Item 1

Questions | Data structures

You are tasked with developing a bike rental system. The bike rental service uses a system to track bikes as they are rented and returned. The same bike can be rented multiple times a day, as long as it is available. The bike IDs to track daily rentals are recorded in two Python lists: one for rentals and another for returns. The system should help manage tasks like identifying which bikes are currently rented out, checking if a specific bike is available, and handling new rentals and returns.

For the following exercises, consider the lists `bikes_rented` and `bikes_returned`, which store the IDs of the bikes that have been rented and returned throughout the day, respectively. The bike IDs are stored in ascending order of rental/return (i.e., the first bike to be rented/returned is the first element of the list). For example:

```
bikes_rented = ["B002", "B001", "B002", "B003"]
bikes_returned = ["B001", "B002"]
```

In this example, bike B001 was rented and returned (so it is available for renting); bike B002 was rented and returned once, but is currently rented now; bike B003 is currently rented.

Given these lists, answer the following questions.

Which of the following answers the question "What was the first bike to be returned?"

A
```
bikes_returned["B001"]
```

B
```
bikes_returned[1]
```

C
```
bikes_returned[0]
```

D
```
bikes_returned[-1]
```

Which of the following answers the question "What was the last rented bike?"

A
```
bikes_rented["B003"]
```

B
```
bikes_rented[0]
```

C
```
bikes_rented[-1]
```

D
```
bikes_rented[1]
```

Which of the following answers the question "How many times have bikes been returned?"

A
```
len(bikes_rented) - len(bikes_returned)
```

B
```
len(bikes_returned)
```

C
```
len(bikes_returned) - len(bikes_rented)
```

D
```
len(bikes_rented)
```

Which of the following answers the question "Is there any bike currently rented"?

A
```
len(bikes_rented) == len(bikes_returned)
```

B
```
len(bikes_rented) - len(bikes_returned) == 1
```

C
```
len(bikes_returned) - len(bikes_rented) < 0
```

D
```
len(bikes_returned) - len(bikes_rented) > 0
```

Which of the following answers the question "Has anyone rented bike B010?":

A
```
"B010" not in bikes_rented
```

B
```
"B010" in bikes_rented
```

C
```
bikes_rented["B010"] is not None
```

D
```
"B010" in bikes_returned
```

Which of the following answers the question "Is bike B002 currently rented?"

Note: The function `count` returns how many times an element occurs in a list.

A

```
"B002" not in bikes_returned
```

B

```
"B002" in bikes_rented
```

C

```
bikes_rented.count("B002") > bikes_returned.count("B002")
```

D

```
"B002" in bikes_rented and "B002" not in bikes_returned
```

Item 2

Questions | Data structures

Suppose you now store the duration, in minutes, of each rental in a dictionary. The keys in the dictionary are the bike IDs, and the values are lists with the duration of rentals.

For the following exercises, consider the dictionary:

```
rentals = {
    "B001": [15],
    "B002": [45, 30],
    "B003": [],
    "B004": [],
    "B005": [],
}
```

In this example, bike B001 has been rented once for 15 minutes, bike B002 has been rented twice: once for 45 minutes, once for 30 minutes; and bikes B003-B005 have not been rented (or returned) yet.

The dictionary is updated whenever a bike is returned.

Which of the following answers the question "How many times has bike B002 been returned?"

A
```
rentals["B002"]
```

B
```
len(rentals["B002"])
```

C
```
rentals["B002"][0]
```

D
```
rentals["B002"] != []
```

What is the output of the following code?

```
if "B001" in rentals:
  print("Found")
else:
  print("Not found")
if "B002" in rentals:
  print("Found")
elif "B010" in rentals:
  print("Found")
elif "B006" not in rentals:
  print("Not found")
else:
  print("Found")
```

A

```
Found
Found
Not found
```

B

```
Found
Found
```

C

```
Not found
Found
Found
Not found
```

D

```
Not found
Found
Found
Found
```

What is the output of the following code?

```
for bike, durations in rentals.items():
  if len(durations) > 0 and durations[0] > 15:
    print(bike)
```

A

```
B001
B002
B003
```

B

```
B002
```

C

```
B001
B002
```

D

Nothing

What is the output of the following code?

```
i = 0
durations = list(rentals.values())
while i < len(durations):
  if durations[i] != []:
    print(durations[i][0])
  i = i + 1
```

A

It enters an infinite loop printing 45 (after printing 15 once):

```
15
45
45
45
...
```

B

```
15
45
```

C

```
[15]
[15]
[15]
[15]
[15]
```

D

```
B001
B002
```

Which of the following answers the question "How many times were bikes returned?"

A

```
count = 0
while count < len(rentals):
  if len(rentals[count]) > 0:
      count += 1
print(count)
```

B

```
count = 0
for bike in rentals:
    count += len(bike)
print(count)
```

C

```
count = 0
for durations in rentals.values():
    if durations != []:
        count += 1
print(count)
```

D

```
count = 0
for durations in rentals.values():
    count += len(durations)
print(count)
```

Which of the following correctly answers the question "Of the bikes that have been returned, what was the duration of the last rental"?

A

```
for bike in rentals:
    if rentals[bike] != []:
        print(bike)
```

B

```
for bike in rentals:
    if rentals[bike] != []:
        print(len(rentals[bike]))
```

C

```
for bike in rentals:
    if rentals[bike] != []:
        print(rentals[bike][0])
```

D

```
for bike in rentals:
    if rentals[bike] != []:
        print(rentals[bike][-1])
```

Fill in the blanks to answer the question "How many bikes have had at least two rentals with the last one being shorter than 15 minutes?":

```
count = 0
for bike in rentals:
    if [1]          [2]          [3]          :
        count += 1
print(count)
```

Correct answers:

1 len(rentals[bike]) >= 2 2 and 3 rentals[bike][-1] < 15

Fill in the blanks to answer the question "Which bikes have been returned more than once, with at least one rental duration being longer than 20 minutes?"

```
count = 0
for bike in rentals:
  times = [1]
  if [2]          :
    for time in [3]          :
      if [4]          :
        print([5]          )
        break
```

Correct answers:

1 rentals[bike] 2 len(times) > 1 3 times 4 time > 20 5 bike

Item 3

Questions | Data structures

Given the following function for counting how many bikes have had rentals longer than some duration:

```
def rented_longer_than(time):
    ...
```

Which of the following options **does not throw an error** when calling the function?

A
```
rented_longer_than(time=10)
```

B
```
rented_longer_than(20, 30)
```

C
```
rented_longer_than(interval=10)
```

D
```
rented_longer_than()
```

The rental price per minute changes depending on the time of rental, according to the following rules:

- For the first 15 minutes, the price is 0.1EUR/min
- After that, the price is 0.2EUR/min
- For rentals longer than 45 minutes, on top of the rental price, there is an additional fixed charge of 5EUR.

Fill in the blanks to complete the definition of a function named `rental_price` that takes one argument, `duration`, and computes the rental price of the bike for a given duration according to the rules described above.

```
def rental_price( 1 ):
  total_price = 0
  if duration  2  15:
    total_price += 0.1 * duration
  else:
    total_price += 0.1 *  3
    total_price += 0.2 *  4
    if  5 :
      total_price += 5
  return total_price
```

Correct answers:

1 duration 2 <= 3 15 4 (duration - 15) 5 duration > 45

Which of the following correctly defines the function `profit` that takes no arguments and computes the profit of the rentals. This value is obtained by adding the rental price, for the length of each rental, of all the rentals registered in the system.

```
def profit():
  total = 0
  for bike in rentals:
    total += rental_price(bike)
  return total
```

```
def profit():
  total = 0
  for bike in rentals:
    for duration in rentals[bike]:
      total += rental_price(duration)
  return total
```

✔

```
def profit():
  total = 0
  for bike in rentals:
    for duration in rentals[bike]:
      total += rental_price(duration)
    return total
```

```
def profit():
  total = 0
  for bike in rentals:
    for duration in rentals[bike]:
      total += rental_price(duration)
      return total
```

Considering the following function

```
def bike_label(name, reverse=True):
  for bike in rentals:
    if reverse:
      return(f"{bike}: {name}")
    else:
      return(f"{name}: {bike}")
  return "No bikes"
```

What is the output of the following code?

```
bike_label(reverse=False)
```

A

```
No bikes
```

B

```
name: B001
name: B002
name: B003
name: B004
name: B005
```

C

```
B001: name
B002: name
B003: name
B004: name
B005: name
```

D

It throws an error ✔

Item 4

Questions | Data structures

Consider the following class to a bike with its ID:

```
class Bike:
  def __init__(self, bike_id):
    self.bike_id = bike_id
```

Implement the `BikeShare` class to represent a bike rental system as described earlier.

The class attributes are:

- `rentals` : A dictionary with the rental duration of each bike, as described earlier.
- `maintenance` : A list with bike IDs that are currently unavailable due to maintenance.

Fill in the blanks to implement the class constructor. The constructor takes one argument, `bike_ids` , a list with the IDs of bikes in the system, which is used to initialize the `rentals` attribute. All bike ids are added to the dictionary with their value being an empty list (i.e. no rentals yet). The attribute `maintenance` is initialized as empty list.

```
class BikeShare:
  def __init__(  1  ):
      2
      3
    for bike in bike_ids:
        4 [bike] = []
```

Correct answers:

1 self, bike_ids 2 self.maintenance = [] 3 self.rentals = {} 4 self.rentals

Which of the following answers the question "Create a new bike rental system with bikes A001-A004"?

A
```
bikes = BikeShare(self, ["A001", "A002", "A003", "A004"])
```

B
```
bikes = BikeShare("A001", "A002", "A003", "A004")
```

C
```
bikes = BikeShare(bikes=["A001", "A002", "A003", "A004"])
```

D
```
bikes = BikeShare(["A001", "A002", "A003", "A004"])
```

Which of the following correctly defines the method `bike_unavailable` that takes one argument, `bike` (an object of the `Bike` class), and adds the bike ID to the `maintenance` attribute?

A

```
def bike_unavailable(self, bike):
  self.maintenance.append(bike.bike_id)
```

B

```
def bike_unavailable(self, bike):
  self.maintenance.append(bike)
```

C

```
def bike_unavailable(self, bike):
  maintenance.append(bike.bike_id)
```

D

```
def bike_unavailable(bike):
  self.maintenance.append(bike.bike_id)
```

Fill in the blanks to complete the definition of method `working_bikes` in the `BikeShare` class, to return a list with the IDs of available bikes (i.e those that are not out for maintenance). This method takes no arguments.

```
class BikeShare:
  ...
  def working_bikes( 1 ):
    available = []
    for bike in 2 :
      if bike 3 :
        4
    return 5
```

Correct answers:

1 self 2 self.rentals 3 not in self.maintenance 4 available.append(bike)

5 available

What is the output of the following code?

```
bike = Bike("A001")
share = BikeShare(["A001", "A002"])
share.bike_unavailable(bike)
print(share.working_bikes(), share.maintenance()])
```

A `["A002"] ["A001"]`

B `["A001"] ["A002"]`

C `[] []`

D It throws an error ✔