Lecture 5 - Manipulating strings, string methods, dot notation

Learning outcomes

- At the end of this lecture, students should be able to:
  - use dot notation (using string methods with string instances)
  - use string methods: upper(), lower(), strip(), find(), rfind()
  - use the inbuilt functions: min(), max(), round(), abs()

Recap

- From lecture 4
  - Use the len() function to calculate how many characters are in a string
  - Obtain a single character from a string
  - Slice strings
  - Concatenate strings

words = "Prince Charming"
length = len(words)
letter1 = words[3]
letter2 = words[-5]
letter3 = words[len(words) - 2]
letters1 = words[3:6]
letters2 = words[:6]
letters3 = words[6:]
letters4 = words[-3:]

word = letter1 + letter2
word = word + " " + letter3

print(letters1, letters2, letters3, letters4, word)

Dot notation

- Every object type, as well as storing some data, has some defined methods which can be applied to that particular type of object.
- Variables which reference an object are called instances, e.g., in the following code, greeting is a string instance and number is an instance of type int.

```python
words = "Prince Charming"
length = len(words)
letter1 = words[3]
letter2 = words[-5]
letter3 = words[len(words) - 2]
letters1 = words[3:6]
letters2 = words[:6]
letters3 = words[6:]
letters4 = words[-3:]

word = letter1 + letter2
word = word + " " + letter3

print(letters1, letters2, letters3, letters4, word)
```
String methods - upper(), lower()

- The **upper()** method returns a new string object with all the characters converted to upper case. The **lower()** method returns a new string object with all the characters converted to lower case.

- For example,

```python
string = "Hello World"

string_upper = string.upper()
string_lower = string.lower()

print(string, string_lower, string_upper)
```

Notice that there is a total of three string objects.

String methods - find()

- The **find()** method is used to look for the position (index number) of the first occurrence (from the left) of some characters. If the characters are found, the find() method returns the index number, otherwise the find() method returns -1.

- For example,

```python
string = "Hello World"

position1 = string.find("o")
position2 = string.find("z")
position3 = string.find("o" + string[4] + string[5])

print(position1, position2, position3)
```

Notice that there is a total of three string objects.

String methods - rfind()

- The **rfind()** method is used to look for the index position of the last (i.e. rightmost) occurrence of some characters. If the characters are found, the rfind() method returns the index number, otherwise the rfind() method returns -1.

- For example,

```python
string = "Hello World"

position1 = string.rfind("o")
position2 = string.rfind("z")
position3 = string.rfind("orl")
position4 = string.rfind("lro")

print(position1, position2, position3, position4)
```

Notice that there is a total of four string objects.

String methods - strip()

- The **strip()** method returns a new string object with all white space from the beginning and end of the string removed. It does not remove spaces from inside the string.

- For example,

```python
string = "    H e l l o oooo    "

string = string.strip()

print(len(string), len(string.strip()))
```

Notice that there are two string objects.
Exercise

Complete the following program so that it prints the initial from the first name followed by a full stop, a space and followed by the surname. Assume the full name is always two names separated by a single space.

```python
full_name = "Wystan Auden"

initialled_name = first_letter + ". " + last_name
print(initialled_name)
```

W. Auden

Common Python inbuilt functions

- `min()` is an inbuilt function which can be used to find the smallest number from a comma separated set of numbers and `max()` is the inbuilt function which can be used to find the largest number from a comma separated set of numbers, e.g.,

```python
num1 = 32
num2 = 16
smallest = min(num1, num2)
print(smallest)
smallest = min(32.7, 56.4, 3, -1.1, 56.99, -1.2)
print(smallest)
largest = max(num1, num2)
print(largest)
largest = max(32.7, 56.4, 3, -1.1, 56.99, -1.2)
print(largest)
```

- The inbuilt function, `round()`, is used to round numbers to the closest whole number (or rounded to a number of digits after the decimal point), e.g.,

```python
num1 = 32.657123
num2 = 16.48926
num3 = -16.48926
print(round(num1))
print(round(num2))
print(round(num3))
print()
print(round(num1, 2))
print(round(num2, 3))
print(round(num3, 4))
```

- Note that the function, `round()` with a single argument returns an int number and that rounding an int returns the int unchanged, e.g.,

```python
print("round(32.657123, 0): ", round(32.657123, 0))
print("round(16.48926, 0): ", round(16.48926, 0))
print("round(32.657123): ", round(32.657123))
print("round(16.48926): ", round(16.48926))
print("round(24.0, 0): ", round(24.0, 0))
print("round(24.0, 1): ", round(24.0, 1))
print("round(24, 0): ", round(24, 0))
print("round(24.0): ", round(24.0))
print("round(24): ", round(24))
```
**round() - unexpected result**

- Sometimes the round() function seems to give an unexpected result e.g.,

  ```python
  num1 = 1.5
  num2 = 2.5
  num3 = 3.5
  print(round(num1))  # 2
  print(round(num2))  # surprising result
  print(round(num3))  # 4
  ```

- This problem happens because floating point numbers are stored in a finite space, e.g., 0.1 has an infinite number of digits when converted to base 2.

  ![Binary representation of 0.1](image)

  but, when stored in the computer memory, float numbers are assigned exactly 64 bits of space which means that some of the bits are cut off. If 2.5 is actually stored as 2.49999999999999999…99 then it is rounded to 2 (not the expected 3).

  ```python
  round(4.235, 2) gives the number 4.24
  round(4.265, 2) gives the number 4.26
  ```

**Common Python inbuilt functions**

- The **inbuilt function, abs()**, is used to get the absolute value (the magnitude) of a number (int or float), e.g.,

  ```python
  num1 = 32
  num2 = -32
  num3 = abs(16 - 23)
  print("1.", abs(num1))  # 32
  print("2.", abs(num2))  # 32
  print("3.", num3)  # 7
  print("4.", abs(-16.78))  # 16.78
  ```

**Exercise**

- Complete the following program so that it prints the total tax and the net pay rounded to a whole number. The first $14000 is not taxed. The next amount up to $38500 is taxed at 24% and the rest is taxed at 34%.

```python
salary = 54000
no_tax_boundary = 14000
rate1_boundary = 38500
rate1 = 0.24
rate2 = 0.34
#Print the output
print("Salary: $", salary, sep="")
print("Amount to be taxed at: 24%: $", rate1_amount, sep="")
print("Tax at rate 1: $", tax_rate1, sep="")
print("Amount to be taxed at: 34%: $", rate2_amount, sep="")
print("Tax at rate 2: $", tax_rate2, sep="")
print("==================================")
print("Total tax: $", total_tax, sep="")
print()
print("Net pay: $", net_pay, sep="")
print("==================================")
```

**Summary**

- **In Python:**
  - **use dot notation when using string methods with string instances**
  - the string methods: upper(), lower(), strip(), find(), rfind() can be used with string instances
  - Some Python inbuilt functions are: min(), max(), round()
Examples of Python features used in this lecture

greeting = "Hello World"
position1 = greeting.find("o")
position2 = greeting.rfind("o")
position3 = words.find("Z")
position4 = words.rfind("o N")

greeting_lower = greeting.lower()
greeting_upper = greeting.upper()

smallest = min(32.7, 56.4, 3, -1.1, 56.99, -1.2)
largest = max(32.7, 56.4, 3, -1.1, 56.99, -1.2)

num1 = 32.657123
print(round(num1))
print(round(num1, 2))

num2 = abs(20 - num1)
print(num2)