Python Fast

Programming for Learning Machines

Niranjan & Vishva

Meegammana Kumara NP

AlgoHack

ISBN 0-670-01043-6

About

This book is designed to jump start Python programming Students following given exercises will be able to learn and understand the fundamentals of python programming to apply them for various applications..

Printed in Sri Lanka.
First Printing, 2018
ISBN 0-670-01039-6
Shilpa Sayura Foundation
Shilpasayura.org

TABLE OF CONTENT .

Global variables
Logging
Communications
IoT
Advanced

The content is still editing and not perfectly in order

Python: is a high-level, interpreted, interactive, **object-oriented** programming language developed by Dutch programmer Guido van Rossum in 1989,.

Python Applications

Web, GUI, Scientific, Numeric, Software, Database, Network, Games, 3D, Console, , Video, Images, CAD, Vision, Machine Learning, Robotics, Harvesting, Scripting, AI, Analytics, IoT

Download Python
http://www.python.org

Install Python IDLE

If you are using windows add a python path to your PATH environment variable. So you can run python from anywhere.

Move to the python directory.
The normal way to run python code is to create a python program and save it as program_name.py.
Then you can run it from the command line
.

```
$ python program_name.py
```

Simple Math Calculations

Start IDLE
Type idle at prompt

```
Idle starts, now practice your math

2+2
2+2+2+2
8*6
4/2
3+5 *3
2**3
5//2
4%3
```

Using Variables

A variable stores a piece of data in a program.
Variables in memory can be used later.
A variable has a specific name to identify it.
You can store numbers, text, lists in variables.
You can pass variables to different functions.

```
x = 15
x=x+5
x=3
x=x+5
y=10
z=y+x
print(z)

p="Hello"
q="Python"
r=p+ " " + q
print(r)

# This is a comment
```

Pythagoras theorem

import math # use math module

```
x=3
y=4
z=x*x + y *y
```

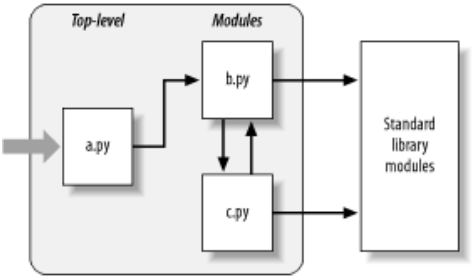
```
math.sqrt(z)
```

Using python Editor to code

Select **New** from the menu to open python editor.

You can code programs here and save as *.py.

You also can import your own and built-in python modules. This is python program architecture.



Reassigning Variables

```
#var1.py
name="Ruvi"
age=18
height=5.6
print(name, age, height)
name="Sindi" # now name is set to
"Sindi"
age=19   # age is now set to 19
height=5.7   # height is now set 5.7
print(name, age, height)
```

You can use an online python editor, spyder or pycharm, sublime text or to code python.

Variables naming rules

- The first character must be a letter or ('_').
- Uppercase, lowercase allowed.
- The rest of the name can consist of letters, ('_') and numbers.
- Variable names are case-sensitive.
- My name and MYNAME are not the same.

Examples :

- Valid names : i, _my_name, name_23, a1b2_c3
- Invalid names: 2things, *name, age+
- Good variable names are important!
- Describe the value: numbats, cats, num_cats

```
# Hard to understand
ir = 0.12
b = 12123.34
i = ir * b

# Easy to understand
interest_rate = 0.12
account_balance = 12123.34
interest_amount = interest_rate *
account_balance
```

Operators

- Operators manipulate the value of operands.
- Arithmetic Operators do calculations.
- Comparison Operators do comparisons.
- Assignment Operators assign values.
- Logical Operators perform logic.

#Arithmetic Operators

```
a=2+3             # Add
b=9-2             # Subtraction
c= 108 * 0.5      # multiplication
c= 6/2            # Division
eight = 2 ** 3    # Exponentiation

x=10
y=2

#How do you get 100 using x,y
z=x**y # Exponentiation
```

Explain output

```
print(4 + 4 )
print(3 - 5)
print(16 / 2)
```

```
print(16 * 4)
print(3 % 2 )

print(2+3*4)
print((2+3)*4)
print(2**10)  # 2^10
print(2**100)

print(6/3) # division
print(7/3)
print(3/6)
print(7%3)  # modular division
print(3%6)
print(7//3) # floor division
print(3//6)
print(1000000000000000 + 101)

print(400000000000000078848488484884*
8888887757548458838358 3)
```

Explain the output

```
y=4
j=2
if ((j<y) or (j!=4)):
    j-=y
    print (j)
else:
    y*=j
```

```
    print (y)
```

#increment x=x+n

```
x = 5
x += 3  # x=x+3
print(x)
```

#decrement x=x-n

```
x -= 1 # x=x-1
print(x)
```

```
x *= 3 is same as x = x * 3
x += 3 is same as x = x + 3
x -= 3 is same as x = x - 3
x *= 3 is same as x = x * 3
x /= 3 is same as x = x / 3
x %= 3 is same as x = x % 3
x //=3 is same as x= x // 3
x **=3 is same as x = x ** 3
```

#AND Sets each bit to 1 if both bits are 1

```
x &= 3 is same as x = x & 3
```

#OR Sets each bit to 1 if one of two bits is 1

```
x |= 3 is same as x = x | 3
```

#XOR Sets each bit to 1 if only one of two bits is 1

```
x ^= 3 is same as x = x ^ 3
```

Variable Types

```
print(20)
print('20')
```

```
The output is the same. But the input data
type is different.
```

```
counter = 100    # integer assignment
miles = 1000.0   # floating point
name = "Nimal"   # string
student=true     # logical assignment
print (miles)
print name
```

```
5/2.0       # divide by float
7/2         # divide by integer
7%2         # modular division gives remainder
7%4
```

int: Integer values

A 32-bit number is made of 32 bits of 1's and 0's.

So 2^32 has 4,294,967,296 possible values.

We use the first bit to represent the sign of negative numbers. We can use 2^31–1=2,147,483,647 signed numbers.
So we can represent -2,147,483,648 to 2,147,483,648.

long: Any integer larger than an int.
float: All floating-point values like 2.4567890333333
str: Strings and characters are the same types. “A” or “ABC”

Strings

```
a="Hello"
print (a)
b=a + " Python"
Print (b)
```

Input and output

```
name = input("Enter Your Name ")
age = input("Enter Your Age :")
print ("Your name is : " + name)
print ("Your age is : " + age)
```

Numbers Input & output

input() reads a line of text.

```
#example a=b+c
number1 = input()  # enter 1
number2=  input()  # enter 2
number3=number1+ number2
print (number3) # check the output

# output is 12 why?
# try entering decimal numbers 1.5 and 2.7
# output is 1.52.7 why?
```

Python Strings:

Strings are set of characters one after another.

```
str = 'Hello Python!'
print (str) # Prints complete string
print (str[0]) # prints first character
print (str[2:5]) # characters from 3rd to 5th
print (str[2:]) # prints from 3rd character
print (str * 2) # Prints string 2 times
print (str + "TEST") # Prints joined string
```

Errors & debugging

Programming is done by human beings. Humans make errors. Programming errors are called bugs

Finding errors in a program and correcting is called debugging.

Syntax errors

Syntax errors occur when programming rules are broken. Using Print instead of print creates an error.

Runtime errors

Errors created by the system. Trying to open a file which does not exist causes a runtime error.

Symantec Errors

The program runs but does not give desired results.
It happens when there is a programming logic error.

Write a program

Accept a number, multiply by 2 and print the result

Accept the radius of a circle and compute the area.
Area Of Circle=3.14 * r * r

Type conversion

```
print('12' + 12)
```

```
TypeError: Can't convert 'int' object to str
implicitly
```

Strings and numbers can’t and together.
They are different types

Convert numbers to strings before joining with string.

```
print('12' + str (12))  # 1212
print(int('12') + 12) # 24
```

The secret of Number Input

All inputs in python are in text format.
Numbers and text are different in a program.
Convert text to obtain the number.
Convert a number to obtain the text.

```
number1 = input()  # enter 1
n1=int(number1)
number2 = input()  # enter 2
n2=int(number2)
n3= n1+ n2
print (n3) # check the output
# output is 3 explain
```

Examples

```
x="2.5"
int(x ) - Converts x to an integer.
long(x) Converts x to a long integer.
float(x) - Converts x to a floating-point
y=65 # y is an integer
str(y) - Converts y to a string.
chr(y) - Converts y to a character.
hex(y) -Converts y to a hexadecimal string.
oct(x) - Converts y to an octal string.
```

Math Library

```
Import math
x=math.sqrt(16)
print(x)
```

Write programs

Accept the base and height of a triangle and compute the area.
return true if the two given integer values are equal or their sum or difference is 5.

1. Calculate distance between the points (x1, y1) and (x2, y2). Formula SquireRoot((x2-x1) * (y2-y1))

2. Solve (x + y) * (x + y)
   Expected Output : (4 + 3) (4+3) = 49

Write Programs

1. Write a Python program to convert degree to radian.

2. Write a Python program to convert radian to degree.

3. Write a Python program to calculate the area of a trapezoid.

4. Write a Python program to calculate surface volume and area of a cylinder.

5. Write a Python program to calculate surface volume and area of a sphere.

6. Write a Python program to calculate arc length of an angle.

Booleans

A boolean variable is like a light switch.
It can only have two values, True or False.
Python stores true as 1 and false as 0
but outputs 'true' or 'false' from print statements
If you input Boolean values, you must input 1 or 0.

```
a = True
b = False
#Boolean expressions return a True or False
c=(b < a)
print(c)
```

Logical comparison operators

The result of a comparison is true or false.

```
A < B       A less than B
A > B       A greater than B
A <= B      A less than or equal to B
A >= B      A greater than or equal to B
A == B      A equal to B
A != B      A not equal to B
A and B     A and B should be true
A or B      A or B should be true
A not B     Negates the truth value
```

Examples

```
print(5==10)
print(10 > 5)
print((5 >= 1) and (5 <= 10))
```

Escaping special characters

```
print('You're good!')
```

```
SyntaxError: invalid syntax
```

' is a special character.
We have to escape it with backslash (\)

```
print('You\'re good!')

print('backslash at the: \\')
print('up\\down')
```

Python Special Characters

\\	Backslash (\)
\'	Single quote (')
\"	Double quote (")
\a	ASCII Bell (BEL)
\b	ASCII Backspace (BS)
\f	ASCII Formfeed (FF)
\n	ASCII Linefeed (LF)
\r	ASCII Carriage Return (CR)
\t	ASCII Horizontal Tab (TAB)

Special Character Challenge

WHAT WOULD YOU EAT "I LIKE CANDY"?!? YOU'RE GOOD!!

```
print("WHAT DO YOU MEAN \"I WANT A RAISE\"?!? YOU\'RE FIRED!!")
```

Write a Python program to print the following Poem

Twinkle, twinkle, little star,
How I wonder what you are!
Up above the world so high,
Like a diamond in the sky.

Output it in one statement

print("Twinkle, twinkle, little star, \n\tHow I wonder what you are! \n\t\tUp above the world so high, \n\t\tLike a diamond in the sky. \n")

% String format operator

Python uses the "%" operator to format a set of variables enclosed in a "tuple" with "argument specifiers", special symbols like "%s" and "%d".

```
name = "Ping"
print("Hello, %s!" % name)

age = 16
print("%s is %d years old." % (name, age))

#more
```

```
print("%x" % 17) #hexadecimal
print("%#x" % 17) #output ox11
print("%03d" % 1) #output 001
print("%+d %+d" % (1, -1)) #sign flag
print("%.1f %.2f" % (3.14, 2.7956)) # decimal
places
print("%i" % 0b1111) # binary to decimals
print("%i" % 0b1111)
print("%x" % 255) # decimal to hex
print("%o" % 15) # decimal to octal
print("%F" % 10.345678) # floating point
print("%g" % 1234567890) # scientific
print("%G" % 1234567890) # scientific
print("%c" % 65) # character string
```

Format() method

Formats the given value(s) inside the string.

```
txt = "Only LKR {price:.2f} !"
print(txt.format(price = 100))

#named indexes:
str1= "She is {fname} and her age is
{age}".format(fname = "Shara", age = 21)
#numbered indexes:
str2= "She is {0} and age is
{1}".format("Shara", 21)
#empty placeholders:
```

```
str3= "She is {} and age is
{}".format("Shara", 21)

print(str1)
print(str2)
print(str3)

#Left aligns the result
txt1 = "We have {:<8} eggs."
print(txt.format(19))

#Right aligns the result
txt1 = "We have {:>8} eggs."
print(txt.format(19))

#Center aligns the result
txt1 = "We have {:^8} eggs."
print(txt.format(19))

#+ indicate if the number is positive or
negative:
txt1 = "Temperature : {:+} and {:+} ."
print(txt.format(-3, 15))

#- indicate if the number is negative:
txt1 = "Temperature : {:+} and {:+} ."
print(txt.format(-3, 15))

#, used as a thousand separator:
```

```
txt1 = "This car is US$ {:,}."
print(txt.format(43800000))

#b converts the number into binary format:
txt1 = "The binary of {0} is {0:b}"
print(txt.format(5))

#b converts the number into decimal format:
txt1 = "We have {:d} eggs."
print(txt.format(0b101))

#e converts the number into scientific format:
txt1 = "Sun is at {:e} km distance."
print(txt.format(1500000))

#fixed point format
txt1 = "The price is US$ {:.2f} ."
print(txt.format(45))
```

Write examples for below formats

```
:o          Octal format
:x          Hex format, lower case
:X          Hex format, upper case
:n          Number format
:%          Percentage format
```

That’s about Strings and Number operations.

String Methods

len()

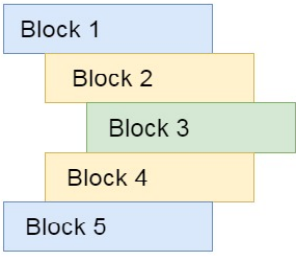
returns the length of the string.

Conditions

Conditionals help us evaluate inputs and make logical decisions to branch program running.

Indenting code blocks

We indent code to indicate a block of code in Python, When indenting we must indent each line of the block




```
import math
n=0
while (n < 16):
    if (n % 2 ==0):
        print(n, "is a even number")
        if (math.sqrt(n) % 2==0):
            print(n, "is also a squired number")
    else:
        print(n,  "is a odd number")
    n+=1
print("End")
```

We indent code to indicate a block of code in Python, When indenting we must indent each line of the block

if statement:

If (rain) I (sleep) else I (work) is a logical statement.

```
x=2
if x < 5:
    print "X < 5"
```

else statement

```
x=3
if x < 5:
   print "X < 5"
else:
   print "X >= 5"
```

if .. elif .. else statement

If (rains) I (sleep) elseif I (work) else I (swim)

```
x=8
if (x=1):
   print ("X = 1" )
elif x=2) :
   print ("X = 2")
elif (x=3) :
   print ("X = 3")
else:
   print ("X > 3 or X < 0")
```

nested if

```
x=12
if (x > 5):
   print (“X > 5”)
if (x < 8):
      print (“x < 8” )
elif (x > 10) :
   print (“x > 10”)
elseif :
   print (“x > 10”)
else:
 print (“X <= 5”)
```

While and if together

```
import math
n=0
while (n < 16):
   if (n % 2 ==0):
      print(n,  "is a even number")
      if (math.sqrt(n) % 2==0):
         print(n,  "is also a squared number")
   else:
      print(n,  "is a odd number")
   n+=1
print("End")
```

Case In Python

Select different paths based on a value

```
age = input("Enter Your Age:")
if (age < 5):
      print "You're too young for python!"
elif age = 6:
      print "you can try python!"
 elif age > 60:
      print "No time for python! Take a nap"
else: # Default case
       print "OK Do python"
```

Test an input number is within 0 to 100 OR 100 to 1000 or 1000 to 2000.

Calculate the sum of three given numbers, if the values are equal then return three times of their sum.

Find whether a given number is odd or even or odd.

Looping

Loops help run a code block repeatedly. The repetition ends when a condition is met.
while (raining) I (stay) at home .
Run 4 rounds to run 400 meters.

For

Repeats a code block specific number of times

```
for x in range(10):
  print(x)
```

How do we calculate the sum of numbers 1 to 10.

```
sum=0
for x in range():
  sum=sum+x

print(sum)
```

While

Repeats a code block while a condition is true.

```
x =4
while (x < 10):
 print( x)
 x=x+1
```

Can you make above loop exit when x is 5
What happens if x=x+1 removed?

Explain this

```
for x in range(0, 100):
    print( x)
```

mile/kilometer conversion

Range function takes three integer parameters from, to and step.

```
km =1.609
   for m in range (1, 10, 1):
      kmpm=m * km
      print("%d m -> %3.2f km" % (m, kmpm))
```

A loop inside a loop

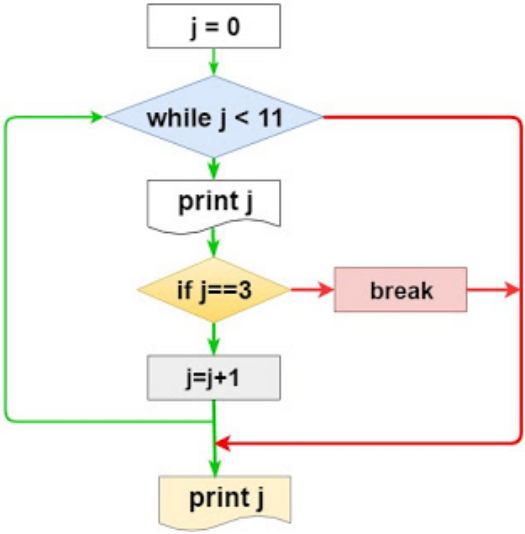
```
for i in range(4):
    for j in range(4):
         print (str(i) + “*” + str(i * j))
```

The break Statement:

Breaks out of current loop

```
for letter in 'Python':
  if letter == 'h':
   break
  print ('Letter:', letter)
```

Breaking a Loop




```
j=0
while (j < 11):
    print("inside while J=" , j)
    if (j == 3):
        break
    j += 1 # increment j by 1 (same as j=j+1)

print("I am free J=", j)
```

Write programs

Count from 1 to 20 in a loop.

Count from 20 to 1 in a loop.

Process numbers 1 to 100 in a loop and print odd numbers.

Guessing game

What does this program do?

```
import random
answer = random.randint(0, 999)

for n in range(0, 10):
    guess = int(input("Guess: "))
    if guess < answer:
        print("Too low!")
    elif guess == answer:
        print("Correct, you got " +
str(answer))
        break
    else:
        print("Too high")
```

Continue Statement:

Returns the control to the top of the loop, rejecting all the remaining statements.

```
for letter in 'Python':     # First Example
   if letter == 'h':
```

```
      continue
   print 'Current Letter :', letter
```

 # Second Example

```
var = 10
while (var > 0):
   print 'Current variable value :', var
   var = var -1

if (var == 5): continue
   print ("Good bye!")
```

Deaf Grandpa

1. Your Grandpa likes you talking. If you type in a normal case, he responds with "HUH?! SPEAK LOUD KID!"
   If you type in capital, he responds with "NO, NOT SINCE 1945!". Every time he responds he forgets the year and tells a random year between 1945 to 1960.
   When you say "BYE" he says "STAY!". But when you say "BYE" 3 times you can leave.

2. Write a program that calculates and prints the value according to the given formula:

Q = Square root of [(2 * C * D)/H]
Following are the fixed values of C and H:
C is 50. H is 30.
D is the input to your program

Operator Precedence

What is the output of following expression

```
x=r%3*c+10/y
print (x)
```

()	Parentheses (grouping)
x[index:index]	Slicing
x[index]	Subscription
x.attribute	Attribute reference
**	Exponentiation
~x	Bitwise not
+x, -x	Positive, negative
*, /, %	Multiplication, division, remainder
+, -	Addition, subtraction
<<, >>	Bitwise shifts
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
in, not in, is, is not, <, <=, >, >=, <>, !=, ==	Comparisons, membership, identity
not x	Boolean NOT

and Boolean AND
or Boolean OR

Workings

```
((r% 3)*c)+(10/y)
((11% 3)*4)+(10/2.5)
(2*4)+(4)
12
```

```
100 - 25 * 3 % 4 the result is 97 how?
25*3 = 75
75 % 4 = 3 (4*18 = 72; remainder is 3)
100 - 3 = 97
```

More

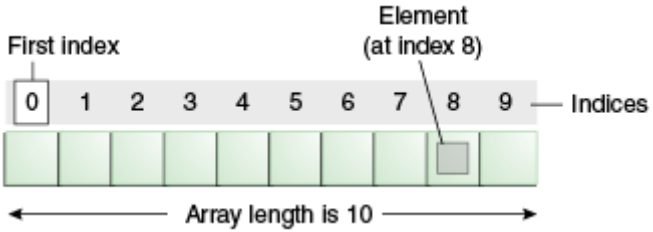
```
a = 20
b = 10
c = 15
d = 5
e = 0

e = (a + b) * c / d       #( 30 * 15 ) / 5
print ("Value of (a + b) * c / d is ",  e)

e = ((a + b) * c) / d     # (30 * 15 ) / 5
print ("Value of ((a + b) * c) / d is ",  e)

e = (a + b) * (c / d);    # (30) * (15/5)
print ("Value of (a + b) * (c / d) is ",  e)
```

```
e = a + (b * c) / d;      #  20 + (150/5)
print ("Value of a + (b * c) / d is ",  e)
```

Lists (Arrays)

Lists can store many values using variable names.
They are also called arrays.
List of names, Names of cities, a price list is an example.
Python gives 3 types of arrays tuples, lists and dictionaries. There elements can be different types

If there are n items in an array
The first element index is 0
The last element index is n-1



Tuples

Tuple is an array with limited features.
Tuples are read only which means Immutable..

```
p = (7, 4) # has two elements
print (p[0]) # prints 7
```

Check whether a given value is in a group of values.

How do you get second element 4

```
tuple = ( 'a', 7, 2.23, 'Sam', 7.2 )
tinytuple = (123, 'Saman')
print (tuple) # Prints complete list
print (tuple[0]) # Prints first element
print (tuple[1:3]) # Prints 2nd to 4th
print (tuple[2:]) # Prints from 3rd element
print (tinytuple * 2)  # Prints list two times
print (tuple+ tinytuple) # Prints Joined lists
```

A tuple is a list of one datatype.

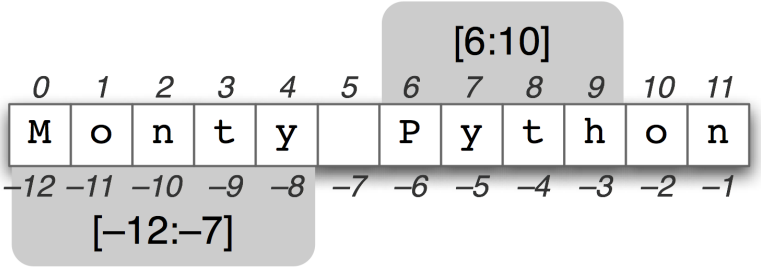
The differences between lists and tuples:
Lists are enclosed in brackets [ ]
List elements and size can be changed.
Tuples are enclosed in parentheses ( )
Tuples cannot be updated.
Tuples are read-only lists

```
tuple1 = (123, 'Niki', "Sugi",9)
print tuple1       # Prints complete list
print tuple1[0]    # Prints first element
print tuple1[1:3]  # Prints from 2nd to 3rd
print tuple1[2:]   # Prints from 3rd element
```

Strings are Arrays

```
Index from rear:    -6  -5  -4  -3  -2  -1
Index from front:    0   1   2   3   4   5
                   +---+---+---+---+---+---+
                   | a | b | c | d | e | f |
                   +---+---+---+---+---+---+
Slice from front:  :   1   2   3   4   5   :
Slice from rear:   :  -5  -4  -3  -2  -1   :
```

Python string is a character array.




```
a="Monty Python"
start, end,step=2,15,2
b=a[start:end] # items start to end-1
c=a[start:]    # items start to end-1
d=a[2:end]      # items from 2 to end-1
e=a[:]         # a copy of whole array
f=a[start:end:step] # start to end-1, by step
```

explain values of

```
a, b,c,d,e, f
```

negative values tell count from the end of the array

```
l=a[-1]    # last item in the array
m=a[-2:]   # last two items in the array
n=a[:-2]   # everything except the last two
items

o=a[::-1]    # all items in the array,
reversed
```

explain output

```
print(l,m,n,o)
```

```
p=a[1::-1]   # the first two items, reversed
q=a[:-3:-1]  # the last two items, reversed
# everything except the last two, reversed
r=a[-3::-1]
```

explain output

```
print(p,q,r)
```

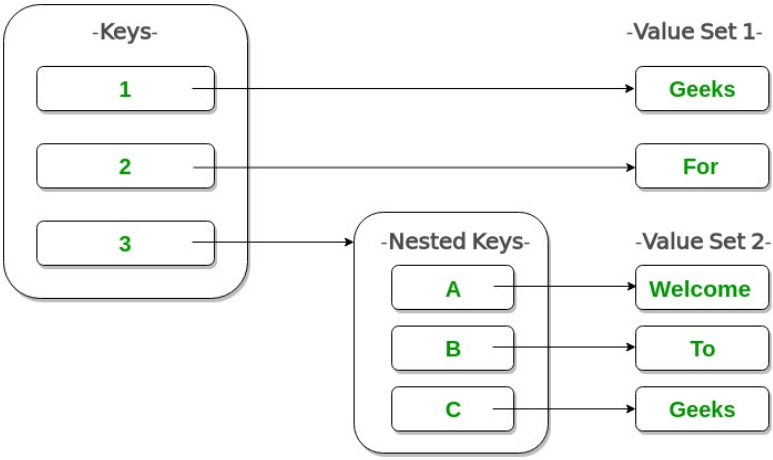
Lists

Python lists, tuples and dictionaries are like arrays. Python lists can have the ability to have elements of different data types.

```
list=["A",2, True, (2,4)]
tuple=(3,2,1, "A", [])
```

```
print(list)
print(tuple)
```

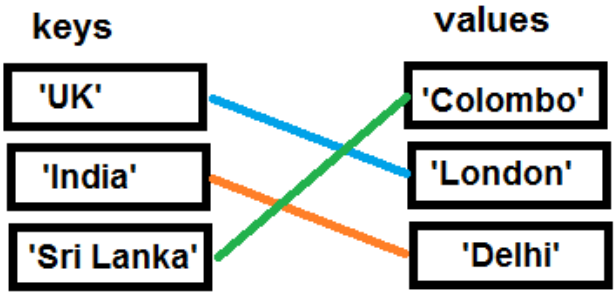
Dictionaries



Dictionary is a list with key, value pairs.

```
dict={"A":1, "B":2, "C":[3,4]}
print(dict)
```

Dictionaries have Elements made of key, value pairs
Every Key is unique. Values can be different.
Values changeable by using the key, hence mutable
Python dictionaries are associative arrays
The elements in the array consist of key-value pairs.



Key is a unique value.
Value can be any python object
Dictionaries are enclosed by curly braces { }
values can be accessed using square braces []

```
dict1 = {} # declare dictionary object
dict1['one'] = "One"    # element
dict1[2]     = "Two"  # another element

print (dict1) # Prints complete dictionary
Print (dict1['one'])# value for key 'one'
print (dict1[1]) # Prints value of 1st element
print (dict1.keys())   # Prints all the keys
print (dict1.values()) # Prints all the values

#The output will be
One
Two
{2: 'Two', 'one': 'One'}
[2, 'one']
['Two', 'One']
```

Another Example

```
P3 {"Kan" : 45, “Anu" : 56, "Col” : 51"}
print(P3[“Kan”])

tinydict = {'name': 'bob','code':6734}
Print (dict['name']) # value for 'name' key
print (dict[2]) # Prints 2nd element
print (tinydict) # Prints complete dictionary
print (tinydict.keys()) # Prints all the keys
print (tinydict.values()) # all the values
```

Write a program which takes 2 digits, X,Y as input and generates a 2-dimensional array with element value in the i-th row and j-th column of the array should be i*j.
Ex.
input 3,5
output [[0, 0, 0, 0, 0], [0, 1, 2, 3, 4], [0, 2, 4, 6, 8]]

Data Science, Analytics and Machine Learning use python Lists and Dictionaries.

Mutable and Immutable

Everything in Python is an object.
All objects in Python can be either mutable or immutable.
Every variable holds an object instance.

When an object is initiated, it gets a unique object id.
Its type is defined at runtime. Once set can't change.
However its state can be changed if it is mutable.
A mutable object can be changed after it is created.
But an immutable object can't be changed after creation.

Objects of built-in types like
int, float, bool, str, tuple, unicode are immutable.
Objects of built-in types like (list, set, dict) are mutable.
Custom classes are generally mutable.

Class	Description	Immutable?
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	✓
dict	associative mapping (aka dictionary)	

Data Type Test

```
x = []
for i in range(0,5):
    x.append(i)
print (x)
print ("-------")
```

```
y = x * 5
print(y)
x = "Hello World!"

print (x)
print ("---------")
print (x * 5)
print (y)
```

Explain output

More String functions

join()

joins a sequence of strings given in arguments

```
str = "_"
str1 = ( "Ping", "Pong", "Kong" )
print ( str.join(str1))
```

strip("string")

delete all the leading and trailing characters

lstrip("string")

delete all the leading characters mentioned in its argument.

rstrip("string")

delete all the trailing characters mentioned in its argument.

```
str = "***sumoateallcakes***"
print ( str.strip('*') )
print ( str.lstrip('*') )
print ( str.rstrip('*') )
```

max("string")

returns the maximum value alphabet from string.

min("string")
returns the minimum value alphabet from string.

```
print(max(str))
print(min(str))
```

replace()

Replaces a string with another string

```
str = "Kandy Randy Andy"
str1 = "Randy"
str2 = "Andy"
print (str.replace( str1, str2, 2))
```

Aligning Strings

We can align strings: left, right, or center using ljust, rjust, and center methods for string objects

```
x="abc"
x.center(20)
Print ( x.center(20, '+')
```

```
s='|'+ x.ljust(10)+ '|'+ x.center(20)+ '|'+
x.rjust(10) + '|'
print(s)
```

string.find(“substring”, start, end)

Finds the position of the substring within a string.
if not found in given range returns -1.
if found returns first occurrence of the sub string

```
str = "the cat ran over bat"
str2 = "ran"
print (str.find(str2))  # output 8
print (str.find( str2, 9) )   # output -1
```

string.rfind(“substring”, start, end)

Returns the position of the last occurrence of the string.

```
print ( str.rfind( str2) )  # output 8
print ( str.rfind( str2,6) ) # output 8
print ( str.rfind( str2,9) ) # output -1
```

Explain following outputs

string.startswith(“prefix”, start, end)
return true if the string starts with a prefix else return false.

string.endswith(“suffix”, start, end)
return true if the string ends with a suffix else return false.

```
str = "the cat ran over bat"
str1="the"
str2 = "bat"
print(str.startswith(str1))
print(str.endswith(str2))
```

swapcase()

swap the cases of string. Upper to lower case and vice versa.

title()

converts the string to its title case

```
str = "GREEK IS sweet"
print(str.lower())
print(str.upper())
print(str.swapcase())
print(str.title)
```

count(“substring”, start, end)

Counts the occurrence of a substring in the whole string.

```
str = "Ping Pong Hong Kong Ping"
print (len(str)) # output 24
print (str.count("Ping")) # output 2
print (str.count("Ping",0,18)) #output 1
print (str.count("Ping",15)) #output 1
print (str.count("Ping",5,15)) #output 0
```

Checking if str has all alphabets

```
print(str.isalpha()):
```

Checking if str1 has all numbers

```
if (str.isalnum()):
```

Checking if str1 has all spaces

```
if (str1.isspace()):
```

Play Random Cricket

```
import random
n_balls=int(input("ENTER NUMBER of balls to be played: "))
n_wickets=int(input("ENTER NUMBER OF WICKETS: "))

print("No of balls to be played = ",n_balls)
score=[]
while(n_balls>0 and n_wickets>0):
```

```
    print("ENTER RUNS (0-6)")
    p1=int(input())
    r=random.randint(0,6)
    if(p1 in range(0,7)):
        print("Player 1 = ",p1)
        print("Random Bot = ", r)
        if(r==p1):
            print("HOWAZZATTT")
            n_wickets-=1
        else:
            score.append(p1)
            print("Score = ",sum(score)," with ",n_balls," remaining
and ",n_wickets," wickets in hand")
    else:
        print("ERROR, ONLY 0-6 runs allowed")
    n_balls-=1
```

Write a program

"99 Bottles of Beer"

This is an anonymous folk song dating to the mid-20th century. It is a traditional reverse counting song. Write a program that prints out the lyrics.

Leap years

Write a program that asks for a starting year and an ending year and output all the leap years between them.

Leap years are years divisible by 4. ex. 1980, 2008. But years divisible by 100 are not leap years (1800 and 1900) unless they are also divisible by 400 (1600 and 2000).

Functions

Functions are blocks of code that we call repeatedly at different places in a program. Functions take some parameters, process and return a value.

```
def sum( a, b ):
#a and b are parameters passed to the function
    print (a+b;)
    Return

# Now you can call sum function
sum(3,2) #call the function with parameters
sum(5,4)
```

Write a function printname(a,b).

a and b are first and last names passed to the function. The function must print fullname.

Write a function multiply(a,b,c).

A, b and c are numbers passed to the function.
The function must return their product.

Define a function max() that takes two numbers as arguments and returns the largest of them. Use if-then-else. Do not use the built in max() function.

Define a function max_of_three() that takes three numbers as arguments and returns the largest of them.

Write a function that takes a character and returns True if it is a vowel, False otherwise.

Write a function to make a list and print

```
def makelist():
   a = [] # list
   for i in range(1, 20):
       a.append(i)
    return a

def printlist(a):
 for i in a:
    print (i)
return

b=makelist()
printlist(b)
```

#for loop with else example

```
mystring= "house"
vowels = set('aeiou')
for a in mystring:
   if (a in vowels):
       print (mystring, "has a vowel", a)
       break
   else:
         print(mystring, "has no vowel")
```

Write a Python program to convert a decimal number to a binary number.

Write a Python function that takes a sequence of numbers and determines if all the numbers are different from each other.

Write a Python program to create all possible strings by using 'a', 'e', 'i', 'o', 'u'. Use the characters exactly once.

Write a Python program to remove and print every third number from a list of numbers until the list becomes empty.

Bit shifting

#NOT Inverts all the bits
Gives two's complement integer representation.

1 is represented as 0000 0001.
When inverted: 1111 1110 it is -2.
15 is 0000 1111
when inverted it is 1111 0000 = -16.
In general, ~n = -n - 1

#Zero fill left shift

```
Shifts left by pushing zeros from the right
and let the leftmost bits fall off

x >>=3 is is same as x = x >> 3

x << y
Returns x with the bits shifted to the left by
y places
print(5<<1)
101 becomes 1010

print(5) #5
print(bin(5)) # 101
print(5<<1) #10
print(bin(5<<1)) #1010
```

#Zero fill right shift

```
Shifts right by pushing zeros from the left
and let the rightmost bits fall off

x <<=3 is same as x = x << 3
```

```
x >> y
Returns x with the bits shifted to the right
by y places
print(5>>1)
101 becomes 10

print(5) #5
print(bin(5)) # 101
print(5>>1) #2
print(bin(5>>1)) #10
```

Python PART II

Python Scope

When a line of code asks for a value of a variable, python searches all available namespaces, in the order of local, global, built-in and module namespaces.

Local namespace :

This limits variable visibility to the current function.
If the function defines a local variable x, or x is an argument x, Python will use this and stop searching.

```
def fn():
    x = "local"
    print(x)

fn()
print ("outside fn")
print(x) #produces error
NameError: name 'x' is not defined
```

Global namespace :

specific to the current module. If the module has defined a variable, function, or class called x, Python will use that and stop searching.

```
x = "global"
```

```
def fn():
   print("x inside fn :", x)

fn()
print("x outside fn :", x)
```

Built-in namespace:

global to all modules. As a last resort, Python will assume that x is the name of a built-in function or variable.

`UnboundLocalError`:

```
x = 1

def fn1():
    x = x * 2 # Error occurs

fn1()
```

Accessing local variable outside the scope

```
def fn():
    y = "local"

fn()
print(y)
```

Using global and local variables locally

```
x = "global"
def fn():
    global x
    y = "local"
    x = x * 2
    print(x)
    print(y)

fn()
```

Global and Local variable in same name

```
x = "A"
def fn():
    x = "B"
    print("local x:", x) # B

fn()
print("global x:", x) # A
```

Local variables are generally faster.

This function takes about 0.3 seconds :

```
def func():
    for i in range(10000000):
```

```
        x = 5
```

func()

The global version, which takes about 0.5 seconds

```
def func():
    global x
    for i in range(1000000):
        x = 5
x=0
func()
```

Python Date & TIme

Python programs do not include all functions by default. It has tons of libraries to do different things. So handling data and time requires "datetime" and "time" libraries . We call them modules. We need to import them into our programs to use their functions.

from datetime import datetime
import time

```
# we import datetime and time modules
# and call their functions

from datetime import datetime
now = datetime.now()
print (now)
# you can get more attributes from now object

print ("Current year:", now.year)
print ("Current month:", now.month)
print ("Current day:", now.day)
print ("Current hour:", now.hour)
print ("Current minute:", now.minute)
print ("Current second:", now.second)
print ("Current microsecond:",
now.microsecond)
```

How many seconds old are you?

```
from datetime import datetime
import time
import math

#find your age in seconds
t0 = datetime(1969, 2, 24, 16, 31,7)

print(t0)
timenow =time.localtime() # gives you time in
seconds from 1970.01.01
timeObj = datetime(tn.tm_year, tn.tm_mon,
tn.tm_mday, tn.tm_hour, tn.tm_min,tn.tm_sec)
print(timeObj)
td=t1-t0
sd=str(td)
dp=sd.split(",") # split
dd=dp[0].split(" ")
dhms=str(dp[1]).split(":")
print(dd)
print(dhms)
```

Write a program that tells you the following:

How many hours are in a year?
How Minutes in a decade?
How many minutes are in a decade?

Did you consider leap years?

Time String to Unix time

```
from datetime import datetime
import time

ttuple1=datetime.strptime('Sun Apr 28 03:36:54
+0000 2019', '%a %b %d %H:%M:%S +0000 %Y')
print(ttuple1)
tu=time.mktime(tm.timetuple())
print(tu)
```

Unix Time to Time String

```
ttuple2=time.localtime(tu)
print(ttuple2)
ts=time.strftime("%a, %d %b %Y %H:%M:%S GMT",
ttuple2)
print(ts)
```

Built in Functions

int() Returns an integer

```
x = int("12")
```

str() Returns a string

```
y = str(124)
```

float() Returns a floating point number

```
z = float(7)
print(z) # 7.0
r = float("12.4567")
print(r)
```

round() rounds a floating point

```
x = round(5.74643, 3)
print(x) # 5.746
x = round(5.74643, 2)
print(x) # 5.46
```

pow() x to the power of y

```
x = pow(2, 3)
print(x) # 8
```

abs() absolute value of a number

```
print(abs(-7.25)) # 7.25
```

bool() Converts to Boolean

```
x = bool(True)
y = bool(0)
```

chr() Character from an Integer

```
x = chr(65)
print(x) # A
```

bin() integer to binary string

```
x = bin(15)
print(x) # 0b1111
```

```
x = bool(0)
print(x) # False
```

ord() Unicode for Unicode character

```
x = ord("ढ")
print(x) # 3461
```

Algorithms with python

Factorial

Factorial of a number is that number times the factorial of that number minus 1.
Factorial (3) =3 * 2 * 1

What is the factorial of 4?
What is the factorial of 5?

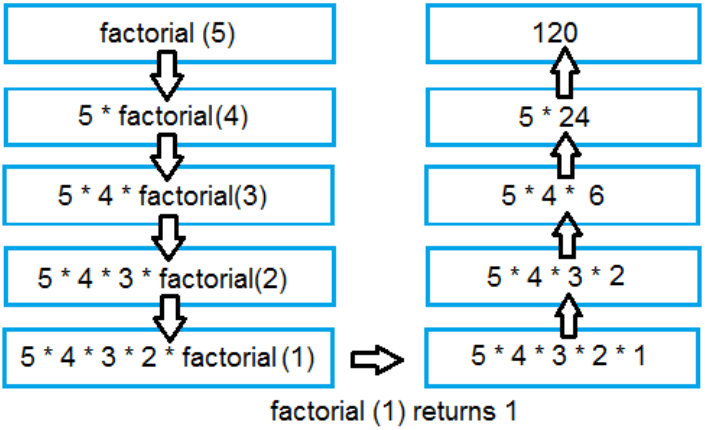
This function calculates factorial of 5

```
def factorial(n):
    f=1
    for i in range(1, n+1):
      f=f*i

    return f

f=factorial(5)
print(f)
```

Recursion

A recursive function calls itself again and again until a solution is found or no more solutions.

This function recursively finds factorial




```
def factorial(n):
    if (n == 0):
        return 1
    else:
        return n * factorial(n - 1)

f=factorial(5) # 1 * 2 * 3 * 4 * 5
print(f)
```

Recursion is an important technique in problem solving.

Sum of numbers with recursion

```
def sumofn(n):
    if (n==1):
        return 1

    return n+sumofn(n-1)

print(sumofn(10)) #output 55
```

Fibonacci Numbers

Fibonacci numbers start from 0 and 1, and the next number is made by adding up the two previous numbers. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …

What is the fibonacci number of element 20?

```
def fib(n):
   if n == 1:
      return 1
   elif n == 0:
      return 0
   else:
      return fib(n-1) + fib(n-2)

print(fib(7))
```

Can you modify the above program to calculate number series, where each number is the sum of 3 previous numbers?

Prime Number Check

Numbers that are not divisible by 1 and itself.

```
def isprime(N,a=2):
    #print (N,a)
    if N <= 2:
        return False
    if (N % a == 0):
        return Fals    if (N > a+1):
        return isprime(N, a + 1)
    else:
        return True

print(isprime(4))
print(isprime(17))
```

Turtle

Turtle is a library for drawing graphics in python.

```
import turtle   # use the turtles library
wn = turtle.Screen() # create graphics window
ax = turtle.Turtle() # create turtle named ax
```

```
ax.forward(150)     # tell alex to move
forward 150 units
ax.left(90)    # turn by 90 degrees
ax.forward(75) # second side of rectangle
```

Draw a hexagon

```
import turtle
polygon = turtle.Turtle()

num_sides = 6
side_length = 70
angle = 360.0 / num_sides

for i in range(num_sides):
    polygon.forward(side_length)
    polygon.right(angle)

turtle.done()
```

Rainbow Benzene

```
import turtle
colors = ['red', 'purple', 'blue', 'green',
'orange', 'yellow']
t = turtle.Pen()
turtle.bgcolor('black')
for x in range(360):
```

```
    t.pencolor(colors[x%6])
    t.width(x/100 + 1)
    t.forward(x)
    t.left(59)
```

Python bytearray()

The bytearray() method returns a bytearray object which is an array of the given bytes.

Syntax

bytearray([source[, encoding[, errors]]])

source (Optional) - source to initialize the array of bytes.

encoding (Optional) - if the source is a string, the encoding of the string.

errors (Optional) - if the source is a string, the action to take when the encoding conversion fails.

```
string = "Python is cool."

#  utf-8 encoded string

arr = bytearray(string, 'utf-8')
print(arr)

#integer
size = 5
```

```
arr = bytearray(size)
print(arr)

#Iterable list
rList = [1, 2, 3, 4, 5]

arr = bytearray(rList)
print(arr)
```

Python bytes()

The bytes() method returns an immutable bytes object initialized with the given size and data.

```
string = "Python is cool."

# utf-8 encoded string
arr = bytes(string, 'utf-8')
print(arr)


#integer
size = 5
arr = bytes(size)
print(arr)

#Iterable list
rList = [1, 2, 3, 4, 5]
```

```
arr = bytes(rList)
print(arr)
```

Event-Driven Programming

Python can respond to events and make things happen.

Keyboard events

```
import turtle

turtle.setup(400,500)  # window size
# Get a reference to the window
wn = turtle.Screen()
ibba = turtle.Turtle() # Create turtle

# event handlers.
def h1():
   ibba.forward(30)
def h2():
   ibba.left(45)
def h3():
   ibba.right(45)
def h4():
    wn.bye()    # Close the turtle window

# wire up keypresses to the handlers
wn.onkey(h1, "Up")
wn.onkey(h2, "Left")
```

```
wn.onkey(h3, "Right")
wn.onkey(h4, "q")

# tell the window to start listening
# If any keys we monitor is pressed,
#call its handler will be called.

wn.listen()
wn.mainloop()
```

Mouse Events

```
import turtle

turtle.setup(400,500)
wn = turtle.Screen()
ibba = turtle.Turtle()

def h1(x, y):
   ibba.goto(x, y)

wn.onclick(h1)  # Wire up the click.
wn.mainloop()
```

1. Write a function to draw a circle on a given radius as input.

2. Write a program to draw a kite with a turtle.

3. Write a program to color the kite with a turtle.

4. Write a program reverse a sentence.

5. Write a password generator in Python.

6. Create a program that will play the “cows and bulls” game with the user. The game works like this:

   Randomly generate a 4-digit number. Ask the user to guess a 4-digit number. For every digit that the user guessed correctly in the correct place, they have a “cow”.

   For every digit the user guessed correctly in the wrong place is a “bull.” Every time the user makes a guess, tell them how many “cows” and “bulls” they have. Once the user guesses the correct number, the game is over.

PART III

Algorithms

Computer science is the study of problems, problem-solving, and the solutions that come out of the problem-solving process.

When we have to solve a complex problem, we need to create models for efficiently solving a problem through a process. The way we solve the problem is an algorithm.

Find the largest among the three input numbers

```
# change the values of n1, n2 and n3
n1 = input("Enter first number: ")
n2 = input("Enter second number: ")
n3 = input("Enter third number: ")

if (num1 >= num2) and (num1 >= num3):
        largest = n1
elif (num2 >= num1) and (num2 >= num3):
        largest = n2
else:
      largest = n3

print("The largest
of",n1,n2,"and",n3,"is",largest)
```

Do we have to change programs for different inputs?
What will happen if we input decimal values?
What happens if we input the same values?

An algorithm is a step by step instruction for carrying out an operation or solving a problem.
Find a number by guessing.

```
print("Think of a number between 0 - 1000")
print(“press enter to begin”)

nmin = 0
nmax = 1000

while nmin + 1 !=nmax:
    division
    guess = (nmin +nmax) // 2
    resp = input("Is your number less than " +
str(guess) + "? y/n ")
    if resp == "y":
       nmax = guess
    elif resp == "n":
       nmin = guess

print("Your number is " + str(nmin))
```

Euclid Greatest Common Divisor Algorithm

```
#non recursive
def gcd(p, q) :
        while (q != 0) :
            temp = q
            q = p % q
            p = temp

        return p
#recursive
def gcdr(p, q):
        if (q == 0) :
            return p
        else:
            p=p % q
            return gcd(q,p )

print(gcd (9,3))
print(gcdr (9,3))
```

Palindrom

Palindrome is a word, phrase, or sequence that reads the same backward, e.g. pop, *madam* or *nurses run*.

```
def palindrome(n):
    if n == 0:
        print(n)
    else:
        print(n)
```

```
        palindrome(n - 1)
        print(n)

palindrome(3)

text = input("Please enter some text: ")
words = text.lower().split(" ")
counts = {}

for word in words:
    if word in counts:
        counts[word] = counts[word] + 1
    else:
        counts[word] = 1

for word in counts:
    print("{}: {}".format(word, counts[word]))
```

Problems

Concatenate all elements in a list into a string and return it.

Define a function that computes the length of a given list or string. do not use the Python len() function built in.

Accept a sequence of comma-separated numbers and generate a list with those numbers.

Accept a filename from the user and print the extension of the file.
abc.txt should print txt

Display the first and last colors from the following list.
color_list = ["Red","Green","White" ,"Black"]

Accept an integer (n) and compute the value of n+nn+nnn.
if n is 5 result is 615

Write a program to print all elements of [4, 44, 2, 3, 5, 7, 13, 3, 34, 25, 19]
That is less than 5.

Make a two-player Rock-Paper-Scissors game. ask inputs, compare them and print out a message to the winner, repeat.
Rules: Rock beats scissors, Scissors beats paper, Paper beats rock.

Ask the user for a number and determine whether the number is prime or not.

Ask the user for a number and determine whether the number is a square number or not.

Ask the user for a number and determine whether the number is a triangle number or not.

Write a program to compute the factorial of a given number.

Generate 10 random numbers n, to create a dictionary that contains (n, n*n)

Write a program which accepts a sequence of space separated numbers and generates a list and a tuple which contains every number.

Array functions

reversed() function

```
a=range(1, 11)
print (a)
b= reversed(a)
print(b)
```

bytearray()
Returns a mutable array of given byte size

```
x = bytearray(3)
print(x) # x = bytearray(3)
y = bytearray(“Hello”)
print(x) #
```

enumerate()
Converts a tuple to a iterable list

```
x = ('A', 'B', 'C') #tuple
y = enumerate(x)
z=list(y)
print(z) # # [(0, 'A'), (1, 'B'), (2, 'C')]
print(z[0][1]) # A
```

reversed()
Returns a reverse iterator

```
a = ["a", "b", "c"]
b = reversed(a)
print(list(b)) # [“c”, “b”, “a”]
```

max()
Returns largest element

```
x = max(15, 10, 12, 7)
print(x) #15
```

min()
Returns smallest element

```
x = max(15, 10, 12, 7)
print(x)  # 10
```

List example

Python list elements can be any data type.

```
myList=[1,2,3,4,5,6]
print(myList[0], myList[5])

myList[0]=100
print(myList[0])

myList[1]="Yasa"
print(myList[1])
```

List Advanced

List are arrays. It means list of values.
A List is read and write ready which means mutable.
The elements can have different types.

```
P2=[2,4,"kandy",4.5]
print(P2[0])
P2[0]=12
print(P2[0])
```

Lists methods

append
Adds an item to the end of a list

```
L=[6,7,9]
L.append(5)
```

insert
Inserts an item at the nth position

```
L=[2,3,4]
n=1
L.insert(n,6)
```

Del
Deletes the item in the nth position

```
L=[3,4,5]
del L[1]
```

Pop

Removes and returns the last item

```
L=[2,3,4]
L.pop()
```

Remove

Removes the first occurrence of item

```
L=["A", "B","C"]
L.remove("B")
```

Index

Returns the index of the first occurrence of `an item`

```
L=["A", "B","C"]
L.index("C")
```

Count

Returns the number of occurrences of item

```
L=["P", "A", "A" , "D"]
L.count("A")
```

Sort

Returns sorted a list

Reverse

Returns a list in reverse order

```
L.reverse()
```

```
list1 = [1,23, 'Ruwi',7]
```

```
print (list1)      # Prints complete list
print (list1[0])   # Prints first element
print (list1[1:3]) # Prints from 2nd to 3rd
print (list[2:])   # Prints from 3rd element

list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']
print (list) # Prints complete list
print (list[0]) # Prints first element
print (list[1:3]) # Prints elements 2 to 4
print (list[2:]) # Prints elements from 3rd
print (tinylist * 2) # Prints list two times
print (list + tinylist) # Prints joined lists
```

sum() returns sum of elements

```
a = (1, 2, 3, 4, 5)
b = [1, 2, 3, 4, 5]
print(sum(a)) # 15
print(sum(b)) #15
```

Sorting Lists

Difference between Sort and Sorted

sort() performs sorting in the list and does not return anything . sorted() returns the sorted list.

```
L1=[5,9,7,11,2]
```

```
L2=sorted(L1)

print("L1 : ", L1) # [5,9,7,11,2]
print ("L2 : ", L2) # [2, 5,7,9, 11]

L3=L1.sort()
print("L3 : " , L3) # None
print("L1 : ", L1) # [2, 5,7,9, 11]
```

Write programs

Take two lists
a = [4, 4, 2, 3, 5, 8, 13, 11, 34, 55, 19]
b = [4, 2, 3, 7, 5, 6, 7, 8, 9, 10, 11, 12, 13]

and write a program that returns a list that contains only elements in both lists

Write programs

Write a program that takes a list of numbers (for example, a = [25, 12, 16, 20, 35]) and makes a new list of only the first and last elements of the given list.

Write a program to find numbers between between 2000 and 3200 which are divisible by 7 but not by 5,

Write a Python program to remove a key from a dictionary.

Write a Python program to get the maximum and minimum value in a dictionary.

Write a Python program to remove duplicates from Dictionary.

Write a Python program to create a dictionary from a string

Write a Python program to match key values in two dictionaries. Go to the editor
Sample dictionary: {'key1': 1, 'key2': 3, 'key3': 2}, {'key1': 1, 'key2': 2}

Python Files

We can manipulate files using a file object.

Files can be opened on read, write, append and binary modes

Opening Files to write

```
f = open("test.txt", "w")
f.write("Creted a file")
f.close()
```

Append to file

```
f = open("test.txt", "a")
f.write("Creted a file")
f.close()
```

Open and Read file

```
f = open("test.txt", "r") # open file
print (f.name) # output file name
print(f.read())
f.close()
```

Open a file for writing

```
fo = open("test.txt", "w+")
fo.write( "Python is cool\n")
fo.close() #Close file
```

Open a file for writing

```
fo = open("test.txt", "a")
```

```
fo.write( "Python is cool 2\n")
fo.close() #Close file
```

Open file in binary mode

```
f=open("test.txt","rb")
fileContent = f.read()
print(fileContent)
f.close()
```

Reading Binary mode

```
f = open("sheep.jpg", 'rb')
while True:
    piece = f.read(1024)
    if not piece:
        break
    print(piece)
f.close()
```

Open CV

```
pip install opencv-python

import sys # to access the system
import cv2
img = cv2.imread("sheep.jpg",
cv2.IMREAD_ANYCOLOR)

while True:
    cv2.imshow("Sheep", img)
```

```
    cv2.waitKey(0)
    sys.exit() # to exit all the processes

cv2.destroyAllWindows() # destroy all windows
```

file opening modes

```
r     read only.
r+    reading and writing. .
w     writing only.
w+    writing and reading.
a     appending.
a+    appending and reading.

rb    read only in binary.
rb+   reading and writing in binary.
wb    writing only in binary .
wb+   writing and reading in binary.
ab    appending in binary.
ab+   appending and reading in binary .
```

r
Open a file for reading only.
The file pointer is placed at the beginning of the file.
This is the default mode.

r+

Opens a file for both reading and writing.
The file pointer will be at the beginning of the file.

w
Open a file for writing only.
Overwrites the file if the file exists. If the file does not exist,
creates a new file for writing.

w+
Opens a file for both writing and reading.
Overwrites the existing file if the file exists.
If the file does not exist, create a new file for reading and writing.

a
Open a file for appending.
The file pointer is at the end of the file if the file exists.
That is, the file is in the append mode.
If the file does not exist, it creates a new file for writing.

a+
Opens a file for both appending and reading.

The file pointer is at the end of the file if the file exists.
The file opens in the append mode. If the file does not exist,
it creates a new file for reading and writing.

rb
Opens a file for reading only in binary format.
The file pointer is placed at the beginning of the file.
This is the default mode.

rb+
Opens a file for both reading and writing in binary format.
The file pointer will be at the beginning of the file.

wb
Opens a file for writing only in binary format.
Overwrites the file if the file exists. If the file does not exist, create a new file for writing.

wb+
Opens a file for both writing and reading in binary format.

Overwrites the existing file if the file exists.
If the file does not exist, create a new file for reading and writing.

ab
Opens a file for appending in binary format.
The file pointer is at the end of the file if the file exists. That is, the file is in the append mode.
If the file does not exist, it creates a new file for writing.

ab+
Opens a file for both appending and reading in binary format.
The file pointer is at the end of the file if the file exists.
The file opens in the append mode. If the file does not exist,
it creates a new file for reading and writing.

Read text lines from a file

```
f = open('test.txt', 'r')
Lines = f.readlines()
count = 0
# Strips the newline character
```

```
for line in Lines:
    count += 1
    print(count, line)
```

Reading Charactors

```
fo = open("test.txt", "r+") #
str = fo.read(6) # read 6 characters
print (str) # output 'Python'
fo.close() # Close file
```

File Positions

tell() method gives current position in file.

```
fo = open("test.txt", "r+")
str = fo.read(6)
print (str)
# Check current position
position = fo.tell()
print (position)
str = fo.read(3)
print (str)
position = fo.tell()
print (position)
# Reposition pointer to beginning
position = fo.seek(0, 0)
str = fo.read(6)
print (str)
fo.close()
```

Read Lines

```
# Open a file
fo = open("myfile.txt", "rw+")
line = fo.readline()
print (line)
line = fo.readline(5) # read 5 characters from
next line
print (line)
# Close opend file
fo.close()

fo = open("myfile.txt", "rw+")
# read all lines to a list till EOF
lines = fo.readlines()
print (lines)
# Close opend file
fo.close()
```

Count frequency of words in a file

```
file = open("word.txt", "r")
wordcount={}
for word in file.read().split():
    if word not in wordcount:
        wordcount[word] = 1
    else:
        wordcount[word] += 1
```

```
for k,v in wordcount.items():
    print (k,v)
file.close();
```

Count frequency of characters is a file.

```
file = open("anyfile.txt", "r", errors='ignore')
charcount={} #dictionary to hold char counts
validchars="abcdefghijklmnopqrstuvwxyz"

for i in range(97,123): # lowercase range
    c=(chr(i)) # the chars a-z
    charcount[c]=0 # initialize count

for line in file:
    words=line.split(" ") # line into words
    for word in words:  # words into chars
      chars=list(word) #convert word to char list
      for c in chars:  # process chars
          if c.isalpha():  # only alpha allowd
              if c.isupper():
                  c=c.lower()
              if c in validchars:
                  charcount[c]+=1

print(charcount) # print list
file.close() # close file
```

Disk Operations

```
#import os library
```

```
import os
# Rename myfile.txt to  myfile2.txt
os.rename( " myfile.txt", "myfile2.txt" )
os.remove("myfile.txt") # remove myfile.txt
os.mkdir("newdir") # Create a directory
# Change a directory to "/home/newdir"
os.chdir("/home/newdir")
#return current working directory
os.getcwd()
os.rmdir( "/home/newdir" ) #remove directory
```

Python bitwise shift operators.

x << y

Returns x with the bits shifted to the left by y places. the new bits on the right-hand-side are zeros. This is the same as multiplying x by 2**y.

5<<2 yeilds 20

x >> y

Returns x with the bits shifted to the right by y places. This is the same as dividing x by 2**y.

5 >> 2 yields 1

x & y
Does a "bitwise and". Each bit of the output is 1 if the corresponding bit of x AND of y is 1, otherwise it's 0.

x | y
Does a "bitwise or". Each bit of the output is 0 if the corresponding bit of x AND of y is 0, otherwise it's 1.

~ x

Returns the complement of x - the number you get by switching each 1 for a 0 and each 0 for a 1. This is the same as -x - 1.

x ^ y
Does a "bitwise exclusive or". Each bit of the output is the same as the corresponding bit in x if that bit in y is 0, and it's the complement of the bit in x if that bit in y is 1.

Just remember about that infinite series of 1 bits in a negative number, and these should all make sense.

Write programs

Write a Python program to get file creation and modification date/times.

Write a Python program to sort files by date in a directory.

Write a Python program to check if a file path is a file or a directory.

Write a Python program to extract the filename from a given path.

Write a Python program to get the size of a file.

Write a Python program to copy a file to another file.

Write a Python program to save user inputs.

Create a program to generate two files with numbers separated by space and find the numbers that are overlapping.

Define a function sum() and a function multiply() that sums and multiplies (respectively) all the numbers in a list of numbers. For example, sum([1, 2, 3, 4]) should return 10, and multiply([1, 2, 3, 4]) should return 24.

Write a function named sleep_in which takes two parameters: weekday and vacation. The parameter weekday is True if it is a weekday, and the parameter vacation is True if we are on vacation. We sleep in if it is not a weekday or we're on vacation. Return True if we sleep in.

```
sleep_in(False, False) → True
sleep_in(True, False) → False
sleep_in(False, True) → True
```

We have two monkeys, a and b, and the parameters a_smile and b_smile indicate if each is smiling. We are in trouble if they are both smiling or if neither of them is smiling. Return true if we are in trouble.

Given an int n, return the absolute difference between n and 21, except return double the

absolute difference if n is over 21.

```
diff21(19) → 2
diff21(10) → 11
diff21(21) → 0
```

Talking parrot

We have a loud talking parrot. The "hour" parameter is the current hour in the range 0 to 23. We are in trouble if

the parrot is talking and the hour is before 7 or after 20. Return True if we are in trouble.

```
parrot_trouble(True, 6) → True
parrot_trouble(True, 7) → False
parrot_trouble(False, 6) → False
```

Given an int n, return True if it is within 10 of 100 or 200. Note: abs(num) computes the absolute value of a number.

```
near_hundred(93) → True
near_hundred(90) → True
near_hundred(89) → False
```

Write a function sum that takes a list of numbers and returns the sum of all the numbers in the list.

Write a function product that takes a list of numbers and returns the product of all the numbers in the list.

What should the product of the empty list be?
Can you optimize this if you find a zero in the list?

eval() Runs Code Within Program

```
x = 'print(55)'
y = 'print(4+3)'
eval(x) # 55
eval(y) # 7
```

exec() Executes the specified code

```
x = 'a = 3\nb=a*2\nprint(b)'
exec(x) # 6
```

hash() Returns hash value

```
print(hash(18)) #18
print(hash(18.1)) #230584300921372690
```

hex() Integer to Hexadecimal

```
x = hex(255)
print(x) # oxff
```

oct() integer to octal

```
x = oct(15)
print(x) #0o17
```

Error Handling

What is an Exception?

An exception is an error that happens during execution of a program. When that error occurs, Python generates an exception that we can handle to avoid program crash.

```
import sys

a=1
b="S"
try:
    c=a+b
except:
    print(sys.exc_info())
    print("---")
finally:
    print("====")

print("End")
```

Nested Dictionary

```
people = {1: {'name': 'J', 'age': '28'},
2: {'name': 'M', 'age': '21'}}

print(people)
#access values
print(people[1]['name'])
print(people[1]['age'])
people[3] = {}
people[3]['name'] = 'N'
people[3]['age'] = '19'
people[3]['city'] = 'Kandy'
print(people)
```

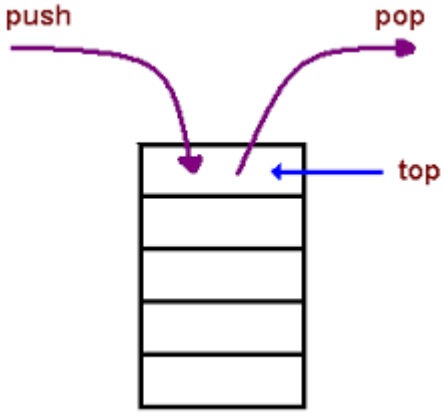
Matrix

```
# a is 2-D matrix with integers
a = [[8,7,8,9,5],
     [8,5,8,7,8],
     [8,8,8,9,9]]

print(a[0])
print(a[0][1])
print(a[1][2])
```

More Algorithms

Stack

Stack uses a list with LIFO or Last In First Out model.




```
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop()

    def is_empty(self):
        return (self.items == [])
```

A stack is a generic data structure, which means that we can add any type of item to it.

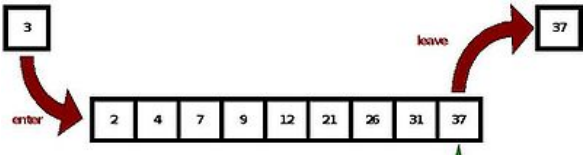
```
s = Stack()
s.push(54)
s.push(45)
s.push("+")
```

We can use is_empty and pop to remove and print all of the items on the stack:

```
while not s.is_empty():
    print(s.pop(), end=" ")
```

Queue

Python queue works like day to day queue .We add items to the end of list and remove items from the front. It is called the First-in-First out method.




```
class Queue:

  def __init__(self):
      self.queue = list()

  def add(self,val):
```

```
  # Insert method to add element
      if val not in self.queue:
          self.queue.insert(0,val)
          return True
      return False

  def size(self):
      return len(self.queue)

  def remove(self):
      if len(self.queue)>0:
          return self.queue.pop()
      return ("No elements in Queue!")
Q = Queue()
Q.add("A")
Q.add("B")
Q.add("C")
print(Q.size())
print(Q.remove())
print(Q.remove())
print(Q.remove())
print(Q.size())
```

Explain output

Binary Search

Given a sorted list of strings, we need to find the index of an

element of a given value. We can do it with a serial search

```
def search(xs, x):
    for i in range(len(xs)):
        if xs[i] == x:
            return i
    return None
L=[3,5, 7, 9, 11, 2, 34, 12, 9]
n=search (L, 9)
print(n)
```

This method is not very efficient.

```
def search(L, target):
  min = 0
  max = len(L)-1
  avg = int((min+max)/2)

  print (L, target, avg)
  while (min < max):
    if (L[avg] == target):
      return avg
    elif (L[avg] < target):
      return avg + 1 + search(L[avg+1:], target)
    else:
      return search(L[:avg], target)

  return avg

L=[3,5, 7, 9, 11, 2, 34, 12, 9]
n=search(L, 9)
print(n)
```

Merge sort

Merge Sort is a divide and conquer algorithm.
It divides a list into two halves, then recursively calls itself for the two halves and merge sorted halves.

```
def mergeSort(L1):
    print("Splitting ",L1)
    if len(L1)>1:
        mid = len(L1)//2
        left = L1[:mid]
        right = L1[mid:]

        mergeSort(left)
        mergeSort(right)

        i,j,k=0,0,0

        while i < len(left) and j <
len(right):
            if left[i] < right[j]:
                L1[k]=left[i]
                i=i+1
            else:
                L1[k]=right[j]
                j=j+1
            k=k+1
```

```
        while i < len(left):
            L1[k]=left[i]
            i=i+1
            k=k+1

        while j < len(right):
            L1[k]=right[j]
            j=j+1
            k=k+1
    print("Merging ",L1)

L1 = [54,26,93,17,77,31,44,55,20]
mergeSort(L1)
print(L1)
```

You can implement merge sort using the merge function from the previous exercise. The result of this function should be the same as the result of the sorted function:

```
def merge_sort(xs):
    # A sorted copy of xs
```

Hint: In Python, if you want to split an array in two you can do:

```
first_half = xs[:(len(xs) // 2)]
second_half = xs[(len(xs) // 2):]
```

Selection sort

The selection sort improves on the bubble sort by making only one exchange for every pass through the list

```
def selectionSort(L):
   for fill_slot in range(len(L)-1,0,-1):
       positionOfMax=0
       for location in range(1,fill_slot+1):
           if L[location]>L[positionOfMax]:
               positionOfMax = location

       temp = L[fill_slot]
       L[fill_slot] = L[positionOfMax]
       L[positionOfMax] = temp

L = [54,26,93,17,77,31,44,55,20]
selectionSort(L)
print(L)
```

Explain how Selection sort works?

List generation

```
import random
letters = []
```

```
for l in 'Python':
    letters.append(l)
print(letters)

list1 = [ 0 for i in range(10)]
print(list1)

def rnd():
    return random.randint(0,9)

list2 = [ [rnd() for i in range(6)] for j in
range(4)]
print(list2)

list1 = [y for y in range(100) if ((y % 2 ==
0) and (y % 5 == 0))]
print(list1)
```

Flattening lists

Lists can contain lists,
take a list [[a, b, c, ...], [d, e, f, ...], ...] #and flatten it to [a, b, c, ... , d, e, f,

```
def flatten(xs):
    # return a flattened copy of the list
    flattened = []
    for x in xs:
        flattened += x
    return flattened

flatten([], [])
flatten([[1, 2], [3, 4]])
flatten([[], [1, 2], [3, 4])
```

Pairwise sum

#Given two lists xs and ys, return a new list zs
where each element is the sum of the corresponding elements # in xs and ys,

`sum2([1, 2, 3], [10, 11, 12]) == [11, 13, 15].`
You may assume that the lists are the same length.

```
def sum2(xs, ys):
    zs = [] # new list
    for i in range(0, len(xs)):
        zs.append(xs[i] + ys[i])
    return zs

sum2([], [])
sum2([1, 2], [3, 4])
sum2([-10, 10, 20], [10, -10, -20])
```

#mean of a list

```
def mean(xs):
```

```
    # return the mean of the list of elements
   s = 0
   for x in xs:
        s = s + x
    return s / len(xs)

mean([1, 2])
mean(range(0, 10))
mean([-1, 0, 1])
```

Divisibility

#Given a number n,
#compute a list of all the integers x
#divisible by 3 or 5 that are 0 <= x < n.
#example, three_or_five(10) == [0, 3, 5, 6, 9].

```
def threes_or_fives(n):
    xs = []
    for x in range(0, n):
        if x % 3 == 0 or x % 5 == 0:
            xs.append(x)
    return xs

threes_or_fives(0)
threes_or_fives(3)
threes_or_fives(5)
threes_or_fives(6)
threes_or_fives(10)
```

Filtering a list

#Given a list of numbers, return a copy of the list that only
#includes the even numbers.

```
def filtered_even(xs):
    # Return a copy of the list that only includes
even number
    ys = []
    for x in xs:
        if x % 2 == 0:
            ys.append(x)
    return ys

filtered_even([])
filtered_even([0])
filtered_even([0, 2, 3])
filtered_even([10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
```

More
https://github.com/shilpasayura/algohack/tree/master/code-1mwtt

Counting frequency items in a list

```
l = ['1','3','5','5','5','3','6','5']
print(l.count('5')) #
```

#finding frequency of all elements

```
s = [] # empty list for unique elements
for x in l:
       if x not in s:
          s.append(x) # add to list s

print(s) # unique elements
#find frequency of elements in list
for x in s :
   print(x, " - ", l.count(x))
```

Sorting lists

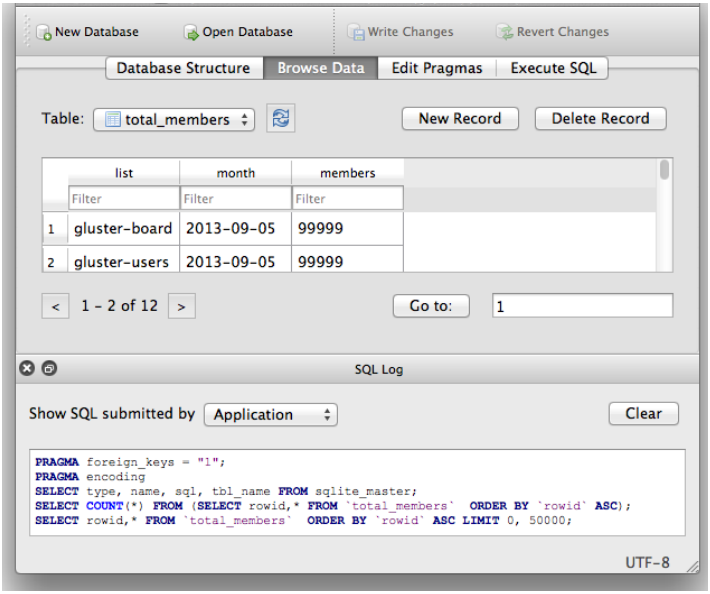
For the second set of the exercises you'll need to be able to sort a list into ascending order. To do this in Python you need to use the sorted function:

```
b = [ "J", "P", "G", "R" ]
b2 = sorted(b) # == [ "G", "J", "P", "R" ]

nums = [13, 56, 26, 2, 12, 12, 2, 4]
nums2 = sorted(nums) # [2,2,4,12,12,13,26, 56]
```

Database Access

sqlite3 is part of the Python Standard Library.
sqlite_file can be anywhere on our disk.

You can use DB browser to create and manipulate SQLite databases. https://sqlitebrowser.org/




```
import sqlite3

try:
    # connect if available else create
    conn= sqlite3.connect('test.db')
    print (conn)
```

```
    print ('connection established')

    #create cursor object
    cursor =conn.cursor()

    sql1="""
    CREATE TABLE users (
    uid INTEGER PRIMARY KEY AUTOINCREMENT
    NOT NULL, name text);
    """
    print ("Table created")

    sql2="INSERT INTO users (name)
VALUES('Ama');"
    sql3="INSERT INTO users (name)
VALUES('Bima');"
    cursor.execute(sql2)
    cursor.execute(sql3)
    conn.commit() # write to db permanently
    sql4="SELECT * FROM users;"
    cursor.execute(sql4)
    # process returned record set
    for row in cursor.fetchall():
        # row is a tuple
        #print(row, row[2])
        uid, name = row
        print(uid, name)

     conn.close()
```

```
except (sqlite3.Error,) as e:
    print(e)
finally:
    print("end")
```

MySQL

When connecting to mySQL database we need to give host, database, user and password to connect to the database.

```
import MySQLdb # mySQL module
host="localhost"
db="mysql_database"
user="mysql_userid"
password="mysql_user_password"
db = MySQLdb.connect(host,user, passwd,db)
```

User defined database errors

```
import sqlite3

class MyError(Exception):
     def __init__(self, value):
         self.value = value
try:
    conn= sqlite3.connect('database56.sqlite')
    print (conn)
    print ('connection established')
except sqlite3.Error as e:
```

```
    print ('Error message:', e.value)
    raise MyError('Could not connect to db: '
+ e.value)
```

Most used SQLite Commands

Select - retrieves data from a table

```
SELECT * FROM table;

SELECT age, name FROM student;
```

Where - specifies a criteria for for retrieving data

```
SELECT age, name FROM student WHERE age > 10;

SELECT age, name FROM student WHERE age > 10
AND age < 20;
```

Order - sorts the results returned.

```
SELECT name, age FROM student ORDER BY age
DESC;

SELECT name, age FROM student ORDER BY name
ASC, age DESC
```

Join - combines related data from one or more tables.

```
SELECT age, name, height FROM student LEFT
JOIN heights USING (name);
```

Alias

Alias temporarily renames a table

```
SELECT A.age FROM student as A;

SELECT age AS student_age FROM student;
```

Union

Allows you to append rows to each other.

```
SELECT age, name FROM customers
UNION
SELECT age, name FROM staff;
```

Insert

```
INSERT INTO student(name, age) VALUES('Ama',
21);
```

Update

Change specific rows.

```
UPDATE student SET name = 'Bima', age = 22;
UPDATE student SET name = 'Ama', age = 21
WHERE name = 'Bima';
```

Delete - Delete records

```
DELETE FROM student WHERE name = 'Ama';
```

Create Table

```
CREATE TABLE student (
  id INTEGER
  name TEXT,
  age, INTEGER,
  PRIMARY KEY(id)
);
```

Alter Table - Modifies table structure

```
ALTER TABLE student ADD height integer;
```

Drop Table

remove table!

```
DROP TABLE student;
```

Python List Comprehension

```
my_list = []
for x in range(10):
  my_list.append(x * 2)

print(my_list)

Generate lists
nums = [1, 2, 3, 4, 5]
letters = ['A', 'B', 'C', 'D', 'E']

#this comprehension combines two lists
nums_letters = [[m,n ] for m in nums for n in letters]
print(nums_letters)



iter_string = "some text"
comp_list = [x for x in iter_string if x !=" "]
print(comp_list)



dict = {x:chr(65+x) for x in range(1, 11)}
type(dict)
print(dict)
```

Iterators

Iterator in Python is simply an object that can be iterated upon. An object which will return data, one element at a time. list, tuple, string etc. are iterables.

```
# define a list
my_list = [4, 7, 0, 3]

# get an iterator using iter()
my_iter = iter(my_list)

# iterate through it using next()
print(next(my_iter))
print(next(my_iter))
print(my_iter.__next__())
print(my_iter.__next__())
# This will raise error, no items left
next(my_iter)
```

Generator Expressions

```
gen_exp = (x ** 2 for x in range(10) if x % 2 == 0)
for x in gen_exp:
  print(x)
```

Python generators are a simple way of creating iterators. All the work we mentioned above is automatically handled by generators in Python.

Both yield and return will return some value from a function.

Once the function yields, the function is paused and the control is transferred to the caller.

```
def gen():
    yield 'A '
    yield 'B'
    yield 'C'

g = gen()
print (g)
print (''.join(g))

# A simple generator function
def my_gen():
    n = 1
    print('This is printed first',1)
    # Generator function contains yield statements
    yield n

    n += 1
    print('This is printed second',2)
    yield n
```

```
    n += 1
    print('This is printed at last',3)
    yield n


a = my_gen()

# We can iterate through the items using next().
next(a)
# Once the function yields, the function is paused and
the control is transferred to the caller.
print("1 done")
# Local variables and their states are remembered
between successive calls.
next(a)
print("2 done")
next(a)
print("3 done")
```

Generator that reverses a string.

```
def rev_str(my_str):
    length = len(my_str)
    for i in range(length - 1, -1, -1):
        yield my_str[i]


# For loop to reverse the string
```

```
for char in rev_str("hello"):
    print(char)
```

Generator Expression

```
 Initialize the list
my_list = [1, 3, 6, 10]

# square each term using list comprehension
list_ = [x**2 for x in my_list]

# same thing can be done using a generator expression
# generator expressions are surrounded by parenthesis
()
generator = (x**2 for x in my_list)

print(list_)
print(generator)

# Initialize the list
my_list = [1, 3, 6, 10]

a = (x**2 for x in my_list)
print(next(a))
print(next(a))
print(next(a))
print(next(a))
next(a)
```

https://www.programiz.com/python-programming/generator

Python Modules

Check whether a given value is in a group of values.

Concatenate all elements in a list into a string and return it.

Define a function that computes the length of a given list or string. do not use the Python len() function built in.

We can develop our own code libraries to modularize our program. To do so we distribute our code into multiple python files. They are our modules.

Each file can import other files as modules and call their functions and access variables as given below.

```
modulename.functionname()
modulename.variable

#hellomodule.py

def hello(name):
  print("Hello, " + name)

A=40
P1 = {
  "name": "Ama",
  "age": 21,
  "country": "Sri Lanka"
```

```
}

#main.py

import hellomodule

hellomodule.hello("Ama")
print(hellomodule.A)

name = hellomodule.P1["name"]
age = hellomodule.P1["age"]

print(name, age)
```

Import only the P1
Dictionary from the module:

```
from hellomodule import P1

print (P1["name"])

#this gives an error
hellomodule.hello("Ama")
```

Module Search Path

When python executes `import mymod` statement,
It searches mymod.py in current directory and the directories in the Python PATH environment variable

Installation directories,

The search path is accessible from `sys.path`

Module Alias

```
import mod as mymod
mymod.a
```

dir() Returns built in name spaces

Python __main__

```
When Python interpreter reads a source file,
it does two things:
1. Sets few special variables like __name__
2. Executes all code top down in the file.

__name__ is main when we run it directly.
if another program imports it, the code to
test the module will not run.
```

```
print("before import")
import math

print("before functionA")
def functionA():
    print("Function A")
```

```
print("before functionB")
def functionB():
    print("Function B
{}".format(math.sqrt(100)))

print("before __name__ guard")
if __name__ == '__main__':
    functionA()
    functionB()
print("after __name__ guard")
```

Python Packages

When we are developing an application with many modules, we can organize them as packages.

Packages are hierarchical structured module namespaces using dot notation. So we can avoid collisions between global variable names and module names.

we place `mod1.py` and `mod2.py` in the package directory. Then call them as.

```
import pakage.mod1, pkg.mod2

pakage.mod1.anyfunction()
```

Package Initialization

If a file named __init__.py is present in a package directory, when a package or a module in the package is imported.

__init__.py

```
print(f'Invoking __init__.py for {__name__}')
A = ['A', 'B', 'C']
```

now when mod1 package is imported, global list A is initialized:

We can use __init__.py to automatically import modules.

Importing from a package

from <package_name> import *

Subpackages

Packages can contain nested subpackages.

```
import pkg.sub_pkg1.mod1
```

Built in sys and time packages

```
import sys
```

```
import time

def slowprint(s):
    for c in s + '\n' :
        sys.stdout.write(c)
        sys.stdout.flush()
        time.sleep(5. / 100)

slowprint(''' \033[93m
[01] python
[02] python2
[03] python-dev
[04] python3
[05] php
[23] coreutils ''')
slowprint('''\033[96m
This Command for access Storage in Termux
[00] termux-setup-storage''')
```

PIP

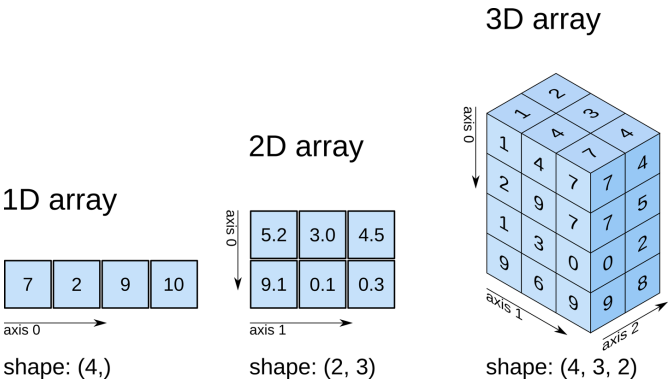
Pip is a python package manager. Allows you to download and install python modules.
Download get-pip.py to your python folder
Open a command prompt
Run the following command:

```
python get-pip.py
```

Pip is now installed!

NumPy

Numpy is a multidimensional array library.
All elements are the same type.
They are indexed by a tuple of positive integers.
In NumPy dimensions are called axes.



Coordinates of a point in 3D space is [1, 2, 1] and has 1 axis.
That axis has 3 elements in it.
So we say it has a length of 3.

Following array has 2 axes.
[[1., 0., 0.],
 [0., 1., 2.]]
first axis has a length of 2
The second axis has a length of 3.

NumPy's array class is called ndarray.

ndarray.ndim
the number of axes or dimensions of the array.

ndarray.shape
A matrix with n rows and m columns, shape will be (n,m).

ndarray.size
the total number of elements of the array. This is a product of the elements of shape.

ndarray.dtype
an object describing the type of the elements in the array.

ndarray.itemsize
the size in bytes of each element of the array.

```
import numpy as np
a = np.arange(15).reshape(3, 5)
a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
a.shape
a.ndim
a.dtype.name
a.itemsize
a.size
type(a)
```

```
b = np.array([6, 7, 8])
b
type(b)
```

Array Creation
There are several ways to create arrays.

```
import numpy as np
a = np.array([2,3,4])
a.dtype

b = np.array([1.2, 3.5, 5.1])
b.dtype
dtype('float64')

a = np.array(1,2,3,4)    # WRONG
a = np.array([1,2,3,4])  # RIGHT
```

array transforms sequences of sequences into two-dimensional arrays, sequences of sequences of sequences into three-dimensional arrays, and so on.

```
b = np.array([(1.5,2,3), (4,5,6)])
B
```

function zeros creates an array full of zeros

```
np.zeros( (3,4) )
```

```
np.ones( (2,3,4), dtype=np.int16 )
# dtype can also be specified
np.empty( (2,3) )

# uninitialized, output may vary
```

functions

```
np.arange( 10, 30, 5 )

np.arange( 0, 2, 0.3 )# it accepts float

from numpy import pi
np.linspace( 0, 2, 9 ) # 9 numbers from 0 to 2
# evaluate function at lots of points
x = np.linspace( 0, 2*pi, 100 )
f = np.sin(x)
```

Aggregate functions

np.sum	Compute sum of elements
np.prod	Compute product of elements

np.mean Compute mean of elements
np.mode Compute mode of elements
np.median Compute median of elements
np.std Compute standard deviation
np.var Compute variance

np.percentile Compute rank of elements

```
import numpy as np
from scipy import stats
a=np.array([[6,1,2],[3,3,4],[7,4,5]])
print(a)
print("Sum :", np.sum(a))
print("Product",np.prod(a))
print ("Mean :", np.mean(a))
print ("Mode :", stats.mode(a))
print("Median", np.median(a))
print("Standard Deviation :", np.std(a))
print("Variance :", np.var(a))
print("Percentile :", np.percentile(a, 50))
```

What will happen if we use [6,1,2,3,3,4,7,4,5] array?

Python Imaging
Pillow module helps image manipulation in Python.

```
$ pip install Pillow
```

```
from PIL import Image, ImageFilter
#Read image
im = Image.open( 'image.jpg' )
#Display image
im.show()

#Applying a filter to the image
im_sharp = im.filter( ImageFilter.SHARPEN )
#Saving the filtered image to a new file
im_sharp.save( 'image_sharpened.jpg', 'JPEG' )

#Splitting the image into its respective
bands, i.e. Red, Green,
#and Blue for RGB
r,g,b = im_sharp.split()

#Viewing EXIF data embedded in image
exif_data = im._getexif()
```

see examples of the Pillow library

Open Source Computer Vision

Open Source Computer Vision known as OpenCV. It is more advanced than pillow.
it uses numpy and cv2 modules.

```
$ pip install numpy
```

```
import cv2
#Read Image
img = cv2.imread('testimg.jpg')
#Display Image
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()

#Applying Grayscale filter to image
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#Saving filtered image to new file
cv2.imwrite('graytest.jpg',gray)
```

See OpenCV python examples

Python Problems

Define a class which has at least two methods:
getString: to get a string from console input
printString: to print the string in upper case.

Write a Python class named Rectangle constructed by a length and width and a method which will compute the area of a rectangle.

Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

Write a Python class to find the three elements that sum to zero from a set of n real numbers. - Go to the editor
Input array : [-25, -10, -7, -3, 2, 4, 8, 10]
Output : [[-10, 2, 8], [-7, -3, 10]]

Write a Python class to find a pair of elements from a given array whose sum equals a specific target number. - Go to the editor
Input: numbers= [10,20,10,40,50,60,70], target=50
Output: 3, 4

Write a program that accepts a space separated words and prints the words in a comma-separated sequence after sorting them alphabetically.

Write a program that accepts sequence of lines as input and prints the lines after making all characters in the sentence capitalized.

Write a program that accepts a sequence of whitespace separated words as input and prints the words after removing all duplicate words and sorting them alphanumerically.

Write a program which accepts a sequence of comma separated 4 digit binary numbers as its input and then check whether they are divisible by 5 or not. The numbers that are divisible by 5 are to be printed in a comma separated sequence.

Write a program, which will find all such numbers between 1000 and 3000 (both included) such that each digit of the number is an even number.
The numbers obtained should be printed in a comma-separated sequence on a single line.

Write a program that accepts a sentence and calculate the number of letters and digits.

Write a program that accepts a sentence and calculate the number of uppercase letters and lowercase letters.

Write a program that computes the value of a+aa+aaa+aaaa with a given digit as the value of a.

Use a list comprehension to square each odd number in a list. The list is input by a sequence of comma-separated numbers.

Write a program that computes the net amount of a bank account based on a transaction log from console input. The transaction log format is shown as following:
D 100
W 200

A website requires the users to input username and password to register. Write a program to check the validity of password input by users.
Following are the criteria for checking the password:
1. At least 1 letter between [a-z]
2. At least 1 number between [0-9]
1. At least 1 letter between [A-Z]
3. At least 1 character from [$#@]
4. Minimum length of transaction password: 6
5. Maximum length of transaction password: 12

Write a program to sort the (name, age, height) tuples by ascending order where name is string, age and height

are numbers. The tuples are input by console. The sort criteria is:
1: Sort based on name;
2: Then sort based on age;
3: Then sort by score.

Define a class with a generator which can iterate the numbers, which are divisible by 7, between a given range 0 and n.

A robot moves in a plane starting from the original point (0,0). The robot can move toward UP, DOWN, LEFT and RIGHT with a given step. The trace of robot movement is shown as the following:
UP 5
DOWN 3
LEFT 3
RIGHT 2
The numbers after the direction are steps. Please write a program to compute the distance from the current position after a sequence of movement and original point. If the distance is a float, then just print the nearest integer.

Write a program to compute the frequency of the words from the input. The output should output after sorting the key alphanumerically.

Define a function which can generate and print a list where the values are square of numbers between 1 and 20 (both included).

With a given tuple (1,2,3,4,5,6,7,8,9,10), write a program to print the first half values in one line and the last half values in one line.

Write a program which accepts a string as input to print "Yes" if the string is "yes" or "YES" or "Yes", otherwise print "No".

Write a program which can filter even numbers in a list by using the filter function. The list is: [1,2,3,4,5,6,7,8,9,10].

Write a program which can map() to make a list whose elements are squares of elements in [1,2,3,4,5,6,7,8,9,10].

Write a program which can map() and filter() to make a list whose elements are squares of even numbers in [1,2,3,4,5,6,7,8,9,10].

Define a class named Circle which can be constructed by a radius. The Circle class has a method which can compute the area.

Define a class named Rectangle which can be constructed by a length and width. The Rectangle class has a method which can compute the area.

We have some email addresses in the "username@companyname.com" format, please write a program to print the user name of a given email address.

Write a program which accepts a sequence of words separated by whitespace as input to print the words composed of digits only.

Write a program to compute 1/2+2/3+3/4+...+n/n+1 with a given n input by console (n>0).

Write a program to compute:

f(n)=f(n-1)+100 when n>0
and f(0)=1

with a given n input by console (n>0).

Write a binary search function which searches an item in a sorted list. The function should return the index of the element to be searched in the list.

Please write a program to print the running time of execution of "1+1" for 100 times.

By using list comprehension, please write a program generate a 3*5*8 3D array whose each element is 0.

Using list comprehension, please write a program to print the list after removing the value 24 in [12,24,35,24,88,120,155].

Write a program which counts and prints the numbers of each character in a string input by console.

Resources
https://www.freecodecamp.org/news/the-python-handbook/

AlgoHack : Python Fast

Get into Python. Jump start fundamental Python programming and move fast to advanced Python. This book helps Python beginners to understand key concepts by exploring simple examples and doing challenges to build a strong foundation to develop Industry level Python programming skills .

Niranjan Meegammana is an award winning innovator of ICT and educator specializing in Localization, web, mobile, e learning and IoT technologies.

Vishva Kumara is an enterprise software engineer and architect specializing in local language integrated web, mobile,IoT, big data technologies..

Published by : Shilpa Sayura Foundation
ISBN : 0-670-01039-6

AlgoHack

ශිල්පසයුර Google for Education CSSL COMPUTER SOCIETY OF SRI LANKA

AlgoHack aims to teach Computer Science and Programming to young people, initiated by Shilpa Sayura Foundation, supported by GOOGLE RISE and Computer Society of Sri Lanka.