Introduction to Python

Lecture 3: Functions and Classes

Daniel Kadyrov

July 6th, 2023

Functions

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.
- Functions are used to perform certain actions, and they are important for reusing code
- Functions are defined using the `def` keyword.
- The `def` keyword is followed by the function name, parentheses `()`, and a colon `:`.
- The `return` keyword is used to return a value from the function.

Functions

The following code snippet defines a function that prints a string passed to it as an argument in reverse order:

```
1  def reverse_string(string):
2      print(string[::-1])
```

The function can be called as follows:

```
1  reverse_string("Hello World!")
```

The output of the function call is:

```
1  !dlroW olleH
```

Functions

The following code snippet defines a function that splits a sentence into a list of words, capitalizes each word in the list, and returns the capitalized sentence:

```
1  def capitalize_sentence(sentence):
2      words = sentence.split()
3      capitalized_words = []
4      for word in words:
5          capitalized_words.append(word.capitalize())
6      return " ".join(capitalized_words)
```

Functions

The function can be called as follows:

```
1  sentence = "Code is like humor. When you have to explain it, it'
       s bad."
2  new_sentence = capitalize_sentence(sentence)
```

The output of the function call is:

```
1  Code Is Like Humor. When You Have To Explain It, It's Bad.
```

A recursive function is a function that calls itself during its execution. This enables the function to repeat itself several times, outputting the result and the end of each iteration.

The following example defines a recursive function that calculates the factorial of a number. Mathematically, a factorial is expressed the following way:

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 1$$

For example, $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$.

This can be expressed in Python as follows:

```
1  def factorial(n):
2      if n == 1:
3          return 1
4      else:
5          return n * factorial(n-1)
```

The function can be called as follows:

```
1  f = factorial(5)
2  print(f)
```

```
120
```

A lambda function is a small anonymous function. It can take any number of arguments, but can only have one expression.

The following code snippet defines a lambda function that takes a number as an argument and returns the square of that number:

```
1  square = lambda x: x**2
```

The function can be called as follows:

```
1  square(5)
```

The output of the function call is: `25`