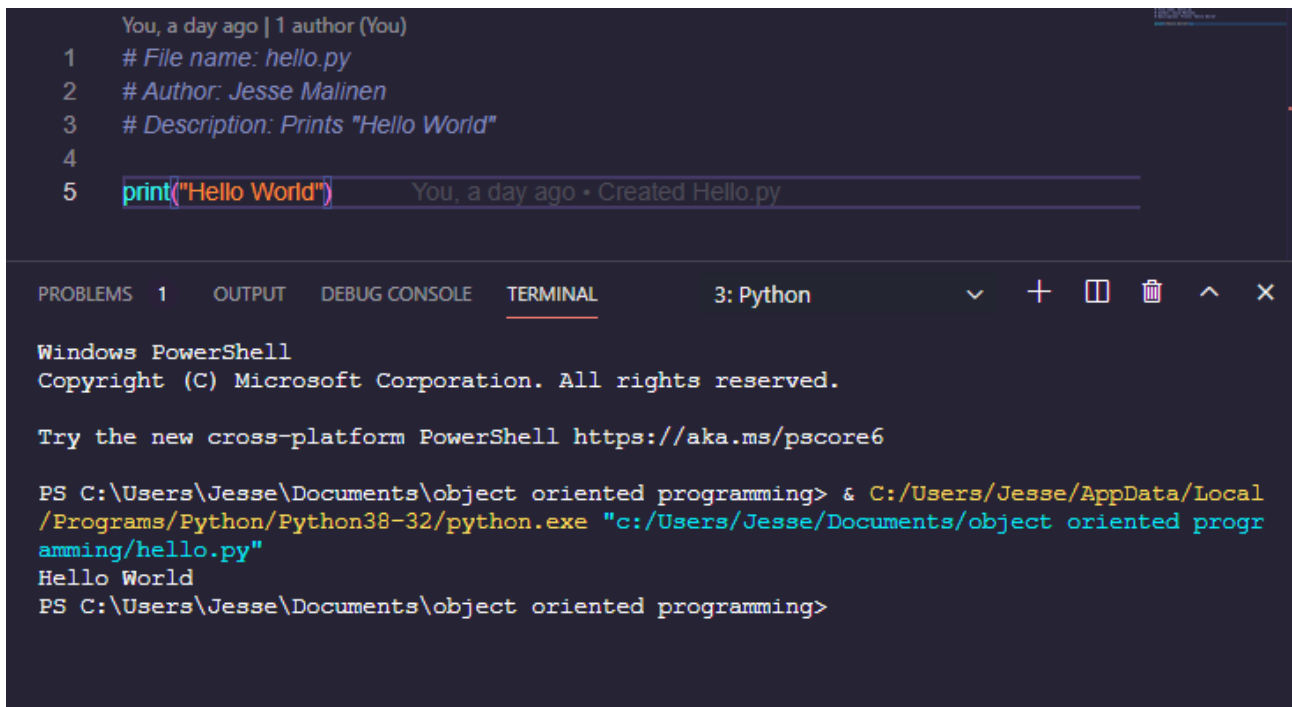


**Exercise 1 – Jesse Malinen** (Remember to fill your first and last name both here and in the file name)

1. print hello



The screenshot shows a code editor with a dark theme. The editor displays a Python file named 'hello.py' with the following content:

```
You, a day ago | 1 author (You)
1 # File name: hello.py
2 # Author: Jesse Malinen
3 # Description: Prints "Hello World"
4
5 print("Hello World")
```

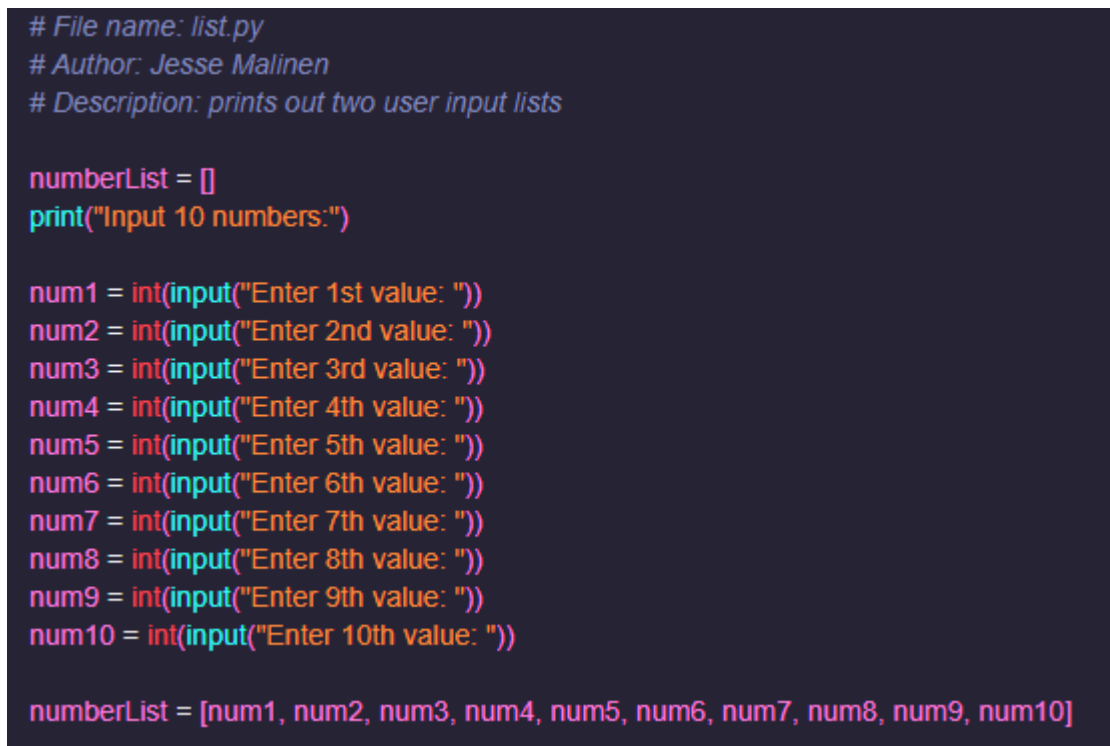
Below the editor is a terminal window titled '3: Python'. It shows the command prompt running the script:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Jesse\Documents\object oriented programming> & C:/Users/Jesse/AppData/Local/Programs/Python/Python38-32/python.exe "c:/Users/Jesse/Documents/object oriented programming/hello.py"
Hello World
PS C:\Users\Jesse\Documents\object oriented programming>
```

2. List of 10 numbers & strings, print lists, random numbers
3. arrange lists



The screenshot shows a code editor with a dark theme. The editor displays a Python file named 'list.py' with the following content:

```
# File name: list.py
# Author: Jesse Malinen
# Description: prints out two user input lists

numberList = []
print("Input 10 numbers:")

num1 = int(input("Enter 1st value: "))
num2 = int(input("Enter 2nd value: "))
num3 = int(input("Enter 3rd value: "))
num4 = int(input("Enter 4th value: "))
num5 = int(input("Enter 5th value: "))
num6 = int(input("Enter 6th value: "))
num7 = int(input("Enter 7th value: "))
num8 = int(input("Enter 8th value: "))
num9 = int(input("Enter 9th value: "))
num10 = int(input("Enter 10th value: "))

numberList = [num1, num2, num3, num4, num5, num6, num7, num8, num9, num10]
```

```
wordList = []
```

```
print("Input 10 words:")
```

```
w1 = input("Enter 1st word: ")
```

```
w2 = input("Enter 2nd word: ")
```

```
w3 = input("Enter 3rd word: ")
```

```
w4 = input("Enter 4th word: ")
```

```
w5 = input("Enter 5th word: ")
```

```
w6 = input("Enter 6th word: ")
```

```
w7 = input("Enter 7th word: ")
```

```
w8 = input("Enter 8th word: ")
```

```
w9 = input("Enter 9th word: ")
```

```
w10 = input("Enter 10th word: ")
```

```
wordList = [w1, w2, w3, w4, w5, w6, w7, w8, w9, w10]
```

```
print(numberList)
```

```
print(wordList)
```

```
numberList.sort()
```

```
print("Sorted number list: ", numberList)
```

```
wordList.sort()
```

```
print("Sorted word list: ", wordList)
```

```
import random
```

```
numberList = random.sample(range(1,101), 10)
```

```
print("Here's a list of 10 random numbers:")
```

```
print(numberList)
```

```
PS C:\Users\Jesse\Documents\object oriented programming> & C:/Users/Jesse/AppData/Local/Programs/Python/Python38-32/python.exe "c:/Users/Jesse/Documents/object oriented programming/list.py"
```

```
Input 10 numbers:
```

```
Enter 1st value: 1
```

```
Enter 2nd value: 2
```

```
Enter 3rd value: 3
```

```
Enter 4th value: 4
```

```
Enter 5th value: 5
```

```
Enter 6th value: 26
```

```
Enter 7th value: 790
```

```
Enter 8th value: 112
```

```
Enter 9th value: 65
```

```
Enter 10th value: 34
```

```
Input 10 words:
```

```
Enter 1st word: hey
```

```
Enter 2nd word: you
```

```
Enter 3rd word: I
```

```
Enter 4th word: don't
```

```
Enter 5th word: want
```

```
Enter 6th word: to
```

```
Enter 7th word: be
```

```
Enter 8th word: with
```

```
Enter 9th word: you
```

```
Enter 10th word: today
```

```
[1, 2, 3, 4, 5, 26, 790, 112, 65, 34]
```

```
['hey', 'you', 'I', "don't", 'want', 'to', 'be', 'with', 'you', 'today']
```

```
Sorted number list: [1, 2, 3, 4, 5, 26, 34, 65, 112, 790]
```

```
Sorted word list: ['I', 'be', "don't", 'hey', 'to', 'today', 'want', 'with', 'you', 'you']
```

```
Here's a list of 10 random numbers:
```

```
[31, 45, 67, 9, 68, 43, 65, 98, 39, 37]
```

```
PS C:\Users\Jesse\Documents\object oriented programming>
```

4. integer program
5. added functionality
6. further added functionality

```
numList = []

def integreader():
    while True:
        num = int(input("Input an integer (0 terminates):"))
        if num == 0:
            break
        else:
            numList.append(num)

    print("Terminated")
    print("Negative integers in the list: ", (sum(1 for num in numList if num < 0)))

def evenreader():
    even_sum = 0
    even_sum = sum(1 for num in numList if num % 2 == 0)
    print("Even integers in the list: ", (even_sum))

def divreader():
    div_sum = 0
    div_sum = sum(num for num in numList if num > 0 and num % 3 == 0)

    print("Sum of positive integers divisible by 3: ", (div_sum))

integreader()
evenreader()
divreader()
```

You, 3 hours ago • changed integers.py into functions

```
PS C:\Users\Jesse\Documents\object oriented programming> & C:/Users/Jesse/AppData/Local/Programs/Python/Python38-32/python.exe "c:/Users/Jesse/Documents/object oriented programming/integers.py"
Input an integer (0 terminates):2
Input an integer (0 terminates):3
Input an integer (0 terminates):4
Input an integer (0 terminates):6
Input an integer (0 terminates):9
Input an integer (0 terminates):-3
Input an integer (0 terminates):-6
Input an integer (0 terminates):-2
Input an integer (0 terminates):-18
Input an integer (0 terminates):0
Terminated
Negative integers in the list: 4
Even integers in the list: 6
Sum of positive integers divisible by 3: 18
PS C:\Users\Jesse\Documents\object oriented programming>
```

7. -

8. RPS Game

```
# File name: RPSGame.py
# Author: Jesse Malinen
# Description: Simple Rock-Paper-Scissors game

# import random to get the AI choice
import random

# defining the main function
def RPSGame():

    # score tallies
    ai_win_count = 0
    player_win_count = 0
    ties = 0

    # main loop
    while True:
        choices = ['rock', 'paper', 'scissors']
        ai_choice = random.choice(choices) # ai chooses one randomly from list

        user_choice = input("(rock, paper, scissors) Type your choice: ").strip().lower()
        print()
        ai_wins = 'The AI wins!'
        player_wins = 'You win!'

        # determining winner, adding to tallies with +=1
        print(f'You played {user_choice}, the computer played {ai_choice}')
        if user_choice == 'scissors' and ai_choice == 'rock' or \
            user_choice == 'paper' and ai_choice == 'scissors' or \
            user_choice == 'rock' and ai_choice == 'paper':
            print(ai_wins)
            ai_win_count += 1

        elif user_choice == 'rock' and ai_choice == 'scissors' or \
            user_choice == 'scissors' and ai_choice == 'paper' or \
            user_choice == 'paper' and ai_choice == 'rock':
            print(player_wins)
            player_win_count += 1

        else:
            if user_choice == ai_choice:
                print("Its a draw!")
                ties += 1

        print(f'Computer: {ai_win_count} - You: {player_win_count} - Ties: {ties}')

    RPSGame()
```

```
(rock, paper, scissors) Type your choice: rock ]  
  
You played rock, the computer played paper  
The AI wins!  
Computer: 1 - You: 0 - Ties: 0  
(rock, paper, scissors) Type your choice: paper  
  
You played paper, the computer played scissors  
The AI wins!  
Computer: 2 - You: 0 - Ties: 0  
(rock, paper, scissors) Type your choice: scissors  
  
You played scissors, the computer played rock  
The AI wins!  
Computer: 3 - You: 0 - Ties: 0  
(rock, paper, scissors) Type your choice: rock  
  
You played rock, the computer played rock  
Its a draw!  
Computer: 3 - You: 0 - Ties: 1  
(rock, paper, scissors) Type your choice: rock  
  
You played rock, the computer played paper  
The AI wins!  
Computer: 4 - You: 0 - Ties: 1  
(rock, paper, scissors) Type your choice: rock  
  
You played rock, the computer played paper  
The AI wins!  
Computer: 5 - You: 0 - Ties: 1  
(rock, paper, scissors) Type your choice: paper  
  
You played paper, the computer played rock  
You win!  
Computer: 5 - You: 1 - Ties: 1  
(rock, paper, scissors) Type your choice: 
```

9. –

10. Explain terms

- a. Procedural programming:  
Procedural programming is as it's name suggests – procedural. Focused on creating modular chains of subroutines that can call on each other, heavily supported by CPU architecture. Call – return functionality.
- b. Functional Programming:  
Replaces procedures with functions. Similarly allows the reuse of code in different parts of the program, also supports call-return functionality. Where procedural programming focuses on chains of tasks, functional programming focuses on complex expressions that only depends on each other through arguments and return values. More modular. Doesn't rely on looping.
- c. Object Oriented Programming:  
Focuses on objects instead of chains of functions. Objects may contain both data (in the form of fields, attributes and/or properties) and code (procedures named methods). Objects can modify the data of themselves, and programs consist of objects interacting with each other. Most modern, widely used languages are object oriented (Python, JS, C++/C#)
- d. Class (in programming):  
Classes are definitions for data formats and available procedures for the type of object it is. Classes can also contain data and modify said data in itself.
- e. Object (in programming):  
Objects are instances of classes, usually corresponding to stuff like shopping carts and customer profiles within web pages.
- f. Instance (in programming)  
Instance variables contain data that belong to specific objects. Every object can have the same data, but in it's own copy of an instance. Instance methods on the otherhand belong entirely to individual objects and can access aforementioned instance variables for the object they are tied to.
- g. Encapsulation (in programming):  
Encapsulation aims to keep the data & functions of an object hidden and safe from misuse. Objects that don't allow other code to access their data are encapsulated. This is a basic form of "encryption" so to speak, where certain data can only be accessed with certain rights or keys.

## Commits on Jan 17, 2021

started arithmetic script blankdev committed 18 minutes ago	f0fb9c5	<>
changed functional order blankdev committed 29 minutes ago	f9600a5	<>
finished simple rps game blankdev committed 44 minutes ago	87c4466	<>
changed integers.py into functions blankdev committed 3 hours ago	de661ad	<>
added div_sum blankdev committed 4 hours ago	8c25984	<>
added functionality to count even numbers blankdev committed 4 hours ago	33a4447	<>
created integers.py for task 4 blankdev committed 5 hours ago	c854845	<>

## Commits on Jan 16, 2021

Added list sorting blankdev committed yesterday	561150d	<>
Created list.py with 3 required list prints blankdev committed yesterday	224f02a	<>
Created Hello.py blankdev committed yesterday	efe5b54	<>

**Self-assessment:**

These tasks were simple but seemed harder than they were due to lack of practice with exercises like these. So much of my time with Python since the beginner course has been spent with building web apps with flask that creating some of the most basic functions seemed lost to me. I did not complete task 7 as my mathematical skills were lacking to the point where I could not form the formulas to find out the first term in the series based on the last and the divider. That is a fault I am fully aware of and will fix down the road. I also skipped task 9 purely due to time constraints as I did not schedule properly while being fully aware that most of my week and the entire weekend would be spent in meetings and training sessions for the student councils etc.

Overall great exercises that gave a moderate challenge.

Link to Git Repo: <https://github.com/blank-py/Object-Oriented-Programming>