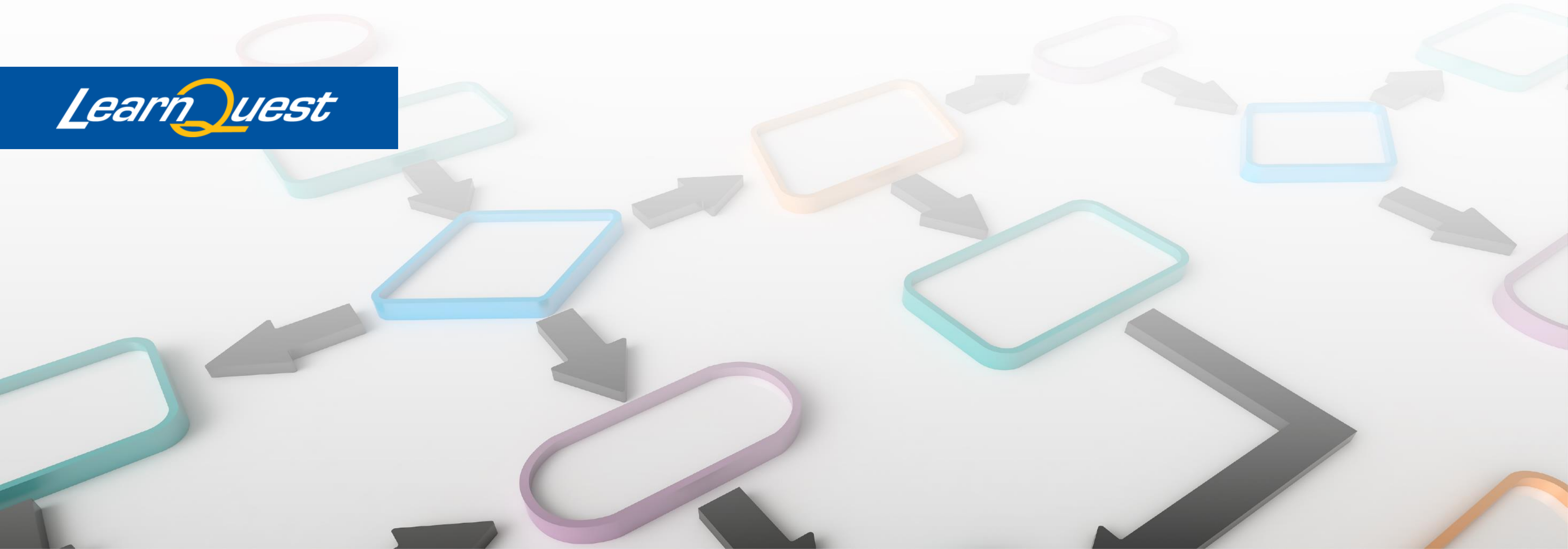
Intermediate Python

2nd Course in Python Scripting for DevOps Specialization



Dates and Advanced Strings

In the first module of this course, we extend our skills by working with dates and time values. We will also expand our abilities by looking at how to work with advance string operations.

1

Learning Objectives

Dates and Advanced Strings

Upon completion of this module, learners will be able to:

- Develop computer programs that utilize the string operations in the execution of the program
- Develop computer programs that utilize date and time data types in the execution of the program
- Develop computer programs that format numbers
- Develop computer programs that format dates and times

Lesson 1

Working with dates and times values

In this lesson we look at how to develop computer programs that utilize date and time data types in the execution of the program

DateTime Module

To create a date, we can use the datetime() class (constructor) of the datetime module.

The datetime() class requires three parameters to create a date: year, month, day.

- import datetime
- x = datetime.datetime (2021, 2, 26)

The datetime() class requires five parameters to create a datetime: year, month, day, hour, minute, second.

- import datetime
- x = datetime.datetime (2021, 2, 26,14,22)

strftime() and strptime() Format Codes

Note: Examples are based on datetime.datetime(2013, 9, 30, 7, 6, 5)

Code	Meaning	Example
%a	Weekday as locale's abbreviated name.	Mon
%A	Weekday as locale's full name.	Monday
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	1
%d	Day of the month as a zero-padded decimal number.	30
%-d	Day of the month as a decimal number. (Platform specific)	30
%b	Month as locale's abbreviated name.	Sep
%B	Month as locale's full name.	September
%m	Month as a zero-padded decimal number.	9
%-m	Month as a decimal number. (Platform specific)	9
%y	Year without century as a zero-padded decimal number.	13
%Y	Year with century as a decimal number.	2013
%H	Hour (24-hour clock) as a zero-padded decimal number.	7
%-H	Hour (24-hour clock) as a decimal number. (Platform specific)	7
%I	Hour (12-hour clock) as a zero-padded decimal number.	7
%-I	Hour (12-hour clock) as a decimal number. (Platform specific)	7
%p	Locale's equivalent of either AM or PM.	AM
%M	Minute as a zero-padded decimal number.	6
%-M	Minute as a decimal number. (Platform specific)	6

strftime() and strptime() Format Codes Codes

Note: Examples are based on datetime.datetime(2013, 9, 30, 7, 6, 5)

Code	Meaning	Example
%S	Second as a zero-padded decimal number.	5
%-S	Second as a decimal number. (Platform specific)	5
%f	Microsecond as a decimal number, zero-padded on the left.	0
%z	UTC offset in the form +HHMM or -HHMM (empty string if the object is naive).	
%Z	Time zone name (empty string if the object is naive).	
%j	Day of the year as a zero-padded decimal number.	273
%-j	Day of the year as a decimal number. (Platform specific)	273
%U	Week number of the year (Sunday as the first day of the week) as a zero padded decimal number. All days in a new year preceding the first Sunday are considered to be in week 0.	39
%W	Week number of the year (Monday as the first day of the week) as a decimal number. All days in a new year preceding the first Monday are considered to be in week 0.	39
%c	Locale's appropriate date and time representation.	Mon Sep 30 07:06:05 2013
%x	Locale's appropriate date representation.	9/30/2013
%X	Locale's appropriate time representation.	7:06:05
%%	A literal '%' character.	%

strptime

Returns a datetime corresponding to date_string, parsed according to format

```
dt = datetime.strptime("21/11/06 16:30", "%d/%m/%y %H:%M")
```

Getting Current Date and Time

```
import datetime
```

```
x = datetime.datetime.now()
```

strftime

Returns a string representing the time, controlled by an explicit format string

```
dtstring = now.strftime("%d/%m/%y %H:%M")
```

Lesson 1 Review

Python uses the same format codes to convert a datetime to a String and from a String

Using the format String you can pull any part of a datetime value

The now method returns the current date and time value

Lesson 2

Advanced string operations

In this lesson we look at how we can develop computer programs that utilize more advanced string operations in the execution of the program

Slicing Strings

Get the characters from position 2 to position 5 (not included):

```
b = "Hello, Coursera!“
print(b[2:5])
```

Get the characters from the start to position 5 (not included):

```
b = "Hello, Coursera!“
print(b[:5])
```

Get the characters from position 2, and all the way to the end:

```
b = "Hello, Coursera!“
print(b[2:])
```

Modifying Strings

The upper() method returns the string in upper case:

```
x = "Hello, Coursera!“
print(x.upper())
```

The lower() method returns the string in lower case:

```
x = "Hello, Coursera!“
print(x.lower())
```

The strip() method removes any whitespace from the beginning or the end:

```
x = “ Hello, Coursera! “
print(x.strip())
```

Replacing Substrings

The replace() method replaces a string with another string:

```
x = "Hello, Coursera!“
print(x.replace(“Coursera”,”World”))
```

Lesson 2 Review

Slicing operations allow us to cut up a string into parts

Strings have methods that return a modified version of the string

The replace method can be used to replace multiple parts of strings

Lesson 3

Formatting Data

In this lesson we look at how we can develop computer programs that format numbers, dates and times

Format Method

The format() method takes unlimited number of arguments, and are placed into the respective placeholders:

```
one= 2
two= 51
three= 99.95
myorder = "I want {} items of type {} for {} dollars."
print(myorder.format(one, two, three))
```

Name Component

- Format Syntax: {[<name>][!<conversion>][:<format_spec>]}
- The <name> component is the first portion of a replacement field.
- <name> indicates which argument from the argument list is inserted into the Python format string in the given location. It’s either a number for a positional argument or a keyword for a keyword argument.

Conversion

- Format Syntax: {[<name>][!<conversion>][:<format_spec>]}
- Python can format an object as a string using three different built-in functions:

str()	-	!s
repr()	-	!r
ascii()	-	!a

Format Specification

- Format Syntax: {[<name>][!<conversion>][:<format_spec>]}
- <format_spec> represents the guts of the Python .format() method's functionality. It contains information that exerts fine control over how values are formatted prior to being inserted into the template string.
  - [[<fill>]<align>][<sign>][#][0][<width>][<group>][.<prec>][<type>]

Format Specification - Subcomponents

Subcomponent	Effect
:	Separates the <format_spec> from the rest of the replacement field
<fill>	Specifies how to pad values that don't occupy the entire field width
<align>	Specifies how to justify values that don't occupy the entire field width
<sign>	Controls whether a leading sign is included for numeric values
#	Selects an alternate output form for certain presentation types
0	Causes values to be padded on the left with zeros instead of ASCII space characters
<width>	Specifies the minimum width of the output
<group>	Specifies a grouping character for numeric output
.<prec>	Specifies the number of digits after the decimal point for floating-point presentation types, and the maximum output width for string presentations types
<type>	Specifies the presentation type, which is the type of conversion performed on the corresponding argument

Type Specification

Value	Presentation Type
b	Binary integer
c	Single character
d	Decimal integer
e or E	Exponential
f or F	Floating point
g or G	Floating point or Exponential
o	Octal integer
s	String
x or X	Hexadecimal integer
%	Percentage

Width

- <width> specifies the minimum width of the output field:
- '{0:8s}'.format(‘hello’)
- '{0:8d}'.format(123)

Precision

- .<prec> specifies the number of digits after the decimal point for floating point presentation types:
- '{0:8.2f}'.format(1234.5678)
- '{0:8.4f}'.format(1.23)

First Programming Assignment

Part 1

Write a program that reads in a number and prints the date that number of days from now in this format:
Saturday, February 6, 2021

Part 2

Write a program that reads in numbers until a -1 is entered and prints the sum of all numbers entered in decimal format with two digits after the decimal.

- For example, if the user enters 5000 10 15 -1 the program should display 5025.00
  (Each number will be separated by a carriage return)

Lesson 3 Review

The format function allows replacement parameters to be inserted into strings

The type subcomponent of the format specifier can be used to specify the data type of the parameter

The precision subcomponent of the format specifier can be used to control the number of decimal places