

# پایتون جامع

محمد حسین مقدم خواه

[Mhmk.nova@proton.me](mailto:Mhmk.nova@proton.me)

جلسه سوم

# Topics

- Conditionals
- While Loop
- List Datatype
- For Loop



# Control Flow

- If - Elif - Else Structure
- Shorthand If
- Shorthand If-Else
- Match Case (Python +3.10)

## References:

- <https://www.geeksforgeeks.org/python-if-else/>
- <https://www.geeksforgeeks.org/python-match-case-statement/>
- [https://www.w3schools.com/python/python\\_conditions.asp](https://www.w3schools.com/python/python_conditions.asp)

## Exercises

- <https://csiplearninghub.com/python-if-else-conditional-statement-practice/>





```
1 num_1 = 10
2 num_2 = 20
3
4 if num_1 > num_2:
5     print("num_1 is greater than num_2")
6 elif num_1 < num_2:
7     print("num_2 is greater than num_1")
8 else:
9     print("num_1 and num_2 are equal")
10
11 """
12 OUT => num_2 is greater than num_1
13 """
```

# If – Elif – Else Structure



```
1 num_1 = 10
2 num_2 = 20
3
4 if num_1 > num_2: print("num_1 is greater than num_2")
5 """
6 OUT => num_2 is greater than num_1
7 """
```

# Shorthand If Statement

اگر عمل متقابل شرط ما یک خطی باشد میتوانیم شرط خود را در یک خط بنویسیم.



```
1 num_1 = 10
2 num_2 = 20
3
4 print("num_1 is greater than num_2") if num_1 > num_2 else print("num_2 is greater than num_1")
5 """
6 OUT => num_2 is greater than num_1
7 """
```

# Shorthand If-Else

همچنین در پایتون برای حالت شرط If – Else هم حالت خلاصه داریم.

# Match Case (Python +3.10)

در پایتون 3.10 به بالا مفهوم Match – Case نیز اضافه شده که مانند ساختار Switch – Case در سایر زبان ها است.

```
1  usr_input = int(input("Enter a number: "))
2
3  match usr_input:
4      case 1:
5          print("One")
6      case 2:
7          print("Two")
8      case 3:
9          print("Three")
10     case _:
11         print("Unknown")
12
13 """
14 IN => 2
15 OUT => Two
16 -----
17 IN => 5
18 OUT => Unknown
19 """
```



# While Loop Statement

References:

- [GeekForGeeks](#)
- [W3School](#)

[Exercises](#)

```
mirror_mod = modifier_ob.  
#set mirror object to mirror  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
  
print("please select exactly  
  
-- OPERATOR CLASSES ----  
  
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```



# While Loop:

نوعی حلقه کنترل شده است که تا زمانی که شرط آن True باشد، حلقه ادامه میابد. همچنین برای انجام عملیات در پایان حلقه، از else استفاده میکنیم.



```
1 num = 10
2
3 while num > 0:
4     print(num)
5     num -= 1
6 else:
7     print("Done")
8
9
10 """
11 OUT => 10, 9, 8 , 7, 6, 5, 4, 3
12 """
```

# Break & Continue

از `break` برای شکستن حلقه و  
`continue` برای پرش کیس فعلی و رفتن  
روی کیس بعدی استفاده میشود.

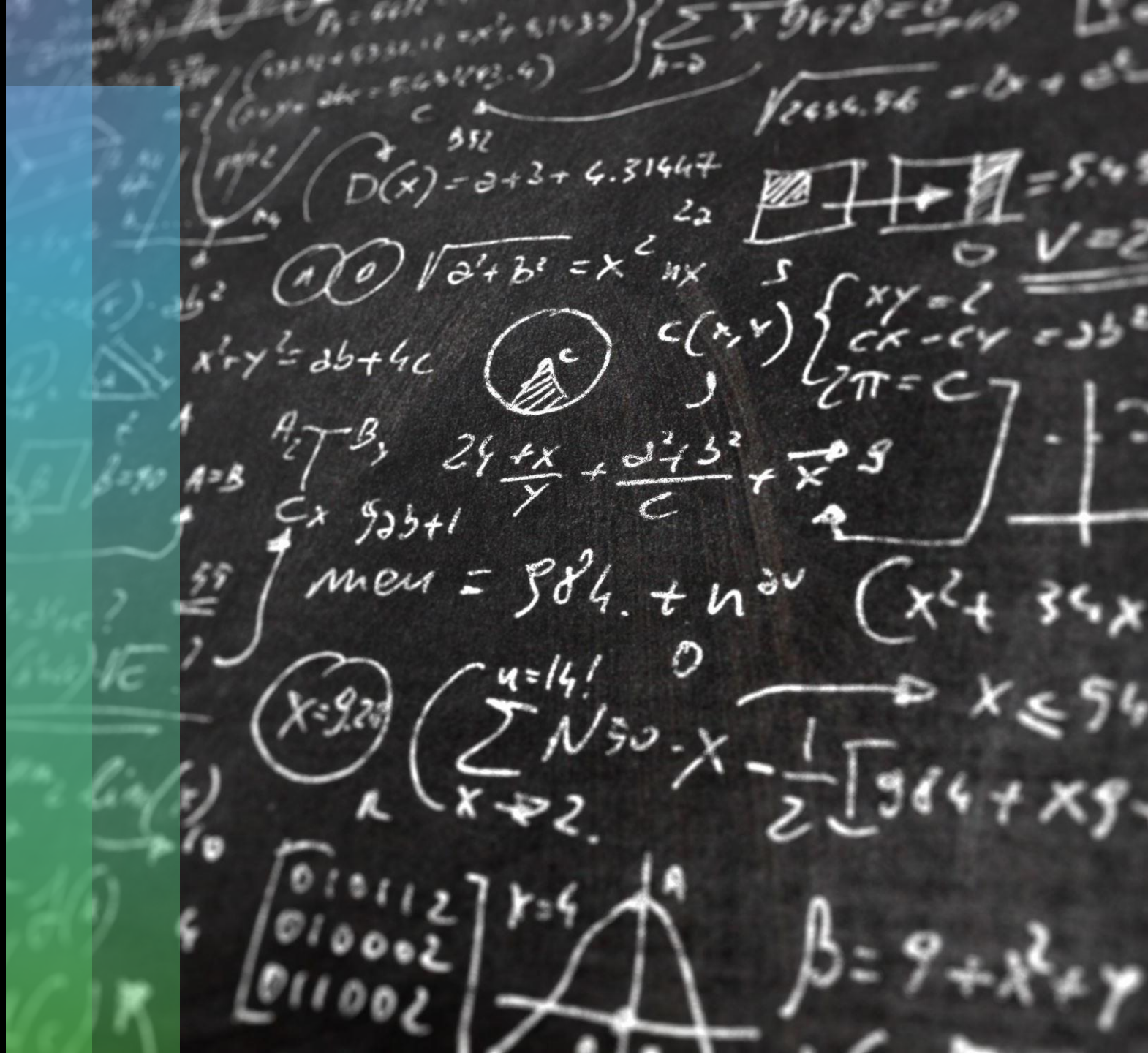


```
1 num = 10
2
3 while num > 0:
4     print(num)
5     num -= 1
6     if num == 5:
7         continue
8     elif num == 3:
9         break
10
11 """
12 OUT => 10, 9, 8, 7, 6, 4
13 """
```

# List

## References:

- [GeekForGeeks](#)
- [W3School](#)





```
1 list_1 = [1, 3.14, "hello", True, 3 + 4j, [1, 2, 3]]
2
3 print(list_1)
4
5 """
6 OUT => [1, 3.14, "hello", True, (3 + 4j), [1, 2, 3]]
7 """
```

# List DataType

دیتا تایپ دیگری که در پایتون داریم، **list** است که آرایه ای از مقادیر مختلف است که:

- ترتیب دارند
- میتواند دیتا تایپ های مختلف را در خود جای دهد
- قابل تغییر است.

# Range() Function

از این تابع برای ساخت لیست از یک مقدار تا مقدار دیگر استفاده میکنیم که ورودی های آن عبارتند از:

1. مقدار شروع

2. مقدار پایان

3. قدم پیشروی

```
1 list_1 = list(range(1, 10, 2))
2
3 print(list_1)
4
5 """
6 OUT => [1, 3, 5, 7, 9]
7 """
```

# Iterables DataTypes

دیتا تایپ هایی که میتوانیم روی آنها حلقه بزنیم که تا اینجا خوانده ایم عبارتند از:

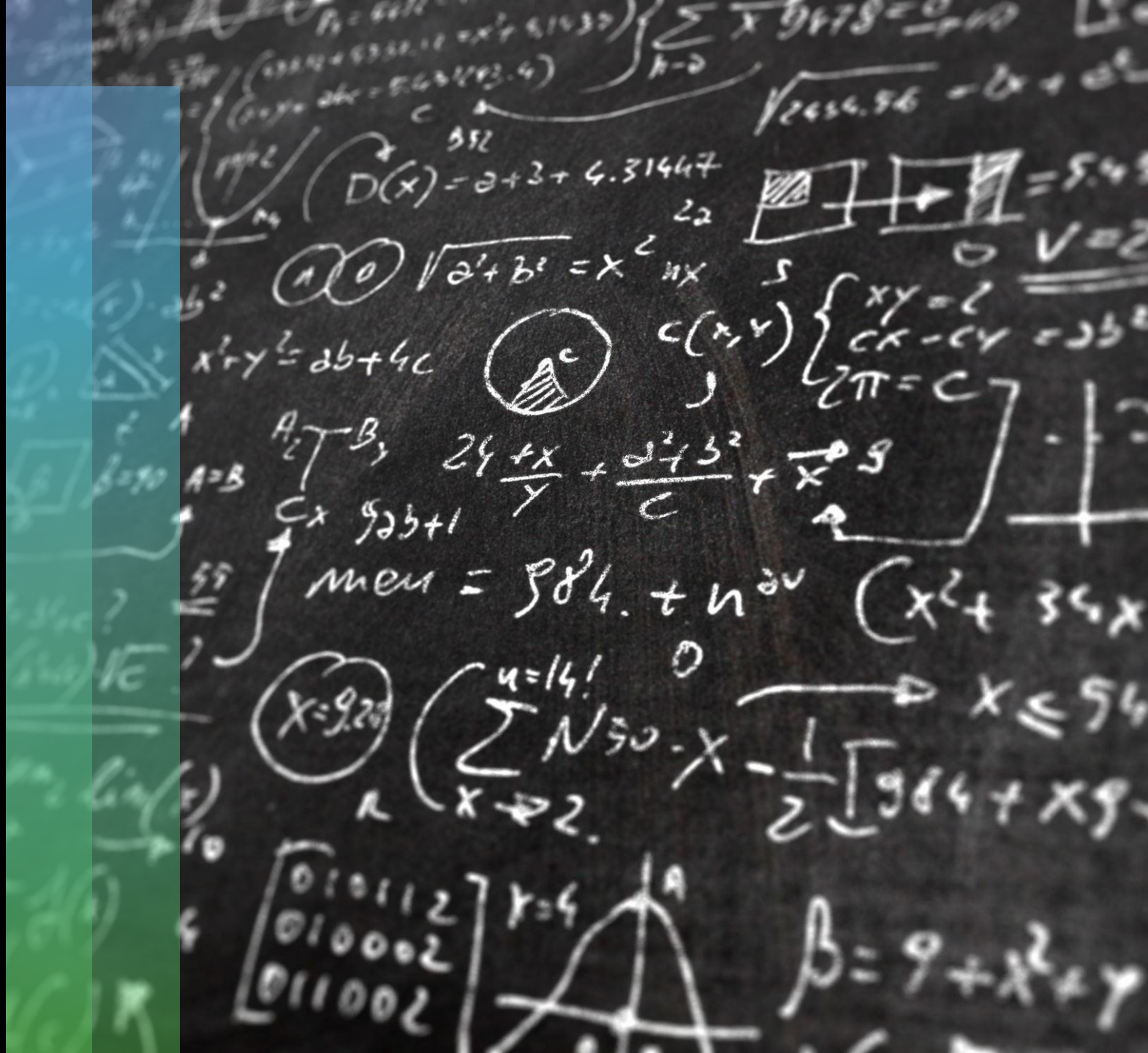
- String: در واقع این نوع داده، لیستی از کارکتر هاست
- List



# For Loop Statement

References:

- [GeekForGeeks](#)
- [W3School](#)  
[Exercises](#)





# For Loop

از عملگر `for` برای ایجاد حلقه استفاده میکنیم.

همچنین در انتهای حلقه میتوانیم از `else` استفاده کنیم.

مانند بقیه ساختارهای حلقه از `break` و `continue` هم میتوانیم استفاده کنیم.



```
1 for i in "Hello":
2     print(i)
3 else:
4     print("Done1")
5
6 for i in [1, 2, 3, 4, 5]:
7     print(i)
8 else:
9     print("Done2")
10
11 for i in range(1, 10):
12     print(i)
13 else:
14     print("Done3")
15
16 """
17 OUT => H, e, l, l, o, Done1
18 1, 2, 3, 4, 5, Done2
19 1, 2, 3, 4, 5, 6, 7, 8, 9, Done3
20 """
```

# Session 3 Ended.

To Be Continued...