山东大学

SHANDONG UNIVERSITY

课程名称

实验标题

姓名　学号　班级

日期

课程名称

实验标题

成员

成员 1　学号 1　班级 1
成员 2　学号 2　班级 2
成员 3　学号 3　班级 3

分工情况

成员 1　成员 1 负责内容
成员 2　成员 2 负责内容
成员 3　成员 3 负责内容

日期

摘要

本实验报告模板基于高鹏鸿学长的实验报告模板实现，在 GNU 通用公共许可证下发布。

本模板按照我的个人喜好，采用较为简洁清爽的风格，去除了高鹏鸿学长的模板中关于页眉页脚的相关内容。此外，将实现代码块的宏包更改为 `minted` ，以实现对多种语言语法高亮的支持，但也同时带来一些使用上的不便，需要对编译器及环境做一定配置，具体内容在正文部分中有所介绍。

本实验报告模板的默认内容将介绍如何使用该 LaTeX 实验报告模板来编写实验报告，包括了一些基本的使用方法，如插入图片、插入代码、插入表格、插入公式、插入参考文献、插入超链接、插入脚注、绘制图像等，和此模板专有的方法，如插入封面等，以及一些个性化相关的内容。

如果在使用本模板时遇到任何问题，或对本模板有任何改进建议，请通过 edgeworthlau@outlook.com 联系我。

目录

摘要 **I**

目录 **II**

1 基本使用 **1**

1.1 段落与空行 1

1.2 逐项输出环境 1

1.3 分栏 2

2 插入图片 **3**

2.1 插入图片 3

2.2 标题与标签 3

3 插入代码 **3**

3.1 插入行内代码 5

3.2 插入行间代码 5

4 插入表格 **7**

5 插入公式 **8**

5.1 插入行内公式 8

5.2 插入行间公式 8

6 插入参考文献 **9**

6.1 插入内部参考文献 9

6.2 插入外部参考文献 10

7 插入超链接 **10**

8 插入脚注 **11**

9 图表绘制 **11**

9.1 绘制流程图 11

9.2 绘制树图 12

9.3 绘制拓扑图 14

9.4 绘制时序图 15

9.5 绘制其他图形 16

9.6 使用 Python 绘图 16

10 插入封面 **18**

11 个性化 **18**
11.1 字体 18
11.2 颜色 18
11.3 页边距 20
11.4 代码高亮 20
11.5 超链接 21

12 宏包 **21**

1 基本使用

1.1 段落与空行

在代码中直接输入文字即可新建一个段落。段落默认有首行缩进，若需取消一个段落的首行缩进，可在段落文字前添加 `\noindent`。

在文字间使用空行分隔不同段落，请注意该空行不会渲染输出至文档中。若需要在文档中的段落间添加空行，可在对应位置添加 `\vspace{0.5cm}`。

下面以代码 1 中的代码为例，展示段落与空行相关命令的使用方法。

代码 1: 段落与空行示例代码

```
这是段落一，该段未作任何设置。

\noindent 这是段落二，该段取消了首行缩进。

\vspace{0.5cm}

这是段落三，该段与前一段之间添加了空行。
```

代码 1 中的代码经过编译后，输出结果如下所示。（为了更好地区分示例内容，后续大部分示例代码的输出内容均用两条分割线包裹。）

这是段落一，该段未作任何设置。
这是段落二，该段取消了首行缩进。

这是段落三，该段与前一段之间添加了空行。

1.2 逐项输出环境

使用 `\begin{itemize}...\end{itemize}` 环境可实现逐项输出。在环境中，使用 `\item` 命令标记项，`\subitem` 命令标记子项，`\subsubitem` 命令标记二级子项。

下面以代码 2 中的代码为例，展示逐项输出环境的使用方法。

代码 2: 逐项输出示例代码

```
\begin{itemize}
    \item 第一项
    \item 第二项
    \subitem 第二项子项
    \subsubitem 第二项二级子项
    \item 第三项
\end{itemize}
```

代码 2 中的代码经过编译后，输出结果如下所示。

- 第一项
- 第二项

  第二项子项

  第二项二级子项
- 第三项

1.3 分栏

使用 `\begin{multicols}{n}...\end{multicols}` 环境可实现分栏输出，其中 `n` 为分栏数。在环境中，使用 `\columnbreak` 命令可手动切换至下一栏。

更多使用方式可以查看 Overleaf 的文档。

下面以代码 3 中的代码为例，展示分栏环境的使用方法。

代码 3: 分栏示例代码

```
\begin{multicols}{2} % 指定分栏数为 2
    这是左栏

    hello world

    \columnbreak % 手动切换至下一栏

    这是右栏

    你好世界
\end{multicols}
```

代码 3 中的代码经过编译后，输出结果如下所示。

这是左栏
hello world

这是右栏
你好

---

2 插入图片

2.1 插入图片

使用 `\includegraphics[options]{figname}` 命令可插入图片，其中 `options` 为图片选项，`figname` 为图片文件名。

使用 `\begin{figure}[htbp]...\end{figure}` 环境可实现对图片的浮动设置，其中 `htbp` 用于设置图片位置，`h` 表示 here，`t` 表示 top，`b` 表示 bottom，`p` 表示 page。

下面以代码 4 中的代码为例，展示插入图片相关命令的使用方法。

代码 4: 插入图片示例代码

```
\begin{figure}[htbp]
    \begin{center}
        \includegraphics[width=0.8\textwidth]{logo.jpg}
        % width 调整图片大小
        \caption{Logo of SDU}
        \label{fig:1}
    \end{center}
\end{figure}
```

代码 4 中的代码经过编译后，输出结果如图 1 所示。

2.2 标题与标签

使用 `\caption{title}` 命令可添加标题，其中 `title` 为标题内容。

使用 `\captionof{type}{title}` 命令可为已知类型的内容添加标题，其中 `type` 为类型，`title` 为标题内容，最终生成的标题会包含：类型前缀、编号以及标题内容。

使用 `\label{key}` 命令可添加标签，其中 `key` 为标签名。

上述命令在插入其他内容时也常使用。

3 插入代码

本模板使用 `minted` 宏包来实现对多种语言语法高亮的支持。由于 `minted` 宏包依赖于 Python 的 `Pygments` 包，且需要在编译时指定 `-shell-escape` 选项，故需要对编译器及环境做一定配置。

下面的配置以 vscode 为例，其他编辑器类似。

打开 vscode 后，按下 `Ctrl + Shift + P`，在命令面板中输入 `settings`，选择 `Preferences: Open User Settings (JSON)`，在打开的 `settings.json` 文件中找到 latex-workshop.latex.tools 选项，在所使用的编译器的参数中添加 `-shell-escape` 和 `-8bit` 两项，如图 2 所示。

其中，`-shell-escape` 选项用于允许编译器调用 `Pygments` 环境；`-8bit` 选项用于解决 `minted` 宏包在中文环境下的编码问题，以避免在缩进处渲染奇怪的符号。

图 1: Logo of SDU

```
43      "latex-workshop.latex.tools": [
44          {
45              "args": [
46                  "-synctex=1",
47                  "-pdfxe",
48                  "-interaction=nonstopmode",
49                  "-file-line-error",
50                  "-outdir=%OUTDIR%",
51                  "-shell-escape",
52                  "-8bit",
53                  "%DOC%"
54              ],
55              "command": "latexmk",
56              "env": {},
57              "name": "xelatexmk"
58          },
59      ],
```

图 2: vscode 配置示例

此外，在使用 minted 宏包前，还需在环境中安装 Pygments 环境。使用 pip 或使用系统包管理器可完成安装，具体的安装步骤依不同的系统环境及版本可能有所变化，请搜索互联网以获得最新、最准确的安装步骤。

下面仅展示 minted 宏包的简单使用方法，更多使用方式可以查看 minted 宏包的官方文档。

3.1 插入行内代码

使用 \mintinline{language}{code} 命令可插入行内代码，其中 language 为代码语言，code 为代码内容。

下面以代码 5 中的代码为例，展示插入行内代码相关命令的使用方法。

代码 5: 插入行内代码示例代码

```
这是一个插入了行内代码的示例段落，其中包含了 \mint{python}{print("Hello,
↪  world!")} 这行代码。
```

代码 5 中的代码经过编译后，输出结果如下所示。

这是一个插入了行内代码的示例段落，其中包含了 print("Hello, world!") 这行代码。

3.2 插入行间代码

使用 \begin{minted}{language}...\end{minted} 环境可插入行间代码，其中 language 为代码语言。

使用 \inputminted{language}{filename} 命令可插入外部代码文件，其中 language 为代码语言，filename 为代码文件名。

下面以代码 6 中的代码为例，展示插入行间代码相关命令的使用方法。

代码 6: 插入行间代码示例代码

```
% 插入行间代码
\begin{minted}{python}
def extended_gcd(a, b):
    if b == 0:
        return (a, 1, 0)
    else:
        d, x, y = extended_gcd(b, a % b)
        return (d, y, x - (a // b) * y)

def mod_inverse(a, m):
    d, x, y = extended_gcd(a, m)
    if d != 1:
        raise ValueError("Modular inverse does not exist")
```

```
    else:
        return x % m
\end{minted}

% 插入外部代码文件
\inputminted{python}{codes/example.py}
```

请注意，在使用 minted 展示 LaTeX 代码时，代码中的 \end{minted} 会使编译器无法分辨代码的结束位置，导致编译失败。为解决此问题，可将待展示的 LaTeX 代码写入一个文件中，以插入外部代码文件展示。

代码 6 中的代码经过编译后，输出结果如代码 7 所示。

代码 7: 插入行间代码输出结果

```python
def extended_gcd(a, b):
    if b == 0:
        return (a, 1, 0)
    else:
        d, x, y = extended_gcd(b, a % b)
        return (d, y, x - (a // b) * y)

def mod_inverse(a, m):
    d, x, y = extended_gcd(a, m)
    if d != 1:
        raise ValueError("Modular inverse does not exist")
    else:
        return x % m
```

```python
def extended_gcd(a, b):
    if b == 0:
        return (a, 1, 0)
    else:
        d, x, y = extended_gcd(b, a % b)
        return (d, y, x - (a // b) * y)


def mod_inverse(a, m):
    d, x, y = extended_gcd(a, m)
    if d != 1:
        raise ValueError("Modular inverse does not exist")
    else:
        return x % m
```

4 插入表格

使用 `\begin{tabular}{set}... \end{tabular}` 环境可插入表格，其中 `set` 为表示表格的列数和对齐方式的字符串，`|` 表示在表格中添加单竖线，`||` 表示在表格中添加双竖线，`c` 表示居中对齐，`l` 表示左对齐，`r` 表示右对齐。在环境中，使用 `\hline` 可添加横线，使用 `&` 可分隔单元格，使用 `\\` 可换行。

下面以代码 8 中的代码为例，展示插入表格相关命令的使用方法。

代码 8: 插入表格示例代码

```
\begin{center}
    \captionof{table}{表格示例 1}
    \label{tab:tab1}
    \begin{tabular}{|c|c|c|}
        \hline
        A & B & C \\
        \hline
        1 & 2 & 3 \\
        \hline
        4 & 5 & 6 \\
        \hline
        7 & 8 & 9 \\
        \hline
    \end{tabular}
\end{center}

\begin{center}
    \captionof{table}{表格示例 2}
    \label{tab:2}
    \begin{tabular}{r||l c|}
        \hline
        AA & BB & C \\
        \hline
        1 & 2 & 3 \\
        4 & 5 & 6 \\
        \hline
        \hline
        7 & 8 & 9 \\
    \end{tabular}
\end{center}
```

代码 8 中的代码经过编译后，输出结果如表 1 和 表 2 所示。

表 1: 表格示例 1

A	B	C
1	2	3
4	5	6
7	8	9

表 2: 表格示例 2

AA	BB	C
1	2	3
4	5	6
7	8	9

5 插入公式

如果不熟悉 LaTeX 的公式语法，可以使用 CodeCogs 来生成公式的代码。

5.1 插入行内公式

使用 `$...$` 或 `\(...\)` 插入行内公式。
下面以代码 9 中的代码为例，展示插入行内公式相关命令的使用方法。

代码 9: 插入行内公式示例代码

```
这是一个插入了行内公式的示例段落，其中包含了 $a^2 + b^2 = c^2$ 和 \((a+b)^2 = a^2 + 2ab + b^2\) 这两个公式。
```

代码 9 中的代码经过编译后，输出结果如下所示。

这是一个插入了行内公式的示例段落，其中包含了 $a^2 + b^2 = c^2$ 和 $(a+b)^2 = a^2 + 2ab + b^2$ 这两个公式。

5.2 插入行间公式

使用 `$$...$$` 或 `\[...\]` 或 `\begin{equation}...\end{equation}` 插入行间公式。
下面以代码 10 中的代码为例，展示插入行间公式相关命令的使用方法。

代码 10: 插入行间公式示例代码

```
这是一个插入了行间公式的示例段落，
$$
    \int_{0}^{1} x^2 \mathrm{d} x = \frac{1}{3}
$$
```

```
\[
    \int_{0}^{1} x^2 \mathrm{d} x = \frac{1}{3}
\]
\begin{equation}
    \int_{0}^{1} x^2 \mathrm{d} x = \frac{1}{3}
\end{equation}
在这个段落中，以三种不同的方式插入了相同的公式。
```

代码 10 中的代码经过编译后，输出结果如下所示。

这是一个插入了行间公式的示例段落，

$$\int_0^1 x^2 \mathrm{d}x = \frac{1}{3}$$
$$\int_0^1 x^2 \mathrm{d}x = \frac{1}{3}$$
$$\int_0^1 x^2 \mathrm{d}x = \frac{1}{3} \tag{1}$$

在这个段落中，以三种不同的方式插入了相同的公式。

---

6 插入参考文献

6.1 插入内部参考文献

使用 `\label{key}` 定义标签，其中 `key` 为标签名。

使用 `\ref{key}` 引用标签，其中 `key` 为带引用标签的标签名，使用时需手动添加标签类型前缀。

此外，本模板还使用 `cleveref` 宏包对引用命令进行了重定义，使用 `\cref` 命令可实现对标签的引用，使用时无需再手动添加标签类型前缀。本模板已默认将“listing”“figure”“table”三个前缀重定义为“代码”“图”“表”，相关设置可以在 `report-style.sty` 文件中查看，可自行修改或增加其他类型前缀的重定义。

完成对标签的定义和引用后，在输出的文件中点击引用即可跳转至被引用位置。

下面以代码 11 中的代码为例，展示插入内部参考文献相关命令的使用方法。

代码 11: 插入内部参考文献示例代码

```
这是一个插入了内部参考文献的示例段落，其中包含了对图\ref{fig:1}和\cref{lst:7}的引
↪ 用。
```

代码 11 中的代码经过编译后，输出结果如下所示。

这是一个插入了内部参考文献的示例段落，其中包含了对图1和代码 7的引用。

6.2 插入外部参考文献

使用 `\begin{thebibliography}{set}...\end{thebibliography}` 环境定义外部参考文献，其中 `set` 为设置参考文献编号方式，一般设置为 `99`，指定参考文献的项目按照数字进行编号且编号最大值为 99。在环境中，使用 `\bibitem{key}...` 定义参考文献，其中 `key` 为参考文献的键，随后可添加文献的相关信息。一般来讲，应在附录部分定义参考文献。

使用 `\cite{key}` 命令可插入外部参考文献，其中 `key` 为参考文献的键。

下面以代码 12 中的代码为例，展示插入外部参考文献相关命令的使用方法。

代码 12: 插入外部参考文献示例代码

```
\begin{thebibliography}{99}
    \bibitem{1} 参考文献 1
    \bibitem{2} 参考文献 2
\end{thebibliography}

这是一个插入了外部参考文献的示例段落，其中包含了对参考文献 1\cite{1}和参考文献 2\cite{2}的引用。
```

代码 12 中的代码经过编译后，输出结果如下所示。

参考文献

[1] 参考文献 1

[2] 参考文献 2

这是一个插入了外部参考文献的示例段落，其中包含了对参考文献 1 [1] 和参考文献 2 [2] 的引用。

---

7 插入超链接

使用 `\href{url}{text}` 命令可插入超链接，其中 `url` 为链接地址，`text` 为链接文本。

下面以代码 13 中的代码为例，展示插入超链接相关命令的使用方法。

代码 13: 插入超链接示例代码

```
\href{https://www.sdu.edu.cn/}{山东大学}是中国著名的综合性大学，位于山东省济南市。
```

代码 13 中的代码经过编译后，输出结果如下所示。

山东大学是中国著名的综合性大学，位于山东省济南市。

8 插入脚注

使用 `\footnote{text}` 命令可插入脚注，其中 `text` 为脚注内容。
下面以代码 14 中的代码为例，展示插入脚注相关命令的使用方法。

代码 14: 插入脚注示例代码

```
这是一个插入了脚注的示例段落，其中包含了脚注\footnote{这是一个脚注}，本页的末尾位置将展示脚注的具体内容。
```

代码 14 中的代码经过编译后，输出结果如下所示。

这是一个插入了脚注的示例段落，其中包含了脚注[1]，本页的末尾位置将展示脚注的具体内容。

9 图表绘制

使用 TikZ 包可以绘制各种图像，例如流程图、树图等。
下面仅做简单的示例，详细的使用方式可以查看 Overleaf 的文档。

9.1 绘制流程图

下面以代码 15 中的代码为例，展示绘制流程图的方法。

代码 15: 绘制流程图示例代码

```
\begin{figure}[htbp]
    \centering
    \begin{tikzpicture}[node distance=1.5cm]
        \usetikzlibrary{shapes,arrows}

        \tikzstyle{startstop} = [rectangle, rounded corners, minimum width=3cm, minimum height=1cm, text centered, draw=black, fill=red!30]
        \tikzstyle{io} = [trapezium, trapezium left angle=70, trapezium right angle=110, minimum width=3cm, minimum height=1cm, text centered, draw=black, fill=blue!30]
        \tikzstyle{process} = [rectangle, minimum width=3cm, minimum height=1cm, text centered, draw=black, fill=orange!30]
```

[1] 这是一个脚注

```
        \tikzstyle{decision} = [diamond, minimum width=3cm, minimum
        ↪  height=1cm, text centered, draw=black, fill=green!30]
        \tikzstyle{arrow} = [thick,->,>=stealth]

        \node (start) [startstop] {Start};
        \node (in1) [io, below of=start] {Input};
        \node (pro1) [process, below of=in1] {Process 1};
        \node (dec1) [decision, below of=pro1, yshift=-0.5cm] {Decision 1};
        \node (pro2a) [process, below of=dec1, yshift=-0.5cm] {Process 2a};
        \node (pro2b) [process, right of=dec1, xshift=2cm] {Process 2b};
        \node (out1) [io, below of=pro2a] {Output};
        \node (stop) [startstop, below of=out1] {Stop};

        \draw [arrow] (start) -- (in1);
        \draw [arrow] (in1) -- (pro1);
        \draw [arrow] (pro1) -- (dec1);
        \draw [arrow] (dec1) -- node[anchor=east] {yes} (pro2a);
        \draw [arrow] (dec1) -- node[anchor=south] {no} (pro2b);
        \draw [arrow] (pro2b) |- (pro1);
        \draw [arrow] (pro2a) -- (out1);
        \draw [arrow] (out1) -- (stop);
    \end{tikzpicture}
    \caption{流程图}
    \label{fig:flowchart}
\end{figure}
```

代码 15 中的代码经过编译后，输出结果如图 3 所示。

9.2 绘制树图

下面以代码 16 中的代码为例，展示绘制树图的方法。

代码 16: 绘制树图示例代码

```
\begin{figure}[htbp]
    \centering
    \begin{tikzpicture}[sibling distance=10em,
        every node/.style = {shape=rectangle,
            rounded corners, draw, align=center,
            top color=white, bottom color=blue!20}]
        \node {Root}
```

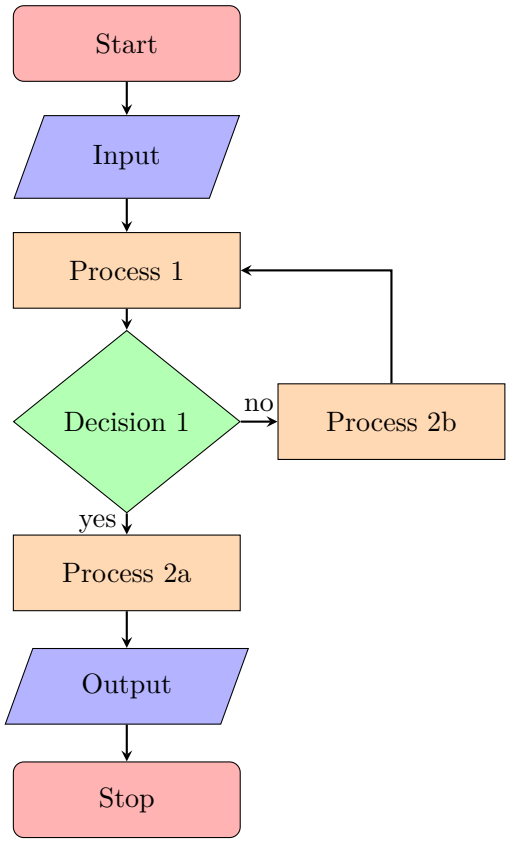
Start
Input
Process 1
Decision 1
no
Process 2b
yes
Process 2a
Output
Stop

图 3: 流程图

```
        child { node {Child 1}
            child { node {Grandchild 1} }
            child { node {Grandchild 2} }
        }
        child { node {Child 2}
            child { node {Grandchild 3} }
            child { node {Grandchild 4} }
        };
    \end{tikzpicture}
    \caption{树图}
    \label{fig:tree}
\end{figure}
```

代码 16 中的代码经过编译后，输出结果如图 4 所示。

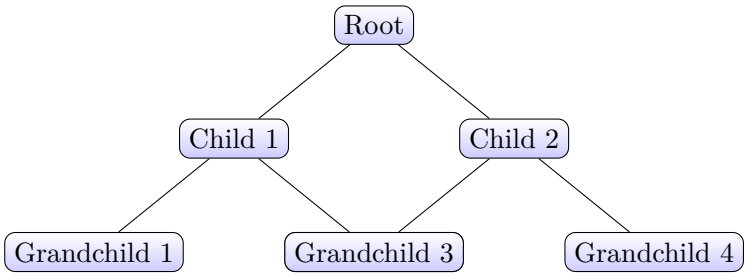

图 4: 树图

9.3 绘制拓扑图

下面以代码 17 中的代码为例，展示绘制拓扑图的方法。

代码 17: 绘制拓扑图示例代码

```
\begin{figure}[htbp]
    \centering
    \begin{tikzpicture}
        \usetikzlibrary{positioning}

        \node (A) {A};
        \node (B) [right=of A] {B};
        \node (C) [below=of A] {C};
        \node (D) [below=of B] {D};

        \draw (A) -- (B);
```

```
        \draw (A) -- (C);
        \draw (B) -- (D);
        \draw (C) -- (D);
    \end{tikzpicture}
    \caption{拓扑图}
    \label{fig:topology}
\end{figure}
```

代码 17 中的代码经过编译后，输出结果如图 5 所示。

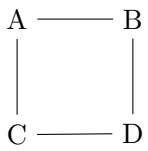

图 5: 拓扑图

9.4 绘制时序图

下面以代码 18 中的代码为例，展示绘制时序图的方法。

代码 18: 绘制时序图示例代码

```
\begin{figure}[htbp]
    \centering
    \begin{tikzpicture}
        \draw (0,0) -- (0,1) -- (1,1) -- (1,0) -- (2,0) -- (2,1) -- (3,1) --
        ↪  (3,0) -- (4,0);
    \end{tikzpicture}
    \caption{时序图}
    \label{fig:timing}
\end{figure}
```

代码 18 中的代码经过编译后，输出结果如图 6 所示。

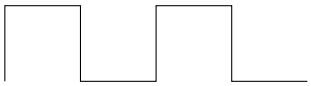

图 6: 时序图

9.5 绘制其他图形

下面以代码 19 中的代码为例，展示绘制其他图形的方法。

代码 19: 绘制其他图形示例代码

```latex
\begin{figure}[htbp]
    \centering
    \begin{tikzpicture}
        \draw (0,0) circle [radius=1];
    \end{tikzpicture}
    \caption{其他图形}
    \label{fig:other}
\end{figure}
```

代码 19 中的代码经过编译后，输出结果如图 7 所示。

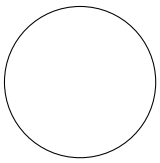

图 7: 其他图形

9.6 使用 Python 绘图

使用 Python 的 `matplotlib` 包绘图，将绘制结果输出为 pdf 文件，此时绘制的图表保存为矢量图，从而保证了图表的清晰度。随后，在需要使用图表时，按照插入图片的方式插入 pdf 文件即可。

代码 20 展示了使用 Python 的 `matplotlib` 包绘制图表的示例代码。

代码 20: 使用 Python 绘图示例代码

```python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.backends.backend_pdf as pdf_backend
from scipy.stats import f

degrees_of_freedom = [[1, 2], [2, 1], [5, 2], [100, 1], [100, 100]]
x = np.linspace(0, 5, 1000)
plt.figure(figsize=(12, 6))

for df in degrees_of_freedom:
```

```
    label = f'df={df[0]},{df[1]}'
    pdf = f.pdf(x, df[0], df[1])
    cdf = f.cdf(x, df[0], df[1])

    plt.subplot(1, 2, 1)
    plt.plot(x, pdf, label=label)

    plt.subplot(1, 2, 2)
    plt.plot(x, cdf, label=label)

plt.subplot(1, 2, 1)
plt.title("PDF - F-distribution")
plt.xlabel("x")
plt.ylabel("Density")
plt.legend()

plt.subplot(1, 2, 2)
plt.title("CDF - F-distribution")
plt.xlabel("x")
plt.ylabel("Probability")
plt.legend()

plt.tight_layout()
with pdf_backend.PdfPages('F_distribution_plots.pdf') as pdf: # 保存为 PDF 文
↪ 件
    pdf.savefig()
```

下面以代码 21 中的代码为例，展示插入 Python 绘制的图表的方法。

代码 21: 插入 Python 绘制的图表示例代码

```
\begin{figure}[htbp]
    \begin{center}
        \includegraphics[width=0.8\textwidth]{imgs/plots.pdf}
        \caption{F 分布}
        \label{fig:F}
    \end{center}
\end{figure}
```

代码 21 中的代码经过编译后，输出结果如图 8 所示。

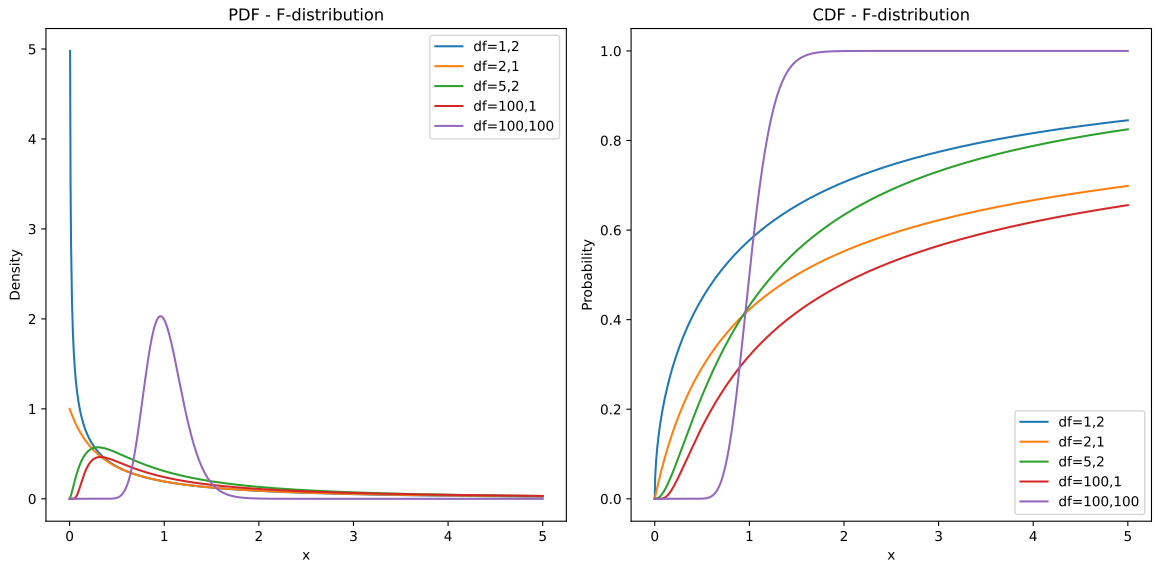

图 8: F 分布

10 插入封面

本模板提供了两种封面样式，分别为“单人报告封面”和“小组报告封面”。其中，“单人报告封面”包含了课程名称、实验名称、姓名、学号、班级、提交日期等信息；“小组报告封面”包含了课程名称、实验名称、组员信息（姓名、学号、班级）、组员分工情况、提交日期等信息。

使用本模板时，请根据情况选择封面样式，并将信息填入对应的位置中。

使用 `\singlecover` 命令可插入单人报告封面，使用 `\groupcover` 命令可插入小组报告封面。

11 个性化

11.1 字体

本模板的中文默认字体为宋体，英文默认字体为 CM Roman，等宽默认字体为 Tex ecltt8。

使用 `fontspec` 宏包可以自定义字体。

使用 `\setmainfont{fontname}` 命令可以设置正文字体（同时设置中文和英文），使用 `\setmonofont{fontname}` 命令可以设置等宽字体，其中 `fontname` 为字体名称。

使用 `fontspec` 宏包设置的字体应已安装在系统环境中，字体名称为字体家族名称，而非字体文件名。

11.2 颜色

使用 `xcolor` 宏包可以自定义颜色。

使用 `\definecolor{colorname}{colortype}{colorvalue}` 命令定义颜色，其中 `colorname` 为颜色名称，`colortype` 为颜色类型，`colorvalue` 为颜色值。常用的颜色类型有：`rgb`（RGB 颜


山东大学
SHANDONG UNIVERSITY
课程名称
实验标题
姓名 学号 班级
日期
山东大学
SHANDONG UNIVERSITY
课程名称
实验标题
成员
成员 1 学号 1 班级 1
成员 2 学号 2 班级 2
成员 3 学号 3 班级 3
分工情况
成员 1 成员 1 负责内容
成员 2 成员 2 负责内容
成员 3 成员 3 负责内容
日期

图 9: 两种封面样式

色）、cmyk（CMYK 颜色）、bahs（HTML 颜色）。

使用 \textcolor{color}{text} 命令可以设置文本颜色，其中 color 为颜色名称，text 为文本内容。

下面以代码 22 中的代码为例，展示自定义颜色的示例代码。

代码 22: 自定义颜色示例代码

```
\definecolor{mycolor}{rgb}{12,125,200}
\textcolor{mycolor}{这段文字的颜色为自定义颜色。}
```

代码 22 中的代码经过编译后，输出结果如下所示。

这段文字的颜色为自定义颜色。

11.3 页边距

使用 geometry 宏包可以自定义页边距。

使用 \geometry{left=...,right=...,top=...,bottom=...} 命令可以设置页边距，其中 left、right、top、bottom 可以分别设置上、下、左、右的页边距。

下面以代码 23 中的代码为例，展示自定义页边距示例代码，本模板也设置为下面的样式。

代码 23: 自定义页边距示例代码

```
\geometry{
    left=3.17cm,
    right=3.17cm,
    top=2.54cm,
    left=2.54cm
}
```

11.4 代码高亮

本模板使用 minted 宏包实现代码高亮。

minted 宏包依赖于 Python 的 Pygments 包实现代码高亮，而 Pygments 包支持数百种编程语言、模板语言与标记语言，因而大部分情况下无需担心语言不支持的问题。

使用 \usemintedstyle{style} 命令可设置代码高亮风格，其中 style 为风格名称。

使用 \setminted{options} 命令可设置代码选项，其中 options 为选项列表。常用选项有：linenos（显示行号）、breaklines（自动折行）、frame=single（添加边框）、tabsize=4（设置制表符宽度）、encoding=utf8（设置编码）。

更多选项可以查看 minted 宏包的官方文档

下面以代码 24 中的代码为例，展示 minted 宏包选项的设置方法，本模板也设置为下面的样式。

代码 24: minted 宏包选项示例代码

```
\setminted{
    linenos,
    breaklines,
    frame=single,
    tabsize=4,
    encoding=utf8,
}
```

11.5 超链接

使用 \hypersetup{...} 命令可以对超链接样式做个性化设置。其中，... 为设置项列表。常用设置项有：colorlinks（设置链接颜色）、linkcolor（设置内部链接颜色）、citecolor（设置引用链接颜色）、urlcolor（设置 URL 链接颜色）。更多设置项可以查看 hyperref 宏包的官方文档。

下面以代码 25 中的代码为例，展示超链接样式设置的示例代码，本模板也设置为下面的样式。

代码 25: 超链接样式设置示例代码

```
\hypersetup{
    hidelinks,
    colorlinks=true,
    linkcolor=blue,
    filecolor=blue,
    urlcolor=blue,
    citecolor=cyan,
}
```

12 宏包

本模板使用了以下宏包。

宏包	说明
abstract	设置摘要
amsmath	插入数学公式
amsfonts	插入数学字体
amssymb	插入数学符号
authblk	处理作者信息
booktabs	生成表格
caption	图表标题设置
cite	管理参考文献引用
cleveref	重定义引用
ctex	提供中文支持
float	设置图片位置
fontspec	设置字体
graphicx	图片插入
geometry	设置页边距
hyperref	超链接
minted	插入代码
multicol	提供多栏排版支持
sectsty	设置章节标题
siunitx	插入单位
subcaption	插入子图表标题
tikz	绘制图表
xcolor	设置自定义颜色