Python Code Snippets

Table Of Contents

- seed

Which Linux version am I running?

```
import sys

def RunningPython3():
    ##print(sys.version_info)
    if sys.version_info[0] == 3:
        return True
    return False

py3 = RunningPython3()
```

Import Tkinter or tkinter?

```
import sys

if sys.version_info.major is 3:
    import tkinter as tk
    import tkinter.font as tkf
    py3 = True
else:
    import Tkinter as tk
    import tkFont as tkf
    py3 = False

or

import sys

if sys.version_info.major is 3:
    from tkinter import *
    py3 = True
else:
    from Tkinter import *
    py3 = False

or

try:
    import tkinter as tk
    import tkinter.font as tkf
except:
    import Tkinter as tk
    import tkFont as tkf
```

Prompt the user for input

```
def GetUserInput(prompt,py3):
    if py3:
        return input(prompt)
    else:
        return raw_input(prompt)
```

Pause program

```
def Pause(py3):
    print('')
    GetUserInput('Press enter to continue ',py3)
```

Get and test user input

```
while True:

    value = GetUserInput('Enter an integer ',py3)

    sval = value.strip()

    if sval == '':
        break

    if sval.isdigit() != True:
        print('')
        print('Illegal value entered({})'.format(sval))
        Pause(py3)
        continue

    ival = int(sval)
```

Convert String to a Number

```
# ---------------------------------------------------------------------
# ---- Function: convert a string to a float
# ---------------------------------------------------------------------

def is_float(s):
    try:
        n = float(s)
        return (True,n)
    except:
        return (False,0.0)

# ---------------------------------------------------------------------
# ---- Function: convert a string to an integer
# ---------------------------------------------------------------------

def is_int(s):
    try:
        n = int(s)
        return (True,n)
    except:
        return (False,0)

def is_integer(s):
        return is_int(s)

# ---------------------------------------------------------------------
# ---- is a number (integer, float, scientific notation)
# ---------------------------------------------------------------------

def is_a_number(s):

    tf,n = is_int(s)
    if tf:
        return True

    tf,n = is_float(s)
    if tf:
        return True

    return False
```

Clear the terminal screen

```
import os
import platform

def ClearScreen():
    if platform.system() == 'Linux':
        os.system('clear')
    elif platform.system() == 'Windows':
        os.system('clear')
    else:
        os.system('cls')
```

Display environment variables

```
import os

env = os.environ

# str contains ';' separated values
def DisplayPath(str):
    strlist = str.split(';')
    for s in strlist:
        print(s)

# env is a dictionary (k:v)
for k,v in env.items():
    print('----------------------------------------')
    print('key: {}'.format(k))
    if k == 'PATH':
        DisplayPath(v)
    else:
        print('{}:{}'.format(k,v))
```

Command line arguments

```
import sys

for arg in sys.argv:
    print(arg)
```

Create 1D and 2D arrays (lists)

```
# ---- create a 1D array initialized to 99

arr = [99 for i in range(cols)]

  or

arr = [99]*cols

  or

# ---- initialize to different values
a = [0,1,2,3,4]
arr = [a[i] for i in range(3)]
```

```
# ---- create a 2D array (rows,cols) initialized to -1
# ---- with independent column lists (arrays)

arr = [[-1 for i in range(cols)] for j in range(rows)]

# ---- demonstrate independent column lists (different IDs)
for row in range(rows):
    print(id(arr[row]))
```

RegExp - String match RegExp?

```
# ---------------------------------------------------------------------
# test if a string matches one of a list or tuple of regular expressions
#
# Regular expressions use the backslash character ('\') to indicate
# special forms or to allow special characters to be used without
# invoking their special meaning. This collides with Python's usage
# of the same character for the same purpose in string literals. The
# solution is to use Python's raw string notation for regular
# expression patterns. Backslashes are not handled in any special
# way in a string literal prefixed with 'r'.  r"\n" is a
# two-character string containing '\' and 'n'.
#                                          (from Python documentation)
#
# For example to match html files: r'\.html$' or '\\.html$'
# ---------------------------------------------------------------------

import re

def StringMatchPattern(patterns,str):

    for p in patterns:
        if re.search(p,str,re.IGNORECASE):
            return True
    return False

MATCHPATTERNS   = [ r'\.html$', r'\.pdf$', r'\.png$', r'\.txt$', r'\.css$', r'^xyz' ]

if StringMatchPattern(MATCHPATTERNS,"xyz.txt")
    print("Matched Pattern")
```

Convert a string to a list

```
# uppercase the text string and
# convert it to a list so it can be modified

lst = list(txt.upper())
```

Convert a list to a string

```
# turn a list into a string

s = ''.join([str(x) for x in lst])
```

Read a file

```
# read the file one line at a time

with open('Peter_Jones.md','r') as f:
    for line in f:
        line = line.strip()
        print(line)
```

```
# read the whole file

f = open("Tom_Jones.md", "r")
print(f.read())
f.close()
```

```
# read the file 24 bytes at a time

with open("Judy_Jones.md", "r") as f:
    print(f.read(24))
```

```
# read the file 1 byte at a time using a generator

def read_bytes(fobj,nbytes):
    while True:
        byt = fobj.read(nbytes)
        if not byt: return
        yield byt

with open('Mary_Jones.md','r') as fobj:
    for c in read_bytes(fobj,1):
        print(c)
```

Write a file

```
# 'with' has the advantage that the file is closed, even if an exception is raised

data = ['one','two','three','four']

lc = 0

with open(outfile,'w') as fout:
    for s in data:
        fout.write(s + '\n')
    lc += 1

print(f'{lc} lines written to file {outfile}')
```

```
outFile = open('sample.txt','w')
for i in range(10):
    outFile.write('Line  number {}\n'.format(i))
outFile.close()
```

Test a file

```
import os.path

# ---- test if a path (file or directory or link) exists
# ---- (return False for broken symbolic links)

os.path.exists('filename')
os.path.exists('dirname')

try:
    f = open('filename')
    f.close()
execpt FileNotFoundError:
    print('file does not exist')

# ---- test if path is a file

os.path.isfile('filename')

# ---- test if path is a directory

os.path.isdir('dirname')

# ---- test if path is a symbolic link

os.path.islink('linkname')

# ---- test if a file exists and is accessible

try:
    open('filename')
except IOError:
    print('File not found or not accessible')
```

```
# pathlib provides an object oriented interface to paths

import pathlib

path = pathlib.Path('filename')        # create path object

path.exists()
path.is_file()
path.is_dir()
path.is_symlink()
```

Reg-Exp - Search a String

```
import re

pattern = r'^$'

inFile = open(file,'r')

for line in inFile:

    if re.search(pattern,line):
        print('found a match')
    elif:
        print('did not find a match')

inFile.close()
```

Note: see regexp01.py code example

RegExp - Replace a Sub-String

```
# Python Doc: re.sub(pattern,repl,string,count=0,flags=0)
#             (if no match is found, the input string is returned)

import re

s0 = 'abc_data.dat'

s1 = re.sub(r'_data','',s0,flags=re.IGNORECASE)

s2 = re.sub(r'\.dat$','.txt',s1,flags=re.IGNORECASE)

s3 = s2.lower()

print(s3)
```

Note: see regexp02.py code example

Convert Commas to Spaces Then Split String

This allows parameters to be separated by both commas and spaces.

```
s = 'a,b , c d e f'

lst = s.replace(',',' ').split()

for x in lst:
    print(x)
```

File/Directory exists

```
import os

# returns a Boolean (True or False)

if os.path.exists("/a/b/c"):
    print("It is a directory or a file")

if os.path.isdir("/a/b/c"):
    print("It is a directory")

# create a directory if it does not exist

if not os.path.isdir("/a/b/c"):
    print("Creating new dirrectory")
    os.system("mkdir /a/b/c")
```

Notes:
1. see os.path.normpath documentation (os.path - Common pathname manipulations)
2.Check out the pathlib module for an object oriented approach

Create multiple dirs at one time

this code assumes "if path exists it is a directory"

```
import os

def MakeDirs(path):
    if not os.path.isdir(path):
        ##print("Creating new directories")
        ##print("path: {}".format(path))
        os.makedirs(path)
    else:
        ##print("Directories already exist")
        ##print("path: {}".format(path))
    return True
```

```
path = "./x/x/x"

MakeDirs(path)
```

this code does not assume "if path exists it is a directory"

```
import os

def MakeDirs(path):
    if os.path.exists(path):
        if not os.path.isdir(path)
            return False
    else:
        os.makedirs(path)
    return True
```

Notes:
1. see os.path.normpath documentation (os.path - Common pathname manipulations)

Get a list of files in a directory

```
import re
import os

dir = './'        # must end in '/'

# file name regular expression patterns
file_regx = []
file_regx.append(re.compile(r'\.html$'))
file_regx.append(re.compile(r'\.pdf$'))
file_regx.append(re.compile(r'\.png$'))
file_regx.append(re.compile(r'\.py$'))
file_regx.append(re.compile(r'\.txt$'))
file_regx.append(re.compile(r'\.css$'))

# ---------------------------------------------------------------------
# ---- get a list of files that match a regular expression
# ---------------------------------------------------------------------

def get_list_of_files(dir,file_regx):

    # --- get a list of entries in the directory

    entries = os.listdir(dir)

    # --- collect all directory entries that match
    # --- a regular expression

    files = []

    for f in entries:

        ff = dir + f

        if os.path.islink(ff): continue

        if not os.path.isfile(ff): continue

        for pat in file_regx:
            if pat.search(f):
                files.append(ff)
                break

    files.sort()

    return files

# ---------------------------------------------------------------------
# ---- main -----------------------------------------------------------
# ---------------------------------------------------------------------

# ---- does the directory to search/process exists?

if not os.path.isdir(dir):
    print('No directory found')
    quit()

# ---- fix the directory name string (if we need too)
# ---- it must end in '/' or be empty
```

```
if len(dir) > 0:
    if not re.search('\/$',dir):
        dir = dir + '/'

files = get_list_of_files(dir,file_regx)

print(f'files list length = {len(files)}')
```

Access Dictionary Using Sorted Keys

```
# ------------------------------------------------------------------
# process file names
#   dictionary key   = file name
#   dictionary value = path + file name
#
# this code has a problem if duplicate file names are found
# in different directories and placed in the dictionary
# (reverse keys and values or use path + file name as key)
# ------------------------------------------------------------------

def ProcessFileNameDictionary(dct):

    c = 0                       # initialize file name count

    for k in sorted(dct.keys()):

        print('key: {}, value: {}'.format(k,dct[k]))

        c += 1                  # increment file name count
```

New Dictionary - Switch Keys and Values

```
# ================================================================
# Reverse keys and Values (duplicates will be lost)
# ================================================================

# ---- test dictionary

dct = { 'aa':2, 'b':3, 'd':0, 'e':20, 'a':2, 'dd':9 }
print()
print(dct)

# ---- one way

new_dct_1 = dict(sorted(dct.items(), key=lambda item: item[1]))
print()
print(new_dct_1)

# ---- another slightly different way

new_dct_2 = {w:dct[w] for w in sorted(dct,key=dct.get,reverse=True)}
print()
print(new_dct_2)
```

Ask the user to make a selection (Menu)

```
# ---------------------------------------------------------------------
# ask user to make a selection
#
# Note: regexp should be a raw string
#       for example: AskUserToSelect(slist,r'\.py$')
# ---------------------------------------------------------------------

def AskUserToSelect(stitle,slist):

    l = len(slist)

    max = 30
    if l > max:
        print('')
        print('Error: Selection list is to long')
        print('len = {} - returning \'\''.format(max))
        return ''

    while True:

        # --- display list menu

        ClearScreen()

        print('==================================================')
        print(stitle)
        print('==================================================')
        print('##  File')
        print('--  ----------------------------------------')

        i = 0
        while i << l:
            print(' {}  {}'.format(i,slist[i]))
            i += 1

        # --- ask the user to select

        print('')
        sel = GetUserInput('Enter index of selection: ')

        # --- valid selection?

        if sel == '':          # no selection?
            return('')

        if sel.isdigit() != True:
            print('')
            print('Illegal selection entered ({})'
                    .format(sel))
            Pause()
            continue

        idx = int(sel)

        if idx < 0 or idx >= l:
            print('')
            print('Selection out of range ({})'
                   .format(idx))
            Pause()
```

```
            continue

        break

    # --- return selection

    return slist[idx]
```

```
title = "Select a Vacation Tour"

selections = [ "England - Lake Country",
               "France  - Wine Country",
               "Africa  - Safari",
               "Japan   - Shrines",
               "India   - Taj Mahal",
               "None" ]

selection = AskUserToSelect(title,selections)
```

Get/Set Object Attributes

```
class x():

    def set_value(self,v):
        self.value = v

    def get_value(self):
        return self.value

a = x()                         # create object

a.set_value(55)                 # set attribute
print(a.get_value())            # get attribute

setattr(a,'value',10)           # set attribute
print(getattr(a,'value'))       # get attribute

a.value = 100                   # set attribute
print(a.value)                  # get attribute
```

Note: If an attribute is not found, setattr() creates the attribute an assigns the value to it. This is only possible if the object implements the __dict__() method. You can check all the attributes of an object by using the dir() function.

Add function to object

```
def funcx(s):                    # external function
    print(s)

class classy():                  # class definition
    def funcy(self,s):
        print(s)

a = classy()                     # create object

a.zz = funcx                     # add function to object

a.funcy('builtin obj function') # execute built in function

a.zz('added obj function')      # execute added obj function
```

Resize Image

```
#!/usr/bin/python3
# ====================================================================
# resize an image
# ====================================================================

from PIL import Image

infile     = 'example.png'
outfile    = 'zzzzzzz.png'
new_height = 420

# ---- open image

image = Image.open(infile)

print()
print('---- input image -----------')
print(f'file  : {infile}')
print(f'width : {image.size[0]}')
print(f'height: {image.size[1]}')

# --- create a "new_width" based on "new_height"

new_width = int(float(image.size[0]) *
                (new_height / float(image.size[1])))

# ---- create a resized image

new_size = (new_width,new_height)

new_image = image.resize(new_size,resample=Image.BICUBIC)

## -------------------------------------------------------
## ---- see PIL.Image documentation for "resize" parameter
## ---- new_image = image.resize(new_size,Image.NEAREST)
## ---- (see FYI Links below)
## -------------------------------------------------------

# ---- save new image

new_image.save(outfile)

print()
print('---- output image ----------')
print(f'file : {outfile}')
print(f'width : {new_image.size[0]}')
print(f'height: {new_image.size[1]}')
```

Resize and display an image using PySimpleGUI

```
#!/usr/bin/python3
# ======================================================================
# Resize and display an image using PySimpleGUI
# ----------------------------------------------------------------------
# also try sg.Image('image file',size=(300,300))
# ----------------------------------------------------------------------
# From: stackoverflow.com/questions/67079155/displaying-an-image-
#       using-pysimplegui-without-having-to-use-an-event-listener
# ======================================================================

from PIL import Image, ImageTk
import PySimpleGUI as sg

filename = 'example.png'

# ---- Resize PNG file to size (300, 300)

size = (300, 300)
im = Image.open(filename)
im = im.resize(size, resample=Image.BICUBIC)

sg.theme('DarkGreen3')

layout = [
    [sg.Image(size=(300, 300), key='-IMAGE-')],
]
window = sg.Window('Window Title',layout,margins=(0,0),finalize=True)

# ---- Convert im to ImageTk.PhotoImage after window finalized

image = ImageTk.PhotoImage(image=im)

# ---- update image in sg.Image

window['-IMAGE-'].update(data=image)

# ---- event loop

while True:

    event, values = window.read()
    if event == sg.WIN_CLOSED:
        break

window.close()
```

argv and args

```
#!/usr/bin/python3
# ======================================================================
# comand line arguments
# ======================================================================

import sys

for arg in sys.argv:
    print(arg)
```

```
#!/usr/bin/python3
# ======================================================================
# function arguments - using the unpacking operator (*) and (**)
# ======================================================================

def funcA(*args):
    print(f'funcA *args is {type(args)}')
    print(args)

def funcB(**kwargs):
    print(f'funcA **kwargs is {type(kwargs)}')
    print(kwargs)

def funcC(*args,**kwargs):
    print(args)
    print(kwargs)


funcA(1,2,3,4,'A','B','C','D')

print('---------------------------------')

funcB(a=1,b=2)

try:
    funcB(99,98,a=1,b=2)
except Exception as e:
    print(str(e))

funcC(99,98,a=1,b=2)

print("funcC(a=1,b=2,99,98")
print("SyntaxError: positional argument follows keyword argument")

print("funcC(99,a=1,98,b=1)")
print('SyntaxError: positional argument Follows keyword argument')
```

bytes and bytearray

```
#!/usr/bin/python3
# ======================================================================
# test/play with bytes and bytearray data types
# Note: data type bytes are immutable
#       data type bytearray is mutable
# ======================================================================

print()
print('---- test bytes data type ----------------------------------')

a = [0,1,2,127,255,256]
print()
print(f'a = {a}')
try:
    x = bytes(a)
    print()
except Exception as e:
    print(str(e))

print()
a = [0,1,2,127,255]
print(f'a = {a}')
try:
    x = bytes(a)
    print(f'x = {x}')
    print('Try: x[0] = 45')
    x[0] = 45
except Exception as e:
    print(str(e))
print(f'x = {x}')

a = [0,1,2,127,255,ord('a')]
print()
print(f'a = {a}    (ord value of character \'a\')')
try:
    x = bytes(a)
    print()
except Exception as e:
    print(str(e))

print()
print('---- test bytearray data type ------------------------------')

a = [0,1,2,127,255,256]
print()
print(f'a = {a}')
try:
    x = bytearray(a)
    print()
except Exception as e:
    print(str(e))

print()
a = [0,1,2,127,255]
print(f'a = {a}')
try:
    x = bytearray(a)
    print(f'x = {x}')
    print('Try: x[0] = 45')
```

```
    x[0] = 45
except Exception as e:
    print(str(e))
print(f'x = {x}')

print()
print(f'(66 ord value of character \'B\')')
print(f'(97 ord value of character \'a\')')
a = [0,1,2,66,127,ord('a')]
for i in range(len(a)):
    print(f'[{i}] {type(a[i])}  {a[i]}')
try:
    x = bytearray(a)
    print()
except Exception as e:
    print(str(e))
print(f'x = {x}')

print()
print(['a','b','c'])
print(f'(97 ord value of character \'a\')')
print(f'(98 ord value of character \'b\')')
print(f'(99 ord value of character \'c\')')
a = [ord('a'),ord('b'),ord('c')]
print(f'a = {a}')
x = bytearray(a)
for i in range(len(a)):
    print(f'[{i}] {type(a[i])}  {a[i]}')
try:
    x = bytearray(a)
    print()
except Exception as e:
    print(str(e))
print(f'x = {x}')
```

Convert string to bytearray

```
a = bytearray(b'Mary had a little lamb.')
for x in a:
    print(x, end=', ')
print()

print('-------------------------------------------')
print()

txt = 'Mary had a little lamb.'
a = bytearray(txt.encode('utf-8'))
for x in a:
    print(x, end=', ')
print()
```

```
a = bytearray(b'Mary had \x263A a little lamb.')
print(a)
for x in a:
    print(f'{x:x} ({chr(x)})', end=', ')
print()
print(f'len={len(a)}')
print()

print('-------------------------------------------')
print()

txt = 'Mary had \u263A a little lamb.'
print(txt)
a = bytearray(txt.encode('utf-8'))
for x in a.decode('utf-8'):
    print(f'{ord(x):x} ({x})', end=', ')
print()
print(f'len={len(txt)}')
print()
```

Sort a List of Tuples

Each item in the list is a tuple. (list of tuples.) This code snippets creates a new list sorted by the second item in each tuple.

```
for idx,data in enumerated(old_list):
    new_list = sorted(data[1], key = lambda i: i[1])
```

2D Matrix (of all zeros)

```
row = 8
col = 8
mtx = [[0 for i in range(col)] for j in range(row)]

r = 4               # row index
c = 6               # col index
mtx[r][c] = 100
```

Title String

```
# ---- title string function --------------------

def title_str(title,length):
    if not title:
        t = '-' * length
    elif len(title) > length:
        t = title[:length]
    else:
        t = title + '-' * (length - len(title))
    return t

# ---- usage example ----------------------------

def display_data(data):
    print(title_str('---- data ',70))
    for x in data:
        print(x)
    print(title_str('',70))
```

JSON File Read/Write

```
# ---------------------------------------------------------------------
# ---- write JSON file
# ---------------------------------------------------------------------

def output_json_file(outfile,jdata):
    with open(outfile,'w') as ofile:
        ofile.write(json.dumps(jdata))

# ---------------------------------------------------------------------
# ---- read JSON file
# ---------------------------------------------------------------------

def input_json_file(infile):
    with open(infile,'r') as ifile:
        jdata = json.load(ifile)
        return jdata
```

Range Function for Floats

```
# ---- range function for floats

def frange(x,y,inc):
    while x < y:
        yield x
        x += inc
```

Start/End Web Page

```
start_of_web_page = '''\
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<meta name="author" content="{author}" />
<link rel="stylesheet" href="{css1}" />
<link rel="stylesheet" href="{css2}" />
<link rel="icon"          href="favicon.png" type="image/png" />
<link rel="shortcut icon" href="favicon.png" type="image/png" />
<title>{pagetitle}</title>
{style}
</head>
<body>
<header>
<center>{headertitle}</center>
</header>
<div class="indent12">
<p>'''


end_of_web_page = '''\
</p>
</div>
<footer>
<modate>Last Modified: {today}</modate>
</footer>
</body>
</html>'''


mystyle = '''\
<style>
modate { font-size: 0.8em; margin-left: 12px; }
</style>'''
```

```
import datetime as dt


def fstr(web_page,params):
    return web_page.format(**params)

print()
x = fstr(start_of_web_page,
    {'pagetitle'  :'Page Title',
     'author'     :'George',
     'css1'       :'style1.css',
     'css2'       :'style2.css',
     'style'      :'mystyle',
     'headertitle':'header Title'})
print(x)

print()
x = fstr(end_of_web_page,
    {'today': dt.datetime.now().strftime("%B %d,%Y %H:%M:%S")})
print(x,end='')
```

Try/Except

```
try:
    # Some Code
except:
    # Executed if error in the try block

---- for example ----------

try:
    x = 100
    y = 0
    result = x / y
except ZeroDivisionError:
    print("Sorry ! You are dividing by zero ")

---- for example ----------

try:
    x = 100
    y = 0
    result = x / y
except Exception as e:
    print(str(e))
```

```
try:
    # Some Code
except:
    # Executed if error in the try block
else:
    # execute if no exception
```

```
try:
    # Some Code
except:
    # Executed if error in the try block
else:
    # execute if no exception
finally:
    # Some code .....(always executed)
```

List Comprehension

```
# ---- process a list of integers get a list of
# ---- every other integer

elements = [1,2,3,4,5,6,7,8,9,0,10,11,12,13,14,15,16,17]

l = [element for index,element in enumerate(elements) if index % 2 == 1]

print(l)
```

Console Input UTF-8

```
#!/usr/bin/python3
# ============================================================
# Python3 - input UTF-8 characters
# test input example: abc\u00c5xyz
# Note: Python2 requires the "raw_input" function
# ============================================================

while True:

    x = input('"Enter a string containing UTF-8 characters:  ')

    x = x.strip()

    if not x: break

    print(x)
```

Random

```
from random import shuffle

names: list[str] = ['Tom', 'Judy', Jason',
                    'Barbara', 'Marie', 'Gretchen']

# ---- randomize a list (shuffle the list in place)

shuffle(names)
print(names)
```

```
from random import random, randint, randrange

# ---- random (random float in the range 0.0 <= X < 1.0)

value: float = random()
print(f'random() is {value}')

# ---- randint (random integer upper bound is included)

values: list[int] = [randint(10,20) for _ in range(5)]
print(randint() = {values})

# ---- randrange (random elements from range, upper bound is excluded)

values2: list[int] = [randrange(0,3) for _ in range(5)]
print(randrange() = {values2})

values3: list[int] = [randrange(0,10,2) for _ in range(5)]
print(randrange(with step = 2) = {values3})
```

```
from random import choice, choices

names: list[str] = ['Tom', 'Judy', Jason',
                    'Barbara', 'Marie', 'Gretchen']

# ---- select a random element from the list

print(f'choice() = {choice(names)')

# ---- select several random elements from the list
# ---- (the same element may be selected more that once)

print(f'choices() = {choices(names,k=3)}')

# ---- use selection weights (sum to 1.0)
# ---- (percentage probability a give element
# ----  will appear in the selected list)
weights: tuple = (.10, .20, .05, .25, .21, .29)
print(f'choices() = {choices(names,k=3,weights=weights)}')
```

```
from random import sample

# ---- the similar to choices except it will only
# ---- draw each element once from the list
# ---- every element in the list is considered unique
# ---- (e.g. [10, 10, 6, 7, 8] are all unique values)

print(samples(range(100),k=10)

# ---- something else we can do

colors: list[str] = ['r','g','b']
print(sample(colors,k=5,counts=(10,20,5)))
```

```
from random import seed

# ---- seed allows us to get consistent results back
# ---- (very important for testing)

seed(1491)  # generate the same random values each time
```

Last Modified: May 2024