

**Python Object Oriented Programming**

Assignment 1

# Python Object Oriented Programming

### Software Development

|  |  |
| --- | --- |
| image001.png | Python Object Oriented Programming |

### Pledge of Honour

You are required to include the following as documentation at the beginning of every assessment:

Student name: First\_name Last\_name

Student ID: 123456789

Pledge of Honour: I pledge by honour that this program is solely my own work.

### Plagiarism

Any course work presented for assessment must be your own work. Copying or paraphrasing someone else’s work be it published, unpublished or off the internet, without clearly acknowledging it constitutes plagiarism and is considered to be academic misconduct. You are required to sign an assignment declaration stating it is your own work. You may receive a zero for part or all of the assessment submitted in first instance. Repeated incidents of plagiarism or cheating could result in you being removed from the course or the programme.

### Marking Criteria of Assignment 1

This is how the questions are marked in Assignment 1:

1. To achieve the full marks on each question, your program must be complete and work correctly. This means:
   1. The program must do what the question says.
   2. The program must make the same output as the Example Output.
2. Your program solutions need to have these documentation comments at the beginning:
   1. Student name and ID
   2. Pledge of Honour declaration
3. Your program should use the Programming Best Practices at the end of this document.

### Assignment 1

**Learning outcomes:** The course work questions cover Learning Outcomes 1 – 3

**Aims:** The course work questions aim to practice these topics:

* Basic input and output operations
* Problem solving techniques
* Use of built-in classes
* Coding standards
* Debugging and testing by using features of Integrated Development Environment (IDE)

**Weighting:** 15%

**Mark allocation on questions:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Question No. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Total |
| Marks | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 15 |

**Due date:** see timetable on Moodle

**Marking process:**

Upload your solutions (as a zip file) to Moodle.

Question 1

|  |
| --- |
| Formatting escape sequence |
| * Learning outcomes: 1&3 * Relevant topics: output, escape sequence characters, output formatting * Suggested time to complete: Week 1 * Workbook topic: *Formatted Output* |

Make a program that will show the output in the picture below. You will need to use the Escape Sequence characters to format the text correctly.

Remember: the backslash **\** and the double quotes **“** are both special characters. To make them show in a string as normal characters, both need a backslash in front of them. For example, \\ makes one backslash as a normal character, and \" makes a double quotes character.

Do not make the white space using the spacebar. You need to format the output using templates like:

template = **"{0:<10},{1:>15}"**result = template.format(**"Hello"**, **"friend"**)  
print(result) *# output: Hello , friend*

Make this output:

Escape sequence Description

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\n New line character

\t Tab character

\" Double quote character

Question 2

|  |
| --- |
| Circle arithmetic calculation |
| * Learning outcomes: 1&3 * Relevant topics: input, output, arithmetic operators, composite formatting * Suggested time to complete: Week 2 * Workbook topic: *Input Statements, Formatted Output* |

Write a program that asks from the user for a number (data type: float) as the radius of a circle. Calculate area and perimeter of the circle and show the results in a table format. The results should be rounded to two decimal places by using composite formatting. Here is the formula for calculating circle area and perimeter.

area = radius \* radius \* π

perimeter = radius \* 2 \* π

where π = 3.1416

Make this output:

Enter a radius: 8.5

Radius Area Perimeter

-------------------------------

8.5 226.98 53.41

Question 3

|  |
| --- |
| Calculate grade |
| * Learning outcomes: 1&3 * Relevant topics: input, output, selection * Suggested time to complete: Week 3 * Workbook topic: Decision Making |

Write a program that asks from the user for a score. If the user input mark is outside 0 – 100, show an error message. Otherwise, a grade is calculated using this table:

|  |  |
| --- | --- |
| **Score** | **Grade** |
| 0-50 (including 0 but not 50) | D |
| 50-70 (including 50 but not 70) | C |
| 70-80 (including 70 but not 80) | B |
| 80-100 (including both 80 and 100) | A |

Example output:

Enter the score: 104

Please enter a score between 0 and 100

Enter the score: 77

Grade is B

Question 4

|  |
| --- |
| Card letter user input validation |
| * Learning outcomes: 1&3 * Relevant topics: input, output, iteration * Suggested time to complete: Week 4 * Workbook topic: Loops |

Write a program that asks the user to input the letter of a play card. The user input must be either K, Q, J or A. Your program should let the user type in either upper or lower case. You can use .lower() to make the input lower case. Use a WHILE loop to ask the user for K, Q, J or A until they enter K, Q, J or A. When the user enters K, Q, J or A, use an IF-ELIF statement to output the correct card.

|  |  |
| --- | --- |
| **Letter** | **Message** |
| K | King |
| Q | Queen |
| J | Jack |
| A | Ace |

Example output:

Enter a card letter: x

Try again.

Enter a card letter: F

Try again.

Enter a card letter: j

Jack

Question 5

|  |
| --- |
| Getting and calculating a list of temperatures |
| * Learning outcomes: 1&3 * Relevant topics: input, output, iteration, lists * Suggested time to complete: Week 5 * Workbook topic: List and Dictionaries, Loops |

The New Zealand Weather Met Office uses sensors around the country to record temperatures at different locations.

Write a program where the user will enter some temperature data. Your program will first ask the user how many sensors there are. Using a FOR loop get the temperatures from the user and put them in a **list**.

You can use the number of sensors to get the correct amount of temperatures. Your program should calculate and show the average temperature, maximum temperature and the number of temperatures between 10 and 20.

Example output:

How many sensors are currently collecting data?: 5

Enter temperature 1: 16.5234

Enter temperature 2: 12.7

Enter temperature 3: 22.76

Enter temperature 4: 10

Enter temperature 5: 19.33

Average temperature: 16.26

Maximum temperature: 22.76

Number of temperatures between 10 and 20: 4

Question 6

|  |
| --- |
| Guess my secret number |
| * Learning outcomes: 1&3 * Relevant topics: input, output, iteration, lists * Suggested time to complete: Week 5 * Workbook topic: Loops, Lists |

Write a program that makes a secret number using a random number between 0 and 20. You will need to use:

**import** random  
  
rand = random.randint(0, 20)

The program then asks the user to enter 4 numbers that are separated by commas. Your program should then split the comma-separated numbers into a **list**. Remember that the values in the list will be strings. You need to convert them to integers.

Use a FOR loop to go through the list and see if it has the secret number:

Example output:

Guess my secret number between 0 and 20 using 4 guesses separated by commas: 3,6,14,18

You won! My secret number was: 3

Guess my secret number between 0 and 20 using 4 guesses separated by commas: 1,5,6,9

You lost! My secret number was: 4

Question 7

|  |
| --- |
| Count the intersection of two lists |
| * Learning outcomes: 1&3 * Relevant topics: input, output, iteration, lists * Suggested time to complete: Week 5 * Workbook topic: Loops, Lists |

Write a program that asks the user to enter 2 lines of numbers (data type: double), and count the total number of values that are in both **lists**. The 2 numbers will be separated by commas **,**

The 2 lists can be different sizes. You can use 2 nested FOR loops to look at both lists and count the number of values that are in both lists.

In the example below, the first line has 5 numbers and the second list has 8 numbers and there are 2 numbers that are in both lists.

Example output:

Enter the first line of numbers separated by commas: 4.56,5,6.82,8.0,10.547

Enter the second line of numbers separated by commas: 3.67,4.11,4.9,5,6.77,10.547,11.6234,15.4

There are 2 numbers that are in both lists

Question 8

|  |
| --- |
| Dictionary operations |
| * Learning outcomes: 1&3 * Relevant topics: input, output, iteration, lists * Suggested time to complete: Week 5 * Workbook topic: Loops, Dictionaries |

|  |  |
| --- | --- |
| **Country** | **Temperature** |
| Australia | 22.5 |
| China | 36.4 |
| Malaysia | 38.4 |
| New Zealand | 18.2 |

Write a program that makes a **dictionary** with the data in the table above.

Your program should print out the country, then a colon **:** then the temperature.

Your program will then calculate and show the average temperature, maximum temperature and the number of temperatures between 20 and 30.

Example output:

Malaysia: 38.4

China: 36.4

Australia: 22.5

New Zealand: 18.2

Average temperature: 28.88

Maximum temperature: 38.40

Number of temperatures between 20 and 30: 1

### Help

Programming Best Practices

Code readability is one of the first things we learn as programmers. A program is only written once but will be looked at many times by you or other people later. It is important to make your code readable and understandable. Here are some best practices when writing readable code.

1. **Indentation** is the whitespace characters you put at the beginning of a line of code. Indentation is important in Python. Here is an example:

Good indentation:

x = 5  
  
**if** x == 5:  
 print(**"x is equal to 5"**)

NOT good indentation **(error)**:

x = 5  
  
**if** x == 5:  
print(**"x is equal to 5"**)

1. **Code spacing**. This is another making code able to be read easily. Here are some examples of code spacing.

name = **"John Smith"** *# Good*name=**"John Smith"** *# NOT good*print(**"Hello world"**) *# Good*print (**"Hello world"**) *# NOT good*a = b + c *# Good*a=b+c; *# NOT good*a = b - c *# Good*a = b-c *# NOT good*a = b \* 2 + c / 3 *# Good*a = b\*2 + c/3 *# NOT good*

1. **Comments**. Try not to write comments that do not need to be written. Remember, you do not have to write a comment for every line of code. Here are 2 examples, where the comments in the first is good, and the number of comments in the second example is too large.

**def** GoodComments():  
  
 *# Calculate and print area of a square* l = int(input(**"Enter length of side: "**))  
  
 **if** l <= 0:  
 print(**"Please enter a side greater than 0"**)  
 **else**:  
 print(**"Area is {0}"**.format(l \* l))  
  
  
  
  
**def** TooManyComments():  
  
 *# Get length of side of the square from user* l = int(input(**"Enter length of side: "**))  
  
 *# If the length is less than or equal to 0* **if** l <= 0:  
 *# Show error message* print(**"Please enter a side greater than 0"**)  
 **else**: *# If the length is greater than 0  
 # Calculate and print the area of the square* print(**"Area is {0}"**.format(l \* l))

1. **Naming variables and methods**. The names of variables and methods are important programming. The names help you and others understand what the code does. For example, the variable name max\_score is a much better name than xyz. The second aspect of naming scheme is that you should have word boundaries in the name.

You should also keep the same style of naming. There are two options, **camelCase** and **underscores**. The **underscores** option is recommended for the Python language. For example:

my\_dog, max\_temperature