```
Invoke Python interactive Interpreter

$ python3.7
Python 3.7.5 (default, Nov 20 2019, 09:21:52)
[GCC 9.2.1 20191008] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> print('hellow world')
hellow world
>>>
>>> # single line comment
...
>>> ''' multi
... line
... comment'''
' multi\nline\ncomment'

Install pycharm IDE
===================

sudo snap install pycharm-community --classic

Program1

~/PycharmProjects/Program1$ cat firstproject.py
print("Hellow World")
print('yes')
print('hello\'s world')
print('Hello said\"You are welcome\"')

escape charactor is "\"

Variables in Python
==================

a = 6
b = 5
print(a)
print(b)
print(a * b)

name = "Peter"  strings in ' or "
na =' Paul'
print(name+' ' na)
print(name,na)

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
 Python has thirty-something keywords (and every now and
 again improvements to Python introduce or eliminate one or two):
 and    as  assert  break   class   continue
 def    del     elif    else    except  exec
 finally    for     from    global  if  import
 in     is  lambda  nonlocal    not     or
 pass   raise   return  try     while   with
 yield  True    False   None
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>


Strings and Variables in Python
===============================

a = "3"
b = '2'

print(a + b)    gives "32" string
print(int(a) + int(b))    gives 5, convert to interger

Accepting input from users in Python
====================================

name = input("What is your name\n")
print("Welcome",name)


Operators
===========

+   Addition: adds two operands     x + y
```

```
-   Subtraction: subtracts two operands     x - y
*   Multiplication: multiplies two operands     x * y
/   Division (float): divides the first operand by the second   x / y
//  Division (floor): divides the first operand by the second   x // y
%   Modulus: returns the remainder when first operand is divided by the second  x % y


Python supports the usual logical conditions from mathematics:

    Equals: a == b
    Not Equals: a != b
    Less than: a < b
    Less than or equal to: a <= b
    Greater than: a > b
    Greater than or equal to: a >= b


Python will give you an error if you skip the indentation.
The number of spaces is up to you as a programmer, but it has to be at least one.
---------------------------------------------------------------------------------

age = 17

if(age == 18):
    print("you are 18")

elif(age > 18):
    print("you are over 18")

else:
    print("enter valid number")


For loop
=========

for i in range(5):
   print(i)


While loop
============ break and continue

i = 0
while(i<5):
   print(i)
   i = i + 1

Break Statement
===============

i = 1
while(i<=100)
    print(i)
    if(i == 7):
       break
    i = i+1

while 1:
    name = input("Please enter your name...\t)
    if(name=='quit'):
        break



Continue Statement
==================

With the continue statement we can stop the current iteration, and continue with the next:

 This will start printing at 50 -

i = 1
while(i<=100):
   i = i+1
   if(i <=50):
       continue
   print(i)
```

```
String Properties
=================

https://www.w3schools.com/python/python_strings.asp


List methods[] in Python
========================
https://www.w3schools.com/python/python_ref_list.asp

Note: Python does not have built-in support for Arrays, but Python Lists can be used instead


Tuples () in Python
================

A tuple is a collection which is ordered and unchangeable. In Python tuples are written with round brackets.


Dictionaries in Python
======================

A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly
brackets, and they have keys and values.

dictionary = {"key":"value","key":"value"}

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

Functions in Python
==================

def NAME( PARAMETERS ):
    STATEMENTS

print(dir(__builtins__))   # prints all the functions.

print(help(pow))  prints help on pow function

Define a function
================

def my_function():   # def is the keyword, my_function is name of the function.
    print("hellow from a function") # what the function does


Arguments are specified after the function name, inside the parentheses.
You can add as many arguments as you want, just separate them with a comma.

def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")  # call the function, passing an arguement
my_function("Tobias")
my_function("Linus")


Modules in Python
=================

print(help("modules")) # see a list of all modules


Password validation
===================

password = 'hellopen'

for a in range(3):
    psw = input("enter password:\t")
    b = 2
    print("a is", a)
    print("b is", b)
    if(psw == password):
        print("Access Granted")
        break
    else:
        print("access denighed", b-a)
        if(b-a != 0):
            print("try again...")
        continue
================================

Write to a file
===============

tx = open('/home/dan/Desktop/first.txt', 'w')
tx.write('This is what I just wrote to my file using python')
tx.close()

Read from a file
================

a = open('/home/dan/Desktop/first.txt', 'r')
print(a.read())

  Another example
  ===============

a = open('/home/dan/Notes/Python_Master_Class/Python_Notes', 'r')
print(a.read())


Rename a file
=============

import os #     os - OS routines for NT or Posix depending on what system we're on.

print(os.rename('/home/dan/Desktop/first.txt', '/home/dan/Desktop/NewFirst.txt'))

Remove a file
=============

import os

print(os.remove('/home/dan/Desktop/NewFirst.txt'))

Create a folder
===============

import os

print(os.mkdir('/home/dan/Desktop/pythonfolder'))
print(os.getcwd())

Remove a directory
==================

import os

print(os.rmdir('/home/dan/Desktop/pythonfolder'))
print(os.getcwd())

Exception handling - How to handle errors
=========================================

a = int(input('Insert 1st number\t'))
b = int(input('insert 2nd number\t'))

try:
    c = a/b  # error if b = 0
except:
    print('you cannot divide',a,'by',b)
else:
    print(c)


Print formatting
===============
name = 'Hellow'
age = 50
grade = 71.9

print('Student Name is %s and his age is %d and his grade is %.2f'%(name, age,grade))

# %s is for string
# %d is for integer
# %f is for float
```

Function format (.format)
=========================

```
name = 'Hellow'
age = 50
grade = 71.9

print('Student Name is {}and his age is {} and his grade is {}'.format(name, age,grade))

print('Student Name is {0}and his age is {1} and his grade is {2}'.format(name, age,grade))
#                       ^ index           ^ index             ^ index of list

for i in range(1,11):
    print("{:5}{:5}{:5}".format(i,i*i,i*i*i))
#             ^ spaces for output in list
```

```
  Output:

/usr/bin/python3.7 /home/dan/PycharmProjects/Program1/firstproject.py
Student Name is Hellowand his age is 50 and his grade is 71.9
Student Name is Hellowand his age is 50 and his grade is 71.9
    1    1    1
    2    4    8
    3    9   27
    4   16   64
    5   25  125
    6   36  216
    7   49  343
    8   64  512
    9   81  729
   10  100 1000

Process finished with exit code 0
```

======================================
Object Oriented Programing in Python 8
======================================

Class
=====

```
class employee:
    def staff(self,name,salary):   # method / function. staff is the name
        self.name = name           # define name
        self.salary = salary       # define salary
        print('Employee\'s name is {} and his salary is {}'.format(name,salary))  # format method


em = employee()   # Call the class
em.staff('Paul',1000)

print(em.name)
print(em.salary)
```

```
  Output:

/usr/bin/python3.7 /home/dan/PycharmProjects/Program1/firstproject.py
Employee's name is Paul and his salary is 1000
Paul
1000

Process finished with exit code 0
```

Constructor Method
==================

```
class employee:
    def staff(self,name,salary):   # method / function. staff is the name
        self.name = name           # define name
        self.salary = salary       # define salary
        print('Employee\'s name is {} and his salary is {}'.format(name,salary))  # format method

    def __init__(self):
        print('This is a constructor method')

em = employee()   # Call the class
em.staff('Paul',1000)

print(em.name)
print(em.salary)

  Output:

/usr/bin/python3.7 /home/dan/PycharmProjects/Program1/firstproject.py
This is a constructor method  <<< prints out 1st because it calls itself
Employee's name is Paul and his salary is 1000
Paul
1000

Process finished with exit code 0


class employee:
    def __init__(self,name,salary):   # method / function. Constructor calls itself
        self.name = name           # define name
        self.salary = salary       # define salary
        print('Employee\'s name is {} and his salary is {}'.format(name,salary))  # format method



em = employee('Paul',1000)   # Call the class and pass arguements


   OUTPUT:

/usr/bin/python3.7 /home/dan/PycharmProjects/Program1/firstproject.py
Employee's name is Paul and his salary is 1000

Process finished with exit code 0


Inheritance
===========

class employee:
    def __init__(self,name,salary):
        self.name = name           # define name
        self.salary = salary       # define salary
        print('Employee\'s name is {} and his salary is {}'.format(name,salary))  # format method

class company(employee):       #  company inherits from employee
    def staff(self,name,salary):
        employee.__init__(self,name,salary)

class profession(employee):    # profession inherits from employee
    def developer(self,name,salary):
        employee.__init__(self,name,salary)


em = company('Hello',950)            #  company inherits from employee
pr = profession('Ejike',1000000000)  # profession inherits from employee


    OUTPUT:

/usr/bin/python3.7 /home/dan/PycharmProjects/Program1/firstproject.py
Employee's name is Hello and his salary is 950
Employee's name is Ejike and his salary is 1000000000

Process finished with exit code 0


Overloading
===========
class employee:
    def __init__(self,name,salary):   # method / function. staff is the name
        self.name = name           # define name
        self.salary = salary       # define salary
    def __str__(self):
        return ('Employee\'s name is {} and his salary is {}' .format(self.name, self.salary))

class company(employee):  # inherit from employee and add age
    def __init__(self,name,salary,age):
        employee.__init__(self,name,salary)
        self.age = age
    def __str__(self):
        return (employee.__str__(self) + ' and his age is: {}' .format(self.age))

class profession(employee):  # inherit from employee and add budget
    def __init__(self,name,salary,budget):
        employee.__init__(self,name,budget)
        self.budget = budget
    def __str__(self):
        return (employee.__str__(self) + ' and his budget is: {}' .format(self.budget))

co = company('Peter',1000,24)
pro = profession('James',500,1000)
print(co)
print(pro)

    OUTPUT:

/usr/bin/python3.7 /home/dan/PycharmProjects/Program1/firstproject.py
Employee's name is Peter and his salary is 1000 and his age is: 24
Employee's name is James and his salary is 1000 and his budget is: 1000

Process finished with exit code 0


==============================================================================
Graphical User Interface (GUI) in Python
==============================================================================

First GUI
=========

from tkinter import * # had to install - sudo apt-get install python3-tk

root = Tk()
root.title('First GUI')
root.mainloop()

    OUTPUT:

 A small window with the title "First GUI"

GUI size and position
=====================

from tkinter import *

root = Tk()
root.title('First GUI')
root.geometry('200x100+700+700')  # x axis size X y axis size + x axis position + y axis position
root.mainloop()


Add Labels to GUI window
========================

from tkinter import *

root = Tk()
root.title('First GUI')
root.geometry('500x500+300+200')
mylabel = Label(text="First Text") # Label is a class
mylabel.pack()  # call pack method - takes mylabel and places in center of window
root.mainloop()

  -- or --
```

```
from tkinter import *

root = Tk()
root.title('First GUI')
root.geometry('500x500+300+200')
mylabel = Label(text="First Text").pack()
#         class                   method
root.mainloop()


from tkinter import *

root = Tk()
root.title('First GUI')
root.geometry('500x500+300+200')
mylabel = Label(text="First Text",fg='blue',bg='yellow').pack()
mylabel2 = Label(text="Second Text",fg='red',bg='green').pack() # need different varable
root.mainloop()

Change Label Positions
======================

.pack method places label at the center.

.place method takes x,y position

mylabel = Label(text="First Text",fg='blue',bg='yellow').place(x=100,y=100)
                                                        -------------------

from tkinter import *

root = Tk()
root.title('First GUI')
root.geometry('500x500+300+200')
mylabel = Label(text="First Text",fg='blue',bg='yellow').place(x=100, y=100)
mylabel2 = Label(text="Second Text",fg='red',bg='green').place(x=200, y=100)
root.mainloop()


.grid takes row and column

mylabel = Label(text="First Text",fg='blue',bg='yellow').grid(row=0, column=0)
                                                        ----------------------

use sticky to position N, S, E, or W to align text

mylabel = Label(text="First Text",fg='blue',bg='yellow').grid(row=0, column=0, sticky='W')
                                                                               ---------

from tkinter import *

root = Tk()
root.title('First GUI')
root.geometry('500x500+300+200')
mylabel = Label(text="First Text",fg='blue',bg='yellow').grid(row=0, column=0, sticky='W')
mylabel2 = Label(text="Second Text",fg='red',bg='green').grid(row=1, column=0)
root.mainloop()


Add Button
==========

Btn1 = Button(text='Sumit', fg='red', bg='purple').pack()

NOTE:
_tkinter.TclError: cannot use geometry manager pack inside . which already has slaves managed by grid


from tkinter import *

root = Tk()
root.title('First GUI')
root.geometry('500x500+300+200')
mylabel = Label(text="First Text",fg='blue',bg='yellow').pack()
mylabel2 = Label(text="Second Text",fg='red',bg='green').pack()
Btn1 = Button(text='Sumit', fg='red', bg='purple').pack()
Btn2 = Button(text='Open', fg='white', bg='blue').pack()
root.mainloop()
```

```
Add Function to Button
======================

from tkinter import *

root = Tk()
def btnf1():  # define function
    Label(text="Save Python programming", fg='blue', bg='red',font = 12).pack()
def btnf2():
    Label(text="Open button in Python", fg='blue', bg='yellow',font = 12).pack()


root.title('First GUI')
root.geometry('500x500+300+200')
mylabel = Label(text="First Text",fg='blue',bg='yellow').pack()
mylabel2 = Label(text="Second Text",fg='black',bg='green').pack()
Btn1 = Button(text='Sumit', fg='black', bg='purple', command = btnf1,font = 12).pack()
Btn2 = Button(text='Open', fg='white', bg='blue', command = btnf2,font = 12).pack()
root.mainloop()


Text Box
==========


from tkinter import *

root = Tk()
def btnf1():  # define function
    txt1 = txt.get()
    Label(text=txt1, fg='blue', bg='red',font = 12).pack()
def btnf2():
    Label(text="Open button in Python", fg='blue', bg='yellow',font = 12).pack()


txt = StringVar()

root.title('First GUI')
root.geometry('500x500+300+200')
mylabel = Label(text="First Text",fg='blue',bg='yellow').pack()
mylabel2 = Label(text="Second Text",fg='black',bg='green').pack()
Btn1 = Button(text='Sumit', fg='black', bg='purple', command = btnf1,font = 12).pack()
mytext = Entry(textvariable = txt).pack()
Btn2 = Button(text='Open', fg='white', bg='blue', command = btnf2,font = 12).pack()
root.mainloop()


Multiple GUIs in one program
============================

from tkinter import *

root = Tk()
mywin = Tk()
def btnf1():  # define function
    txt1 = txt.get()
    Label(root,text=txt1, fg='blue', bg='red',font = 12).pack()
def btnf2():
    Label(mywin,text="Open button in Python", fg='blue', bg='yellow',font = 12).pack()


txt = StringVar()

root.title('First GUI')
mywin.title('Second GUI')
root.geometry('500x500+100+200')
mywin.geometry('500x500+650+200')
mylabel = Label(text="First Text",fg='blue',bg='yellow').pack()
mylabel2 = Label(mywin,text="Second Text",fg='black',bg='green').pack()
Btn1 = Button(mywin,text='Sumit', fg='black', bg='purple', command = btnf1,font = 12).pack()
mytext = Entry(textvariable = txt).pack()
Btn2 = Button(mywin,text='Open', fg='white', bg='blue', command = btnf2,font = 12).pack()

root.mainloop()
mywin.mainloop()
```

```
https://realpython.com/python-gui-tkinter/

Add Menu to Window
==================

from tkinter import *

root = Tk()
def btnf1():  # define function
    txt1 = txt.get()
    Label(text=txt1, fg='blue', bg='red',font = 12).pack()
def btnf2():
    Label(text="Open button in Python", fg='blue', bg='yellow',font = 12).pack()


txt = StringVar()
root.title('First GUI')
root.geometry('500x500+300+200')
mylabel = Label(text="First Text",fg='blue',bg='yellow').pack()
mylabel2 = Label(text="Second Text",fg='black',bg='green').pack()
Btn1 = Button(text='Sumit', fg='black', bg='purple', command = btnf1,font = 12).pack()
mytext = Entry(textvariable = txt).pack()
Btn2 = Button(text='Open', fg='white', bg='blue', command = btnf2,font = 12).pack()

Chooser = Menu()
Chooser.add_cascade(label="File")
Chooser.add_cascade(label="Edit")
Chooser.add_cascade(label="Navigate")
Chooser.add_cascade(label="Code")
Chooser.add_cascade(label="Run")
Chooser.add_cascade(label="Tools")
Chooser.add_cascade(label="Help")


root.config(menu=Chooser)

root.mainloop()


Add Menu Items to Menus
=======================

from tkinter import *

root = Tk()
def btnf1():  # define function
    txt1 = txt.get()
    Label(text=txt1, fg='blue', bg='red',font = 12).pack()
def btnf2():
    Label(text="Open button in Python", fg='blue', bg='yellow',font = 12).pack()


txt = StringVar()
root.title('First GUI')
root.geometry('500x500+300+200')
mylabel = Label(text="First Text",fg='blue',bg='yellow').pack()
mylabel2 = Label(text="Second Text",fg='black',bg='green').pack()
Btn1 = Button(text='Sumit', fg='black', bg='purple', command = btnf1,font = 12).pack()
mytext = Entry(textvariable = txt).pack()
Btn2 = Button(text='Open', fg='white', bg='blue', command = btnf2,font = 12).pack()

Chooser = Menu()
itemone = Menu()
itemone.add_command(label='New Project')
itemone.add_command(label='Save')
itemone.add_command(label='Navigate')
itemone.add_command(label='Close')

itemtwo = Menu()
itemtwo.add_command(label='Copy')
itemtwo.add_command(label='Cut')
itemtwo.add_command(label='Past')
itemtwo.add_command(label='Delete')

Chooser.add_cascade(label="File",menu=itemone)
Chooser.add_cascade(label="Edit",menu=itemtwo)
Chooser.add_cascade(label="Navigate")
Chooser.add_cascade(label="Code")
Chooser.add_cascade(label="Run")
Chooser.add_cascade(label="Tools")
Chooser.add_cascade(label="Help")


root.config(menu=Chooser)

root.mainloop()
```

Add Functionality to menu item
==============================

```
from tkinter import *

root = Tk()
def btnf1():  # define function
    txt1 = txt.get()
    Label(text=txt1, fg='blue', bg='red',font = 12).pack()
def btnf2():
    Label(text="Open button in Python", fg='blue', bg='yellow',font = 12).pack()
def gui():
    gu = Tk()
    gu.title("New Project")
    gu.mainloop()
def save():
    Label(text="Project Saved", fg='black', bg='green', font=12).pack()

txt = StringVar()
root.title('First GUI')
root.geometry('500x500+300+200')
mylabel = Label(text="First Text",fg='blue',bg='yellow').pack()
mylabel2 = Label(text="Second Text",fg='black',bg='green').pack()
Btn1 = Button(text='Sumit', fg='black', bg='purple', command = btnf1,font = 12).pack()
mytext = Entry(textvariable = txt).pack()
Btn2 = Button(text='Open', fg='white', bg='blue', command = btnf2,font = 12).pack()

Chooser = Menu()
itemone = Menu()
itemone.add_command(label='New Project',command = gui)
itemone.add_command(label='Save',command = save)
itemone.add_command(label='Navigate')
itemone.add_command(label='Close')

itemtwo = Menu()
itemtwo.add_command(label='Copy')
itemtwo.add_command(label='Cut')
itemtwo.add_command(label='Past')
itemtwo.add_command(label='Delete')

Chooser.add_cascade(label="File",menu=itemone)
Chooser.add_cascade(label="Edit",menu=itemtwo)
Chooser.add_cascade(label="Navigate")
Chooser.add_cascade(label="Code")
Chooser.add_cascade(label="Run")
Chooser.add_cascade(label="Tools")
Chooser.add_cascade(label="Help")


root.config(menu=Chooser)

root.mainloop()
```

Create Messgebox
================

```
from tkinter import *
import tkinter.messagebox

root = Tk()
def btnf1():  # define function
    txt1 = txt.get()
    Label(text=txt1, fg='blue', bg='red',font = 12).pack()
def btnf2():
    Label(text="Open button in Python", fg='blue', bg='yellow',font = 12).pack()
def gui():
    gu = Tk()
    gu.title("New Project")
    gu.mainloop()
def save():
    Label(text="Project Saved", fg='black', bg='green', font=12).pack()
def mbox():
    tkinter.messagebox.showinfo('Save','Do you want to open this file?')
def dele():
    de = tkinter.messagebox.askquestion('Delete','Want to close this file?')
    if de == 'yes':
        root.destroy()

txt = StringVar()
root.title('First GUI')
root.geometry('500x500+300+200')
mylabel = Label(text="First Text",fg='blue',bg='yellow').pack()
mylabel2 = Label(text="Second Text",fg='black',bg='green').pack()
Btn1 = Button(text='Sumit', fg='black', bg='purple', command = btnf1,font = 12).pack()
mytext = Entry(textvariable = txt).pack()
Btn2 = Button(text='Open', fg='white', bg='blue', command = btnf2,font = 12).pack()

Chooser = Menu()
itemone = Menu()
itemone.add_command(label='New Project',command = gui)
itemone.add_command(label='Save',command = save)
itemone.add_command(label='Open',command = mbox)
itemone.add_command(label='Close',command = dele)

itemtwo = Menu()
itemtwo.add_command(label='Copy')
itemtwo.add_command(label='Cut')
itemtwo.add_command(label='Past')
itemtwo.add_command(label='Delete',command = dele)

Chooser.add_cascade(label="File",menu=itemone)
Chooser.add_cascade(label="Edit",menu=itemtwo)
Chooser.add_cascade(label="Navigate")
Chooser.add_cascade(label="Code")
Chooser.add_cascade(label="Run")
Chooser.add_cascade(label="Tools")
Chooser.add_cascade(label="Help")


root.config(menu=Chooser)

root.mainloop()
```

Create a digit counter
======================

```
from tkinter import *

counter = 0
def digit_counter(mylabel):
    counter = 0
    def digit():
        global counter
        counter += 1
        mylabel.config(text=str(counter))
        mylabel.after(1000,digit)
    digit()

root = Tk()
root.title('Digit Counter')
mylabel = Label(fg='red', font=200)
mylabel.pack()
digit_counter(mylabel)
btn = Button(text='Terminate', width=50,command=root.destroy)
btn.pack()
root.mainloop()
```

Create a color chooser
======================

```
from tkinter import *
from tkinter.colorchooser import *

root = Tk()
root.geometry('250x100')

def mycolor():
    color = askcolor()
    mylabel = Label(text='This is your Preferred color', bg=color[1]).pack()
    print(color)

btn = Button(text='Choose color', command=mycolor).pack()

root.mainloop()
```