

**NPTEL MOOC**

# **PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON**

**Week 5, Lecture 2**

**Madhavan Mukund, Chennai Mathematical Institute**

**<http://www.cmi.ac.in/~madhavan>**



# Interacting with the user

- \* Program needs to interact with the user
  - \* Receive input
  - \* Display output
- \* Standard input and output
  - \* Input from keyboard
  - \* Output to screen





# Reading from the keyboard

- \* Read a line of input and assign to `userdata`

```
userdata = input()
```

- \* Display a message prompting the user

```
userdata = input("Enter a number")
```

- \* Add space, newline to make message readable

```
userdata = input("Enter a number: ")
```

```
userdata = input("Enter a number:\n")
```



# Reading from the keyboard

- \* Input is always a string, convert as required

```
userdata = input("Enter a number")  
usernum = int(userdata)
```



# Reading from the keyboard

- \* Use exception handling to deal with errors

```
while(True):  
    try:  
        userdata = input("Enter a number: ")  
        usernum = int(userdata)  
    except ValueError:  
        print("Not a number. Try again")  
    else:  
        break
```



# Printing to screen

- \* Print values of names, separated by spaces

```
print(x,y)  
print(a,b,c)
```

- \* Print a message

```
print("Not a number. Try again")
```

- \* Intersperse message with values of names

```
print("Values are x:", x, "y:", y)
```



# Fine tuning `print()`

- \* By default, `print()` appends new line character `'\n'` to whatever is printed
- \* Each `print()` appears on a new line
- \* Specify what to append with argument `end="..."`

```
print("Continue on the", end=" ")  
print("same line", end=". \n")  
print("Next line.")
```

Continue on the same line.  
Next line.



# Fine tuning `print()`

- \* By default, `print()` appends new line character '`\n`' to whatever is printed
- \* Each `print()` appears on a new line
- \* Specify what to append with argument `end="..."`

```
print("Continue on the", end=" ")  
print("same line", end=".\n")  
print("Next line.")
```

Add space,  
no new line

Continue on the same line.  
Next line.



# Fine tuning `print()`

- \* By default, `print()` appends new line character `'\n'` to whatever is printed
- \* Each `print()` appears on a new line
- \* Specify what to append with argument `end="..."`

```
print("Continue on the", end=" ")
print("same line", end=". \n")
print("Next line.")
```

Add space,  
no new line

Add full stop,  
new line

Continue on the same line.  
Next line.



# Fine tuning `print()`

- \* Items are separated by space by default

```
(x,y) = (7,10)
print("x is",x,"and y is",y, ".")
```

x is 7 and y is 10 .

- \* Specify separator with argument `sep="..."`

```
print("x is ",x," and y is ",y, ".", sep="")
```

x is 7 and y is 10.



# Formatting print

- \* May need more control over printing
  - \* Specify width to align text
  - \* Align text within width — left, right, centre
  - \* How many digits before/after decimal point?
- \* See how to do this later



# Summary

- \* Read from keyboard using `input()`
  - \* Can also display a message
- \* Print to screen using `print()`
  - \* Caveat: In Python 2, `()` is optional for `print`
- \* Can control format of `print()` output
  - \* Optional arguments `end="..."`, `sep="..."`
  - \* More precise control later