

python



# Python Beginners workshop

Day 4

# Colorama module

This module needs to be installed via pip: `pip install colorama`

```
from colorama import Fore, Style, Back, init
init(autoreset=True)
```

```
print(f'{Fore.RED}Hello') # prints red text
```

```
print(f'{Back.GREEN}Hello') # prints text with green background
```

```
print(f'{Style.BRIGHT}Hello') # prints brighter text
```

You can look at the [documentation](#) to see what colours are available for use

The 6 subject marks of 30 students from a class need to be input into a program, and the average of each student needs to be calculated, along with their overall grade. Write a program to output the name of the student, their marks, their average and their grade. Store all these values in a data structure of your choice. Use the grade boundaries below.

Average of student	Grade assigned
90 and above	A
80 to 90	B
70 to 80	C
60 to 70	D
50 to 60	E
49 and below	F

Some shortcuts...

# Assignment shortcut

To declare multiple variables at a time, you can put them all in one line:

```
a = 0
```

```
b = 1
```

```
c = 2
```

```
d = 3
```



```
a, b, c, d = 0, 1, 2, 3
```

# Sending multiple returns from a function

When returning multiple values from a function, you can get those values into multiple variables using the same shortcut assignment method:

```
def something():  
    # does something  
    x, y, z = 0, "hello", True  
    return x, y, z # returning 3 different values  
  
a, b, c = something() # getting those values into 3 variables
```

# Giving default values for function arguments

I can provide an argument to a function, and I can give that argument a default value:

```
def something(argument1=10):  
    # do something  
    return
```

```
something(3) # does something with 3
```

```
something() # no argument given,
```

```
# so it takes default value which is 10
```

# Quick functions

The `sum()` function can find the sum of numbers in a list

The `min()` function can find the smallest number in a list

The `max()` function can find the biggest number in a list

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(sum(arr)) # prints 55
print(min(arr)) # prints 1
print(max(arr)) # prints 10
```

# Conditions

No need to check if `x == True` or `x == False`. Just do this:

```
if x == True:  
    # do something  
    print("something")
```



```
if x:  
    # do something  
    print("something")
```

```
if x == False:  
    # do something  
    print("something")
```



```
if not x:  
    # do something  
    print("something")
```

Any doubts?

# The Challenge Problem

# Hand Cricket!!

- User vs. Comp
- 1 wicket match
- No ties, only win or lose(if you don't reach the target, you lose)
- "Odd or Eve" toss - take 2 numbers, add them and see if it is odd or even  
Ex: I call odd, toss is  $3+6=9$ , 9 is odd so I win the toss
- Accept "o", "odd", "eve", "even" or "e" for toss; ignore capitalization
- Accept only "bat" or "bowl" for opting to bat/bowl; ignore capitalization
- Have to opt to bat or bowl(depending on who wins toss)
- Runs are only from 1 to 6; no dot balls, wide, no balls etc.
- After 1st innings, print out the target
- At the end of the match, print out the winner

# Code specs

- Must have at least 1 function of your own
- Must use at least 1 module
- Must use at least 1 loop(while or for) and 1 conditional
- Must use at least 3 f-strings
- Must tell what is going on(tell when toss is happening, proper input prompts, tell when each innings is etc.)
- Coloured output:
  - If the player wins the toss, print it in **green**. If the computer won the toss, print it in **red**.
  - The target should be printed out in a color of your choice(except **red** and **green**)
  - If the player wins, print in **green**. If the computer wins, print in **red**
  - If there is a wicket, print **“THATS OUT!!!”** in **red** text

# Hints

- You might want to use a while loop to check if a user is out; otherwise continue the innings
- You can use f-strings to output the target, match result etc.
- Conditions will be useful in checking who won the toss, whether the bowler and batsman put the same number etc.
- Do some error checking: make sure the user only inputs a number from 1 to 6 for the runs
- The random module can help you make the computer chose a random number
- The colorama module will help you with the coloured text
- Use the break keyword to exit a while loop randomly(not a good idea to use this too often, but once or twice is ok)
- You can go online if you need any help with anything, or ask us

# Bonus challenge

- You can choose the number of wickets(if user wins toss, computer chooses wickets. If comp wins toss, user chooses wickets). But only from 1 to 10 wickets allowed
- Store every batsman's score somewhere
- Output the scorecard like this: "Batsman 1: 30" and so on
- If someone scored a 50(or more) show them in **green** in the scorecard
- Show the win margin(how many runs or wickets the winner won by)
- Highlight changes in **blue**(when showing that 1st innings has started, showing that toss is happening, showing a caption for the scorecard/result etc.)

# Code specs

- Same as before, but must also use at least one list/dictionary/class
- Coloured output:  
Same as before, but also show the difference between batsman and bowler by highlighting them in different colours(batsman is always same colour, bowler is always in same colour, but different from each other). You cannot use red or green for this

# Hints

- No need for dictionaries or fancy classes; you can just store the batsmen's individual scores in a simple list
- When printing out the scorecard, use a for loop to go over your list
- Remember that indexes start from 0, but the first batsman will be called "Batsman 1" or "Batter 1" so you can add one to the index while printing
- Change the while loop so it checks if all the wickets have fallen or not
- To calculate the win margin, you will need to know who won, whether they won while chasing or defending, and how many wickets were left(if the winner was chasing) or the difference in the score(if the winner was defending)

# Grading criteria

1. **Validity of submission** - code must meet all the specifications required (extra things it does have no effect on the grade)
2. **Code efficiency** - minimize redundant/unnecessary code, good code structure
3. **Readability** - your code should be easy to read, with gaps between different parts of your code, and comments to explain what some code is doing
4. **User-friendliness** - does the user know what is going on?
5. **Adherence to Code of Cyber Ethics** - are you plagiarising? Or purposely making an infinite loop? Or using malicious code? If so, you might be **penalised**

Total is out of 40 points, with 10 points for each criteria.

Top 10 submissions will get **Certificates of Excellence**

Everyone else will get Certificates of Participation(provided that they attended at least 2 out of the 3 workshop sessions)

Unfortunately, IB Yr 1 and CBSE 12 students will **not** be eligible for Certificates of Excellence, so that other students in lower grades have a better chance of getting these certificates as a token of appreciation from us for their hard work.

Any doubts?