```
#!/usr/bin/env python
from PyQt4.QtGui import QTextDocument, QPrinter, QApplication
import os
import sys
import logging
import argparse

try:
    import pygments
    from pygments import lexers, formatters, styles
except ImportError as ex:
    print('\nCould not import the required "pygments" module:\n{}'.format(
        ex))
    sys.exit(1)

__version__ = '1.2.0'


def logger(func):
    def log_wrap(self, ifile=None, ofile=None, size="A4"):
        logging.getLogger().name = "code2pdf> "
        logging.getLogger().setLevel(logging.INFO)
        func(self, ifile, ofile, size)
    return log_wrap


class Code2pdf:

    """
            Convert a source file into a pdf with syntax highlighting.
    """
    @logger
    def __init__(self, ifile=None, ofile=None, size="A4"):
        self.size = size
        if not ifile:
            raise Exception("input file is required")

        self.input_file = ifile
        if ofile:
            self.pdf_file = ofile
        else:
            self.pdf_file = ifile.split('.')[0] + ".pdf"

    def highlight_file(self, linenos=True, style='default'):
        """ Highlight the input file, and return HTML as a string. """
        try:
            lexer = lexers.get_lexer_for_filename(self.input_file)
        except pygments.util.ClassNotFound:
            # Try guessing the lexer (file type) later.
            lexer = None

        try:
            formatter = formatters.HtmlFormatter(
                linenos=linenos,
                style=style,
                full=True)
        except pygments.util.ClassNotFound:
            print("\nInvalid style name: {}\nExpecting one of:\n    {}".format(
                style,
                "\n    ".join(sorted(styles.STYLE_MAP))))
            sys.exit(1)

        try:
            with open(self.input_file, "r") as f:
                content = f.read()
                if lexer is None:
                    try:
                        lexer = lexers.guess_lexer(content)
                    except pygments.util.ClassNotFound:
                        # No lexer could be guessed.
                        lexer = lexers.get_lexer_by_name("text")
        except EnvironmentError as exread:
            print("\nUnable to read file: {}\n{}".format(
                self.input_file,
                exread))
            sys.exit(1)

        return pygments.highlight(content, lexer, formatter)

    def init_print(self, linenos=True, style="default"):
        app = QApplication(sys.argv)  # noqa
        doc = QTextDocument()
        doc.setHtml(
```

```
            self.highlight_file(linenos=linenos, style=style)
        )
        printer = QPrinter()
        printer.setOutputFileName(self.pdf_file)
        printer.setOutputFormat(QPrinter.PdfFormat)
        if self.size.lower() == "a2":
            printer.setPageSize(QPrinter.A2)
        elif self.size.lower() == "a3":
            printer.setPageSize(QPrinter.A3)
        elif self.size.lower() == "a4":
            printer.setPageSize(QPrinter.A4)

        printer.setPageMargins(15, 15, 15, 15, QPrinter.Millimeter)
        doc.print_(printer)
        logging.info("PDF created at %s" % (self.pdf_file))


def get_output_file(inputname, outputname=None):
    """ If the output name is set, then return it.
        Otherwise, build an output name using the current directory,
        replacing the input name's extension.
    """
    if outputname:
        return outputname

    inputbase = os.path.split(inputname)[-1]
    outputbase = "{}.pdf".format(os.path.splitext(inputbase)[0])
    return os.path.join(os.getcwd(), outputbase)


def parse_arg():
    parser = argparse.ArgumentParser(
        description=(
            "Convert given source code into .pdf with syntax highlighting"),
        epilog="Author:tushar.rishav@gmail.com"
    )
    parser.add_argument(
        "filename",
        help="absolute path of the python file",
        type=str)
    parser.add_argument(
        "-l",
        "--linenos",
        help="include line numbers.",
        action="store_true")
    parser.add_argument(
        "outputfile",
        help="absolute path of the output pdf file",
        nargs="?",
        type=str)
    parser.add_argument(
        "-s",
        "--size",
        help="PDF size. A2,A3,A4,A5 etc",
        type=str,
        default="A3")
    parser.add_argument(
        "-S",
        "--style",
        help="the style name for highlighting.",
        type=str,
        default="default",
        metavar="NAME")
    parser.add_argument(
        "-v",
        "--version",
        action="version",
        version="%(prog)s v. {}".format(__version__))
    return parser.parse_args()


def main():

    args = parse_arg()
    pdf_file = get_output_file(args.filename, args.outputfile)
    pdf = Code2pdf(args.filename, pdf_file, args.size)
    pdf.init_print(linenos=args.linenos, style=args.style)
    return 0

if __name__ == "__main__":
    sys.exit(main())
```