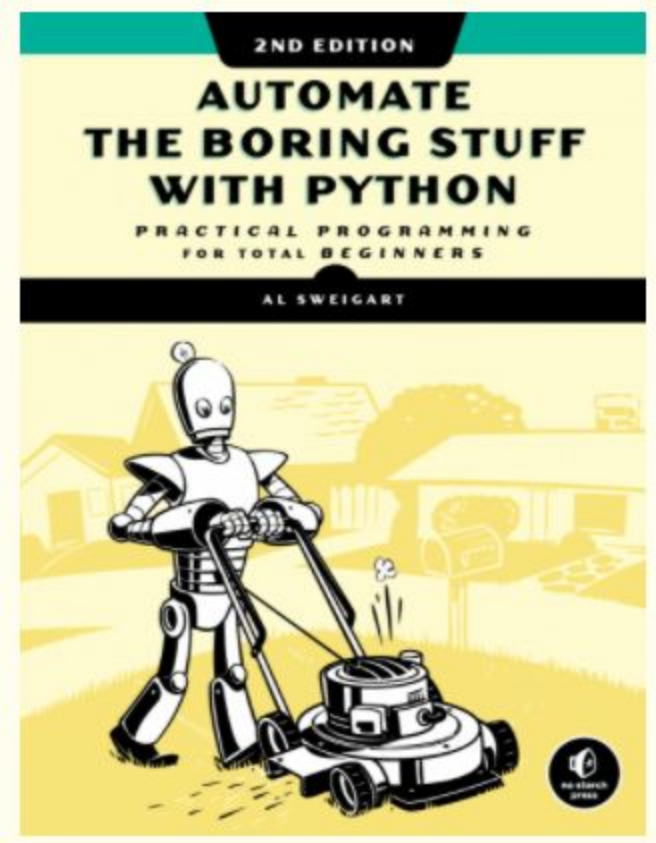
Course Introduction and syllabus



https://automatetheboringstuff.com/

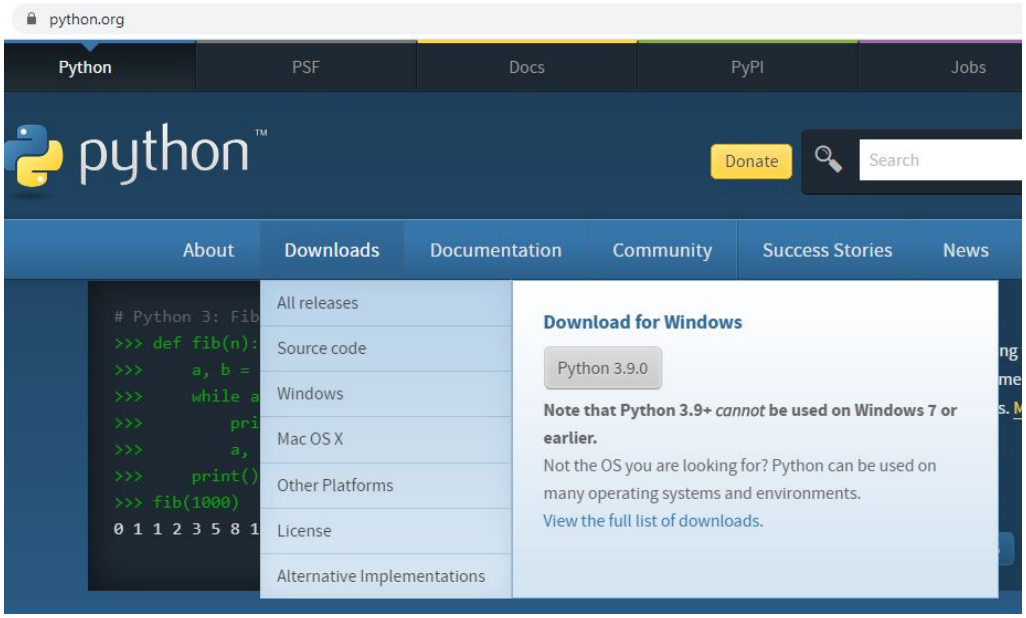
What we'll learn

- Automate tasks on computers by writing simple Python programs.
- Write programs that can do text pattern recognition with "regular expressions".
- Programmatically generate and update Excel spreadsheets.
- Parse PDFs and Word documents.
- Crawl web sites and pull information from online sources.
- Write programs that send out email notifications.
- Use Python's debugging tools to quickly figure out bugs in your code.
- Programmatically control the mouse and keyboard to click and type for you.

Source: https://www.udemy.com/course/automate/learn/lecture/3309062?start=0#overview

Installations:

Python: https://www.python.org/



Python Interpreter or IDE:

- Built in**:** IDLE (used in this course)
- Third party: https://codewith.mu/, https://www.jetbrains.com/pycharm/, many others

Basics

Expressions, Statements (one line codes)

- Start IDLE
- Lets get familiar with
    - Shell, scripts
    - Expressions
    - Data types
    - Variables

Exercise: Open IDLE and print Hello Word on the shell terminal.

Exercise: Find 30 time 345.4 on the terminal

Exercise: Do ¾ + 5 in the terminal

Exericse: Do 1+ 2/3 in the terminal

Exericse: Do (1+2)/3 in the terminal

Exericse: Try to add a string and a number

Excercise: What do you think will happen if I type

```
'Python' + 20
```

Excercise: What do you think will happen if I type

```
'Python' + 'is cool'
```

This is called **string concatenation**

Exercise: Try this expression

```
"India" *3
```

Exercise: Try running

```
"hello"/5
```

Exercise: Guess the output and verify

```
'What' + '?'*10
```

Variables:

Saving values in computer memory so that we don't have to type or calculate again and again.

```
fName = "Ajit"
print(name)
```

```
lName = name + "Kumar"
fullName = fName + " " + lName
```

```
print(fullName)
```

Exercise: Try and evaluate these lines of explain what is happening

```
a = 10
print (a)
a = 20 # variables can be reused
print(a)
```

Exercise: Try this. Explain the output of this line

```
a = 10
a = a + 23
print(a)
```

Recap:

- Where to start programming in Python?

- IDLE
    - Interactive shell
    - File editor

- Data type:
    - 4 is an **int** datatype
    - 4.0 is a **float** datatype
    - “Hello” or “4” is a **string** datatype
- Expressions/statements: Any single line command we run on shell
    - print(“hello world”)
- Variables:
    - a = 3.0, etc
    - b = “coding is boring”

Writing codes in File Editor and Running:

Multiple commands are passed to computer in one shot. The lines will be executed on line at a time.

Exercise: Create a new python file, type the code below and run.

```
print( "Hello file editor")
```

Exercise: Create a new python file, type the code below and run.

```
a = 20
b = 30
X = a + b
print(x)
```

Exercise: Create a new python file, type the code below and run.

```
print("Hello. I can double any number. Type any number of your choice:")
number = input()
print("You printed: ", number)
print("Its double is:", 2*number)
```

Noting something weird? Guess the reason and try to fix it. Feel free to use Google.

Exercise: Create a new python file, type the code below and run.

```
print("I can count how many characters are there in your name. What is your
name?")
name = input()
length = len(name)
print("you have ", length, " characters in your name")
```

Exercise: In the codes, add some human readable comments but which will be ignored by python. Why do we need comments?

New functions learned: (play around with it)

int(), str(),

Exercise: Write a program, which asks the user for their age, and prints out how old they will be after 5 years.

Program Flow Control

What if scenarios

Exercise: Write a program which
- Asks user to input his/her name and sex
- The greets with a hello “Mr. X” or “Mrs. X” depending on the response

Exercise: Write a program which
- Asks user to input an number
- The program checks if the number is even or odd.
- Prints out different message in each case.

Exercise: Write a program which
- Ask user to input an email address.
- Checks if the input is a valid email address or not. (By looking for “@” character)
- Throughs out some warning kind of message if it is not a valid email address.

Exercise: Write a program which asks user to create password which must be
- Of 8 < lenth < 16
- Must have at least one of these special characters: @, !
- Check if the password is valid
- Throughs out some warning kind of message if it is not valid

Recap:
- Boolean Data Type: True, False
- Comparison Operators: ==, !=, <, >, <=, >=
- == is comparison, = is assignment
- Boolean Operators: and, or, not

Repeat an action over and over again

- **for** loop
- **while** loop

Exercise: Type and run this code in a new file. Explain line by line

```
name = ''
while name != 'Ajit':
    print("who is the best coder in the world?")
    name = input()
print("Yes you are right. ", name, " is the best coder in the world")
```

Exercise: Type and run this code in a new file. Explain line by line

```
number = 1
while (number %2) != 0:
    number = input("enter an even number:")
    number = int(number)
print("ok")
```

Exercise: Write a for-loop program for printing which prints out each letter of the given string in a separate line.

```
text = "Hello for loop"
n = len(text)
for i in range(n):
    print(text[i])
```

Exercise: Repeat the above exercise to print every alternate characters.

Exercise: Repeat the above exercise to print and print each character

- And stop if the program hits a **space** character (if any)

break & continue

Exercise: Study these codes and try to guess the use of **break** and **continue** keywords.

```
text = "Hello for loop"
for c in text:
    if c == ' ':
        break
    print(c)
```

```
text = "Hello for loop"
for c in text:
    if c == ' ':
        continue
    print(c)
```

Exercise: Write a progam to find the sum of first n numbers

```
n = 100

total = 0
for i in range(1,n+1):
    total = total + (i)

print("Sum of first ", n, "natural number = ", total)
```

Exercise: Modify the above program to find the square, cube, square roots, etc of first n numbers.

Functions

Functions are pre-written codes which we can use repeatedly by simply calling their name.

Python's built in standard functions

We don't need to import anything to use these.

We have already seen some examples of fucntions

```
int(); str(); print()
```

Exercise: File opening, reading and writing

- Create a text file, say demo.txt.
- Write two three meaningless lines in it.
- In the same folder, create an empty file **fileReading.py**
- Write code in fileReading.py which will read the data from demo.txt and print in IDLE shell.

```
f = open("demo.txt", "r")  # "r" for reading, "w" for overwriting,
                           #   "" for appending
fileContent = f.read()
print(fileContent)
```

Exercise: Add something more to this file

```
f = open("demo.txt", "a")
f.write("hello how are you?\n")
f.close()
```

Exercise Run the above code without `"\n"`

For some functions which are **pre-installed.** We need to **import** them

Exercise: Import **math** module

```
import math
print(math.pi)
```

```
import math
x = ???? # find square root of 2
```

Third-party module:

Need to install and **then import**

PANDAS

Exercise: Try to run this code which reads and excel file in the current folder.

Install pandas

```
pip install pandas
```

Run this code to read an excel file in the current directory.

```
import pandas as pd
df = pd.read_excel("testxl.xlsx")
```

Exercise: print only the **name** column on the screen

Exercise: Create a new columns, twicing the marks and save as a new file.

Writing our own functions

Exercise: Guess the output

```
def myPrint():
    print("ha ha ha")
    print("hi hi hi")
    print("ho ho ho")

myPrint()
```

Exercise: Guess the output

```
def myPrint_n_Times(n):
    for i in range(n):
        print("pak pak")

myPrint_n_Times(10)
```

Exercise: Guess the output

```
myPrint_n_Times()
```

Exercise: Guess the output

```
myPrint_n_Times(10, 20)
```

Exercise: Guess the output

```
def get_n_th_character(s,n):
    return s[n]


c = get_n_th_character("India", 2)
print(c)
```

Exercise: Write a function which accepts a string as input, and returns the **first** character of the string

```
def get_1st_character(s):
    return # your code here

print(get_1st_character("love"))
```

Exercise: Modify the above code to get the last character of the input string.

Global and local scopes

Example:

```
x = 0 # global variable
def testGlobal():
    x = 3 # local variable
    print(x)

testGlobal()
print(x)
```

Example

```
x = 0
def testFun():
    y = 3

testFun()
print(y) # error
```

Example

```
x = 0
```

```
def testFun():
    x = x +  1 # error
testFun()
```

Example:

```
x = 0
def testFun():
    global x
    x = x +  1 # error

testFun()
print(x)
```

Exceptional Handling

A way to let the program continue even if some error happened

Example:

```
y = '3'

try:
    print("x" + y)
    print("all seems ok")
except:
    print("some error happened")

# then rerun with y = 3
```

Exercise: Guess the number

```
# This is a guess the number game.
import random
secretNumber = random.randint(1, 20)
print('I am thinking of a number between 1 and 20.')
# Ask the player to guess 6 times.
"""

YOUR CODE HERE

"""
```

Exercise: The Collatz Sequence

Write a function named collatz() that has one parameter named number. If number is even, then collatz() should print number // 2 and return this value. If number is odd, then collatz() should print and return 3 * number + 1.

Then write a program that lets the user type in an integer and that keeps calling collatz() on that number until the function returns the value 1.

(Amazingly enough, this sequence actually works for any integer—sooner or later, using this sequence, you'll arrive at 1! Even mathematicians aren't sure why. Your program is exploring what's called the Collatz sequence, sometimes called "**the simplest impossible math problem**.")

Lists

Variables which contains one or more other objects.

```
pets = ['cat', 'dog', 'cow', 'horse']
morePets = ['chicken', 'goat']
```

```
# list concatenation
allPets = pets + morePets
```

```
# gets items from list
allPets[2] # 3rd entry

allPets[1:3] # sublist
allPets[-1] # last list


# check if an item is in the list
print('goat' in allPets)
print('snake' in allPets)
```

```
# for loop with list
pets = ['cat', 'dog', 'horse', 'goat', 'chicken']

# loop over list
for p in pets:
    print(p)
```

```
# or this with index
for i, p in enumerate(pets):
    print(i,p)
```

```
# list of numbers
x = list(range(1,100, 2))
```

List methods

```
pets = ['cat', 'dog', 'horse', 'goat', 'chicken']


# position of 'cat'
i = pets.index('cat')
print(i)
```

```
# position of something that does not exist
i = pets.index('snake')
print(i)
```

```
# other useful method
# insert()
# append()
# remove()
```

```
spam = [2,3,-1,2.4,6.2,-2]
spam.sort()
print(spam)
```

```
pets.sort()
print(pets)
```

```
pets.sort(reverse=True)
print(pets)
```

Dictionary

```
#Create and print a dictionary:

thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

print(thisdict)
```

Exercise:

- Get an excel sheet from https://https://data.gov.in/
- Read the sheet in your python program
- Convert the data into a dictionary

Regular Expressions

Pattern matching, advanced search, eg.

- Find all phone numbers in a page
- Find all email addresss

```
# read file content and
f = open("resources/sampleText.txt", "r")
fileContent = f.read()
```

```
# import regex module
import re
```

```
# suppose we are searching for all 4 digit numbers
numberRegEx = re.compile(r'\d\d\d\d')

# find first 4 digit number
mo = numberRegEx.search(fileContent)
#print(mo.group())

# find all 4 digit number
mo = numberRegEx.findall(fileContent)
#print(mo)
```

```
# exericse : find all dates dd/mm/yyyy in the sample text
dateRegEx = re.compile(r'\d\d/\d\d/\d\d\d\d')
mo = dateRegEx.findall(fileContent)
#print(mo)
```

```
## group within match
dateRegEx = re.compile(r'(\d\d)/(\d\d)/(\d\d\d\d)')
mo = dateRegEx.findall(fileContent)
##print(mo)
```

```
## piping
## group within match
dateRegEx = re.compile(r'\d\d/\d\d/(2020|2019)')
mo = dateRegEx.search(fileContent)
#print(mo.group())
```

```
# match non-fixed length pattern
#? zero or one appearence
cRegEx = re.compile(r'a?b')
mo = cRegEx.search("bcd")
#print(mo.group())
```

```
phoneNumberRegEx = re.compile(r'(\+91|0)?(\d\d\d\d\d\d\d\d\d\d)')
mo = phoneNumberRegEx.findall("blah blah +918285906156 pak pak  08285906157
quaq quad 9068900900")
#print(mo)
```

```
#+ 1 more times
wordRegEx = re.compile(r'[a-zA-Z]+')
mo = wordRegEx.findall("156 pak Padak  08285906157 quaq quad 9068900900")
#print(mo)
```

```
numRegEx = re.compile(r'\d+')
mo = numRegEx.findall("156 pak Padak  08285906157 quaq quad 9068900900")
#print(mo)
```

```
#* zero or more
numRegEx = re.compile(r'Can I call you\?*')
mo = numRegEx.findall("blah blah bla Can I call you???")
#print(mo)
```

```
# {n} exact number of match
phoneNumberRegEx = re.compile(r'(\+91|0)?(\d{10})')
mo = phoneNumberRegEx.findall("blah blah +918285906156 pak pak  08285906157
quaq quad 9068900900")
#print(mo)
```

```
# {m,n} m to n repeatting
phoneNumberRegEx = re.compile(r'\d{3,6}')
mo = phoneNumberRegEx.findall("blah blah 18 pak pak  0828 quaq quad
906890")
#print(mo)
```

```
# {m,n} m to n repeatting - non-greedy match with ?
phoneNumberRegEx = re.compile(r'\d{3,6}?')
mo = phoneNumberRegEx.findall("blah blah 18 pak pak  0828 quaq quad
906890")
#print(mo)
```

Character class
\d : is a shorthand character class that matches digits.
\w: matches "word characters" (letters, numbers, and the underscore).
\s : matches whitespace characters (space, tab, newline).

The uppercase shorthand character classes
\D, \W, and \S match charaters that are not digits, word characters, and whitespace.

You can make your own character classes with square brackets: [aeiou]

^caret makes it a negative character class, matching anything not in the brackets:
[^aeiou]

```
#^xyz begins with
# xyz$ ends with xyz
```

```
# dot
# . : anything except newline
atRegEx = re.compile(r'.*\.pdf')
mo = atRegEx.findall(fileContent)
#print(mo)
```

Exercise:

```
#Find all email addresses
```

Exercise:

```
#Find all hyperlinks: https://
```

Multiline search

re.DOTALL

```
atRegEx = re.compile(r'.*', re.DOTALL)
mo = atRegEx.search("i hate this\n non sense")
#print(mo.group())
```

ignore cases (capital/small)

re.I

```
#atRegEx = re.compile(r'.*', re.I)
```

Exericse

Find and replace all numbers with **

```
phoneNumberRegEx = re.compile(r'(\+91|0)?(\d{10})')
textInput = "blah blah +918285906156 pak pak  08285906157 quaq quad
9068900900"
newText = phoneNumberRegEx.sub('**',textInput)
print(newText)
```

Exericse

```
# Find and replace all emails with **
```

Exercise

google re.VERBOSE