Appendix

Table I shows the results of models fine-tuned by various context and docstring combinations (corresponding to Section III-D in our main pdf submission).

Table II shows the performance of three models across six runnable levels on WenwangBench with excluding functions with the none return value (corresponding to Section VI in our main pdf submission).

Figure 1, Figure 2, and Figure 3 show the examples of code generated by the various models (corresponding to Section V-C in our main pdf submission).

I. Appendix Table

TABLE I: Performance of models fine-tuned by various context and docstring combinations

Fine-tune Prompt Type	Inference Prompt Type	pass@1	pass@5	pass@10
AllContext+Docs	AllContext+Docs	**14.43%**	**22.29%**	**25.65%**
Only Docs	Only Docs	12.09%	17.99%	20.87%
Docs+AllContext	Docs+AllContext	13.04%	21.31%	24.78%
Docs+AllContext+Docs	Docs+AllContext+Docs	14.00%	20.65%	23.91%
OracleContext+Docs	OracleContext+Docs	13.57%	20.86%	24.35%
Docs+OracleContext	Docs+OracleContext	13.74%	20.26%	23.04%

TABLE II: Performance of three models across six runnable levels on WenwangBench with excluding functions with none return value

Runnable Level	Model	pass@1	pass@5	pass@10
self_contained	CodeGen	17.14%	34.82%	37.14%
	PanGu-Coder	21.14%	34.92%	40.00%
	WenwangCoder	**28.00%**	**39.85%**	**42.86%**
slib_runnable	CodeGen	8.21%	20.04%	25.00%
	PanGu-Coder	**12.50%**	**24.89%**	**32.14%**
	WenwangCoder	12.14%	20.55%	25.00%
plib_runnable	CodeGen	6.84%	20.36%	26.32%
	PanGu-Coder	**13.16%**	**25.13%**	**31.58%**
	WenwangCoder	11.05%	18.30%	21.05%
class_runnable	CodeGen	8.89%	14.66%	16.67%
	PanGu-Coder	11.39%	12.50%	13.89%
	WenwangCoder	**13.61%**	**17.92%**	**19.44%**
file_runnable	CodeGen	7.64%	14.45%	16.36%
	PanGu-Coder	10.36%	13.23%	14.55%
	WenwangCoder	**10.73%**	**16.42%**	**20.00%**
project_runnable	CodeGen	0%	0%	0%
	PanGu-Coder	0%	0%	0%
	WenwangCoder	0%	0%	0%
over_all	CodeGen	8.85%	18.17%	20.83%
	PanGu-Coder	12.08%	18.62%	21.88%
	WenwangCoder	**13.59%**	**20.14%**	**22.92%**

II. Case Study

Figure 1 shows the code generated by the three models with all context and docstring as inputs. The code generated by CodeGen and Pangu-Coder fails to pass the test cases in the WenwangBench, but the code generated by Wenwang-Coder can pass the test cases.

```
“With import re sys pyfs fs namaste
file NORMALIZATIONS ; add_object_args(parser) ; add_shared_args(parser) ;
check_shared_args(args) ; next_version(version) ; remove_first_directory(path) ;
make_unused_filepath(filepath,used,separator) ; find_path_type(path) ; ”
```

All Context

```
"Next version identifier following existing pattern.\n\nMust deal with both zero-
prefixed and non-zero prefixed versions."
```

Docstring

```
def next_version(version):
    m = re.match(r'''v((\d)\d*)$''', version)
    if not m:
        raise ObjectException("Bad version '%s'" % version)
    next_n = int(m.group(1)) + 1
    if m.group(2) == '0':
        # Zero-padded version
        next_v = ('v0%0' + str(len(version) - 2) + 'd') % next_n
        if len(next_v) != len(version):
            raise ObjectException("Version number overflow for zero-
padded version %d to %d" % (version, next_v))
        return next_v
    # Not zero-padded
    return 'v' + str(next_n)
```

Oracle Code

```
def next_version(version):
    for version in range(1, 10):
        yield version
        yield int(version)
```

☒ *CodeGen*

```
def next_version(version):
    return version + 1
```

☒ *PanGu-Coder*

```
def next_version(version):
    m = re.match(r'^v(\\d+)$', version)
    if not m:
        raise ValueError('Unsupported version: {}'.format(version))
    number = int(m.group(1)) + 1
    return 'v{}'.format(str(number))
```

☑ *WenwangCoder*

Fig. 1: Case Study I

Figure 2 shows the code generated by four different models with only docstring as input, where CodeGen and Pangu-Coder failed the test cases of WenwangBench. Significantly, the WenwangCoder can pass the test cases of WenwangBench even if only docstring is available.

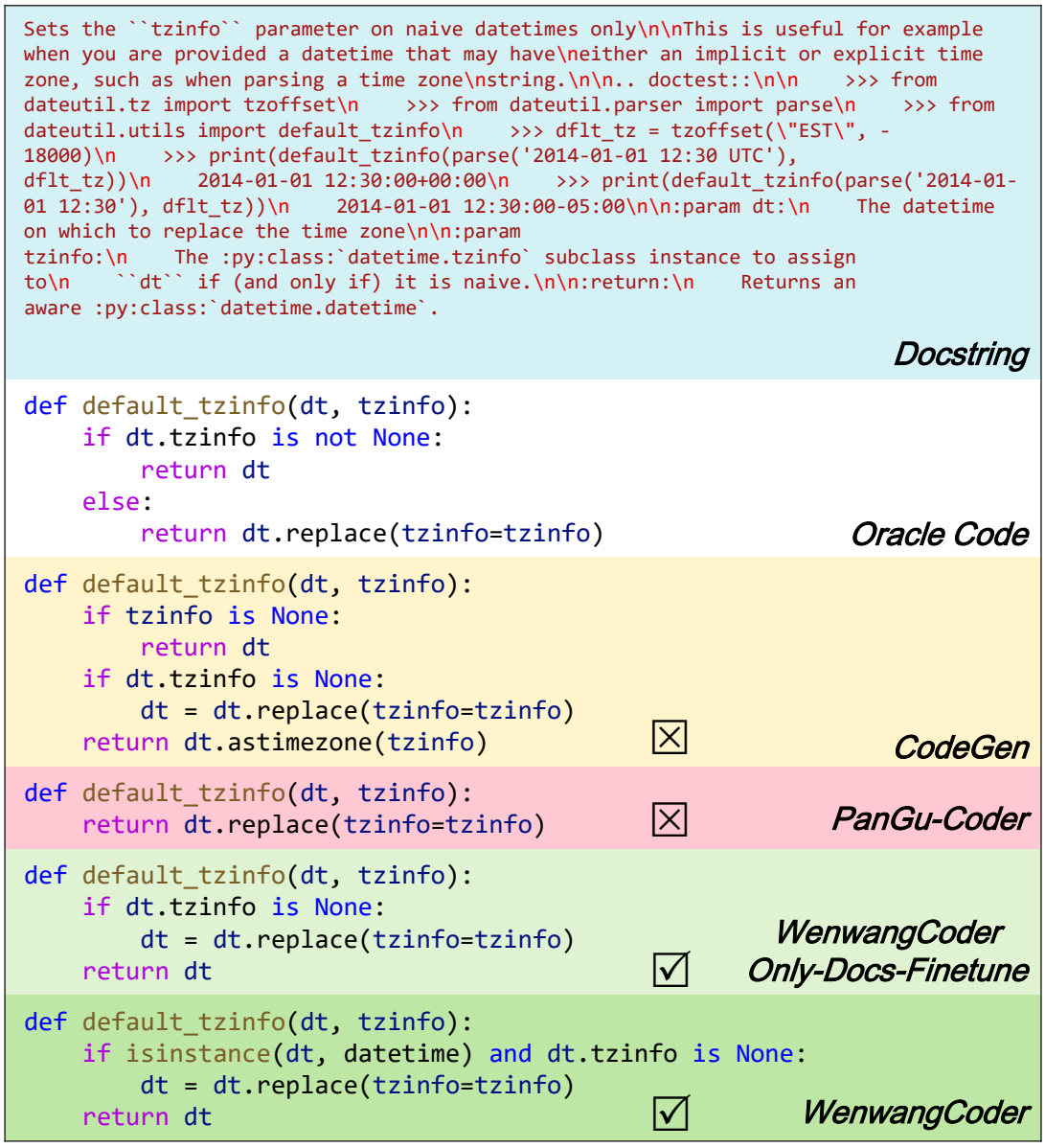

Fig. 2: Case Study II

Figure 3 shows the case where WenwangCoder failed the test cases with only docstring as input, and the other two models passed. However, as we can see, the code generated by WenwangCoder is similar to oracle code with only a few errors and still conforms to what docstring expresses.

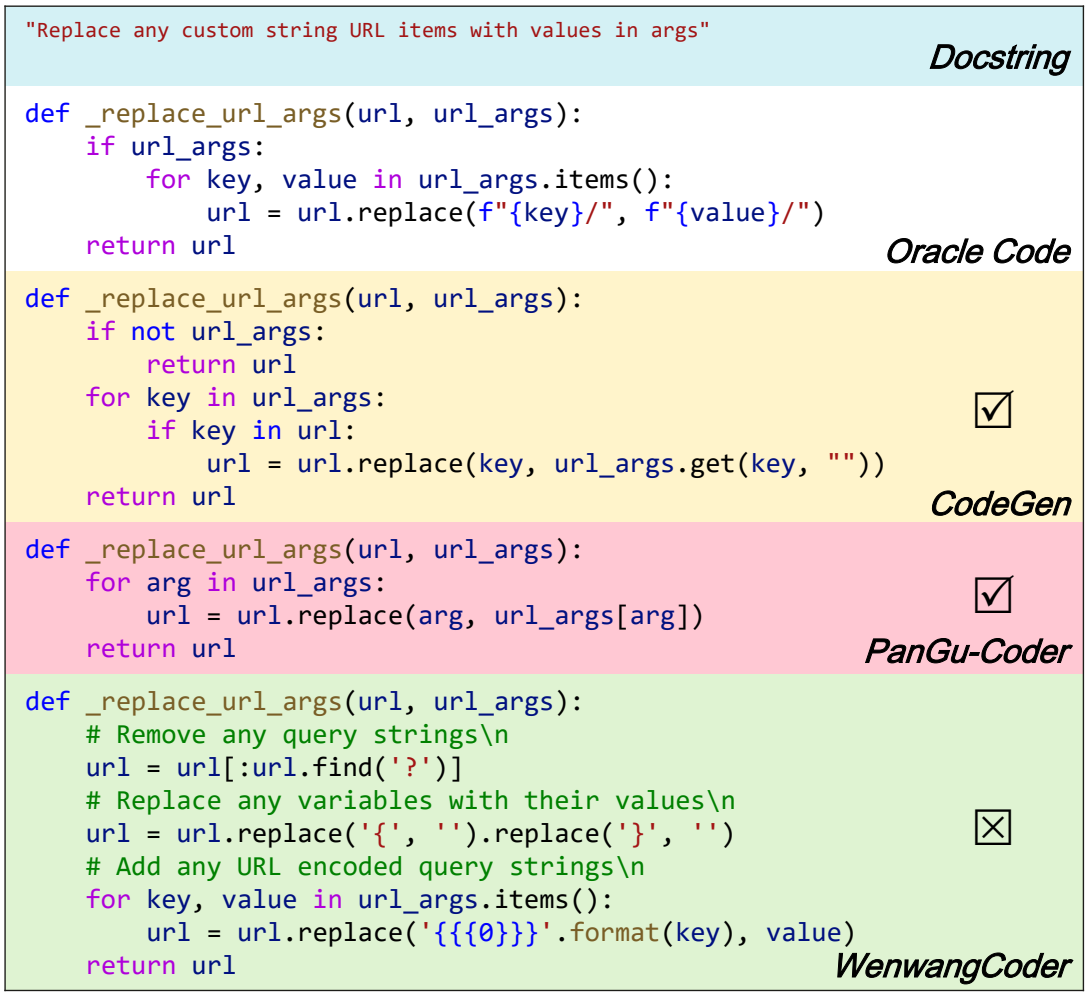

Fig. 3: Case Study III