.
- Example:

```
name = "John"
age = 25
print(f"{name} is {age} years old.")
```

- Formatting options can be applied inside placeholders using `':'`. For example, formatting numbers.

4 Lists

- A list is a mutable, ordered collection of elements, defined with square brackets `[ ]`.
- Access list elements using their index:

```
my_list = [1, 2, 3]
print(my_list[0])  # Output: 1
```

- Lists can be modified, and new items can be added using the `append()` method.
- Items can be removed using `pop()` or `remove()` methods.

5 Tuples

- Tuples are immutable, ordered collections of elements, defined using parentheses `( )`.
- Example:

```
coordinates = (83.232, 32.321)
```

- Named tuples allow attributes to be accessed using dot notation.

6 Sets

- Sets are unordered collections of unique elements, defined using curly braces { }.
- No repeated elements are allowed.
- Add elements using the `add()` method and remove elements using `remove()` or `pop()`.

7 Dictionaries

- Dictionaries store key-value pairs, defined using curly braces { } with a colon separating keys and values.
- Access items using keys rather than indices.
- Modify dictionaries by assigning new values to keys, and remove entries using the `del` keyword.

```
nhteams = {
1926: 'Detroit Red Wings',
1979: 'Edmonton Oilers',
1927: 'Toronto Maple Leafs'
}
nhteams[1926]= 'Chicago Blackhawks'

print(nhteams)
```