KHWARIZMI
SCIENCE SOCIETY

CODING WITH PYTHON

A COURSE FOR ABSOLUTE BEGINNERS

iDRACK

KHWARIZMI
SCIENCE SOCIETY

iDRACK

https://idrack.org/

https://idrack.org/forum/community/

contact@idrack.org

COURSE OUTLINE

KHWARIZMI SCIENCE SOCIETY

5 WEEKS, SATURDAYS, 2-5 PM

Week 1:

- An introduction to Python
- Installing Python on Windows
- Python Shell

Week 2:

- Saving & Running scripts
- Operators & Variables
- Working with Strings

Week 3:

- Python Collections
- Condition Blocks

Week 4:

- Writing Loops
- Functions in Python

Week 5:

- Introducing Modules
- In-class Assessment

iDRACK

INSTRUCTOR CONTACT

ROOP OMAR

(roop.omar@gmail.com)

We will try to cater to all queries within class timings, but if you feel there is something you need help with later, or were not able to ask during the session, please feel free to drop me an email, and I will get back to you as soon as I can.

KHWARIZMI
SCIENCE SOCIETY

MODULES IN PYTHON

iDRACK

MODULE

KHWARIZMI SCIENCE SOCIETY

- A module is a file containing Python definitions and statements.
- It can define functions, classes, and variables.
- A module can also include runnable code.
- Grouping related code into a module makes the code easier to understand and use.
- It also makes the code logically organized.

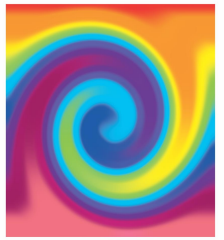
```
# A simple module, calc.py

def add(x, y):
    return (x+y)

def subtract(x, y):
    return (x-y)
```

iDRACK

IMPORTING



- We can use any Python source file as a module by executing an import statement in some other Python source file.
- When the interpreter encounters an import statement, it imports the module if the module is present in the search path.
- For example, to import the module calc.py, we need to put the following command at the top of the script :

```
# importing  module calc.py
import calc

print(add(10, 2))
```

iDRACK

FROM IMPORT

KHWARIZMI SCIENCE SOCIETY

- Python's *from* statement lets you import specific attributes from a module.
- The *from* .. *import* .. has the following syntax

```
# importing sqrt() and factorial from the
# module math
from math import sqrt, factorial

# if we simply do "import math", then
# math.sqrt(16) and math.factorial()
# are required.
print(sqrt(16))
print(factorial(6))
```

iDRACK


```
# importing built-in module math
import math

# using square root(sqrt) function contained
# in math module
print(math.sqrt(25))

# using pi function contained in math module
print(math.pi)

# 2 radians = 114.59 degreees
print(math.degrees(2))

# 60 degrees = 1.04 radians
print(math.radians(60))

# Sine of 2 radians
print(math.sin(2))

# Cosine of 0.5 radians
print(math.cos(0.5))

# Tangent of 0.23 radians
print(math.tan(0.23))

# 1 * 2 * 3 * 4 = 24
print(math.factorial(4))
```

iDRACK


```python
# importing built in module random
import random

# printing random integer between 0 and 5
print(random.randint(0, 5))

# print random floating point number between 0 and 1
print(random.random())

# random number between 0 and 100
print(random.random() * 100)

List = [1, 4, True, 800, "python", 27, "hello"]

# using choice function in random module for choosing
# a random element from a set such as a list
print(random.choice(List))


# importing built in module datetime
import datetime
from datetime import date
import time

# Returns the number of seconds since the
# Unix Epoch, January 1st 1970
print(time.time())

# Converts a number of seconds to a date object
print(date.fromtimestamp(454554))
```

iDRACK

KHWARIZMI
SCIENCE SOCIETY

MODULAR PROGRAMMING

iDRACK

- Modular programming refers to the process of breaking a large, unwieldy programming task into separate, smaller, more manageable subtasks or modules.
- Individual modules can then be cobbled together like building blocks to create a larger application.

- There are several advantages to modularizing code in a large application:

- **Simplicity:** Rather than focusing on the entire problem at hand, a module typically focuses on one relatively small portion of the problem.
- **Maintainability:** Modules are written in a way that minimizes interdependency. This makes it more viable for a team of many programmers to work collaboratively on a large application.
- **Reusability:** Functionality defined in a single module can be easily reused. This eliminates the need to duplicate code.

iDRACK

KHWARIZMI
SCIENCE SOCIETY

PROGRAMMING CHALLENGE

iDRACK

- Write a program to ask the user to enter their school username in the following format:
  - ✓ Year Group Using two digits (e.g. "07" for year 7, "11" for year 11, "00" for staff members)
  - ✓ 1 character for their initial (first letter of their firstname)
  - ✓ The user's lastname
  - ✓ A 2-digit code: "_S" for students, "_T" for teachers, "_A" for admin staff.
- For instance the following usernames are valid usernames:
  - ✓ 07jFox_S
  - ✓ 09kJohnson_S
  - ✓ 11rTaylor_S
  - ✓ 00pJones_T
  - ✓ 00jOliver_A

KHWARIZMI SCIENCE SOCIETY

iDRACK

- Your program should read the username and decide:

  - ✓ If the username is less than 6 characters long the program should ask the user to enter a valid username.
  - ✓ If the username does not contain the character "_" it should also ask the user to enter a valid username.
  - ✓ If the username is valid, the program should decide if the user is a member of staff or a student.
  - ✓ If they are a student the programme should find out their year group.
  - ✓ The program should also display the initial of the student as well as their lastame.
  - ✓ Finally the program should display whether the user is a Student, a Teacher or an Admin member of staff.

KHWARIZMI SCIENCE SOCIETY

iDRACK

KHWARIZMI
SCIENCE SOCIETY

PROGRAMMING CHALLENGE

iDRACK

- Store the name, location, gender, name of the enemy, super-powers of your superhero.
- Don't forget the use of "speech marks".

  - ✓ name = "..."
  - ✓ location = "..."
  - ✓ gender = "..."
  - ✓ enemy = "..."
  - ✓ powers = "..., ..., ..."

- Now save the speed, age and strength of your superhero using numerical values.
- You will not need to use speech marks.

  - ✓ speed =
  - ✓ strength =
  - ✓ age =

KHWARIZMI SCIENCE SOCIETY

iDRACK

- Now use the print command to display a description of our superhero on screen.
- You will combine statements such as "My superhero is called " and variables (such as name) to create a clear description of our superhero.
- To do this we will use the + operator.

- For instance:

"My name is Batman and I live in Gotham City.

I am a businessman by day, and a vigilante by night.

Some also know me as the Dark Knight.

My greatest enemies are Joker, Two-face and Riddler"