```python
from tkinter import *

app = Tk()

app.title("BMI")

hi = Entry(app)
hi.insert(0, "Enter your Height")
hi.grid(row=0, column=0)

wi = Entry(app)
wi.insert(0, "Enter your Weight")
wi.grid(row=1, column=0)


def bmi():
    w = int(wi.get())
    h = int(hi.get())
    return int(w / (h * h) * 10000)


cb = Button(app, text="Calc", command=lambda: cb.config(text="BMI: " + str(bmi())))
cb.grid(row=2, column=0)
app.mainloop()

```

```
0000008::Edison Kinetoscopic Record of a Sneeze (1894)::Documentary|Short
0000010::La sortie des usines Lumière (1895)::Documentary|Short
0000012::The Arrival of a Train (1896)::Documentary|Short
25::The Oxford and Cambridge University Boat Race (1895)::
0000091::Le manoir du diable (1896)::Short|Horror
0000131::Une nuit terrible (1896)::Short|Comedy|Horror
0000417::A Trip to the Moon (1902)::Short|Action|Adventure|Comedy|Fantasy|Sci-Fi
0000439::The Great Train Robbery (1903)::Short|Action|Crime|Western
0443::Hiawatha, the Messiah of the Ojibway (1903)::
0000628::The Adventures of Dollie (1908)::Action|Short
0000833::The Country Doctor (1909)::Short|Drama
0001223::Frankenstein (1910)::Short|Horror|Sci-Fi
0001740::The Lonedale Operator (1911)::Short|Drama|Romance
0002101::Cleopatra (1912)::Drama|History
0002130::L'inferno (1911)::Adventure|Drama|Fantasy|Horror
0002354::Max et Jane veulent faire du théâtre (1911)::Short|Comedy|Romance
0002844::Fantômas - À l'ombre de la guillotine (1913)::Crime|Drama
0003740::Cabiria (1914)::Adventure|Drama|History
0003863::Dough and Dynamite (1914)::Comedy|Short
0004099::His Majesty, the Scarecrow of Oz (1914)::Family|Fantasy|Adventure|Comedy
0004100::His Musical Career (1914)::Short|Comedy
0004101::His New Profession (1914)::Short|Comedy
0004210::Laughing Gas (1914)::Short|Comedy
0004395::The New Janitor (1914)::Short|Comedy
0004413::The Ocean Waif (1916)::Short|Comedy|Drama
0004457::The Patchwork Girl of Oz (1914)::Adventure|Family|Fantasy
0004518::Recreation (1914)::Comedy|Short
0004546::The Rounders (1914)::Comedy|Short
0004936::The Bank (1915)::Comedy|Short
0004972::The Birth of a Nation (1915)::Drama|History|War
0005074::The Champion (1915)::Short|Comedy|Sport
0005078::The Cheat (1915)::Drama
0005530::L'héroïsme de Paddy (1915)::
0005571::A Jitney Elopement (1915)::Short|Comedy
0005960::Regeneration (1915)::Biography|Crime|Drama|Romance
0006177::The Tramp (1915)::Short|Comedy
0006206::Les vampires (1915)::Action|Adventure|Crime|Drama|Mystery|Thriller
0006333::20,000 Leagues Under the Sea (1916)::Action|Adventure|Sci-Fi
0006414::Behind the Screen (1916)::Short|Comedy|Romance
0006437::The Blacklist (1916)::Drama
0006684::The Fireman (1916)::Short|Comedy
0006689::The Floorwalker (1916)::Short|Comedy
0006864::Intolerance: Love's Struggle Throughout the Ages (1916)::Drama|History
0007145::One A.M. (1916)::Comedy|Family|Short
0007162::The Pawnshop (1916)::Comedy|Short
0007264::The Rink (1916)::Comedy|Short
0007340::Shoes (1916)::Drama
0007507::The Vagabond (1916)::Short|Comedy|Romance
0007832::The Cure (1917)::Short|Comedy
0007880::Easy Street (1917)::Short|Comedy
0008133::The Immigrant (1917)::Short|Comedy|Drama|Romance
0008395::Otets Sergiy (1918)::Biography|Drama|History
0009018::A Dog's Life (1918)::Short|Comedy|Drama
009340::The Man Who Woke Up (1918)::
0009678::Take a Chance (1918)::Comedy|Short
0009893::Die Austernprinzessin (1919)::Comedy
0009968::Broken Blossoms or The Yellow Man and the Girl (1919)::Drama|Romance
0010180::Godovshchina revolyutsii (1918)::Documentary
0010193::The Greatest Question (1919)::Drama
0010247::Herr Arnes pengar (1919)::Drama|History
0010258::His Royal Slyness (1920)::Comedy|Short
0010323::Das Cabinet des Dr. Caligari (1920)::Fantasy|Horror|Mystery|Thriller
0010747::Sunnyside (1919)::Comedy|Short
0010806::True Heart Susie (1919)::Comedy|Drama|Romance
0010930::Young Mr. Jazz (1919)::Comedy|Short
0011130::Dr. Jekyll and Mr. Hyde (1920)::Drama|Horror|Sci-Fi
0011267::Headin' Home (1920)::Biography|Comedy|Drama
```

```
0011439::The Mark of Zorro (1920)::Adventure|Romance|Western
0011508::Neighbors (1920)::Short|Comedy|Romance
0011541::One Week (1920)::Short|Comedy
0011607::Prästänkan (1920)::Comedy|Drama|Horror
0011656::The Scarecrow (1920)::Comedy|Short|Family
0011717::The Son of Tarzan (1920)::Action|Adventure
0011841::Way Down East (1920)::Drama|Romance
0011870::Within Our Gates (1920)::Drama|Romance
0012224::The Goat (1921)::Comedy|Short
0012278::The 'High Sign' (1921)::Short|Comedy
0012349::The Kid (1921)::Comedy|Drama|Family
0012364::Körkarlen (1921)::Drama|Fantasy|Horror
0012494::Der müde Tod (1921)::Drama|Fantasy|Thriller
0012532::Orphans of the Storm (1921)::Drama|History|Romance
0012651::Schloß Vogeloed (1921)::Crime|Drama|Horror|Mystery
0012675::The Sheik (1921)::Adventure|Drama|Romance
012844::White and Unmarried (1921)::
0013025::Cops (1922)::Short|Comedy|Family
0013086::Dr. Mabuse, der Spieler (1922)::Crime|Mystery|Thriller
0013099::The Electric House (1922)::Short|Comedy
0013140::Foolish Wives (1922)::Drama|Thriller
0013257::Häxan (1922)::Documentary|Fantasy|Horror
0013427::Nanook of the North (1922)::Documentary
0013442::Nosferatu (1922)::Fantasy|Horror
0013486::Pay Day (1922)::Comedy|Short
0013571::Salomé (1922)::Biography|Drama|History|Horror
0013626::La souriante Madame Beudet (1923)::Drama
0013741::Das Weib des Pharao (1922)::Drama|History
0013858::The Balloonatic (1923)::Short|Comedy
0014142::The Hunchback of Notre Dame (1923)::Drama|Horror|Romance
0014341::Our Hospitality (1923)::Comedy|Romance|Thriller
0014390::En rackarunge (1924)::Drama
0014417::La roue (1923)::Drama
0014429::Safety Last! (1923)::Comedy|Thriller
0014497::Souls for Sale (1923)::Comedy|Drama|Romance
0014532::The Ten Commandments (1923)::Biography|Drama|Fantasy
0014538::Three Ages (1923)::Comedy
0014624::A Woman of Paris: A Drama of Fate (1923)::Drama|Romance
0014664::Alice's Spooky Adventure (1924)::Animation|Short|Comedy
0014872::Entr'acte (1924)::Short
0014972::He Who Gets Slapped (1924)::Drama|Romance|Thriller
0015002::Hot Water (1924)::Comedy
0015016::The Iron Horse (1924)::History|Romance|Western
0015039::Kinoglaz (1924)::Documentary
0015064::Der letzte Mann (1924)::Drama
0015163::The Navigator (1924)::Action|Comedy|Romance
0015174::Die Nibelungen: Kriemhilds Rache (1924)::Adventure|Drama|Fantasy
0015175::Die Nibelungen: Siegfried (1924)::Adventure|Drama|Fantasy
0015202::Orlacs Hände (1924)::Crime|Horror|Mystery|Sci-Fi|Thriller
0015233::En piga bland pigor (1924)::Comedy|Drama
0015310::The Sea Hawk (1924)::Adventure|Drama|Romance
0015324::Sherlock Jr. (1924)::Action|Comedy|Romance
0015361::Stachka (1925)::Drama
0015400::The Thief of Bagdad (1924)::Adventure|Family|Fantasy|Romance
0015477::West of Hot Dog (1924)::Short|Western|Comedy
0015532::Die Abenteuer des Prinzen Achmed (1926)::Animation|Adventure|Fantasy|Romance
0015624::The Big Parade (1925)::Drama|Romance|War
0015648::Bronenosets Potyomkin (1925)::Drama|History
0015673::Shakhmatnaya goryachka (1925)::Comedy|Short
0015768::Du skal ære din hustru (1925)::Drama
0015772::De adelaar (1925)::Action|Comedy|Drama|History|Romance
0015864::The Gold Rush (1925)::Adventure|Comedy|Drama|Family
0015881::Greed (1924)::Drama|Thriller|Western
0016039::The Lost World (1925)::Adventure|Fantasy|Sci-Fi
0016172::Oh, Doctor! (1925)::Comedy
0016220::The Phantom of the Opera (1925)::Horror
0016230::The Pleasure Garden (1925)::Drama|Romance
```

```
0016332::Seven Chances (1925)::Comedy|Romance
0016361::Smouldering Fires (1925)::Drama
0016544::The Wizard of Oz (1925)::Comedy|Family|Fantasy|Adventure
0016600::Along Came Auntie (1926)::Comedy|Short
0016654::The Black Pirate (1926)::Adventure|Action
0016747::Crazy Like a Fox (1926)::Short|Comedy
0016847::Faust: Eine deutsche Volkssage (1926)::Drama|Fantasy|Horror
0016903::45 Minutes from Hollywood (1926)::Short|Comedy
016954::Hell's Four Hundred (1926)::

```

```python
lst = map(int, input().split())

for i in lst:
    if i * i % 8 == 0:
        print(i)

```

```python
from requests import get
from bs4 import BeautifulSoup


def helper(uri="/chart/top"):
    return get("https://www.imdb.com" + uri, headers={"User-Agent": "Mozilla"}).content


soup = BeautifulSoup(helper(), "html.parser")

movies = soup.select(".cli-children")
for movie in movies[:10]:
    print(movie.select("h3")[0].text)  # NAME
    print(movie.select("span")[0].text)  # YEAR
    sumry = BeautifulSoup(
        helper(movie.select("a")[0].get("href")), "html.parser"
    )  # LINK for PLOT
    print(sumry.find("span", attrs={"data-testid": "plot-xl"}).text)  # PLOT
    print()

```

```python
str_ = input("Enter a string: ")
count = dict()

for i in str_:
    if i in count:
        count[i] += 1
    else:
        count[i] = 1

for i in count.items():
    print(f"{i[0]} : {i[1]}")

```

```
Hello world
Test File
Bye world
```

```python
from turtle import *

n_sides = int(input("How many sides do you want: "))
s_side = int(input("Size of side: "))

color("green")

fillcolor("red")

begin_fill()

for _ in range(n_sides):
    fd(s_side)
    lt(360 / n_sides)

end_fill()

mainloop()

```

```python
n_hour = int(input("Enter number of hours: "))
n_wage = int(input("Enter number of wage/hr: "))
mul = 1

if n_hour < 40:
    mul = 1
elif n_hour < 60:
    mul = 1.5
else:
    mul = 2

print(n_hour * n_wage * mul)

```

```python
class BankAccount:
    def __init__(self, balance):
        self.balance = balance

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient funds")
            return

        self.balance -= amount
        self.vbalance()

    def deposit(self, amount):
        self.balance += amount
        self.vbalance()

    def vbalance(self):
        print("\nBalance: ", self.balance)
        print()
        return self.balance


bank = BankAccount(int(input("Enter balance: ")))
while True:
    choice = input("Enter choice\n1. Withdraw\n2. Deposit\n3. Balance\n4. Exit: ")
    if choice == "1":
        amount = int(input("Enter amount: "))
        bank.withdraw(amount)
    elif choice == "2":
        amount = int(input("Enter amount: "))
        bank.deposit(amount)
    elif choice == "3":
        bank.vbalance()
    elif choice == "4":
        break
    else:
        print("Invalid choice")

```

```
{
 "cells": [
  {
   "cell_type": "code",
   "id": "initial_id",
   "metadata": {
    "collapsed": true,
    "ExecuteTime": {
     "end_time": "2024-08-10T11:44:49.967184Z",
     "start_time": "2024-08-10T11:44:49.962787Z"
    }
   },
   "source": "print \"HI\"",
   "outputs": [
    {
     "ename": "SyntaxError",
     "evalue": "Missing parentheses in call to 'print'. Did you mean print(...)? (205863140.py, line 1)",
     "output_type": "error",
     "traceback": [
      "\u001B[1;36m  Cell \u001B[1;32mIn[2], line 1\u001B[1;36m\u001B[0m\n\u001B[1;33m    print \"HI\"\u001B[0m\n\u001B[1;37m    ^\u001B[0m\n\u001B[1;31mSyntaxError\u001B[0m\u001B[1;31m:\u001B[0m Missing parentheses in call to 'print'. Did you mean print(...)?\n"
     ]
    }
   ],
   "execution_count": 2
  },
  {
   "metadata": {
    "ExecuteTime": {
     "end_time": "2024-08-10T11:44:55.428090Z",
     "start_time": "2024-08-10T11:44:55.405372Z"
    }
   },
   "cell_type": "code",
   "source": [
    "lst = [1,2,3]\n",
    "print(lst[5])"
   ],
   "id": "70dd73560a148e15",
   "outputs": [
    {
     "ename": "IndexError",
     "evalue": "list index out of range",
     "output_type": "error",
     "traceback": [
      "\u001B[1;31m---------------------------------------------------------------------------\u001B[0m",
      "\u001B[1;31mIndexError\u001B[0m                                Traceback (most recent call last)",
      "Cell \u001B[1;32mIn[3], line 2\u001B[0m\n\u001B[0;32m      1\u001B[0m lst \u001B[38;5;241m=\u001B[39m [\u001B[38;5;241m1\u001B[39m,\u001B[38;5;241m2\u001B[39m,\u001B[38;5;241m3\u001B[39m]\n\u001B[1;32m----> 2\u001B[0m \u001B[38;5;28mprint\u001B[39m(\u001B[43mlst\u001B[49m\u001B[43m[\u001B[49m\u001B[38;5;241;43m5\u001B[39;49m\u001B[43m]\u001B[49m)\n",
      "\u001B[1;31mIndexError\u001B[0m: list index out of range"
     ]
    }
   ],
   "execution_count": 3
  },
  {
   "metadata": {
    "ExecuteTime": {
     "end_time": "2024-08-10T11:44:59.700925Z",
     "start_time": "2024-08-10T11:44:59.688850Z"
    }
   },
   "cell_type": "code",
   "source": "int(\"Hi\")",
   "id": "b377d4c55f21db42",
   "outputs": [
    {
     "ename": "ValueError",
     "evalue": "invalid literal for int() with base 10: 'Hi'",
     "output_type": "error",
     "traceback": [
      "\u001B[1;31m---------------------------------------------------------------------------\u001B[0m",
      "\u001B[1;31mValueError\u001B[0m                                Traceback (most recent call last)",
      "Cell \u001B[1;32mIn[4], line 1\u001B[0m\n\u001B[1;32m----> 1\u001B[0m \u001B[38;5;28;43mint\u001B[39;49m\u001B[43m(\u001B[49m\u001B[38;5;124;43m\"\u001B[39;49m\u001B[38;5;124;43mHi\u001B[39;49m\u001B[38;5;124;43m\"\u001B[39;49m\u001B[43m)\u001B[49m\n",
      "\u001B[1;31mValueError\u001B[0m: invalid literal for int() with base 10: 'Hi'"
     ]
    }
   ],
   "execution_count": 4
  },
  {
   "metadata": {
    "ExecuteTime": {
     "end_time": "2024-08-10T11:45:12.726459Z",
     "start_time": "2024-08-10T11:45:12.714546Z"
    }
   },
   "cell_type": "code",
   "source": "lst[\"1\"]",
   "id": "9dd90829f90e23c9",
   "outputs": [
    {
     "ename": "TypeError",
     "evalue": "list indices must be integers or slices, not str",
     "output_type": "error",
     "traceback": [
      "\u001B[1;31m---------------------------------------------------------------------------\u001B[0m",
      "\u001B[1;31mTypeError\u001B[0m                                 Traceback (most recent call last)",
      "Cell \u001B[1;32mIn[5], line 1\u001B[0m\n\u001B[1;32m----> 1\u001B[0m \u001B[43mlst\u001B[49m\u001B[43m[\u001B[49m\u001B[38;5;124;43m\"\u001B[39;49m\u001B[38;5;124;43m1\u001B[39;49m\u001B[38;5;124;43m\"\u001B[39;49m\u001B[43m]\u001B[49m\n",
      "\u001B[1;31mTypeError\u001B[0m: list indices must be integers or slices, not str"
     ]
    }
   ],
   "execution_count": 5
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 2
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython2",
   "version": "2.7.6"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}

```

```python
dictmap = {
    "CBI": "Central Bureau of Investigation",
    "FBI": "Financial Bureau of Investigation",
    "NIA": "National Investigation Agency",
    "SSB": "South State Bureau of Investigation",
    "WPA": "West State Agency",
}

print(dictmap)

dictmap["BSE"] = "Bombay Stock Exchange"

print(dictmap)

dictmap["SSB"] = "Social Security Admin"

print(dictmap)

del dictmap["CBI"]
del dictmap["WPA"]

print(dictmap)

```

```python
from turtle import *

angles = [0, 120, 240]
colors = ["red", "blue", "magenta"]


def shift_turtle(len_, a):
    lt(a)
    fd(len_)
    rt(a)


def draw_triangle(length, depth, index):
    penup()
    if depth == 0:
        pendown()
        for i in range(3):
            fd(length)
            lt(120)
        penup()
        return

    for ind, angle in enumerate(angles):
        color(colors[ind])
        draw_triangle(length / 2, depth - 1, index - 1)
        shift_turtle(length / 2, angle)


draw_triangle(100, 2, 1)
mainloop()

```

```python
def search(arr, key):
    if len(arr) == 0:
        return -1

    mid = len(arr) // 2

    if arr[mid] == key:
        return mid
    elif arr[mid] < key:
        val = search(arr[mid + 1 :], key)
        if val == -1:
            return val
        return mid + val + 1
    else:
        return search(arr[:mid], key)


for i in range(11):
    print(search([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], i))

```

```python
def merge(arr1, arr2):
    arro = []
    l = r = 0

    while l < len(arr1) and r < len(arr2):
        if arr1[l] < arr2[r]:
            arro.append(arr1[l])
            l += 1
        else:
            arro.append(arr2[r])
            r += 1

    arro.extend(arr1[l:])
    arro.extend(arr2[r:])

    return arro


def mergesort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left = arr[:mid]
    right = arr[mid:]

    l_sorted = mergesort(left)
    r_sorted = mergesort(right)

    return merge(l_sorted, r_sorted)


print(mergesort([3, 4, 5, 6, 7, 8, 9, 2, 1]))

```

```python
from turtle import *


def snowflake(length, depth):
    if depth == 0:
        fd(length)
        return

    snowflake(length, depth - 1)
    lt(60)
    snowflake(length, depth - 1)
    rt(120)
    snowflake(length, depth - 1)
    lt(60)
    snowflake(length, depth - 1)


for _ in range(3):
    snowflake(20, 2)
    rt(120)

mainloop()

```

```python
# hr: 100
# daily: 500
# week: 2500
# n : 3 - 5 : 0.7x

fare_list = [100, 500, 2500]
fare = {
    "Hourly": fare_list[0],
    "Daily": fare_list[1],
    "Weekly": fare_list[2],
}

bikes = ["MTB", "Road", "eV"]

while True:
    ch = input("Enter a choice\n1. View fare\n2. View menu\n3. Buy\n4. Exit: ")
    if ch == "4":
        break

    if ch == "1":
        for i in fare.items():
            print(f"{i[0]} : {i[1]}")
        print()
    elif ch == "2":
        for i in bikes:
            print(i)
        print()
    elif ch == "3":
        n_bikes = int(input("Enter number of bikes: "))
        n_fare = int(input("Enter type of fare\n1. Hourly\n2. Daily\n3. Weekly: "))
        mul = 1
        if 3 < n_bikes and n_bikes < 5:
            print("Family discount applied (30% off)")
            mul = 0.7
        print("Amount is", fare_list[n_fare - 1] * mul * n_bikes, "/hr")
    else:
        print("Invalid choice")

```

```python
fileName = input("Enter a file name: ")

with open(fileName) as file:
    l_counter = 0
    ch_counter = 0
    word_counter = 0
    for line in file:
        l_counter += 1
        ch_counter += len(line)
        word_counter += len(line.strip().split())

print(l_counter)
print(ch_counter)
print(word_counter)

```

```python
from mrjob.job import MRJob
from itertools import combinations


class MRMovieHadoop(MRJob):

    def mapper(self, _, line):
        __, movie, genre = line.split("::")
        yield genre, movie

    def reducer(self, genre, movies):
        for i, j in combinations(movies, 2):
            if i == j:
                continue
            yield (i, j), len(set(i.lower()).intersection(set(j.lower())))


MRMovieHadoop.run()

```

```python
from requests import get
from bs4 import BeautifulSoup

html = get("https://en.wikipedia.org/wiki/Sachin_Tendulkar").content

bs = BeautifulSoup(html, "html.parser")

images = bs.find_all("img")

ast = set()
for i in images:
    ast.add(i.get("src"))

for i in ast:
    print(i)

```