

### Question 1.1: Give the names and examples of any 5 data types in Python.

Computers are used to store information permanently in many ways, like a file or a database. Nevertheless, when a program is running needs to store information temporarily as it might need to be manipulated. To do this, a program uses variables, which can store data of different types, called "Data Type", that can do different things. Python has the following main built-in data types by default: **Basic** and **Advanced** (Shown below with examples). The **Basic** Data Types include **Numeric** types, such as **int**, **float**, and **complex**; **Boolean** and **Strings**. The **Advanced** Data Types include **Set**, **Dictionaries** and **Sequence Type** such as **List**, and **Tuple**.

#### Basic Data Types:

1. **Numeric Types (int, float, complex)**: These can be either integers (**int**), known as whole numbers, or floats (**float**), known as decimal numbers, and complex (**complex**).

**Examples:** **int**--> x = 20      **float**--> x = 20.6      **complex**--> x = 5j

2. **Boolean (or bool) Type**: These are True/False values.

**Example**      **bool**--> x = True    **and/or**    x = False

3. **String or Text Type (str)**: These are text values composed of a sequence of characters (storing text in strings), such as letters, numbers, spaces, and symbols. Sets are mutable but not ordered, indexing/slicing and/or duplicate elements.

**str**-->    **Example:** x = "Hello World", "x = 43", x = "^"

#### Advanced-Data Types:

**Sequence Types (list, tuple, range)**: These are collections of data types that can be the same or different.

4. **list** --> a collection of ordered, mutable, dynamic, non-unique elements enclosed in square brackets [] and separated by commas. **Example:** [1, 2, 3, 4, 5] or [1, 'blueberry', 2, 'blackcurrant', 3, 'redcurrant']

5. **tuple**--> a collection of ordered, immutable, non-unique elements enclosed in parentheses (), and faster than lists.

**Example:** ("blueberry", "blackcurrant", "redcurrant")

6. **range**--> a sequence of numbers generated by the range() function.

**Example:** x = range (6)

#### Mapping Types:

7. **dict**--> or dictionaries is a collection of key-value pairs enclosed in curly braces{}

**Example:** {"name": "John", "age": "37"}

8. **Set Types**: An unordered collection of unique elements.

**Example**      **set**--> {"blueberry", "blackcurrant", "redcurrant"}  
                 **frozenset**--> ({"blueberry", "blackcurrant", "redcurrant"})

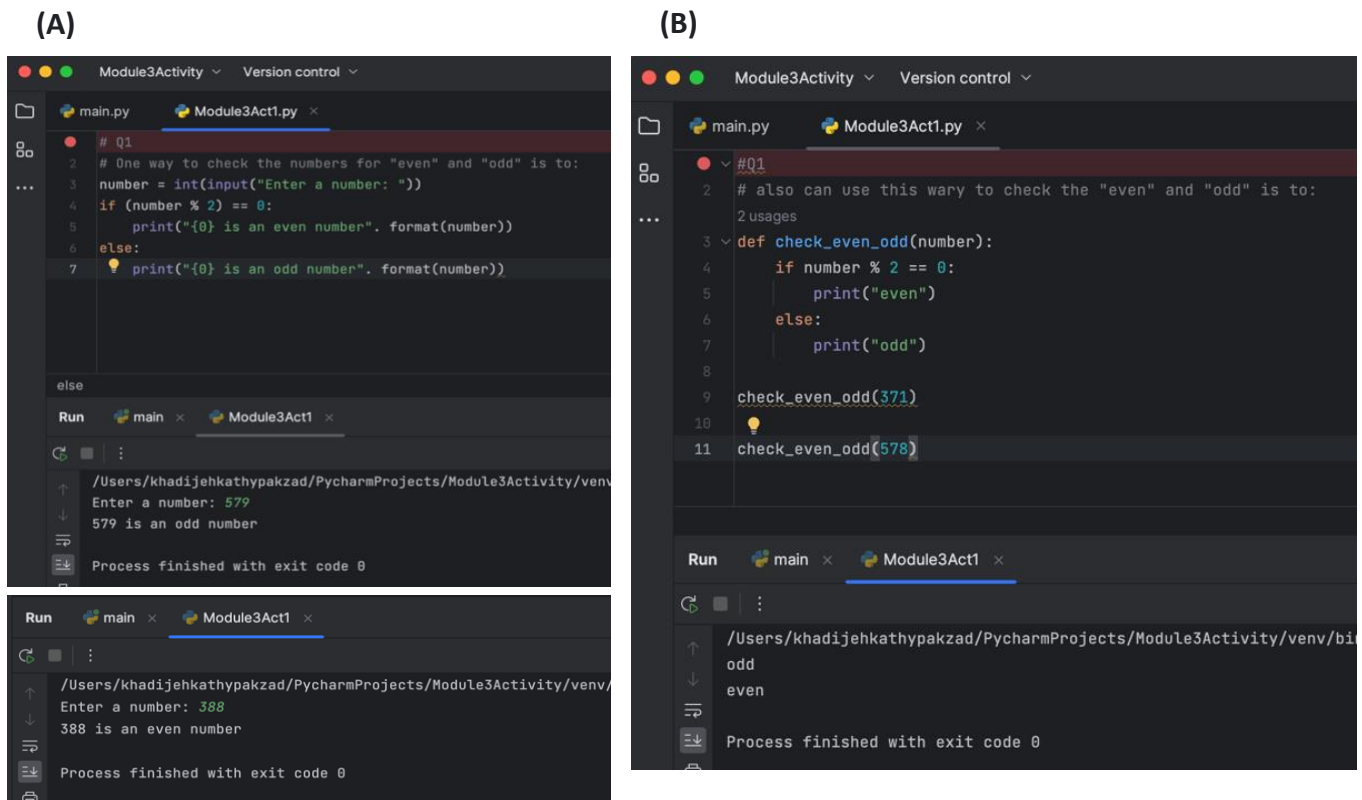
9. **None Type**: a special data type representing the absence of a value

**Example**      **NoneType**--> x = None

### Question 1.2: Create a simple function that takes an integer as an input and prints out "even" if the number is even and "odd" if the number is odd.

Here, we get the input from the user, user types a number, which is a string, so it needs to be converted to an integer and saved it in a variable called number. To see if the number is even or odd, we use the modulus operator to see the remainder of the division of the number divided by two. If this division of our number by 2 is zero means that our number is even. Also, we can format this string to look more informative by calling the format function and formatting this number. However, if the remainder is not zero, it means that our number is odd (**Figure.1. A**).

On the other hand, we can use a simple Python function that takes an integer as input and prints "even" if the number is even and "odd" if the number is odd and then call this function by passing an integer as an argument, and it will print the corresponding result (**Figure.1. B**).

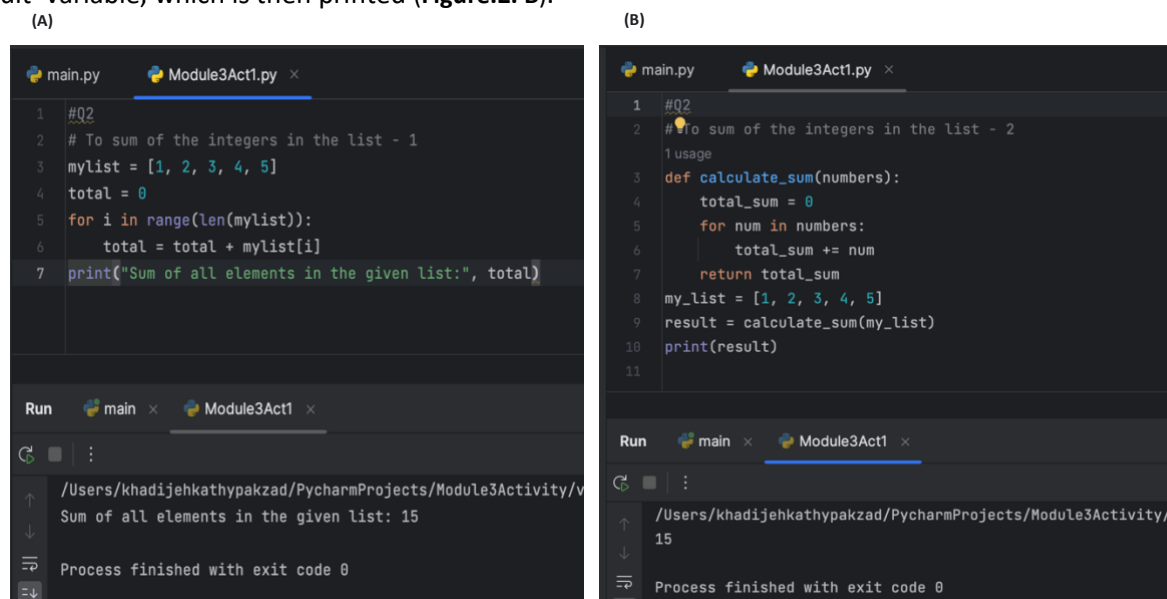


**Figure 1.** Two ways, A and B, of creating simple functions to count even and odd digits in a number in Python.

**Question 2:** Write a Python function that takes a list of integers as input and returns the sum of all the integers in the list. For example, if the input list is [1, 2, 3, 4, 5], the function should return 15, which is the sum of all the integers in the list. To solve this problem, you will need to use data types (lists, integers), loops (for loop), and functions (defining a function to calculate the sum of a list of integers).

There are multiple approaches to solving this problem one would be writing the loop statement with the range function. First, define mylist with the numbers, to extract values from this list and add that value to the total variable. Then using the loop statement, for i (representing index), in the range from zero to the number of elements in the list, using the len method, starting from zero to the end of the list. Finally, print the sum of all elements in the list (**Figure.2. A**).

On the other hand, can call this function by passing a list of integers as an argument, which will return the sum of all the integers (**Figure.2. B**). Here, we initialized a variable 'total\_sum' to 0. Then, using a 'for' loop, we iterate over each element 'num' in the 'numbers' list. Then add each 'num' to the 'total\_sum' variable. Finally, we return the 'total\_sum'. By calling 'calculate\_sum(mylist)', the function will calculate the sum of the numbers in the provided list and store the result in the 'result' variable, which is then printed (**Figure.2. B**).

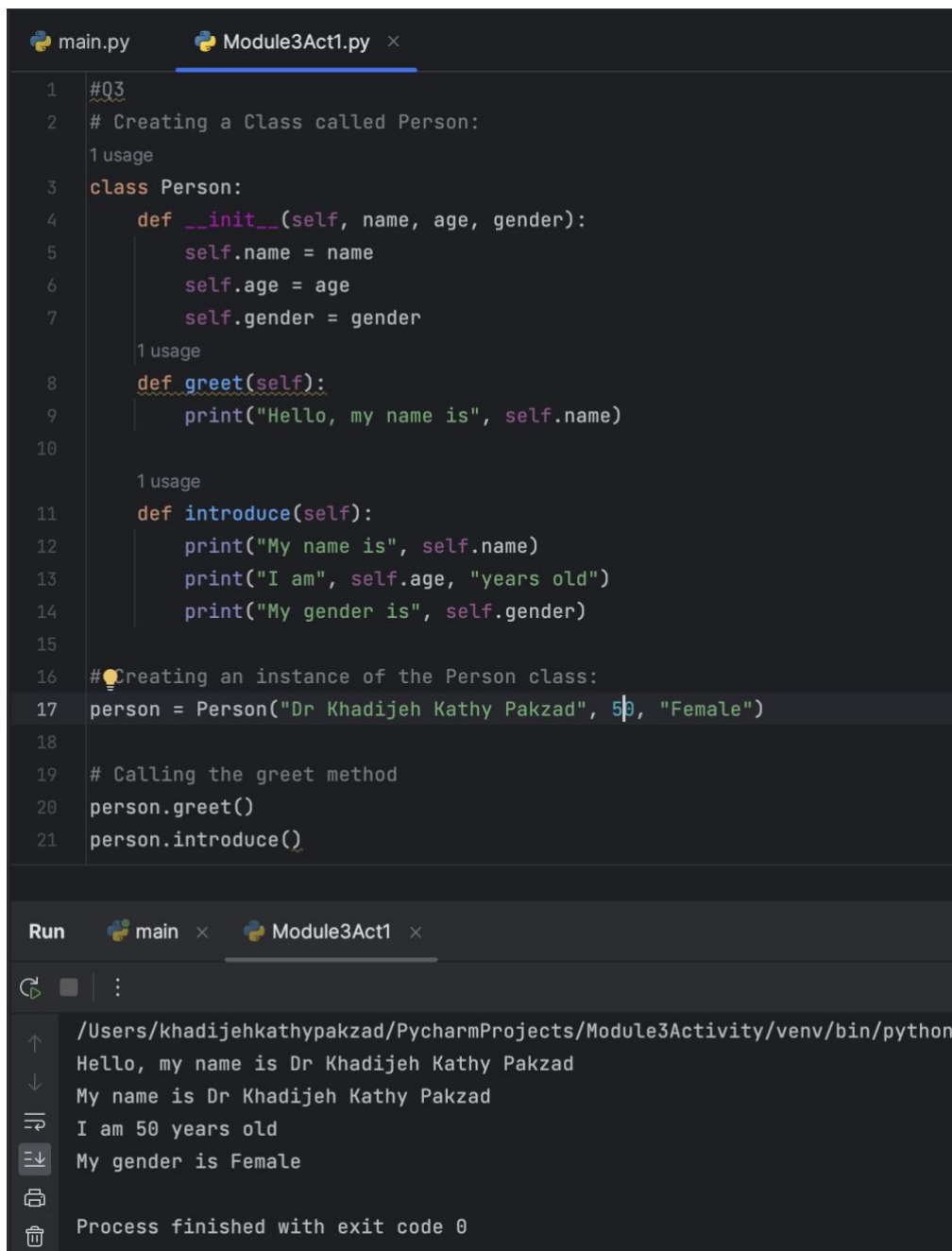


**Figure 2.** Two ways, A and B, of creating simple functions to return the sum of a list of integers as input in Python.

**Question 3:** Create a class called `Person` that has the following attributes: `name` (a string), `age` (an integer), `gender` (a string). The class should have a method called `greet` that takes no input parameters and prints out a greeting message with the person's name.

In this code, we define the `'Person'` class with the `'name'`, `'age'`, and `'gender'` attributes. The `__init__` method is a special method that is called when an instance of the class is created. It initialises the attributes of the object using the provided values. The `'greet'` method is defined within the class, which takes no input parameters. It simply prints a greeting message with the person's name using the `'self.name'` attribute. To use the class, we create an instance of the `'Person'` class called `'person'` with desired attributes. We can then call the `'greet'` method on the `'person'` object, which will print the greeting message with the person's name (**Figure 3**).

In addition, I added a new method called `'introduce'` to the `'Person'` class, that point out the person's name, age, and gender. You can call this method on the `'person'` object after calling the `'greet'` method. By adding the `'introduce'` method, you can now get an introduction including the person's name, age, and gender by calling `'person.introduce()'` (**Figure 3**).



```
main.py  Module3Act1.py x
1  #Q3
2  # Creating a Class called Person:
3  1 usage
4  class Person:
5      def __init__(self, name, age, gender):
6          self.name = name
7          self.age = age
8          self.gender = gender
9  1 usage
10 def greet(self):
11     print("Hello, my name is", self.name)
12
13 1 usage
14 def introduce(self):
15     print("My name is", self.name)
16     print("I am", self.age, "years old")
17     print("My gender is", self.gender)
18
19 # Creating an instance of the Person class:
20 person = Person("Dr Khadijeh Kathy Pakzad", 50, "Female")
21
22 # Calling the greet method
23 person.greet()
24 person.introduce()
```

Run main x Module3Act1 x

/Users/khadijehkathypakzad/PycharmProjects/Module3Activity/venv/bin/python

Hello, my name is Dr Khadijeh Kathy Pakzad

My name is Dr Khadijeh Kathy Pakzad

I am 50 years old

My gender is Female

Process finished with exit code 0

**Figure 3.** Creating `'Person'` class with the specified attributes and `'greet'` method in Python.

- **Question 4:** Use **NumPy** to create a 2D array with the following values: 1, 2, 3, 4, 5, 6. Convert the **NumPy** array into a **Pandas** DataFrame with the column names "A", "B", and "C". Use **Pandas** to select the second row of the DataFrame. Use **Pandas** to select the value in column "B" of the second row.

Here we first import numpy as np and panda as pd. Then create a 2D array with NumPy, convert the array into a DataFrame with column names, then select the second row of the DataFrame as well as the value in the column "B" of the second row (**Figure 4, Input, Output**).

#### Input:

```
main.py  Module3Act1.py x
1  #Q4
2  import numpy as np
3  import pandas as pd
4
5  # Create a 2D array with NumPy:
6  array_2d = np.array([[1, 2, 3],
7                      [4, 5, 6]])
8
9  # Convert the array into a DataFrame with column names:
10 df = pd.DataFrame(array_2d, columns=['A', 'B', 'C'])
11
12 # Select the second row of the DataFrame:
13 second_row = df.iloc[1]
14
15 # Select the value in column "B" of the second row:
16 value_b = df.loc[1, 'B']
17
18 # Print the results:
19 print("DataFrame:")
20 print(df)
21 print("\nSecond Row:")
22 print(second_row)
23 print("\nValue in column B of the second row:", value_b)
```

#### Output:

```
Run  main x  Module3Act1 x
/Users/khadijehkathypakzad/PycharmProjects/Module3Act1
DataFrame:
   A  B  C
0  1  2  3
1  4  5  6

Second Row:
A    4
B    5
C    6
Name: 1, dtype: int64

Value in column B of the second row: 5

Process finished with exit code 0
```

**Figure 4.** Creating a 2D array using NumPy and converted into a Pandas DataFrame and selecting a specific row or value at a specific location.

#### Reference:

1. Python for Data Analysis by Wes McKinney (Third Edition, 2022).
2. Python in 24 hours, Sams Teach Yourself by Katie Cunningham (Second Edition).
3. Plenty of internet searches and YouTube tutorials on Python.