2. Conditional Statements

April 19, 2023

1 Introduction

- Decision making is required when we want to execute a code only if a certain condition is satisfied.

2 `if`

`if` there is exam tomorrow, I will stay at home to study

[1]:
```
exam = True
if exam == True:
    print("I will stay at home to study")
    print("Oh!!!")
print("Not in if block")
```

```
I will stay at home to study
Oh!!!
Not in if block
```

[2]:
```
if exam == True:
    print("I will stay at home to study")
    print("Oh!!!")
print()
```

```
I will stay at home to study
Oh!!!
```

2.1 Syntax for if

```
if boolean_expression :
    statements
    # this block of code will be executed if boolean_expression returns True
```

3 if...else

`if` there is exam tomorrow, I will stay at home to study; `else` I will play football on the playground

[3]:
```
exam = True
if exam == True:
    print("I will stay at home to study")
    print("Oh :(")
else:
    print("I will play football on the playground")
```

```
I will stay at home to study
Oh :(
```

3.1 Syntax for if else

> `if` boolean_expression `:`
>
> > statements
> > # this block of code will be executed if boolean_expression returns `True`
>
> `else:`
>
> > statements
> > # this block of code will be executed if boolean_expression returns `False`

3.2 Example

- Write a program to find whether a number is even or odd.

[4]:
```
num = int(input("Enter a number: "))

if num % 2 == 0:
    print(num, "is even number")
    print("yay!!")
else:
    print(num, "is odd number")
    print("Oh!!")
```

```
Enter a number:  9
```

```
9 is odd number
Oh!!
```

4 Nested `if...else`

> `if` boolean_expression 1 `:`
>
> > statements
> > # this block of code will be executed if boolean_expression 1 returns `True`
>
> `elif` boolean_expression 2 `:`
>
> > statements
> > # this block of code will be executed if boolean_expression 1 returns `False` and boolean_expression 2 returns `True`
>
> `elif` boolean_expression 3 `:`
>
> > statements
> > # this block of code will be executed if boolean_expression 1 & 2 returns `False` and boolean_expression 3 returns `True`
>
> .
> .
> .
>
> `else:`
>
> > statements
> > # this block of code will be executed if all boolean_expression returns `False`

4.1 Example

Write a program to decide whether to watch a movie based on its rating (Rating is given out of 5) - Movie is a must watch if rating is 4 and above - Movie can be watched once if rating is 2.5 and above but less than 4 - Don't watch the movie if rating is less than 2.5

[5]:
```
rating = float(input("Enter movie rating(Out of 5): "))

if rating >= 4:
    print("it is a good move")
    print("Must watch")
elif rating >= 2.5 and rating < 4:
    print("Okay movie")
    print("you can watch it once")
elif rating < 2.5:
    print("Don't watch :(")
```

```
Enter movie rating(Out of 5):  3.5

Okay movie
you can watch it once
```

5 Ternary operator

- It simply allows to test a condition in a single line replacing the multiline if-else making the code compact

statement1 `if` expression `else` statement2

[6]:
```
age = 15
print('kid' if age < 18 else 'adult')
```

```
kid
```