```
(base) PS C:\Users\Abi Rahman> python
Python 3.10.9 | packaged by conda-forge | (main, Jan 11 2023, 15:15:40) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> # 7. Input and Output
>>> # 7.1. Fancier Output Formatting
>>> year = 2016
>>> event = 'Referendum'
>>> f'Results of the {year} {event}'
'Results of the 2016 Referendum'
>>> yes_votes = 42_572_654
>>> no_votes = 43_132_495
>>> percentage = yes_votes / (yes_votes + no_votes)
>>> '{:-9} YES votes  {:2.2%}'.format(yes_votes, percentage)
' 42572654 YES votes  49.67%'
>>> s = 'Hello, world.'
>>> str(s)
'Hello, world.'
>>> repr(s)
"'Hello, world.'"
>>> str(1/7)
'0.14285714285714285'
>>> x = 10 * 3.25
>>> y = 200 * 200
>>> s = 'The value of x is ' + repr(x) + ', and y is ' + repr(y) + '...'
>>> print(s)
The value of x is 32.5, and y is 40000...
>>> # The repr() of a string adds string quotes and backslashes:
>>> hello = 'hello, world\n'
>>> hellos = repr(hello)
>>> print(hellos)
```

```
'hello, world\n'
>>> # The argument to repr() may be any Python object:
>>> repr((x, y, ('spam', 'eggs')))
"(32.5, 40000, ('spam', 'eggs'))"
>>>
>>> #7.1.1. Formatted String Literals
>>> import math
>>> print(f'The value of pi is approximately {math.pi:.3f}.')
The value of pi is approximately 3.142.
>>> table = {'Sjoerd': 4127, 'Jack': 4098, 'Dcab': 7678}
>>> for name, phone in table.items():
...     print(f'{name:10} ==> {phone:10d}')
...
Sjoerd     ==>       4127
Jack       ==>       4098
Dcab       ==>       7678
>>> animals = 'eels'
>>> print(f'My hovercraft is full of {animals}.')
My hovercraft is full of eels.
>>> print(f'My hovercraft is full of {animals!r}.')
My hovercraft is full of 'eels'.
>>>
>>> bugs = 'roaches'
>>> count = 13
>>> area = 'living room'
>>> print(f'Debugging {bugs=} {count=} {area=}')
Debugging bugs='roaches' count=13 area='living room'
>>>
>>> #7.1.2. The String format() Method
>>> print('We are the {} who say "{}!"'.format('knights', 'Ni'))
We are the knights who say "Ni!"
```

```
>>> print('{0} and {1}'.format('spam', 'eggs'))
spam and eggs
>>> print('{1} and {0}'.format('spam', 'eggs'))
eggs and spam
>>> print('This {food} is {adjective}.'.format(
...       food='spam', adjective='absolutely horrible'))
This spam is absolutely horrible.
>>> print('The story of {0}, {1}, and {other}.'.format('Bill', 'Manfred',
...                                       other='Georg'))
The story of Bill, Manfred, and Georg.
>>> table = {'Sjoerd': 4127, 'Jack': 4098, 'Dcab': 8637678}
>>> print('Jack: {0[Jack]:d}; Sjoerd: {0[Sjoerd]:d}; '
...       'Dcab: {0[Dcab]:d}'.format(table))
Jack: 4098; Sjoerd: 4127; Dcab: 8637678
>>> table = {'Sjoerd': 4127, 'Jack': 4098, 'Dcab': 8637678}
>>> print('Jack: {Jack:d}; Sjoerd: {Sjoerd:d}; Dcab: {Dcab:d}'.format(**table))
Jack: 4098; Sjoerd: 4127; Dcab: 8637678
>>>
>>>
>>> for x in range(1, 11):
...     print('{0:2d} {1:3d} {2:4d}'.format(x, x*x, x*x*x))
...
 1   1    1
 2   4    8
 3   9   27
 4  16   64
 5  25  125
 6  36  216
 7  49  343
 8  64  512
 9  81  729
```

```
10 100 1000
>>>
>>>
>>> #7.1.3. Manual String Formatting
>>> for x in range(1, 11):
...     print(repr(x).rjust(2), repr(x*x).rjust(3), end=' ')
...     # Note use of 'end' on previous line
...     print(repr(x*x*x).rjust(4))
...
 1   1    1
 2   4    8
 3   9   27
 4  16   64
 5  25  125
 6  36  216
 7  49  343
 8  64  512
 9  81  729
10 100 1000
>>>
>>> '12'.zfill(5)
'00012'
>>> '-3.14'.zfill(7)
'-003.14'
>>> '3.14159265359'.zfill(5)
'3.14159265359'
>>>
>>>
>>> #7.1.4. Old string formatting
>>> import math
>>> print('The value of pi is approximately %5.3f.' % math.pi)
```

```
The value of pi is approximately 3.142.
>>>
>>>
>>>
>>> #7.2. Reading and Writing Files
>>> f = open('workfile', 'w', encoding="utf-8")
>>> with open('workfile', encoding="utf-8") as f:
...     read_data = f.read()
...
>>> # We can check that the file has been automatically closed.
>>> f.closed
True
>>> f.close()
>>> f.read()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: I/O operation on closed file.
>>>
>>>
>>> #7.2. Reading and Writing Files
>>> f = open('workfile', 'w', encoding="utf-8")
>>> with open('workfile', encoding="utf-8") as f:
...     read_data = f.read()
...
>>> # We can check that the file has been automatically closed.
>>> f.closed
True
>>> f.close()
>>> f.read()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
```

```
ValueError: I/O operation on closed file.
>>>
>>>#7.2.1. Methods of File Objects
>>> f.write('This is a test\n')
15
>>> value = ('the answer', 42)
>>> s = str(value)  # convert the tuple to string
>>> f.write(s)
18
>>> f = open('workfile', 'rb+')
>>> f.write(b'0123456789abcdef')
16
>>> f.seek(5)      # Go to the 6th byte in the file
5
>>> f.read(1)
b'5'
>>> f.seek(-3, 2)  # Go to the 3rd byte before the end
31
>>> f.read(1)
b'4'
>>>
>>>
>>> #7.2.2. Saving structured data with json
>>> import json
>>> x = [1, 'simple', 'list']
>>> json.dumps(x)
'[1, "simple", "list"]'
>>> json.dump(x, f)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "C:\Users\Abi Rahman\miniconda3\lib\json\__init__.py", line 180, in dump
```

```
    fp.write(chunk)
TypeError: a bytes-like object is required, not 'str'
>>> x = json.load(f)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "C:\Users\Abi Rahman\miniconda3\lib\json\__init__.py", line 293, in load
    return loads(fp.read(),
  File "C:\Users\Abi Rahman\miniconda3\lib\json\__init__.py", line 346, in loads
    return _default_decoder.decode(s)
  File "C:\Users\Abi Rahman\miniconda3\lib\json\decoder.py", line 340, in decode
    raise JSONDecodeError("Extra data", s, end)
json.decoder.JSONDecodeError: Extra data: line 1 column 2 (char 1)
>>>
```