Exercises

1) Import the zipfile module and use the extract function to extract a file from a zip archive.
2) Import the sys module and use the exit function to exit the program with an error code of 1.
3) Import the itertools module and use the permutations function to generate all permutations of a list of numbers.
4) Import the collections module and use the defaultdict class to create a dictionary with a default value of 0.
5) Import the heapq module and use the heappop function to remove the smallest item from a heap.
6) Import the bisect module and use the bisect_right function to find the insertion point for an item in a sorted list.
7) Import the math module and use the floor function to round down a number to the nearest integer.
8) Import the random module and use the choice function to randomly select an item from a list.
9) Import the datetime module and use the strftime function to format a datetime object as a string using a custom format.
10) Import the os module and use the mkdir function to create a new directory.

Exercises and solution

1) To extract a file from a zip archive using the zipfile module, you can use the extract function.

```
import zipfile

with zipfile.ZipFile('example.zip', 'r') as zip_ref:
    zip_ref.extract('example_file.txt', 'extracted_files/')
```

This will extract the file "example_file.txt" from the archive "example.zip" to a new directory called "extracted_files".

2) To exit a Python program with an error code of 1 using the sys module, you can use the exit function.

```
import sys

sys.exit(1)
```

This will exit the program with an error code of 1.

3) To generate all permutations of a list of numbers using the itertools module, you can use the permutations function.

```
import itertools

numbers = [1, 2, 3]
permutations = list(itertools.permutations(numbers))

print(permutations)
```

This will generate all permutations of the list [1, 2, 3].

4) To create a dictionary with a default value of 0 using the collections module, you can use the defaultdict class.

```
from collections import defaultdict

my_dict = defaultdict(int)

my_dict['key1'] += 1

print(my_dict)
```

This will create a dictionary with a default value of 0 and increment the value associated with the key 'key1'.

5) To remove the smallest item from a heap using the heapq module, you can use the heappop function.

```
import heapq

my_heap = [4, 2, 1, 3, 5]
heapq.heapify(my_heap)
smallest_item = heapq.heappop(my_heap)

print(smallest_item)
```

This will remove the smallest item from the heap and print it.

6) To find the insertion point for an item in a sorted list using the bisect module, you can use the bisect_right function.

```
import bisect
my_list = [1, 3, 5, 7, 9]
insertion_point = bisect.bisect_right(my_list, 4)
print(insertion_point)
```

This will find the insertion point for the value 4 in the sorted list [1, 3, 5, 7, 9].

7) To round down a number to the nearest integer using the math module, you can use the floor function.

```
import math

x = 3.7
rounded_x = math.floor(x)

print(rounded_x)
```

This will round the number 3.7 down to 3.

8) To randomly select an item from a list using the random module, you can use the choice function.

```
import random

my_list = ['apple', 'banana', 'cherry']
random_item = random.choice(my_list)

print(random_item)
```

This will randomly select an item from the list ['apple', 'banana', 'cherry'].

9) To format a datetime object as a string using a custom format using the datetime module, you can use the strftime function.

```
import datetime

now = datetime.datetime.now()
formatted_date = now.strftime("%Y-%m-%d %H:%M:%S")

print(formatted_date)
```

This will format the current date and time as a string in the format "YYYY-MM-DD HH:MM:SS".

10) To create a new directory using the os module, using mkdir to create new directory

```
import os

# Create a new directory
new_dir = 'new_directory'
os.mkdir(new_dir)

# Check if the directory was created
if os.path.exists(new_dir):
   print(f"{new_dir} was created successfully!")
else:
   print(f"Failed to create {new_dir}.")
```

This code first imports the os module, which provides a number of functions for interacting with the operating system. It then uses the os.mkdir() function to create a new directory with the name "new_directory".