មាតិការ

[មេរៀនទី ១: Introduction to Python 1](#_Toc62896058)

[1.1. Overview 1](#_Toc62896059)

[1.2. Environment Setup 3](#_Toc62896060)

[មេរៀនទី ២: Python-Basic Syntax 7](#_Toc62896061)

[2.1. Basic Syntax 7](#_Toc62896065)

[2.2. Variable Types 9](#_Toc62896067)

[2.3. Basic Operators 38](#_Toc62896068)

[មេរៀនទី ៣ Python-Control Flow Statement 44](#_Toc62896069)

[3.1. Decision Making or Condition 44](#_Toc62896073)

[3.2. Loop Statement 46](#_Toc62896074)

[មេរៀនទី ៤ Functions 49](#_Toc62896075)

[4.1. Basic Functions 49](#_Toc62896080)

[4.2. Scope of Variable 53](#_Toc62896081)

[មេរៀនទី ៥: Modules 56](#_Toc62896082)

[5.1. Build-In Module 56](#_Toc62896088)

[5.2. Define Module 56](#_Toc62896089)

[មេរៀនទី ៦: FILE I/O 60](#_Toc62896090)

[6.1. String vs Byte 60](#_Toc62896091)

[6.2. Opening and Closing File 64](#_Toc62896093)

[6.3. Reading and Writing Files 66](#_Toc62896094)

[6.4. Creating, Renaming and Deleting Files 68](#_Toc62896095)

[6.5. Directory in Python 69](#_Toc62896096)

[6.6. File and Directory Related Methods 69](#_Toc62896097)

[មេរៀនទី ៧: ERRORs AND EXCEPTIONs 76](#_Toc62896098)

[7.1. Exception Handling 76](#_Toc62896106)

[7.2. Debugging Tools 83](#_Toc62896107)

[7.3. Logging Module 84](#_Toc62896108)

[7.4. Python Test 88](#_Toc62896109)

[មេរៀនទី​ ៨: Python - Object Oriented 96](#_Toc62896110)

[8.1. សេចក្តីណែនាំ Introduction to OOP 96](#_Toc62896119)

[8.2. ការបង្កើត Class and Object 97](#_Toc62896120)

[8.3. ការបង្កើត Encapsulation(Data Hiding) 101](#_Toc62896121)

[8.4. ការបង្កើត Inheritance 104](#_Toc62896122)

[8.5. ការបង្កើត Polymorphism 105](#_Toc62896123)

[8.6. Reqular Expressions 108](#_Toc62896124)

[មេរៀនទី​ ៩: Python – GUI Programming 121](#_Toc62896140)

[9.1. សេចក្តីណែនាំ Introduction to GUI 121](#_Toc62896150)

[9.2. GUI-Tkinter Programming 121](#_Toc62896151)

[9.3. GUI-PyQt Programming 179](#_Toc62896152)

# មេរៀនទី ១: Introduction to Python

នៅក្នុងមេរៀននេះ យើងនឹងពន្យល់ពីការចាប់ផ្ដើមជាមួយភាសា Python ដោយមានការណែនាំជាមួយ កម្មវិធីដ៏សាមញ្ញជាច្រើន។

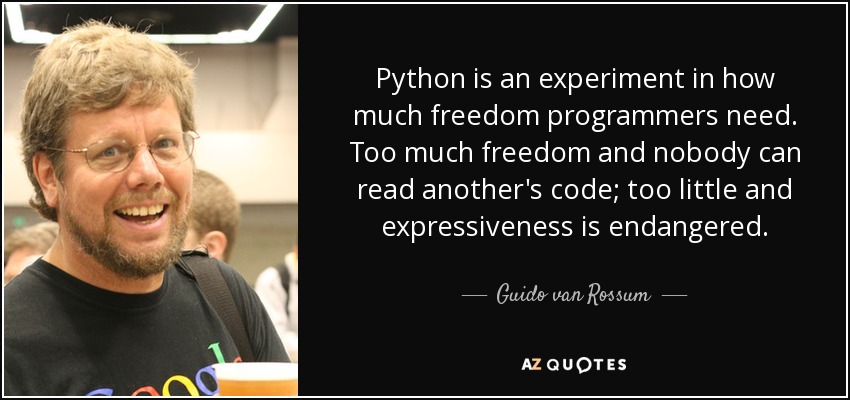
នៅក្នុងមេរៀននេះនឹងបង្ហាញអោយអ្នកមានសមត្ថភាពដូចខាងក្រោមៈ

* + ប្រើប្រាស់ទំរង់ទូទៅ និងរចនាសម្ព័ននៃភាសា Python
  + កំណត់អំពី console input/output (I/O)
  + របៀបតំឡើងកម្មវិធី Python
  + OOP Concept, File IO, GUI Programming

## Overview

Python គឺជាភាសា object-oriented programming language ដែល run លើ server ដើម្បីបង្កើតបានជា web application ដូចជាភាសា web មួយចំនួនផ្សេងទៀតដែរ PERL PHP ASP.NET JSP។ល។

ភាសា Python ត្រូវបានបង្កើតឡើងតាំងពីឆ្នាំ ១៩៩១ ដោយលោក Guido van Rossum នៅNational Research Institute for Mathematics and Computer Science ប្រទេស Netherlands។



គោលបំណងរបស់ Python គឺ៖

* + - web development (server-side),
    - software development,
    - mathematics,
    - system scripting.

**តើ Python អាចធ្វើអ្វីបានខ្លះ?**

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

**ហេតុអ្វីត្រូវប្រើភាសា Python?**

* **Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).**
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

**Python Features?**

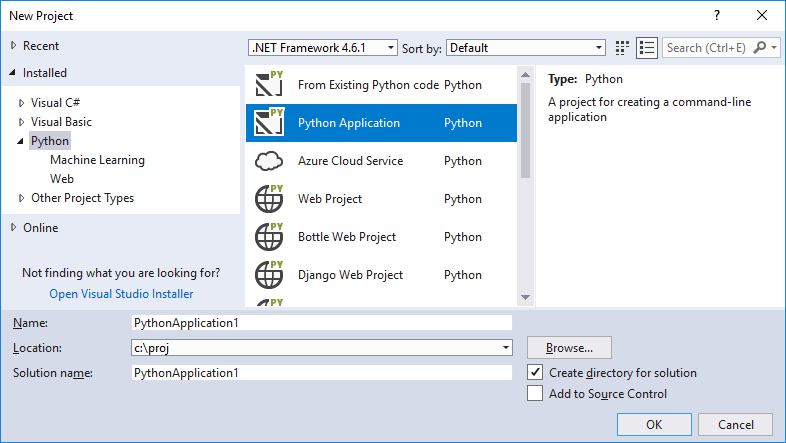
* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and **cross-platform** compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Realeased Version របស់ Python៖

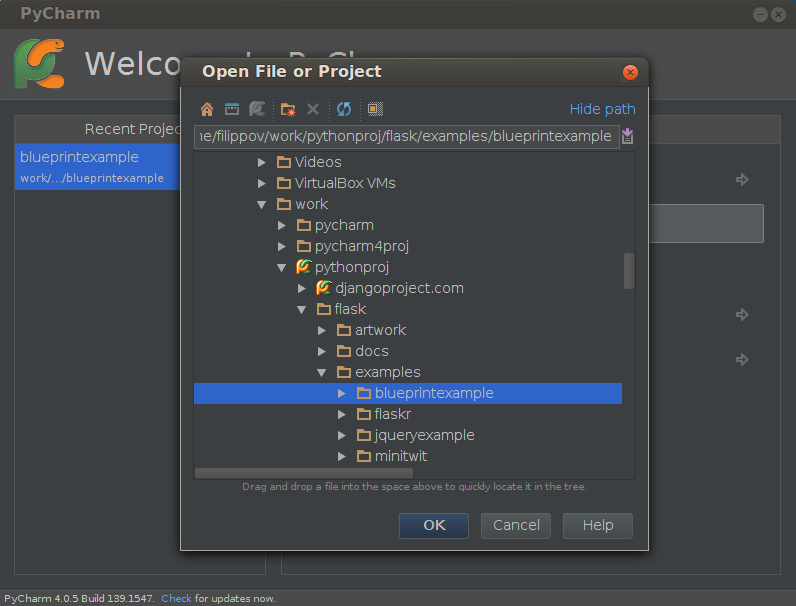
* Python 0.9.0 - February 20, 1991
* Python 0.9.1 - February, 1991
* Python 0.9.2 - Autumn, 1991
* Python 0.9.4 - December 24, 1991
* Python 0.9.5 - January 2, 1992
* Python 0.9.6 - April 6, 1992
* Python 0.9.8 - January 9, 1993
* Python 0.9.9 - July 29, 1993
* Python 1.0 - January 1994
* Python 1.2 - April 10, 1995
* Python 1.3 - October 12, 1995
* Python 1.4 - October 25, 1996
* Python 1.5 - December 31, 1997
* Python 1.6 - September 5, 2000
* Python 2.0 - October 16, 2000
* Python 2.1 - April 17, 2001
* Python 2.2 - December 21, 2001
* Python 2.3 - July 29, 2003
* Python 2.4 - November 30, 2004
* Python 2.5 - September 19, 2006
* Python 2.6 - October 1, 2008
* Python 2.7 - July 3, 2010
* Python 3.0 - December 3, 2008
* Python 3.1 - June 27, 2009
* Python 3.2 - February 20, 2011
* Python 3.3 - September 29, 2012
* Python 3.4 - March 16, 2014
* Python 3.5 - September 13, 2015
* Python 3.6 - December 23, 2016

## Environment Setup

យើងអាចតម្លើងកម្មវិធី Python លើ plateform ផ្សេងៗដូចជា Window, Linux, Mac OS ជាដើម។ បច្ចុប្បន្ននេះលោកអ្នកអាចសរសេរcode Python ជាមួយនឹង Microsoft Visual Studio IDE ក៏បាន ឬក៏ Pycham IDE(professional is license, community is free) ឬ Netbean IDE ឬ Qt Creater IDE ក៏បាន…។



Visual Studio IDE



PyCharm IDE

លោកអ្នកអាច Download Python & PyCharm តាម link ខាងក្រោម៖

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

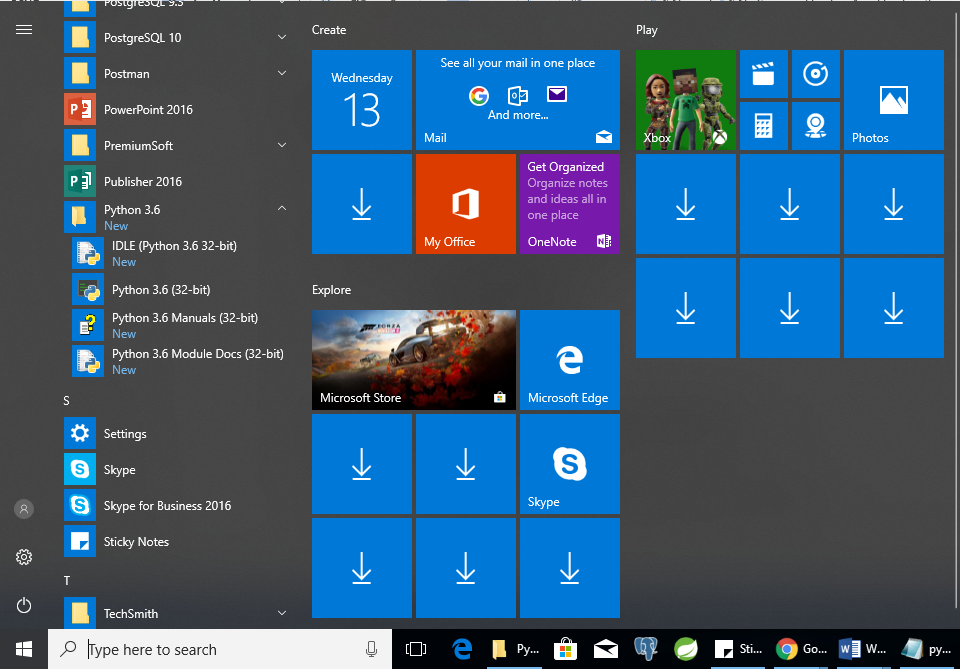
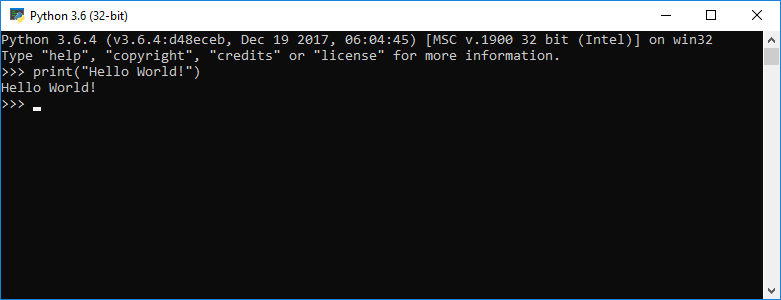
PyCharm : <https://www.jetbrains.com/pycharm/download/#section=windows>

Python Document : <https://www.python.org/doc/>

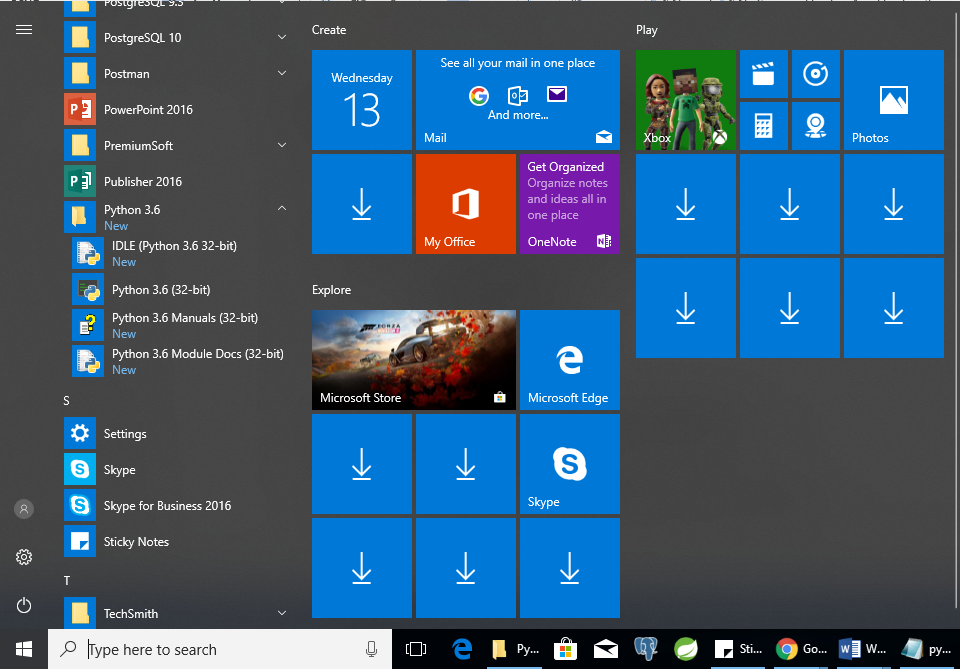
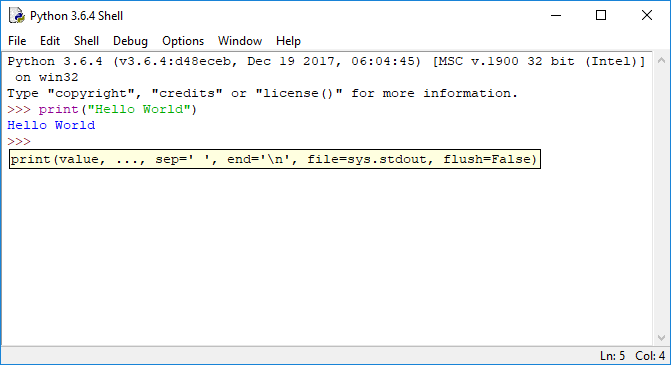
**Installing Python & PyCharm :**

[**https://www.youtube.com/watch?v=puBXxzcWJIQ**](https://www.youtube.com/watch?v=puBXxzcWJIQ)

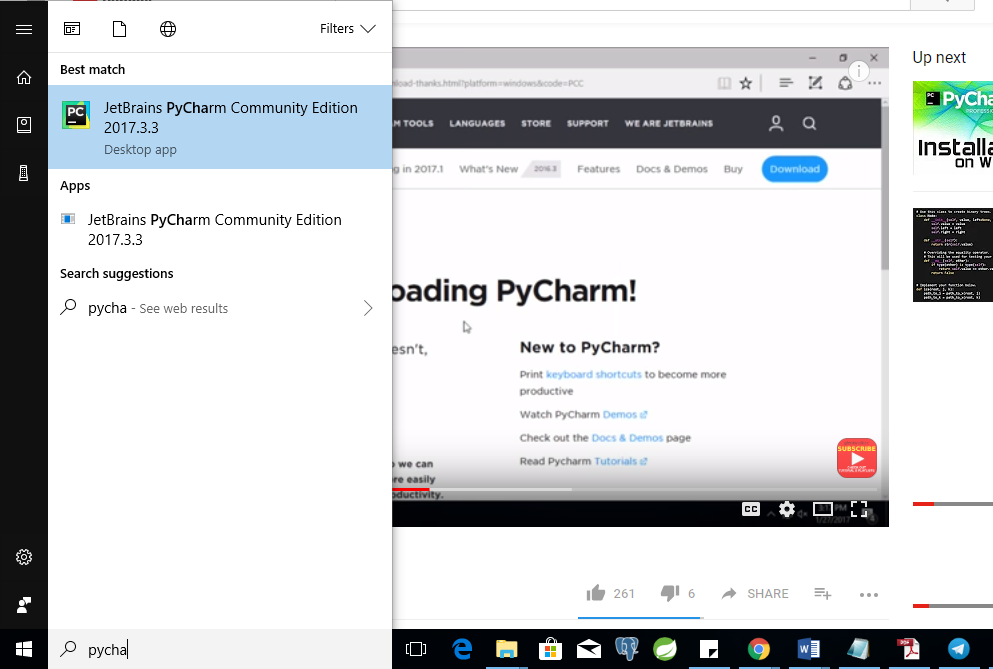
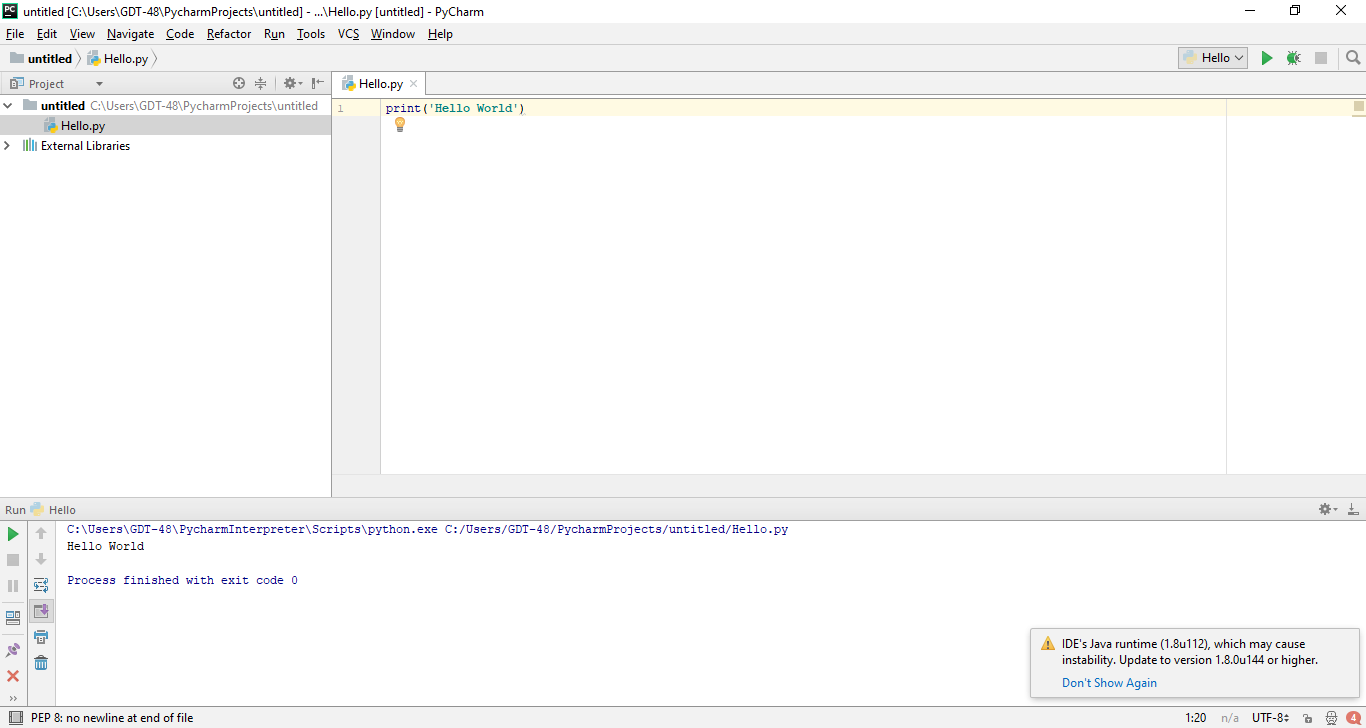
**របៀបដំណើរការកម្មវិធី Python**



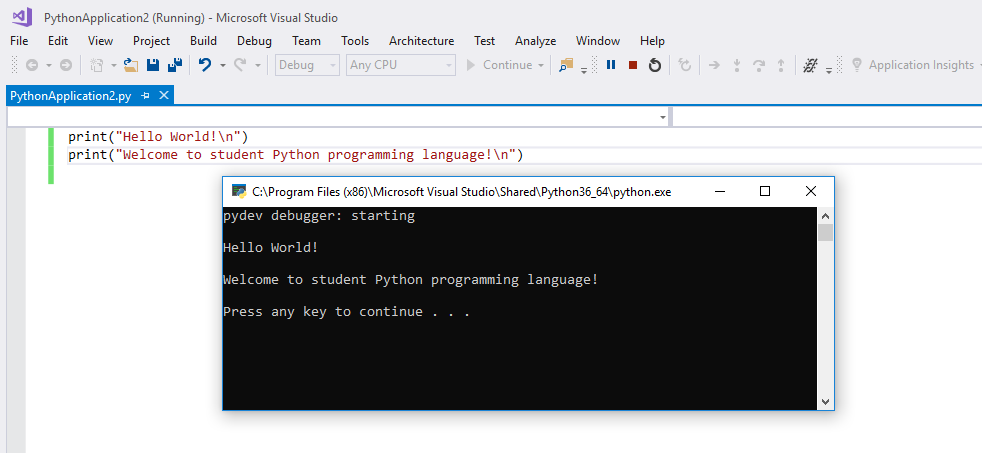
ទី ១- Python-Console



ទី ២- Python-Shell



ទី ៣- PyCharm IDE



ទី ៤- Visual Studio IDE(វាមានស្រាប់មកជាមួយ setup package របស់ visual studio)

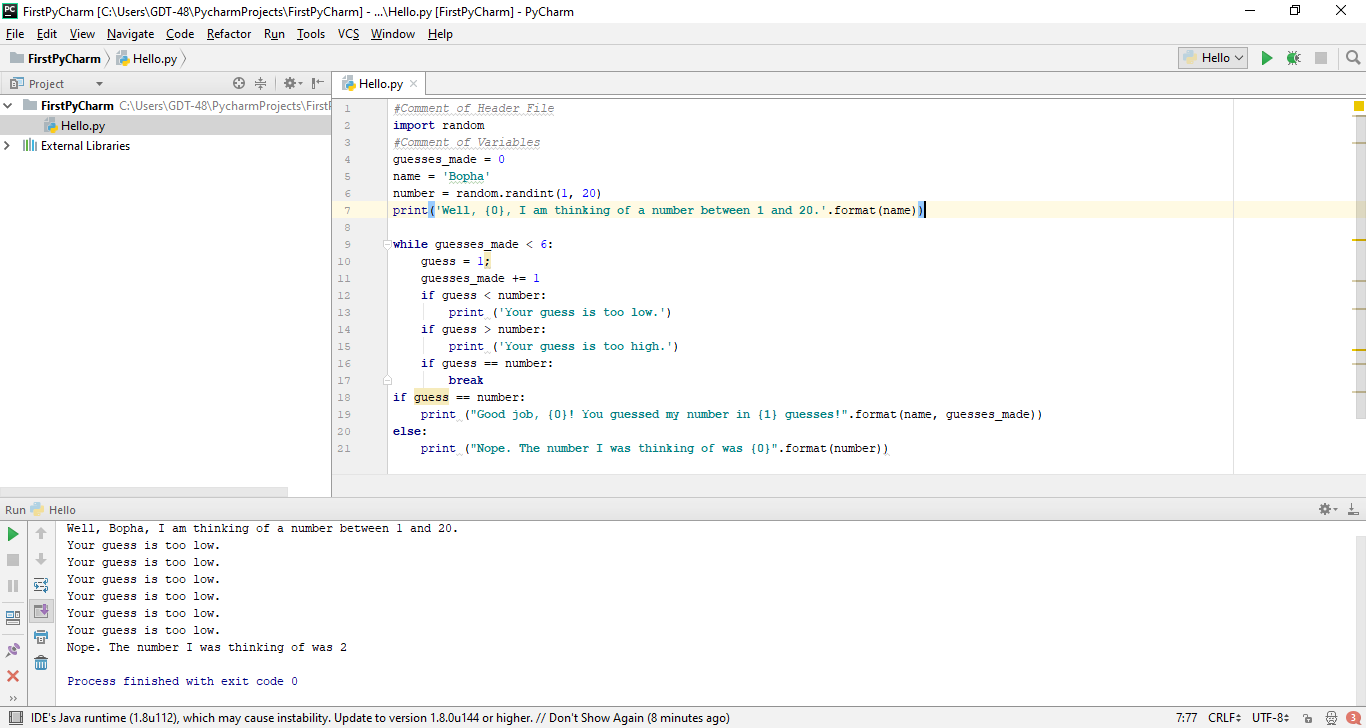
# មេរៀនទី ២: Python-Basic Syntax

នៅក្នុងមេរៀននេះ នឹងបង្ហាញអោយអ្នកមានសមត្ថភាពដូចខាងក្រោមៈ

* ស្គាល់ Header file
* ស្គាល់ Comment
* កំណត់ពីអ្វីទៅដែលហៅថា អថេរ រឺ អញ្ញាតិ ថេរ (Variables vs Constant)
* ស្គាល់ Data Type & built-in types
* ស្គាល់ Build-in function.
* ចេះប្រើប្រាស់ Python operators.



## Basic Syntax



**Using Comment** : គឺជាការបិទ code មិនអោយដំណើរការពេល execute program។ #(hash code) គឺជា comment ក្នុងភាសា python។

#Single Line Comment

""""

Multi-line

Comment in

Python Programming Language.

"""""

**Using Quoatation :** 'single quote' ឬ "double-quote" ត្រូវបានគេប្រើប្រាស់ដើម្បីផ្ទុកនូវតួរអក្សរ។ ''' បង្ហាញ single quote ឯ """ សម្រាប់បង្ហាញ double-quote។

**Using Statement :** គឺជារបៀបនៃការសេរឃ្លាបញ្ជានៅក្នុង Python៖

**Single Statement**

print ('Your guess is too low.') ឬ

print ('Your guess '

'is too low.') ឬ

print ('Your guess ' \

'is too low.')

**Multi-Statement**

name = 'Bopha'

number = random.randint(1, 20) ឬ

name = 'Bopha' ; number = random.randint(1, 20)

**New Line :** \n ត្រូវបានគេប្រើប្រាស់ដើម្បីចុះបន្ទាត់។

**Reserved Words :** ជាពពួក keyword Python ប្រើប្រាស់វាដើម្បីបញ្ជា app ឲ្យដំណើរការធ្វើអ្វីមួយ។ Keyword រួមមានមួយចំនួនដូចខាងក្រោម៖

|  |  |  |
| --- | --- | --- |
| and | exec | not |
| assert | finally | or |
| break | for | pass |
| class | from | print |
| continue | global | raise |
| def | if | return |
| del | import | try |
| elif | in | while |
| else | is | with |
| except | lambda | yield |

**Output :** ដើម្បីបង្ហាញលទ្ធផលគេប្រើ method print()

Ex1: print ('Your guess is too high.')

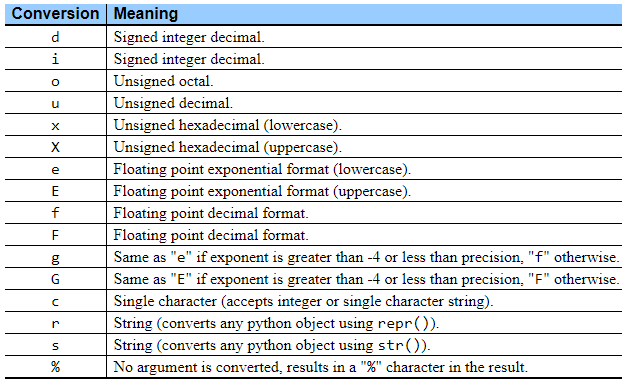
Ex2: print ("Good job, {0}! You guessed my number in {1} guesses!".format(name, guesses\_made))

Ex3: print("I"," Love ", " You ",end='',sep='')//no new line, and no space

**Input :** ដើម្បី scan យកទិន្នន័យតាម keyboard គេប្រើ method input ()

Ex1:

str=input("What is your name:");

print("your name is %s"%str);

Ex2:

print("What is your name:")

str=input();

print("your name is %s"%str);

**Exit App :** ដើម្បីបិទឬចាកចេញពីកម្មវិធីគេប្រើ function exit()។



## Variable Types

**Variable** : គឺជាទីតាំងធម្មតាមួយដែលត្រូវបានគេប្រើសំរាប់ផ្ទុកទិន្នន័យបណ្តោះអាសន្នក្នុងតំបន់ Memory។ យើងអាចផ្ទុកទិន្នន័យទាំងនោះនៅក្នុង **អថេរ រឺ អញ្ញាតិ**​ ដើម្បីយកទិន្នន័យទាំងនោះទៅធ្វើការគណនា រឺក៏ធ្វើជាការបង្ហាញផ្សេងៗ។ ប៉ុន្តែនៅក្នុងភាសា Python, data type របស់វាជា Variant-types ។

ច្បាប់ក្នុងការដាក់ឈ្មោះ variable

* + - ហាមផ្តើមដោយលេខ
    - ហាមផ្តើមដកឃ្លា
    - ហាមប្រើជាមួយនឹងសញ្ញាពិសេសផ្សេងៗ ដូចជា៖ # ? ! & | % @.......
    - ហាមផ្តើមជាន់ keyword

ទី ១ :

guesses\_made = 0

ទី ២ :

age=int(input())

ទី ៣ :

str=input("What is your name:")

**Note:** function **type(varName)** ប្រើសម្រាប់បង្ហាញប្រភេទនៃobject។

**Python Casting :** ប្រើដើម្បីបំលែងពីប្រភេទទិន្នន័យមួយទៅជាប្រភេទទិន្នន័យមួយផ្សេងទៀត។

int(x) : បំលែង x ទៅជា integer

long(x) : បំលែង x ទៅជា long integer

float(x) : បំលែង x ទៅជា float number

str(x) : បំលែង x ទៅជា string

complex(x) : to convert x to a complex number with real part x and imaginary part zero.

complex(x, y) to convert x and y to a complex number with real part x and imaginary part

y. x and y are numeric expressions

Ex1:

x = int(1) # x will be 1

y = int(2.8) # y will be 2

z = int("3") # z will be 3

Ex2:

x = float(1) # x will be 1.0

y = float(2.8) # y will be 2.8

z = float("3") # z will be 3.0

w = float("4.2") # w will be 4.2

Ex3:

x = str("s1") # x will be 's1'

y = str(2) # y will be '2'

z = str(3.0) # z will be '3.0'

**More on Data Type Conversion**

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Description** |
| 1 | **int(x [,base])**  Converts x to an integer. base specifies the base if x is a string. |
| 2 | **long(x [,base] )**  Converts x to a long integer. base specifies the base if x is a string. |
| 3 | **float(x)**  Converts x to a floating-point number. |
| 4 | **complex(real [,imag])**  Creates a complex number. |
| 5 | **str(x)**  Converts object x to a string representation. |
| 6 | **repr(x)**  Converts object x to an expression string. |
| 7 | **eval(str)**  Evaluates a string and returns an object. |
| 8 | **tuple(s)**  Converts s to a tuple. |
| 9 | **list(s)**  Converts s to a list. |
| 10 | **set(s)**  Converts s to a set. |
| 11 | **dict(d)**  Creates a dictionary. d must be a sequence of (key,value) tuples. |
| 12 | **frozenset(s)**  Converts s to a frozen set. |
| 13 | **chr(x)**  Converts an integer to a character. |
| 14 | **unichr(x)**  Converts an integer to a Unicode character. |
| 15 | **ord(x)**  Converts a single character to its integer value.(or convert char to ASCII) |
| 16 | **hex(x)**  Converts an integer to a hexadecimal string. |
| 17 | **oct(x)**  Converts an integer to an octal string. |

**Python Numbers :** មាន៤ប្រភេទគឺ int(signed integer), long(unlimited integer), float(float or real value), complex(complex number)។

Ex:

x = 1 # int

y = 2.8 # float

z = y+2j # complex : j or J is mean square root of -1

a = 2.5e2 # float can be in from of scientific notation

print(x)

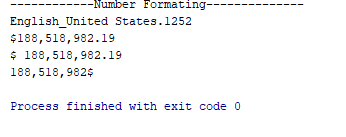
print(y)

print(z)

print(a)

print(type(x))

print(type(y))

print(type(z))

**ឧទាហរណ៍**៖ custom format លេខ

#!C:/Python3.6.4/python

import locale

print('------------Number Formating--------------')

print(locale.setlocale(locale.LC\_ALL,''))

value=188518982.1889

print(locale.currency(value, grouping=True))

print(locale.format\_string('$ %.2f',value, grouping=True))

d\_place=0

print(locale.format('%.\*f',(d\_place,value), grouping=True) + '$')

**Sample Numeric Value:**

|  |  |  |  |
| --- | --- | --- | --- |
| **int** | **long** | **float** | **complex** |
| 10 | 51924361L | 0.0 | 3.14j |
| 100 | -0x19323L | 15.20 | 45.j |
| -786 | 0122L | -21.9 | 9.322e-36j |
| 080 | 0xDEFABCECBDAECBFBAEL | 32.3+e18 | .876j |
| -0490 | 535633629843L | -90. | -.6545+0J |
| -0x260 | -052318172735L | -32.54e100 | 3e+26J |
| 0x69 | -4721885298529L | 70.2-E12 | 4.53e-7j |

**Mathematical Functions(import math, some no need import)**

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Returns ( description )** |
| 1 | [**abs(x)**](https://www.tutorialspoint.com/python/number_abs.htm)  The absolute value of x: the (positive) distance between x and zero. |
| 2 | [**ceil(x)**](https://www.tutorialspoint.com/python/number_ceil.htm)  The ceiling of x: the smallest integer not less than x |
| 3 | [**cmp(x, y)**](https://www.tutorialspoint.com/python/number_cmp.htm)  -1 if x < y, 0 if x == y, or 1 if x > y : compare function |
| 4 | [**exp(x)**](https://www.tutorialspoint.com/python/number_exp.htm)  The exponential of x: ex |
| 5 | [**fabs(x)**](https://www.tutorialspoint.com/python/number_fabs.htm)  The absolute value of x. return as float |
| 6 | [**floor(x)**](https://www.tutorialspoint.com/python/number_floor.htm)  The floor of x: the largest integer not greater than x |
| 7 | [**log(x)**](https://www.tutorialspoint.com/python/number_log.htm)  The natural logarithm of x, for x> 0 |
| 8 | [**log10(x)**](https://www.tutorialspoint.com/python/number_log10.htm)  The base-10 logarithm of x for x> 0. |
| 9 | [**max(x1, x2,...)**](https://www.tutorialspoint.com/python/number_max.htm)  The largest of its arguments: the value closest to positive infinity |
| 10 | [**min(x1, x2,...)**](https://www.tutorialspoint.com/python/number_min.htm)  The smallest of its arguments: the value closest to negative infinity |
| 11 | [**modf(x)**](https://www.tutorialspoint.com/python/number_modf.htm)  The fractional and integer parts of x in a two-item tuple. Both parts have the same sign as x. The integer part is returned as a float. |
| 12 | [**pow(x, y)**](https://www.tutorialspoint.com/python/number_pow.htm)  The value of x\*\*y. |
| 13 | [**round(x [,n])**](https://www.tutorialspoint.com/python/number_round.htm)  **x** rounded to n digits from the decimal point. Python rounds away from zero as a tie-breaker: round(0.5) is 1.0 and round(-0.5) is -1.0. |
| 14 | [**sqrt(x)**](https://www.tutorialspoint.com/python/number_sqrt.htm)  The square root of x for x > 0 |

**Random Number Functions**(ចាប់លេខដោយចៃដន្យ) (import random)

Random numbers ត្រូវបានគេនិយមប្រើប្រាស់សម្រាប់ games, simulations, testing, security, និង privacy applications។ Functions ទាំងនោះរួមមាន៖

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Description** |
| 1 | [**choice(seq)**](https://www.tutorialspoint.com/python/number_choice.htm)  A random item from a list, tuple, or string. print("Radom : ",random.choice((1,2,3)) |
| 2 | [**randrange ([start,] stop [,step])**](https://www.tutorialspoint.com/python/number_randrange.htm)  A randomly selected element from range(start, stop, step) |
| 3 | [**random()**](https://www.tutorialspoint.com/python/number_random.htm)  A random float r, such that 0 is less than or equal to r and r is less than 1 |
| 4 | [**seed([x])**](https://www.tutorialspoint.com/python/number_seed.htm)  Sets the integer starting value used in generating random numbers. Call this function before calling any other random module function. Returns None. |
| 5 | [**shuffle(lst)**](https://www.tutorialspoint.com/python/number_shuffle.htm)  Randomizes the items of a list in place. Returns None. |
| 6 | [**uniform(x, y)**](https://www.tutorialspoint.com/python/number_uniform.htm)  A random float r, such that x is less than or equal to r and r is less than y |

**Trigonometric Functions(អនុគមន៍ត្រីកោណមាត្រ) (import math)**

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Description** |
| 1 | [**acos(x)**](https://www.tutorialspoint.com/python/number_acos.htm)  Return the arc cosine of x, in radians. |
| 2 | [**asin(x)**](https://www.tutorialspoint.com/python/number_asin.htm)  Return the arc sine of x, in radians. |
| 3 | [**atan(x)**](https://www.tutorialspoint.com/python/number_atan.htm)  Return the arc tangent of x, in radians. |
| 4 | [**atan2(y, x)**](https://www.tutorialspoint.com/python/number_atan2.htm)  Return atan(y / x), in radians. |
| 5 | [**cos(x)**](https://www.tutorialspoint.com/python/number_cos.htm)  Return the cosine of x radians. |
| 6 | [**hypot(x, y)**](https://www.tutorialspoint.com/python/number_hypot.htm)  Return the Euclidean norm, sqrt(x\*x + y\*y). |
| 7 | [**sin(x)**](https://www.tutorialspoint.com/python/number_sin.htm)  Return the sine of x radians. |
| 8 | [**tan(x)**](https://www.tutorialspoint.com/python/number_tan.htm)  Return the tangent of x radians. |
| 9 | [**degrees(x)**](https://www.tutorialspoint.com/python/number_degrees.htm)  Converts angle x from radians to degrees. |
| 10 | [**radians(x)**](https://www.tutorialspoint.com/python/number_radians.htm)  Converts angle x from degrees to radians. |

**Mathematical Constants(import math)**

|  |  |
| --- | --- |
| **Sr.No.** | **Constants & Description** |
| 1 | **pi**  The mathematical constant pi. |
| 2 | **e**  The mathematical constant e. |

**Python Strings :**

'hello' is the same as "hello".

**Ex1**: Get the character at position 1:

a = "hello"

print(a[1])

**Ex2**: Substring. Get the characters from index 2 to position 5:

b = "world"

print(b[2:5])

**Ex3**: The strip() method removes any whitespace from the beginning or the end:

a = " Hello, World! "

print(a.strip()) # returns "Hello, World!"

**Ex4**: The len() method returns the length of a string:

a = "Hello, World!"

print(len(a))

**Ex5**: The lower() method returns the string in lower case:

a = "Hello, World!"

print(a.lower())

**Ex6**: The upper() method returns the string in upper case:

a = "Hello, World!"

print(a.upper())

**Ex7**: The replace() method replaces a string with another string:

a = "Hello, World!"

print(a.replace("H", "J"))

**Ex8**: The split() method splits the string into substrings if it finds instances of the separator:

a = "Hello, World!"

print(a.split(",")) # returns ['Hello', ' World!']

**Unicode String :**

print(u"សួស្តីលោកគ្រូ")

**Notation :**

**Escape Characters**

|  |  |  |
| --- | --- | --- |
| **Backslash notation** | **Hexadecimal character** | **Description** |
| \a | 0x07 | Bell or alert |
| \b | 0x08 | Backspace |
| \cx |  | Control-x |
| \C-x |  | Control-x |
| \e | 0x1b | Escape |
| \f | 0x0c | Formfeed |
| \M-\C-x |  | Meta-Control-x |
| \n | 0x0a | Newline |
| \nnn |  | Octal notation, where n is in the range 0.7 |
| \r | 0x0d | Carriage return |
| \s | 0x20 | Space |
| \t | 0x09 | Tab |
| \v | 0x0b | Vertical tab |
| \x |  | Character x |
| \xnn |  | Hexadecimal notation, where n is in the range 0.9, a.f, or A.F |

**String Special Operators(ex: a=”a” , b=”b”)**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Concatenation - Adds values on either side of the operator | a + b will give HelloPython |
| \* | Repetition - Creates new strings, concatenating multiple copies of the same string | a\*2 will give -HelloHello |
| [] | Slice - Gives the character from the given index | a[1] will give e |
| [ : ] | Range Slice - Gives the characters from the given range | a[1:4] will give ell |
| in | Membership - Returns true if a character exists in the given string | H in a will give 1 |
| not in | Membership - Returns true if a character does not exist in the given string | M not in a will give 1 |
| r/R | Raw String - Suppresses actual meaning of Escape characters. The syntax for raw strings is exactly the same as for normal strings with the exception of the raw string operator, the letter "r," which precedes the quotation marks. The "r" can be lowercase (r) or uppercase (R) and must be placed immediately preceding the first quote mark. | print r'\n' prints \n and print R'\n'prints \n |
| % | Format - Performs String formatting | See at next section |

**String Formatting Operator**

print "My name is %s and weight is %d kg!" % ('Zara', 21)

|  |  |
| --- | --- |
| **Format Symbol** | **Conversion** |
| %c | character |
| %s | string conversion via str() prior to formatting |
| %i | signed decimal integer |
| %d | signed decimal integer |
| %u | unsigned decimal integer |
| %o | octal integer12 |
| %x | hexadecimal integer (lowercase letters) |
| %X | hexadecimal integer (UPPERcase letters) |
| %e | exponential notation (with lowercase 'e') |
| %E | exponential notation (with UPPERcase 'E') |
| %f | floating point real number |
| %g | the shorter of %f and %e |
| %G | the shorter of %f and %E |

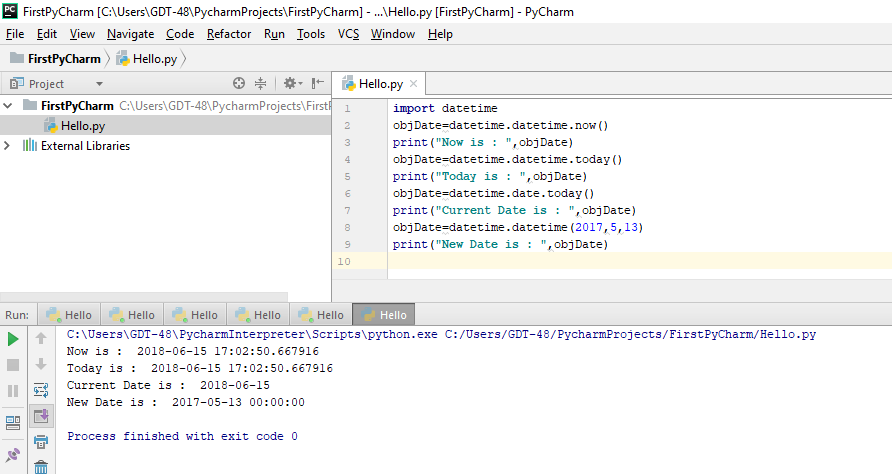
**Other supported symbols and functionality are listed in the following table**

|  |  |
| --- | --- |
| **Symbol** | **Functionality** |
| \* | argument specifies width or precision |
| - | left justification |
| + | display the sign |
| <sp> | leave a blank space before a positive number |
| # | add the octal leading zero ( '0' ) or hexadecimal leading '0x' or '0X', depending on whether 'x' or 'X' were used. |
| 0 | pad from left with zeros (instead of spaces) |
| % | '%%' leaves you with a single literal '%' |
| (var) | mapping variable (dictionary arguments) |
| m.n. | m is the minimum total width and n is the number of digits to display after the decimal point (if appl.) |

**Built-in String Methods**

|  |  |
| --- | --- |
| **Sr.No.** | **Methods with Description** |
| 1 | [**capitalize()**](https://www.tutorialspoint.com/python/string_capitalize.htm)  Capitalizes first letter of string |
| 2 | [**center(width, fillchar)**](https://www.tutorialspoint.com/python/string_center.htm)  Returns a space-padded string with the original string centered to a total of width columns. |
| 3 | [**count(str, beg= 0,end=len(string))**](https://www.tutorialspoint.com/python/string_count.htm)  Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given. |
| 4 | [**decode(encoding='UTF-8',errors='strict')**](https://www.tutorialspoint.com/python/string_decode.htm)  Decodes the string using the codec registered for encoding. encoding defaults to the default string encoding. |
| 5 | [**encode(encoding='UTF-8',errors='strict')**](https://www.tutorialspoint.com/python/string_encode.htm)  Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'. |
| 6 | [**endswith(suffix, beg=0, end=len(string))**](https://www.tutorialspoint.com/python/string_endswith.htm)  Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; returns true if so and false otherwise. |
| 7 | [**expandtabs(tabsize=8)**](https://www.tutorialspoint.com/python/string_expandtabs.htm)  Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided. |
| 8 | [**find(str, beg=0 end=len(string))**](https://www.tutorialspoint.com/python/string_find.htm)  Determine if str occurs in string or in a substring of string if starting index beg and ending index end are given returns index if found and -1 otherwise. |
| 9 | [**index(str, beg=0, end=len(string))**](https://www.tutorialspoint.com/python/string_index.htm)  Same as find(), but raises an exception if str not found. |
| 10 | [**isalnum()**](https://www.tutorialspoint.com/python/string_isalnum.htm)  Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise. |
| 11 | [**isalpha()**](https://www.tutorialspoint.com/python/string_isalpha.htm)  Returns true if string has at least 1 character and all characters are alphabetic and false otherwise. |
| 12 | [**isdigit()**](https://www.tutorialspoint.com/python/string_isdigit.htm)  Returns true if string contains only digits and false otherwise. |
| 13 | [**islower()**](https://www.tutorialspoint.com/python/string_islower.htm)  Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise. |
| 14 | [**isnumeric()**](https://www.tutorialspoint.com/python/string_isnumeric.htm)  Returns true if a unicode string contains only numeric characters and false otherwise. |
| 15 | [**isspace()**](https://www.tutorialspoint.com/python/string_isspace.htm)  Returns true if string contains only whitespace characters and false otherwise. |
| 16 | [**istitle()**](https://www.tutorialspoint.com/python/string_istitle.htm)  Returns true if string is properly "titlecased" and false otherwise. |
| 17 | [**isupper()**](https://www.tutorialspoint.com/python/string_isupper.htm)  Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise. |
| 18 | [**join(seq)**](https://www.tutorialspoint.com/python/string_join.htm)  Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string. |
| 19 | [**len(string)**](https://www.tutorialspoint.com/python/string_len.htm)  Returns the length of the string |
| 20 | [**ljust(width[, fillchar])**](https://www.tutorialspoint.com/python/string_ljust.htm)  Returns a space-padded string with the original string left-justified to a total of width columns. |
| 21 | [**lower()**](https://www.tutorialspoint.com/python/string_lower.htm)  Converts all uppercase letters in string to lowercase. |
| 22 | [**lstrip()**](https://www.tutorialspoint.com/python/string_lstrip.htm)  Removes all leading whitespace in string. |
| 23 | [**maketrans()**](https://www.tutorialspoint.com/python/string_maketrans.htm)  Returns a translation table to be used in translate function. |
| 24 | [**max(str)**](https://www.tutorialspoint.com/python/string_max.htm)  Returns the max alphabetical character from the string str. |
| 25 | [**min(str)**](https://www.tutorialspoint.com/python/string_min.htm)  Returns the min alphabetical character from the string str. |
| 26 | [**replace(old, new [, max])**](https://www.tutorialspoint.com/python/string_replace.htm)  Replaces all occurrences of old in string with new or at most max occurrences if max given. |
| 27 | [**rfind(str, beg=0,end=len(string))**](https://www.tutorialspoint.com/python/string_rfind.htm)  Same as find(), but search backwards in string. |
| 28 | [**rindex( str, beg=0, end=len(string))**](https://www.tutorialspoint.com/python/string_rindex.htm)  Same as index(), but search backwards in string. |
| 29 | [**rjust(width,[, fillchar])**](https://www.tutorialspoint.com/python/string_rjust.htm)  Returns a space-padded string with the original string right-justified to a total of width columns. |
| 30 | [**rstrip()**](https://www.tutorialspoint.com/python/string_rstrip.htm)  Removes all trailing whitespace of string. |
| 31 | [**split(str="", num=string.count(str))**](https://www.tutorialspoint.com/python/string_split.htm)  Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given. |
| 32 | [**splitlines( num=string.count('\n'))**](https://www.tutorialspoint.com/python/string_splitlines.htm)  Splits string at all (or num) NEWLINEs and returns a list of each line with NEWLINEs removed. |
| 33 | [**startswith(str, beg=0,end=len(string))**](https://www.tutorialspoint.com/python/string_startswith.htm)  Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise. |
| 34 | [**strip([chars])**](https://www.tutorialspoint.com/python/string_strip.htm)  Performs both lstrip() and rstrip() on string. |
| 35 | [**swapcase()**](https://www.tutorialspoint.com/python/string_swapcase.htm)  Inverts case for all letters in string. |
| 36 | [**title()**](https://www.tutorialspoint.com/python/string_title.htm)  Returns "titlecased" version of string, that is, all words begin with uppercase and the rest are lowercase. |
| 37 | [**translate(table, deletechars="")**](https://www.tutorialspoint.com/python/string_translate.htm)  Translates string according to translation table str(256 chars), removing those in the del string. |
| 38 | [**upper()**](https://www.tutorialspoint.com/python/string_upper.htm)  Converts lowercase letters in string to uppercase. |
| 39 | [**zfill (width)**](https://www.tutorialspoint.com/python/string_zfill.htm)  Returns original string leftpadded with zeros to a total of width characters; intended for numbers, zfill() retains any sign given (less one zero). |
| 40 | [**isdecimal()**](https://www.tutorialspoint.com/python/string_isdecimal.htm)  Returns true if a unicode string contains only decimal characters and false otherwise. |

**Python Dates & Time** : ក្នុងភាសារបស់ Python Date & Time គឺមិនមែនជា data type នោះទេយើងអាចប្រើប្រាស់វាបានតាមមធ្យោបាយច្រើនផ្សេងៗគ្នា។

**Using datetime class :**

import datetime

objDate=datetime.datetime.now()

print("Now is : ",objDate)

objDate=datetime.datetime.today()

print("Today is : ",objDate)

objDate=datetime.date.today()

print("Current Date is : ",objDate)

objDate=datetime.datetime(2017,5,13)

print("New Date is : ",objDate)

**The strftime() and strptime() Method on datetime class**

import datetime

x = datetime.datetime(2018, 6, 1)

print(x.strftime("%d-%B-%Y"))

#Convert string to datetime

dob=datetime.datetime.strptime("1986-02-02","%Y-%m-%d")

print("DoB=",dob.strftime("%Y-%m-%d"))

|  |  |  |
| --- | --- | --- |
| **Directive** | **Description** | **Example** |
| %a | Weekday, short version | Wed |
| %A | Weekday, full version | Wednesday |
| %w | Weekday as a number 0-6, 0 is Sunday | 3 |
| %d | Day of month 01-31 | 31 |
| %b | Month name, short version | Dec |
| %B | Month name, full version | December |
| %m | Month as a number 01-12 | 12 |
| %y | Year, short version, without century | 18 |
| %Y | Year, full version | 2018 |
| %H | Hour 00-23 | 17 |
| %I | Hour 00-12 | 05 |
| %p | AM/PM | PM |
| %M | Minute 00-59 | 41 |
| %S | Second 00-59 | 08 |
| %f | Microsecond 000000-999999 | 548513 |
| %z | UTC offset | +0100 |
| %Z | Timezone | CST |
| %j | Day number of year 001-366 | 365 |
| %U | Week number of year, Sunday as the first day of week, 00-53 | 52 |
| %W | Week number of year, Monday as the first day of week, 00-53 | 52 |
| %c | Local version of date and time | Mon Dec 31 17:41:00 2018 |
| %x | Local version of date | 12/31/18 |
| %X | Local version of time | 17:41:00 |
| %% | A % character | % |

**Using time class:**

import time

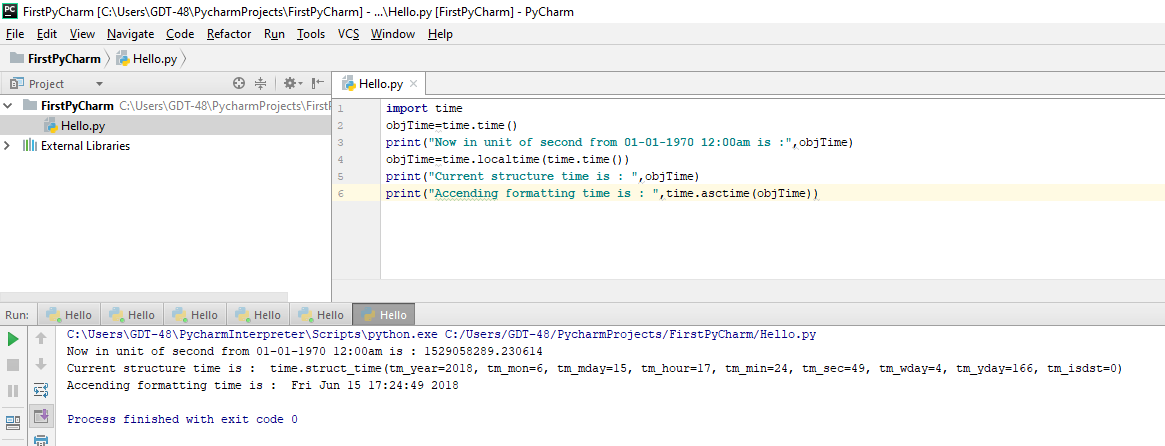
objTime=time.time()

print("Now in unit of second from 01-01-1970 12:00am is :",objTime)

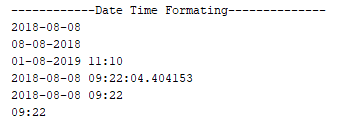
objTime=time.localtime(time.time())

print("Current structure time is : ",objTime)

print("Accending formatting time is : ",time.asctime(objTime))



**ឧទាហរណ៍៖ Custom Date and Time Formating**

#!C:/Python3.6.4/python

import datetime

import time

print('------------Date Time Formating--------------')

today=datetime.date.today()

print(today)

print(today.strftime('%d-%m-%Y'))

print(datetime.datetime(2019,8,1,11,10,11).strftime('%d-%m-%Y %H:%M'))

print(datetime.datetime.now())

print(time.strftime("%Y-%m-%d %H:%M"))

print(time.strftime("%H:%M"))

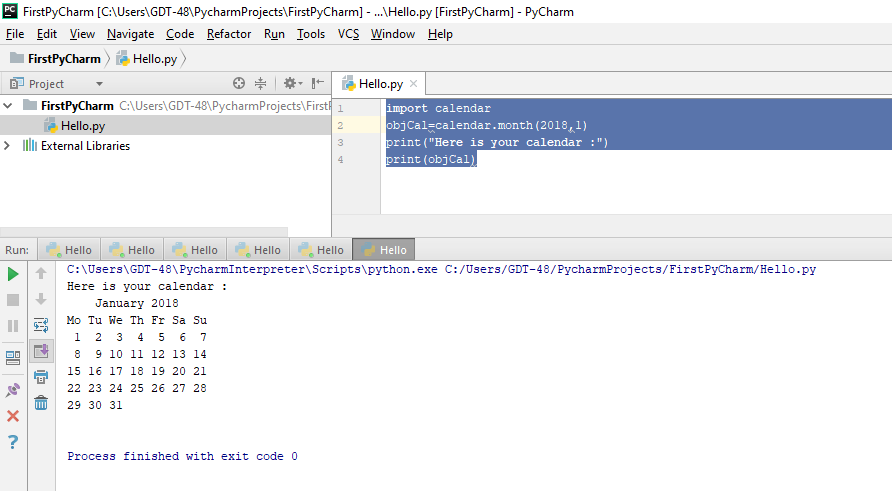
**Using calendar class:**

import calendar

objCal=calendar.month(2018,1)

print("Here is your calendar :")

print(objCal)



**More on Time Function: (import time)**

|  |  |
| --- | --- |
| **Sr.No.** | **Function with Description** |
| 1 | [**time.altzone**](https://www.tutorialspoint.com/python/time_altzone.htm)  The offset of the local DST timezone, in seconds west of UTC, if one is defined. This is negative if the local DST timezone is east of UTC (as in Western Europe, including the UK). Only use this if daylight is nonzero. |
| 2 | [**time.asctime([tupletime])**](https://www.tutorialspoint.com/python/time_asctime.htm)  Accepts a time-tuple and returns a readable 24-character string such as 'Tue Dec 11 18:07:14 2008'. |
| 3 | [**time.clock( )**](https://www.tutorialspoint.com/python/time_clock.htm)  Returns the current CPU time as a floating-point number of seconds. To measure computational costs of different approaches, the value of time.clock is more useful than that of time.time(). |
| 4 | [**time.ctime([secs])**](https://www.tutorialspoint.com/python/time_ctime.htm)  Like asctime(localtime(secs)) and without arguments is like asctime( ) |
| 5 | [**time.gmtime([secs])**](https://www.tutorialspoint.com/python/time_gmtime.htm)  Accepts an instant expressed in seconds since the epoch and returns a time-tuple t with the UTC time. Note : t.tm\_isdst is always 0 |
| 6 | [**time.localtime([secs])**](https://www.tutorialspoint.com/python/time_localtime.htm)  Accepts an instant expressed in seconds since the epoch and returns a time-tuple t with the local time (t.tm\_isdst is 0 or 1, depending on whether DST applies to instant secs by local rules). |
| 7 | [**time.mktime(tupletime)**](https://www.tutorialspoint.com/python/time_mktime.htm)  Accepts an instant expressed as a time-tuple in local time and returns a floating-point value with the instant expressed in seconds since the epoch. |
| 8 | [**time.sleep(secs)**](https://www.tutorialspoint.com/python/time_sleep.htm)  Suspends the calling thread for secs seconds. |
| 9 | [**time.strftime(fmt[,tupletime])**](https://www.tutorialspoint.com/python/time_strftime.htm)  Accepts an instant expressed as a time-tuple in local time and returns a string representing the instant as specified by string fmt. |
| 10 | [**time.strptime(str,fmt='%a %b %d %H:%M:%S %Y')**](https://www.tutorialspoint.com/python/time_strptime.htm)  Parses str according to format string fmt and returns the instant in time-tuple format. |
| 11 | [**time.time( )**](https://www.tutorialspoint.com/python/time_time.htm)  Returns the current time instant, a floating-point number of seconds since the epoch. |
| 12 | [**time.tzset()**](https://www.tutorialspoint.com/python/time_tzset.htm)  Resets the time conversion rules used by the library routines. The environment variable TZ specifies how this is done. |

**More on Time Attribute/Property**

|  |  |
| --- | --- |
| **Sr.No.** | **Attribute with Description** |
| 1 | **time.timezone**  Attribute time.timezone is the offset in seconds of the local time zone (without DST) from UTC (>0 in the Americas; <=0 in most of Europe, Asia, Africa). |
| 2 | **time.tzname**  Attribute time.tzname is a pair of locale-dependent strings, which are the names of the local time zone without and with DST, respectively. |

**More on Calendar Function:**

|  |  |
| --- | --- |
| **Sr.No.** | **Function with Description** |
| 1 | **calendar.calendar(year,w=2,l=1,c=6)**  Returns a multiline string with a calendar for year year formatted into three columns separated by c spaces. w is the width in characters of each date; each line has length 21\*w+18+2\*c. l is the number of lines for each week. |
| 2 | **calendar.firstweekday( )**  Returns the current setting for the weekday that starts each week. By default, when calendar is first imported, this is 0, meaning Monday. |
| 3 | **calendar.isleap(year)**  Returns True if year is a leap year; otherwise, False. |
| 4 | **calendar.leapdays(y1,y2)**  Returns the total number of leap days in the years within range(y1,y2). |
| 5 | **calendar.month(year,month,w=2,l=1)**  Returns a multiline string with a calendar for month month of year year, one line per week plus two header lines. w is the width in characters of each date; each line has length 7\*w+6. l is the number of lines for each week. |
| 6 | **calendar.monthcalendar(year,month)**  Returns a list of lists of ints. Each sublist denotes a week. Days outside month month of year year are set to 0; days within the month are set to their day-of-month, 1 and up. |
| 7 | **calendar.monthrange(year,month)**  Returns two integers. The first one is the code of the weekday for the first day of the month month in year year; the second one is the number of days in the month. Weekday codes are 0 (Monday) to 6 (Sunday); month numbers are 1 to 12. |
| 8 | **calendar.prcal(year,w=2,l=1,c=6)**  Like print calendar.calendar(year,w,l,c). |
| 9 | **calendar.prmonth(year,month,w=2,l=1)**  Like print calendar.month(year,month,w,l). |
| 10 | **calendar.setfirstweekday(weekday)**  Sets the first day of each week to weekday code weekday. Weekday codes are 0 (Monday) to 6 (Sunday). |
| 11 | **calendar.timegm(tupletime)**  The inverse of time.gmtime: accepts a time instant in time-tuple form and returns the same instant as a floating-point number of seconds since the epoch. |
| 12 | **calendar.weekday(year,month,day)**  Returns the weekday code for the given date. Weekday codes are 0 (Monday) to 6 (Sunday); month numbers are 1 (January) to 12 (December). |

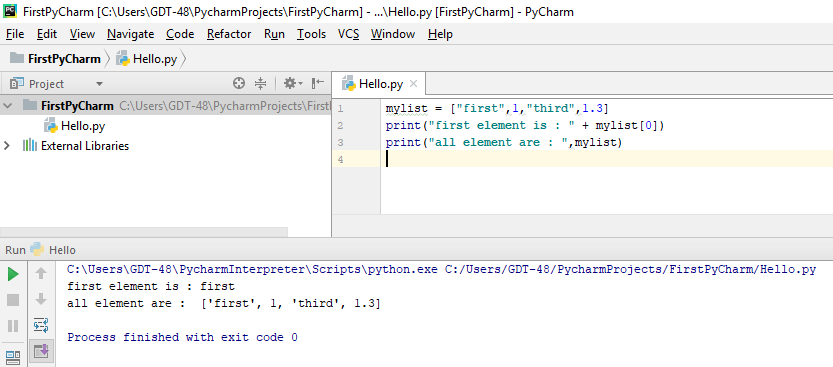
**More on DateTime Type**

-The **pytz** Module

-The **dateutil** Module

**Python Lists :** វាជា Python Collections (Arrays) ឬជា compound data types ដែលមានលំដាប់ អាចផ្លាស់ប្តូរ ហើយអាច duplicate members ។ ជាទូទៅ lists ត្រូវសរសេរក្នុង [ ] ហើយធាតុនីមួយៗរបស់វាត្រូវកាត់ផ្តាច់ពីគ្នាដោយសញ្ញា , ។

ឧទាហរណ៍១



ឧទាហរណ៍២

thislist = ["apple", "banana", "cherry"]

thislist[1] = "blackcurrant"

print(thislist)

ឧទាហរណ៍៣- Using the list() constructor to make a List:

thislist = list(("apple", "banana", "cherry")) # note the double round-brackets

print(thislist)

ឧទាហរណ៍៤- Using the append() method to append an item:

thislist = list(("apple", "banana", "cherry"))

thislist.append("damson")

print(thislist)

ឧទាហរណ៍៥- Accessing Values in Lists

list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5, 6, 7 ];

print ("list1[0]: ", list1[0])

print ("list2[1:5]: ", list2[1:5])

ឧទាហរណ៍៦- Updating Lists

list = ['physics', 'chemistry', 1997, 2000];

print ("Value available at index 2 : ")

print (list[2])

list[2] = 2001;

print ("New value available at index 2 : ")

print (list[2])

ឧទាហរណ៍៧- Delete List Elements

list1 = ['physics', 'chemistry', 1997, 2000];

print (list1)

del (list1[2]);

print ("After deleting value at index 2 : ")

print (list1)

**Basic List Operations**

|  |  |  |
| --- | --- | --- |
| **Python Expression** | **Results** | **Description** |
| len([1, 2, 3]) | 3 | Length |
| [1, 2, 3] + [4, 5, 6] | [1, 2, 3, 4, 5, 6] | Concatenation |
| ['Hi!'] \* 4 | ['Hi!', 'Hi!', 'Hi!', 'Hi!'] | Repetition |
| 3 in [1, 2, 3] | True | Membership |
| for x in [1, 2, 3]: print x, | 1 2 3 | Iteration |

**Built-in List Functions & Methods**

|  |  |
| --- | --- |
| **Sr.No.** | **Function with Description** |
| 1 | [**cmp(list1, list2)**](https://www.tutorialspoint.com/python/list_cmp.htm)  Compares elements of both lists. |
| 2 | [**len(list)**](https://www.tutorialspoint.com/python/list_len.htm)  Gives the total length of the list. |
| 3 | [**max(list)**](https://www.tutorialspoint.com/python/list_max.htm)  Returns item from the list with max value. |
| 4 | [**min(list)**](https://www.tutorialspoint.com/python/list_min.htm)  Returns item from the list with min value. |
| 5 | [**list(seq)**](https://www.tutorialspoint.com/python/list_list.htm)  Converts a tuple into list. |
| Sr.No. | Methods with Description |
| 1 | [list.append(obj)](https://www.tutorialspoint.com/python/list_append.htm)  Appends object obj to list |
| 2 | [list.count(obj)](https://www.tutorialspoint.com/python/list_count.htm)  Returns count of how many times obj occurs in list |
| 3 | [list.extend(seq)](https://www.tutorialspoint.com/python/list_extend.htm)  Appends the contents of seq to list |
| 4 | [list.index(obj)](https://www.tutorialspoint.com/python/list_index.htm)  Returns the lowest index in list that obj appears |
| 5 | [list.insert(index, obj)](https://www.tutorialspoint.com/python/list_insert.htm)  Inserts object obj into list at offset index |
| 6 | [list.pop(obj=list[-1])](https://www.tutorialspoint.com/python/list_pop.htm)  Removes and returns last object or obj from list |
| 7 | [list.remove(obj)](https://www.tutorialspoint.com/python/list_remove.htm)  Removes object obj from list |
| 8 | [list.reverse()](https://www.tutorialspoint.com/python/list_reverse.htm)  Reverses objects of list in place |
| 9 | [list.sort([func])](https://www.tutorialspoint.com/python/list_sort.htm)  Sorts objects of list, use compare func if given |

**Python Tuples :** វាស្រដៀងទៅនឹង list ដែរ តែខុសគ្នាត្រង់ unchangeable, ប្រើ ( ) ។

ឧទាហរណ៍១-

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5 );

tup3 = "a", "b", "c", "d";

tup4 = (); #no element

tup5 = (50,); #1 element must also follow by ,

ឧទាហរណ៍២- Accessing Values in Tuples

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5, 6, 7 );

print ("tup1[0]: ", tup1[0]);

print "tup2[1:5]: ", tup2[1:5];

ឧទាហរណ៍៣- Updating Tuples

tup1 = (12, 34.56);

tup2 = ('abc', 'xyz');

# Following action is not valid for tuples

# tup1[0] = 100;

# So let's create a new tuple as follows

tup3 = tup1 + tup2;

print tup3;

# So the result is like below:

#(12, 34.56, 'abc', 'xyz')

ឧទាហរណ៍៣-Delete Tuple Elements

tup = ('physics', 'chemistry', 1997, 2000)

print(tup)

del tup

print("After deleting tup : ")

print(tup)

ឧទាហរណ៍៤- The tuple() Constructor​ to convert to a tuple

thistuple = tuple(("apple", "banana", "cherry")) # note the double round-brackets

print(thistuple)

**Basic Tuples Operations:**

|  |  |  |
| --- | --- | --- |
| **Python Expression** | **Results** | **Description** |
| len((1, 2, 3)) | 3 | Length |
| (1, 2, 3) + (4, 5, 6) | (1, 2, 3, 4, 5, 6) | Concatenation |
| ('Hi!',) \* 4 | ('Hi!', 'Hi!', 'Hi!', 'Hi!') | Repetition |
| 3 in (1, 2, 3) | True | Membership |
| for x in (1, 2, 3): print x, | 1 2 3 | Iteration |

**Built-in Tuple Functions:**

|  |  |
| --- | --- |
| **Sr.No.** | **Function with Description** |
| 1 | [**cmp(tuple1, tuple2)**](https://www.tutorialspoint.com/python/tuple_cmp.htm)  Compares elements of both tuples. |
| 2 | [**len(tuple)**](https://www.tutorialspoint.com/python/tuple_len.htm)  Gives the total length of the tuple. |
| 3 | [**max(tuple)**](https://www.tutorialspoint.com/python/tuple_max.htm)  Returns item from the tuple with max value. |
| 4 | [**min(tuple)**](https://www.tutorialspoint.com/python/tuple_min.htm)  Returns item from the tuple with min value. |
| 5 | [**tuple(seq)**](https://www.tutorialspoint.com/python/tuple_tuple.htm)  Converts a list into tuple. |

**Python Dictionaries** : គឺជាប្រភេទមួយនៃ collection ដែល unordered, changeable and indexed។ វាត្រូវបានគេសរសេរវានៅក្នុងសញ្ញា { } ហើយធាតុនីមួយៗរបស់វាត្រូវបានកាត់ផ្តាច់ដោយសញ្ញា , ដែលធាតុនីមួយៗនោះត្រូវមានតម្លៃជាគូគឺ key & value ដែល key អាចជា strings, numbers, ឬ tuplesតែត្រូវតែunique រីឯvalueអាចជាប្រភេទផ្សេងៗ។

ឧទាហរណ៍១-

thisdict = {

"apple": "green",

"banana": "yellow",

"cherry": "red"

}

print(thisdict)

ឧទាហរណ៍២- Accessing Values in Dictionary

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print ("dict['Name']: ", dict['Name'])

print ("dict['Age']: ", dict['Age'])

ឧទាហរណ៍៣- The dict() Constructor

thisdict = dict(apple="green", banana="yellow", cherry="red")

# note that keywords are not string literals

# note the use of equals rather than colon for the assignment

print(thisdict)

ឧទាហរណ៍៣- Updating Dictionary

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

dict['Age'] = 8; # update existing entry

dict['School'] = "DPS School"; # Add new entry

print ("dict['Age']: ", dict['Age'])

print ("dict['School']: ", dict['School'])

ឧទាហរណ៍៤-Delete Dictionary Elements

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

del dict['Name']; # remove entry with key 'Name'

dict.clear(); # remove all entries in dict

del dict ; # delete entire dictionary

print ("dict['Age']: ", dict['Age'])

print ("dict['School']: ", dict['School'])

**ឧទាហរណ៍៥**-Using Loop on Dictionary

#!C:/Python3.6.4/python

dict={1:"One",2:"Two",3:"Three",4:"Four",5:"Five",6:"Six",7:"Seven",8:"Egg",9:"Nine"}

print(dict)

for k,v in dict.items():

print(k,"=",0)

print(dict[1])

print("Loop on key")

for d in dict:

print(d)

for k in dict.keys():

print(k)

print("Loop on value")

for v in dict.values():

print(v)

**Built-in Dictionary Functions & Methods**

|  |  |
| --- | --- |
| **Sr.No.** | **Function with Description** |
| 1 | [**cmp(dict1, dict2)**](https://www.tutorialspoint.com/python/dictionary_cmp.htm)  Compares elements of both dict. |
| 2 | [**len(dict)**](https://www.tutorialspoint.com/python/dictionary_len.htm)  Gives the total length of the dictionary. This would be equal to the number of items in the dictionary. |
| 3 | [**str(dict)**](https://www.tutorialspoint.com/python/dictionary_str.htm)  Produces a printable string representation of a dictionary |
| 4 | [**type(variable)**](https://www.tutorialspoint.com/python/dictionary_type.htm)  Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type. |

|  |  |
| --- | --- |
| **Sr.No.** | **Methods with Description** |
| 1 | [**dict.clear()**](https://www.tutorialspoint.com/python/dictionary_clear.htm)  Removes all elements of dictionary *dict* |
| 2 | [**dict.copy()**](https://www.tutorialspoint.com/python/dictionary_copy.htm)  Returns a shallow copy of dictionary *dict* |
| 3 | [**dict.fromkeys()**](https://www.tutorialspoint.com/python/dictionary_fromkeys.htm)  Create a new dictionary with keys from seq and values *set* to *value*. |
| 4 | [**dict.get(key, default=None)**](https://www.tutorialspoint.com/python/dictionary_get.htm)  For *key* key, returns value or default if key not in dictionary |
| 5 | [**dict.has\_key(key)**](https://www.tutorialspoint.com/python/dictionary_has_key.htm)  Returns *true* if key in dictionary *dict*, *false* otherwise |
| 6 | [**dict.items()**](https://www.tutorialspoint.com/python/dictionary_items.htm)  Returns a list of *dict*'s (key, value) tuple pairs |
| 7 | [**dict.keys()**](https://www.tutorialspoint.com/python/dictionary_keys.htm)  Returns list of dictionary dict's keys |
| 8 | [**dict.setdefault(key, default=None)**](https://www.tutorialspoint.com/python/dictionary_setdefault.htm)  Similar to get(), but will set dict[key]=default if *key* is not already in dict |
| 9 | [**dict.update(dict2)**](https://www.tutorialspoint.com/python/dictionary_update.htm)  Adds dictionary *dict2*'s key-values pairs to *dict* |
| 10 | [**dict.values()**](https://www.tutorialspoint.com/python/dictionary_values.htm)  Returns list of dictionary *dict*'s values |

**Python Sets** : គឺជាប្រភេទមួយនៃcollectionដែល unordered និង unindexed។ Python sets ត្រូវបានគេសម្គាល់និងសរសេរវានៅក្នុង { }។ គ្រប់ element ទាំងអស់របស់វាគឺមិន duplicate ទេ។

ឧទាហរណ៍១-

thisset = {"apple", "banana", "cherry"}

print(thisset)

ឧទាហរណ៍២-The set() Constructor

thisset = set(("apple", "banana", "cherry")) # note the double round-brackets

print(thisset)

ឧទាហរណ៍៣-Using the add() method to add an item:

thisset = set(("apple", "banana", "cherry"))

thisset.add("damson")

print(thisset)

ឧទាហរណ៍៣- Using the remove() method to remove an item:

thisset = set(("apple", "banana", "cherry"))

thisset.remove("banana")

print(thisset)

ឧទាហរណ៍៤- Using the len() method to return the number of items:

thisset = set(("apple", "banana", "cherry"))

print(len(thisset))

**ឧទាហរណ៍៥**- ប្រើប្រាស់ loop លើ set

#!C:/Python3.6.4/python

myset={"a","b","c"}

myset.add("d")

print(myset)

for x in myset:

print(x)

## Basic Operators

**Operators** : គឺជាពពួក symbol ឬ keyword ដើម្បីធ្វើការគណនា រៀបចំ ឬប្រតិបត្តិផ្សេងៗទៀតទៅលើ variable និងvalue។ Operators ទាំងនោះមានដូចខាងក្រោម៖

* Arithmetic Operators(ប្រមាណវិធីពិជគណិត ឬនពន្ធសាស្ត្រ)
* Comparison (Relational) Operators(ប្រមាណវិធីប្រៀបធៀប)
* Assignment Operators(ប្រមាណវិធីផ្ទេរតម្លៃ)
* Logical Operators(ប្រមាណវិធីតក្ក)
* Bitwise Operators
* Membership Operators
* Identity Operators

**Python Arithmetic Operators(Ex: a=10, b=20)** :ដើម្បីគណនាលេខក្នុងទំរង់ពិជគណិត

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + Addition | Adds values on either side of the operator. | a + b = 30 |
| - Subtraction | Subtracts right hand operand from left hand operand. | a – b = -10 |
| \* Multiplication | Multiplies values on either side of the operator | a \* b = 200 |
| / Division | Divides left hand operand by right hand operand | b / a = 2 |
| % Modulus | Divides left hand operand by right hand operand and returns remainder | b % a = 0 |
| \*\* Exponent | Performs exponential (power) calculation on operators | a\*\*b =10 to the power 20 |
| // | Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity) − | 9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0 |

**Python Comparison Operators(Ex: a=10, b=20)** :ត្រូវបានគេប្រើដើម្បីធ្វើការប្រៀបធៀប

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | If the values of two operands are equal, then the condition becomes true. | (a == b) is not true. |
| != | If values of two operands are not equal, then condition becomes true. | (a != b) is true. |
| <> | If values of two operands are not equal, then condition becomes true. | (a <> b) is true. This is similar to != operator. |
| > | If the value of left operand is greater than the value of right operand, then condition becomes true. | (a > b) is not true. |
| < | If the value of left operand is less than the value of right operand, then condition becomes true. | (a < b) is true. |
| >= | If the value of left operand is greater than or equal to the value of right operand, then condition becomes true. | (a >= b) is not true. |
| <= | If the value of left operand is less than or equal to the value of right operand, then condition becomes true. | (a <= b) is true. |

**Python Assignment Operators(Ex: a=10,b=20):** ដើម្បីផ្ទេរតម្លៃទៅឲ្យvariable

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Assigns values from right side operands to left side operand | c = a + b assigns value of a + b into c |
| += Add AND | It adds right operand to the left operand and assign the result to left operand | c += a is equivalent to c = c + a |
| -= Subtract AND | It subtracts right operand from the left operand and assign the result to left operand | c -= a is equivalent to c = c - a |
| \*= Multiply AND | It multiplies right operand with the left operand and assign the result to left operand | c \*= a is equivalent to c = c \* a |
| /= Divide AND | It divides left operand with the right operand and assign the result to left operand | c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a |
| %= Modulus AND | It takes modulus using two operands and assign the result to left operand | c %= a is equivalent to c = c % a |
| \*\*= Exponent AND | Performs exponential (power) calculation on operators and assign value to the left operand | c \*\*= a is equivalent to c = c \*\* a |
| //= Floor Division | It performs floor division on operators and assign value to the left operand | c //= a is equivalent to c = c // a |

**Python Bitwise Operators**

វាគណនាលើតម្លៃជា bit។ ឧទាហរណ៍ថា a=60, b=13 នោះតម្លៃក្នុងទំរង់ជាប្រព័ន្ធ binaryរបស់វាគឺ៖

a = 0011 1100

b = 0000 1101

-----------ឧ--------------

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| & Binary AND | Operator copies a bit to the result if it exists in both operands | (a & b) (means 0000 1100) |
| | Binary OR | It copies a bit if it exists in either operand. | (a | b) = 61 (means 0011 1101) |
| ^ Binary XOR | It copies the bit if it is set in one operand but not both. | (a ^ b) = 49 (means 0011 0001) |
| ~ Binary Ones Complement | It is unary and has the effect of 'flipping' bits. | (~a ) = -61 (means 1100 0011 in 2's complement form due to a signed binary number. |
| << Binary Left Shift | The left operands value is moved left by the number of bits specified by the right operand. | a << 2 = 240 (means 1111 0000) |
| >> Binary Right Shift | The left operands value is moved right by the number of bits specified by the right operand. | a >> 2 = 15 (means 0000 1111) |

**Python Logical Operators(Ex: a=10,b=20)** :គេប្រើវាដើម្បីចងភ្ជាប់conditionបញ្ចូលគ្នា

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| and | Returns True if both statements are true | x < 5 and  x < 10 |
| or | Returns True if one of the statements is true | x < 5 or x < 4 |
| not | Reverse the result, returns False if the result is true | not(x < 5 and x < 10) |

**Python Membership Operators**: សម្រាប់ test ឬ check membership នៅក្នុងsequence(strings, lists, or tuples)។

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| in | Evaluates to true if it finds a variable in the specified sequence and false otherwise. | x in y, here in results in a 1 if x is a member of sequence y. |
| not in | Evaluates to true if it does not finds a variable in the specified sequence and false otherwise. | x not in y, here not in results in a 1 if x is not a member of sequence y. |

**Python Identity Operators**:សម្រាប់ប្រៀបធៀប memory location នៃ object ពីរ។

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| is | Evaluates to true if the variables on either side of the operator point to the same object and false otherwise. | x is y, here **is** results in 1 if id(x) equals id(y). |
| is not | Evaluates to false if the variables on either side of the operator point to the same object and true otherwise. | x is not y, here **is not** results in 1 if id(x) is not equal to id(y). |

#!C:/Python3.6.4/python

myset={"a","b","c"}

if(type(myset) is set):

print("is set")

**Python Operators Precedence** : សញ្ញគណនាអតិភាព(ខាងលើអតិភាពមុន)

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| 1 | **\*\***  Exponentiation (raise to the power) ស្វ័យគុណ |
| 2 | **~ + -**  Complement, unary plus and minus (method names for the last two are +@ and -@) |
| 3 | **\* / % //**  Multiply, divide, modulo and floor division |
| 4 | **+ -**  Addition and subtraction |
| 5 | **>> <<**  Right and left bitwise shift |
| 6 | **&**  Bitwise 'AND' |
| 7 | **^ |**  Bitwise exclusive `OR' and regular `OR' |
| 8 | **<= < > >=**  Comparison operators |
| 9 | **<> == !=**  Equality operators |
| 10 | **= %= /= //= -= += \*= \*\*=**  Assignment operators |
| 11 | **is is not**  Identity operators |
| 12 | **in not in**  Membership operators |
| 13 | **not or and**  Logical operators |

# មេរៀនទី ៣ Python-Control Flow Statement

Control Flow Statement ត្រូវបានគេប្រើសម្រាប់ត្រួតពិនិត្យទៅលើដំណើរការរបស់ Program នៅពេលដែល Program កំពុងដំណើរការ។​ ជាទូទៅការត្រួតពិនិត្យលក្ខខណ្ឌនៅក្នុង Program មួយគឺយើងចង់ដឹងពីសកម្មភាពដែលវាបានកើតឡើង។ នៅក្នុងការត្រួតពិនិត្យលក្ខខណ្ឌនៅក្នុង Python Program មានតែពីរប្រភេទគត់ គឺលក្ខខណ្ឌពិត (TRUE) នឹងលក្ខខណ្ឌមិនពិត (FALSE) ។

នៅក្នុងមេរៀននេះ នឹងបង្ហាញអោយអ្នកមានសមត្ថភាពដូចខាងក្រោមៈ

* ការប្រើប្រាស់ Operator ផ្សេងៗដើម្បីធ្វើការគណនាលើ content
* កំណត់ការប្រើប្រាស់លក្ខខ័ណ if else Statements
* កំណត់ការប្រើប្រាស់ continue, break, exit()
* កំណត់ការប្រើប្រាស់ Loop (while, for)



## Decision Making or Condition

Condition Satement គឺជា structure ដែលត្រូវបានគេប្រើប្រាស់សំរាប់បញ្ជា រឺប្រាប់ទៅ system ឲ្យធ្វើអ្វីមួយដោយមានលក្ខ័ណ។

If នឹងត្រូវត្រួតពិនិត្យលក្ខខណ្ឌរបស់ condition

បើ condition ពិតនោះវាអនុវត្ត Statement(s)

របស់វា តែបើមិនពិតទេនោះ គឺវានឹងមិនអនុវត្ត

Statement(s) នោះទេ ។

|  |  |
| --- | --- |
| **Sr.No.** | **Statement & Description** |
| 1 | [**if statements**](https://www.tutorialspoint.com/python/python_if_statement.htm)  An **if statement** consists of a boolean expression followed by one or more statements. |
| 2 | [**if...else statements**](https://www.tutorialspoint.com/python/python_if_else.htm)  An **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is FALSE. |
| 3 | [**nested if statements**](https://www.tutorialspoint.com/python/nested_if_statements_in_python.htm)  You can use one **if** or **else if** statement inside another **if** or **else if**statement(s). |

ឧទាហរណ៍ទី១- Single Statement with Signle line

var = 100

if ( var == 100 ) : print("Value of expression is 100")

print("Good bye!")

ឧទាហរណ៍ទី២- Single Statement with Multi-lines

var = 100

if ( var == 100 ):

print("Value of expression is 100") #ត្រូវចូលបន្ទាត់

print("Good bye!")

ឧទាហរណ៍ទី៣- Multi Statements with Multi-lines

var = 100

if ( var == 100 ):

print("Value of expression is 100") #ត្រូវចូលបន្ទាត់

print("Good bye!") #ត្រូវចូលបន្ទាត់

ឧទាហរណ៍ទី៤-The **elif** keyword is pythons way of saying "if the previous conditions were not true, then do this condition".

a = 33

b = 33

if b > a:

print("b is greater than a")

elif a == b:

print("a and b are equal")

ឧទាហរណ៍ទី៥- The **else** keyword catches anything which isn't caught by the preceding conditions.

a = 200

b = 33

if b > a:

print("b is greater than a")

elif a == b:

print("a and b are equal")

else:

print("a is greater than b")

## Loop Statement

នៅក្នុងការសរសេរកម្មវិធីដើម្បីដោះស្រាយបញ្ហាអ្វីមួយ ជួនកាលគេជួបប្រទះលក្ខខណ្ឌ មួយតែមួយដង ជួនកាលគេជួបប្រទះលក្ខខណ្ឌដដែលច្រើនដង ។ ហើយលក្ខខណ្ឌដដែលៗ ច្រើនដងនេះត្រូវបានគេហៅថា Loops ។ Loops គឺជាការធ្វើសកម្មភាពដដែលរហូតជួបលក្ខខណ្ឌ ណាមួយទើបវាបញ្ចប់សកម្មភាពរបស់វា ។ នៅក្នុង Python Program មាន៖

* while loop
* for loop

Loop Control Statements

|  |  |
| --- | --- |
| **Sr.No.** | **Control Statement & Description** |
| 1 | [**break statement**](https://www.tutorialspoint.com/python/python_break_statement.htm)  Terminates the loop statement and transfers execution to the statement immediately following the loop. |
| 2 | [**continue statement**](https://www.tutorialspoint.com/python/python_continue_statement.htm)  Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating. |
| 3 | [**pass statement**](https://www.tutorialspoint.com/python/python_pass_statement.htm)**(do nothing)**  The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute. |

**The while Loop :** វានៅតែ loop ប្រសិនបើលក្ខ័ណរបស់វានៅតែពិត

i = 1

while i < 6:

print(i) #ត្រូវចូលបន្ទាត់

i += 1#ត្រូវចូលបន្ទាត់

**Python For Loops** : ត្រូវបានគេប្រើប្រាស់ដើម្បី loop(iteration)លើsequence (list, a tuple or a string)។

fruits = ["apple", "banana", "cherry"]

for x in fruits:

print(x) #ត្រូវចូលបន្ទាត់

**The break Statement**

i = 1

while i < 6:

print(i)

if i == 3:

break

i += 1

ឬ

fruits = ["apple", "banana", "cherry"]

for x in fruits:

if x == "banana":

break

print(x)

**The continue Statement**

i = 0

while i < 6:

i += 1

if i == 3:

continue

print(i)

ឬ

fruits = ["apple", "banana", "cherry"]

for x in fruits:

if x == "banana":

continue

print(x)

**The range() Function :** The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

for x in range(6): #loop from 0 to 6

print(x)

ឬ

for x in range(2, 6): #loop from 2 to 6

print(x)

ឬ

for x in range(2, 30, 3): #loop from 2 to 30​ and increment by +1

print(x)

ឬ

# មេរៀនទី ៤ Functions

**Python Functions**

* A function is a block of code which only runs when it is called.
* You can pass data, known as parameters, into a function.
* A function can return data as a result.

**គោលបំណងនៃមេរៀននេះគឺមានដូចខាងក្រោម:**

* **ស្វែងយល់ពីរចនាសម្ព័ន្ធ Structure នៃ អនុគមន៍ Functions ។**
* **ស្វែងយល់ពីភាពខុសគ្នារវាង** **Static និង Instance Functions ។**
* **រៀនបង្កើត Instance Functions ក្នុង object។**
* **ហៅ Instance Functions ចេញពី object ឲ្យដំណើរការ។**
* **យល់ដឹងពីប្រភេទនៃប៉ារ៉ាម៉ែត្រ​ parameters។**



## Basic Functions

**Syntax**

def functionname( parameters ):

"function\_docstring"

function\_suite

return [expression]

**Creating a Function**

def my\_function():

print("Hello from a function")

ឬ

def printme( str ):

"This prints a passed string into this function"

print(str)

return

**Calling a Function**

def my\_function():

print("Hello from a function")

**my\_function()**

ឬ

# Function definition is here

def printme( str ):

"This prints a passed string into this function"

print(str)

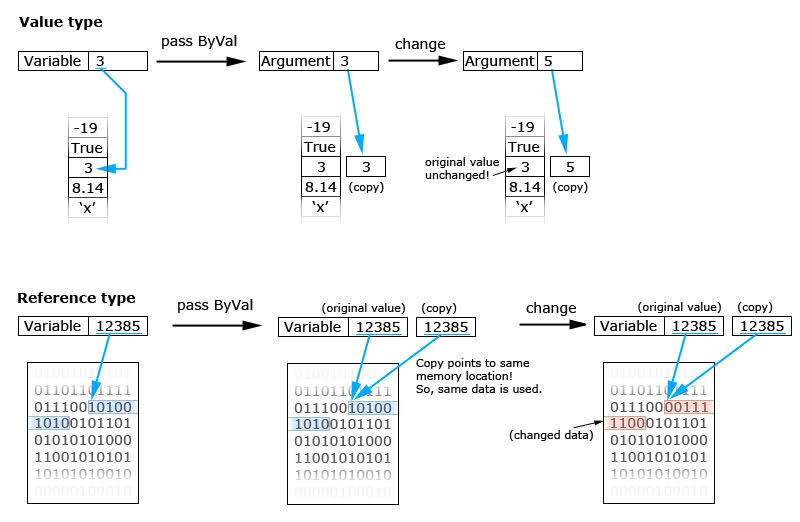
return;

# Now you can call printme function

printme("I'm first call to user defined function!")

printme("Again second call to the same function")

**Pass by reference vs value**

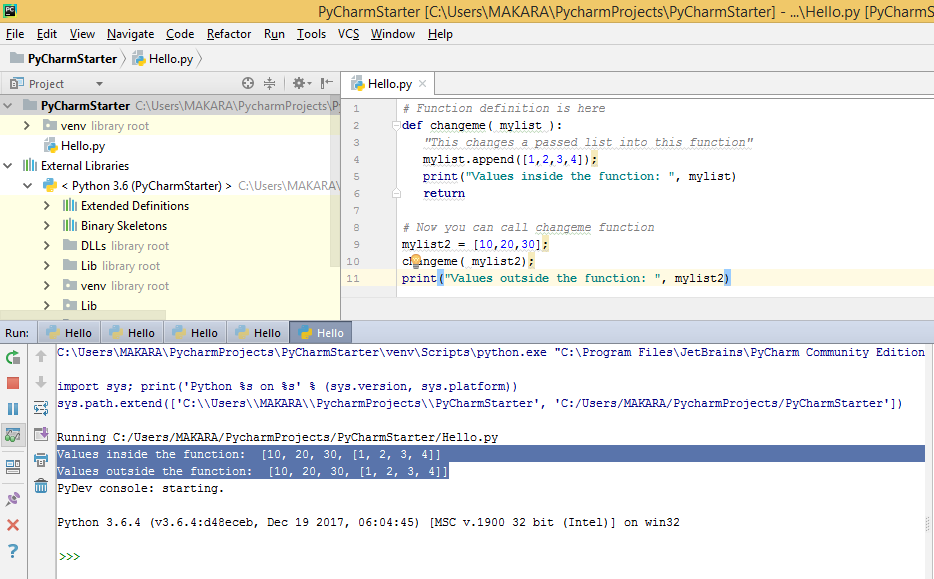


ក្នុងភាសា Python គ្រប់ parameters (arguments) ប្រភេទ អក្សរ លេខ...គឺ Pass by Value ចំណែកឯ List, Dictionary…គឺ Pass by reference(មានន័យថានៅពេលដែរគ្រប់ parametersរបស់functionមានកែប្រែប្រួលនោះវានឹងផ្លាស់ប្តូរតម្លៃរបស់variable ដែលបានផ្តល់តម្លៃទៅឲ្យfunctionនោះដែរ)។

# Function definition is here

def changeme( mylist ):

"This changes a passed list into this function"

 mylist.append([1,2,3,4]);

print("Values inside the function: ", mylist)

return

# Now you can call changeme function

mylist2 = [10,20,30];

changeme( mylist2);

print("Values outside the function: ", mylist2)

# Function definition is here

def changeme( mylist ):

"This changes a passed list into this function"

mylist = [1,2,3,4]; # This would assig new reference in mylist

print("Values inside the function: ", mylist)

return

# Now you can call changeme function

mylist = [10,20,30];

changeme( mylist );

print("Values outside the function: ", mylist)

**Required arguments**:ត្រូវតែផ្តល់តម្លៃ

# Function definition is here

def printme( str ):

"This prints a passed string into this function"

print (str)

return;

# Now you can call printme function

printme()

**Keyword arguments**:

# Function definition is here

def printme( str ):

"This prints a passed string into this function"

print str

return;

# Now you can call printme function

printme( str = "My string")

ឬ

# Function definition is here

def printinfo( name, age ):

"This prints a passed info into this function"

print ("Name: ", name)

print ("Age ", age)

return;

# Now you can call printinfo function

printinfo( age=50, name="miki" )

**Default arguments**: មានតម្លៃ default

# Function definition is here

def printinfo( name, age = 35 ):

"This prints a passed info into this function"

print ("Name: ", name)

print ("Age ", age)

return;

# Now you can call printinfo function

printinfo( age=50, name="miki" )

printinfo( name="miki" )

**Variable-length arguments**: គេប្រើសម្រាប់បោះតម្លៃចូលទៅកាន់ function លក្ខណជា tuple collection។ វាជាឈ្មោះនៃparamenter ដែលផ្តើមដោយសញ្ញា \* ។

Syntax:

def functionname([formal\_args,] \*var\_args\_tuple ):

"function\_docstring"

function\_suite

return [expression]

ឧទាហរណ៍

# Function definition is here

def printinfo( arg1, \*vartuple ):

"This prints a variable passed arguments"

print ("Output is: ")

print (arg1)

for var in vartuple:

print var

return;

# Now you can call printinfo function

printinfo( 10 )

printinfo( 70, 60, 50 )

**Lambda Functions**: សម្រាប់បង្កើតជា anonymous functions(function no name)។

Syntax

lambda [arg1 [,arg2,.....argn]]:expression

ឧទាហរណ៍៖

# Function definition is here

sum = lambda arg1, arg2: arg1 + arg2;

# Now you can call sum as a function

print "Value of total : ", sum( 10, 20 )

print "Value of total : ", sum( 20, 20 )

ឬ

myfunc = lambda i: i\*2

print(myfunc(2))

ឬ

myfunc = lambda x,y: x\*y

print(myfunc(3,6))

**The return Statement**: សម្រាប់return តម្លៃចេញពីអនុគមន៍function និង/ឬ exit function។

# Function definition is here

def sum( arg1, arg2 ):

# Add both the parameters and return them."

total = arg1 + arg2

print ("Inside the function : ", total)

return total;

# Now you can call sum function

total = sum( 10, 20 );

print ("Outside the function : ", total)

## Scope of Variable

**Scope of Variables**:ដែនកំណត់របស់ variable មានពីរ(Global និង Local)

total = 0; # This is global variable.

# Function definition is here

def sum( arg1, arg2 ):

# Add both the parameters and return them."

total = arg1 + arg2; # Here total is local variable.

print ("Inside the function local total : ", total)

return total;

# Now you can call sum function

sum( 10, 20 );

print ("Outside the function global total : ", total)

**Gloable** : ជា keyword ដែលត្រូវបានគេប្រើដើម្បីកំណត់ថាជា gloable variable ក្នុង module ហើយត្រូវបានគេប្រើប្រាស់វាក្នុង function។

**g\_name=None**

def printName():

g\_name='test'#still local variable

print(g\_name)

def printName2():

**global g\_name**#is gloable variable

**g\_name**='test2'

print(**g\_name**)

printName()#test

print(g\_name)#None

printName2()#test2

print(**g\_name**)#test2

**nonlocal** : ជា keyword ដែលត្រូវបានគេប្រើដើម្បីកំណត់ថាជា gloable variable ក្នុង parent function ហើយត្រូវបានគេប្រើប្រាស់វាក្នុង nested function។

def f():

x = 42

def g():

**nonlocal x**

**x = 43**

print("Before calling g: " + str(x))

print("Calling g now:")

g()

print("After calling g: " + str(x))

x = 3

print("x in main: " + str(x)) #x is 3

f()

print("x in main: " + str(x)) #x is 3

# មេរៀនទី ៥: Modules

យើងអាចចាត់ទុក Python Module ថាវាដូចជា code libray ឬជាfile ដែលរួមមាននូវសំនុំ function ជាច្រើនដែលយើងអាចយកវាមកប្រើប្រាស់ក្នុង applicationរបស់យើងបាន។

នៅក្នុងមេរៀននេះ នឹងបង្ហាញអោយអ្នកមានសមត្ថភាពដូចខាងក្រោមៈ

* របៀបបង្កើត module
* របៀបប្រើប្រាស់ module



## Build-In Module

គឺជា Module​ ដែលមានស្រាប់ លោកអ្នកអាចហៅវាមកប្រើប្រាស់បានតាមរយៈ import keyword។

import platform

import sys

x = platform.system()

print(x)

print(platform.\_sys\_version())

sys.exit(0)

## Define Module

យើងអាចបង្កើត module ក្នុង file ដែលមាន extension(\*.py)។

**Create a Module** : mymodule.py

def greeting(name):

print("Hello, " + name)

**Use a Module** : testmodule.py

យើងប្រើប្រាស់ keyword import ដើម្បីហៅ module មកប្រើប្រាស់វិញ។

import mymodule

mymodule.greeting("Jonathan")

ការប្រើប្រាស់ import មានច្រើនសណ្ឋានដូចខាងក្រោម

**The import Statement:**

Syntax

import module1[, module2[,... moduleN]

ឧទាហរណ៍

#import module mymodule

import mymodule

#call a function from mymodule

mymodule.greeting("makara")

**The from...import Statement**: lets you import specific attributes from a module into the current namespace.

from modname import name1[, name2[, ... nameN]]

ឧទាហរណ៍

#import module mymodule

from mymodule import greeting

#call a function from mymodule

greeting("makara")

**The from...import \* Statement** : import យកនូវ attributes ទាំងអស់របស់ moduleណាមួយ

Syntax: from modname import \*

ឧទាហរណ៍

#import module mymodule

from mymodule import \*

#call a function from mymodule

greeting("makara")

greeting2("makara")

**ចំណាំ**៖ នៅពេលដែលអ្នក import module នោះ python នឹង

* ជាតំបូងនឹង search in current directory
* បើរកមិនឃើញ នោះវានឹងទៅស្វែងរកក្នុងគ្រប់ shell variable PYTHONPATH
* បើនៅតែរកមិនឃើញទៀតនោះ Python នឹងទៅ search ក្នុង default path (សម្រាប់ UNIX គឺ /usr/local/lib/python/)
* បើអ្នកចង់មើល path សូមប្រើcode sys.path (វារួមមាននូវ current directory, PYTHONPATH, and the installation-dependent default)
* PYTHONPATH : is an environment variable, consisting of a list of directories. សម្រាប់ window(set PYTHONPATH = c:\python20\lib;) unix(set PYTHONPATH = /usr/local/lib/python)

**Re-naming/Alias a Module**: វិធីប្តូរឈ្មោះ module

import mymodule as mx

a = mx.person1["age"]

print(a)

**The dir( ) Function**: សម្រាប់បង្ហាញនូវ member ទាំងអស់របស់ module

# Import built-in module math

import math

content = dir(math)

print (content)

**Note**: Here, the special string variable \_\_name\_\_ is the module's name, and \_\_file\_\_ is the filename from which the module was loaded.

**The globals() and locals() Functions:**

The **globals() and locals()** functions can be used to return the names in the global and local namespaces depending on the location from where they are called.

If **locals()** is called from within a function, it will return all the names that can be accessed locally from that function.

If **globals()** is called from within a function, it will return all the names that can be accessed globally from that function.

The **return type of both these functions is dictionary**. Therefore, names can be extracted using the keys() function.

# Import built-in module math

import math

globalsDic=globals()

localsDic=locals()

print("Gloable")

print(globalsDic)

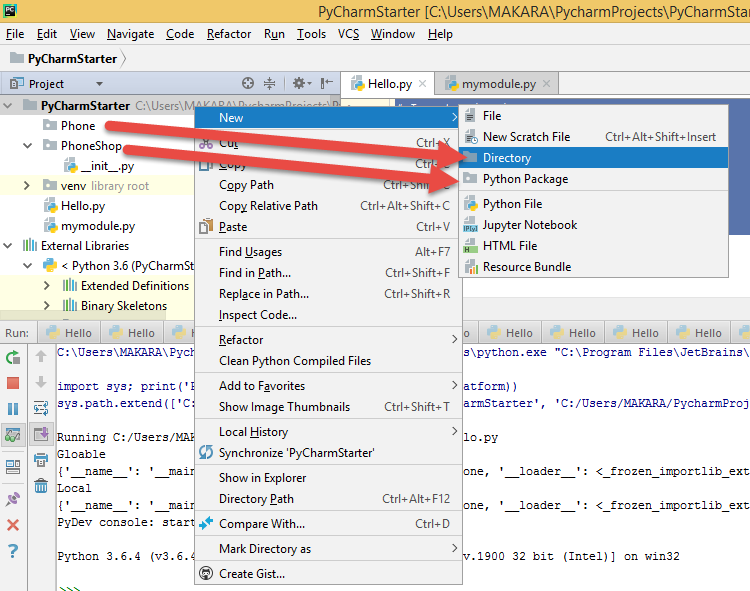
print("Local")

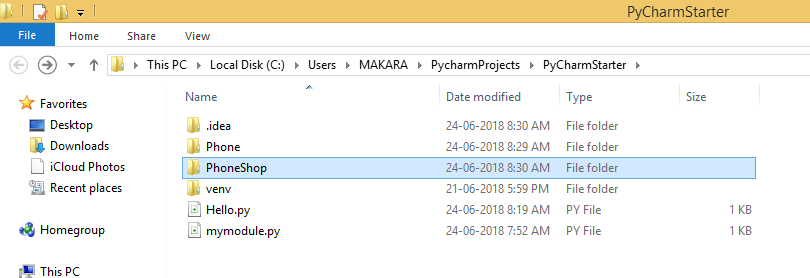
print(localsDic)

**The reload() Function**: ជាទូទៅរាល់ modules ដែលបាន import នៅខាងលើគឺវា execute បានតែម្តងទេ តែយើងអាចបញ្ជាឲ្យវា executeម្តងទៀតបានដោយប្រើដូចខាងក្រោម

reload(module\_name)

**Packages in Python**: A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and subpackages and sub-subpackages, and so on.





ឧទាហរណ៍

ទី១-បង្កើត package **PhoneShop/** ក្នុងproject

ទី២-បង្កើត python file PhoneShop/Pots.py

def Pots():

print "I'm Pots Phone"

ទី៣-បង្កើត python file PhoneShop/ Isdn.py

def Isdn ():

print "I'm Isdn Phone"

ទី៤-បង្កើត python file PhoneShop/ G3.py

def G3 ():

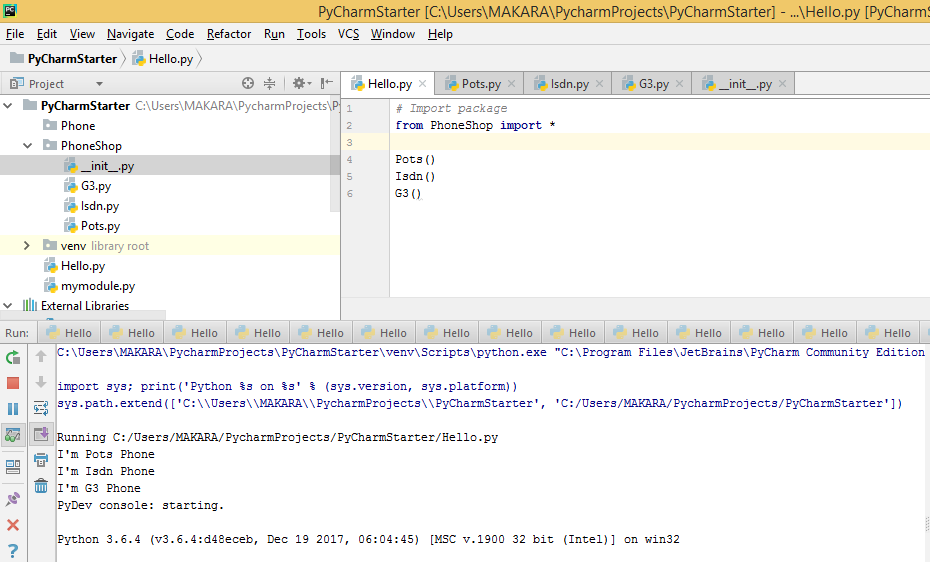
print "I'm G3 Phone"

ទី៥-config code ខាងក្រោមក្នុង file(\_\_init\_\_.py)

from PhoneShop.Pots import Pots

from PhoneShop.Isdn import Isdn

from PhoneShop.G3 import G3

ទី៦-បង្កើត python file ដើម្បី test(Hello.py)

# Import package

from PhoneShop import \*

Pots()

Isdn()

G3()

# មេរៀនទី ៦: FILE I/O

នៅក្នុងមេរៀននេះ នឹងបង្ហាញអោយអ្នកមានសមត្ថភាពដូចខាងក្រោមៈ

* យល់ដឹងពី basic I/O functions
* យល់ដឹងពី create/delete file(flat database)
* យល់ដឹងពី read/write data on file

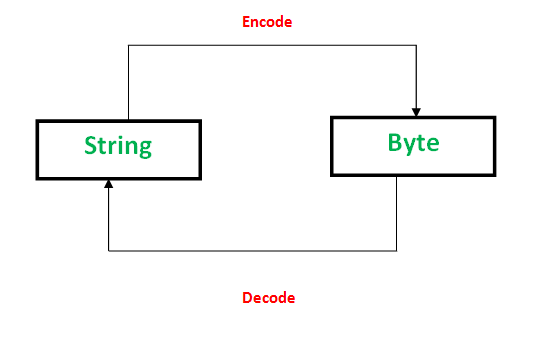
## String vs Byte

ជាតំបូងអ្នកត្រូវស្វែងយល់ពីរបៀបក្នុងការបំលែងទិន្នន័យរវាង stringនិងarray of bytesជាមុនសិន៖

**+ Byte Objects vs String in Python**

In Python 2, both str and bytes are the same typeByte objects whereas in Python 3 Byte objects, defined in Python 3 are “sequence of bytes” and similar to “unicode” objects from Python 2. However, there are many differences in strings and Byte objects. Some of them are depicted below:

* Byte objects are sequence of **Bytes**, whereas Strings are sequence of **characters**.
* Byte objects are in**machine readable** form internally, Strings are only in**human readable** form.
* Since Byte objects are machine readable, they can be **directly stored on the disk**. Whereas, Strings**need encoding**before which they can be stored on disk.



There are methods to convert a byte object to String and String to byte objects.

**Encoding**

PNG, JPEG, MP3, WAV, ASCII, UTF-8 etc are different forms of encodings. An encoding is a format to represent audio, images, text, etc in bytes. Converting **Strings to byte** objects is termed as encoding. This is necessary so that the text can be stored on disk using mapping using **ASCII** or **UTF-8** encoding techniques.

This task is achieved using**encode()**. It take encoding technique as argument. Default technique is “**UTF-8**” technique.

|  |
| --- |
| *# Python code to demonstate String encoding*  *# initialising a String*  *a = 'GeeksforGeeks'*  *# initialising a byte object*  *c = b'GeeksforGeeks'*  *# using encode() to encode the String*  *# encoded version of a is stored in d*  *# using ASCII mapping*  *d = a.encode('ASCII')*  *# checking if a is converted to bytes or not*  *if (d==c):*  *print ("Encoding successful")*  *else : print ("Encoding Unsuccessful")* |

*Output:*

*Encoding successful*

**Decoding**

Similarly, Decoding is process to convert a**Byte object to String**. It is implemented using **decode()** . A byte string can be decoded back into a character string, if you know which encoding was used to encode it. Encoding and Decoding are **inverse** processes.

|  |
| --- |
| *# Python code to demonstate Byte Decoding*  *# initialising a String*  *a = 'GeeksforGeeks'*  *# initialising a byte object*  *c = b'GeeksforGeeks'*  *# using decode() to decode the Byte object*  *# decoded version of c is stored in d*  *# using ASCII mapping*  *d = c.decode('ASCII')*  *# checking if c is converted to String or not*  *if (d==a):*  *print ("Decoding successful")*  *else : print ("Decoding Unsuccessful")* |

Output:

Decoding successful

**+ String និង Bytes ក្នុងគោល ១០(bytes(), encoded(), decoded)**

text10='admin:password'

text10\_bytes=bytes(text10,'utf-8')

print(text10\_bytes)

print(type(text10\_bytes))

text10\_bytes=text10.encode('utf-8')

print(text10\_bytes)

print(type(text10\_bytes))

#convert back to string

print(text10\_bytes.decode('utf-8'))

**+ String និង Bytes ក្នុងគោល ១០(import codecs class)**

import codecs

text10='admin:password'

encoded\_bytes=codecs.encode(obj=text10,encoding='utf-8')

print(encoded\_bytes)

print(type(encoded\_bytes))

decoded\_string=codecs.decode(obj=encoded\_bytes,encoding='utf-8')

print(decoded\_string)

print(type(decoded\_string))

print(encoded\_bytes.decode('utf-8'))

**+ String និង Bytes ក្នុងគោល ១០ និង ៦៤(import base64 class)**

import base64

text='admin:password'

encoded\_bytes=base64.encodebytes(bytes(text,'utf-8'))

print(encoded\_bytes)

print(type(encoded\_bytes))

decoded\_bytes=base64.decodebytes(encoded\_bytes)

print(decoded\_bytes)

print(type(decoded\_bytes))

#convert to string

print(encoded\_bytes.decode('utf-8').replace('\n',''))

print(decoded\_bytes.decode('utf-8'))

**+ String និង Bytes និង Unicode (import encoding class)**

import encodings.utf\_8

encoded\_bytes\_tuple=encodings.utf\_8.encode("afsfsfសសសងសថ")

print(encoded\_bytes\_tuple)

print(type(encoded\_bytes\_tuple))

#get bytes

print(type(encoded\_bytes\_tuple))

print(encoded\_bytes\_tuple[0])

#convert to string

decoded\_string\_tuple=encodings.utf\_8.decode(encoded\_bytes\_tuple[0])

print(decoded\_string\_tuple[0])

print(type(decoded\_string\_tuple[0]))

print(encoded\_bytes\_tuple[0].decode('utf-8'))

\*\***Codecs** class គឺជា build-in class ដែលមានសមត្ថភាពជាច្រើនដូចជា៖

* Encode និង Decode ទិន្នន័យ
* ផ្តល់នូវ sub-class និង function សម្រាប់ការងារលើ file
  + codecs.open(filename, mode='r', encoding=None, errors='strict', buffering=1)
  + codecs.StreamWriter(stream, errors='strict') **មាន function ដូចជា**៖
    - write(object)
    - writelines(list)
    - reset()
  + codecs.StreamReader(stream, errors='strict')**មាន function ដូចជា**៖
    - read([size[, chars[, firstline]]])
    - readline([size[, keepends]])
    - readlines([sizehint[, keepends]])
    - reset()
  + codecs.StreamReaderWriter(stream, Reader, Writer, errors='strict')
* ……………………………………………………………………………………..

For more : <https://docs.python.org/3/library/codecs.html#streamrecoder-objects>



## Opening and Closing File

ក្នុងមេរៀននេះដើម្បីរៀបចំ file Python យើងនឹងប្រើប្រាស់ file object ជាមួយនឹង function Open()តែប៉ុណ្ណោះ។

**The open Function**: សម្រាប់បង្កើត file object មុននឹងធ្វើការ read/write data

Syntax

file object = open(file\_name [, access\_mode][,encoding] [,error] [, buffering],)

* **file\_name**:ទីតាំងរបស់ file
* **access\_mode**: read, write, append, etc(សូមមើល list ខាងក្រោម)
* **buffering**: (buffering value)default គឺតំលៃអវិជ្ជមាន។ If the buffering value is set to 0, no buffering takes place. If the buffering value is 1, line buffering is performed while accessing a file. If you specify the buffering value as an integer greater than 1, then buffering action is performed with the indicated buffer size.

|  |  |
| --- | --- |
| **Sr.No.** | **Modes & Description** |
| 1 | **r**  Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode. |
| 2 | **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. |
| 3 | **r+**  Opens a file for both reading and writing. The file pointer placed at the beginning of the file. |
| 4 | **rb+**  Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file. |
| 5 | **w**  Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| 6 | **wb**  Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| 7 | **w+**  Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing. |
| 8 | **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, creates a new file for reading and writing. |
| 9 | **a**  Opens 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. |
| 10 | **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. |
| 11 | **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. |
| 12 | **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. |

**ឧទាហរណ៍១**-Open a file for read only

f = open("demofile.txt")

ឬ

f = open("demofile.txt", "rt", encoding=’utf-8’)

**The file Object Attributes:**

|  |  |
| --- | --- |
| **Sr.No.** | **Attribute & Description** |
| 1 | **file.closed**  Returns true if file is closed, false otherwise. |
| 2 | **file.mode**  Returns access mode with which file was opened. |
| 3 | **file.name**  Returns name of the file. |
| 4 | **file.softspace**  Returns false if space explicitly required with print, true otherwise. |

ឧទាហរណ៍២-

fo = open("demofile.txt","wb")

print("File Information-----------")

print ("Name of the file: ", fo.name)

print ("Closed or not : ", fo.closed)

print ("Opening mode : ", fo.mode)

**The close() Method:** សម្រាប់ flushes file object ឬ close the file object។

**Syntax:**

fileObject.close();

ឧទាហរណ៍៣-

# Open a file

fo = open("foo.txt", "wb")

print ("Name of the file: ", fo.name)

# Close opend file

fo.close()

## Reading and Writing Files

**The write() Method** : write data into file

Syntax: fileObject.write(string);

# Open a file

fo = open("foo.txt","w+")

fo.write("Python is a great language.\nYeah its great!!\n");

# Close opend file

fo.close()

**Write to an Existing File**

f = open("demofile.txt", "a")

f.write("Now the file has one more line!")

ឬ

f = open("demofile.txt", "w")

f.write("Woops! I have deleted the content!")

Note: "a" - Append - will append to the end of the file, "w" - Write - will overwrite any existing content

**The read() Method**:

Syntax: fileObject.read([count]); បើគ្មាន count នោះវានឹង read ពីចំនុចចាប់ផ្តើមដល់ចប់ តែបើមាននោះវានឹង read ពីកន្លែងចាប់ផ្តើមហើយយកគ្រប់ចំនួនbytesដែលបានកំណត់។

# Open a file

fo = open("foo.txt", "r+")

str = fo.read(10);

print ("Read String is : ", str)

# Close opend file

fo.close()

**File Positions** tell() vs seek():

**The tell() method** tells you the current position within the file; in other words, the next read or write will occur at that many bytes from the beginning of the file.

**The seek(offset[, from])** **method** changes the current file position. The offset argument indicates the number of bytes to be moved. The from argument specifies the reference position from where the bytes are to be moved.

If from is set to 0, it means use the beginning of the file as the reference position and 1 means use the current position as the reference position and if it is set to 2 then the end of the file would be taken as the reference position.

# Open a file

fo = open("foo.txt", "r+")

str = fo.read(10);

print ("Read String is : ", str)

# Check current position

position = fo.tell();

print ("Current file position : ", position)

# Reposition pointer at the beginning once again

position = fo.seek(0, 0);

str = fo.read(10);

print ("Again read String is : ", str)

# Close opend file

fo.close()

**The readline() method**: read 1 line in file

f = open("demofile.txt", "r")

print(f.readline())

ឬ

f = open("demofile.txt", "r")

print(f.readline())

print(f.readline())

ឬ

f = open("demofile.txt", "r")

for x in f:

print(x)

## Creating, Renaming and Deleting Files

**Create a New File:**ត្រូវប្រើ the open() method

* "x" - Create - will create a file, returns an error if the file exist
* "a" - Append - will create a file if the specified file does not exist
* "w" - Write - will create a file if the specified file does not exist

f = open("myfile.txt", "x") ឬ

f = open("myfile.txt", "w")

**The rename() Method:** សម្រាប់ rename ឈ្មោះ file

Syntax: os.rename(current\_file\_name, new\_file\_name)

import os

# Rename a file from test1.txt to test2.txt

os.rename( "test1.txt", "test2.txt" )

**The remove() Method:** លុប file

Syntax: os.remove(file\_name)

import os

# Delete file test2.txt

os.remove("text2.txt")

ឬ

import os

if os.path.exists("demofile.txt"):

os.remove("demofile.txt")

else:

print("The file does not exists")

## Directory in Python

**Create Directory:**ត្រូវប្រើ mkdir() Method

import os

# Create a directory "test"

os.mkdir("test")

**Delete Folder:**ត្រូវប្រើ​ os.rmdir() method

import os

os.rmdir("myfolder")

ឬ

import os

# This would remove "/tmp/test" directory.

os.rmdir( "/tmp/test" )

**The getcwd() Method** : បង្ហាញ current working directory

import os

print(os.getcwd())

**The chdir() Method** : change the current working directory

import os

# Changing a directory to "/home/newdir"

os.chdir("/home/newdir")

## File and Directory Related Methods

More on FILE object:

|  |  |
| --- | --- |
| **Sr.No.** | **Methods with Description** |
| 1 | [**file.close()**](https://www.tutorialspoint.com/python/file_close.htm)  Close the file. A closed file cannot be read or written any more. |
| 2 | [**file.flush()**](https://www.tutorialspoint.com/python/file_flush.htm)  Flush the internal buffer, like stdio's fflush. This may be a no-op on some file-like objects. |
| 3 | [**file.fileno()**](https://www.tutorialspoint.com/python/file_fileno.htm)  Returns the integer file descriptor that is used by the underlying implementation to request I/O operations from the operating system. |
| 4 | [**file.isatty()**](https://www.tutorialspoint.com/python/file_isatty.htm)  Returns True if the file is connected to a tty(-like) device, else False. |
| 5 | [**file.next()**](https://www.tutorialspoint.com/python/file_next.htm)  Returns the next line from the file each time it is being called. |
| 6 | [**file.read([size])**](https://www.tutorialspoint.com/python/file_read.htm)  Reads at most size bytes from the file (less if the read hits EOF before obtaining size bytes). |
| 7 | [**file.readline([size])**](https://www.tutorialspoint.com/python/file_readline.htm)  Reads one entire line from the file. A trailing newline character is kept in the string. |
| 8 | [**file.readlines([sizehint])**](https://www.tutorialspoint.com/python/file_readlines.htm)  Reads until EOF using readline() and return a list containing the lines. If the optional sizehint argument is present, instead of reading up to EOF, whole lines totalling approximately sizehint bytes (possibly after rounding up to an internal buffer size) are read. |
| 9 | [**file.seek(offset[, whence])**](https://www.tutorialspoint.com/python/file_seek.htm)  Sets the file's current position |
| 10 | [**file.tell()**](https://www.tutorialspoint.com/python/file_tell.htm)  Returns the file's current position |
| 11 | [**file.truncate([size])**](https://www.tutorialspoint.com/python/file_truncate.htm)  Truncates the file's size. If the optional size argument is present, the file is truncated to (at most) that size. |
| 12 | [**file.write(str)**](https://www.tutorialspoint.com/python/file_write.htm)  Writes a string to the file. There is no return value. |
| 13 | [**file.writelines(sequence)**](https://www.tutorialspoint.com/python/file_writelines.htm)  Writes a sequence of strings to the file. The sequence can be any iterable object producing strings, typically a list of strings. |

More on OS object

|  |  |
| --- | --- |
| **Sr.No.** | **Methods with Description** |
| 1 | [**os.access(path, mode)**](https://www.tutorialspoint.com/python/os_access.htm)  Use the real uid/gid to test for access to path. |
| 2 | [**os.chdir(path)**](https://www.tutorialspoint.com/python/os_chdir.htm)  Change the current working directory to path |
| 3 | [**os.chflags(path, flags)**](https://www.tutorialspoint.com/python/os_chflags.htm)  Set the flags of path to the numeric flags. |
| 4 | [**os.chmod(path, mode)**](https://www.tutorialspoint.com/python/os_chmod.htm)  Change the mode of path to the numeric mode. |
| 5 | [**os.chown(path, uid, gid)**](https://www.tutorialspoint.com/python/os_chown.htm)  Change the owner and group id of path to the numeric uid and gid. |
| 6 | [**os.chroot(path)**](https://www.tutorialspoint.com/python/os_chroot.htm)  Change the root directory of the current process to path. |
| 7 | [**os.close(fd)**](https://www.tutorialspoint.com/python/os_close.htm)  Close file descriptor fd. |
| 8 | [**os.closerange(fd\_low, fd\_high)**](https://www.tutorialspoint.com/python/os_closerange.htm)  Close all file descriptors from fd\_low (inclusive) to fd\_high (exclusive), ignoring errors. |
| 9 | [**os.dup(fd)**](https://www.tutorialspoint.com/python/os_dup.htm)  Return a duplicate of file descriptor fd. |
| 10 | [**os.dup2(fd, fd2)**](https://www.tutorialspoint.com/python/os_dup2.htm)  Duplicate file descriptor fd to fd2, closing the latter first if necessary. |
| 11 | [**os.fchdir(fd)**](https://www.tutorialspoint.com/python/os_fchdir.htm)  Change the current working directory to the directory represented by the file descriptor fd. |
| 12 | [**os.fchmod(fd, mode)**](https://www.tutorialspoint.com/python/os_fchmod.htm)  Change the mode of the file given by fd to the numeric mode. |
| 13 | [**os.fchown(fd, uid, gid)**](https://www.tutorialspoint.com/python/os_fchown.htm)  Change the owner and group id of the file given by fd to the numeric uid and gid. |
| 14 | [**os.fdatasync(fd)**](https://www.tutorialspoint.com/python/os_fdatasync.htm)  Force write of file with filedescriptor fd to disk. |
| 15 | [**os.fdopen(fd[, mode[, bufsize]])**](https://www.tutorialspoint.com/python/os_fdopen.htm)  Return an open file object connected to the file descriptor fd. |
| 16 | [**os.fpathconf(fd, name)**](https://www.tutorialspoint.com/python/os_fpathconf.htm)  Return system configuration information relevant to an open file. name specifies the configuration value to retrieve. |
| 17 | [**os.fstat(fd)**](https://www.tutorialspoint.com/python/os_fstat.htm)  Return status for file descriptor fd, like stat(). |
| 18 | [**os.fstatvfs(fd)**](https://www.tutorialspoint.com/python/os_fstatvfs.htm)  Return information about the filesystem containing the file associated with file descriptor fd, like statvfs(). |
| 19 | [**os.fsync(fd)**](https://www.tutorialspoint.com/python/os_fsync.htm)  Force write of file with filedescriptor fd to disk. |
| 20 | [**os.ftruncate(fd, length)**](https://www.tutorialspoint.com/python/os_ftruncate.htm)  Truncate the file corresponding to file descriptor fd, so that it is at most length bytes in size. |
| 21 | [**os.getcwd()**](https://www.tutorialspoint.com/python/os_getcwd.htm)  Return a string representing the current working directory. |
| 22 | [**os.getcwdu()**](https://www.tutorialspoint.com/python/os_getcwdu.htm)  Return a Unicode object representing the current working directory. |
| 23 | [**os.isatty(fd)**](https://www.tutorialspoint.com/python/os_isatty.htm)  Return True if the file descriptor fd is open and connected to a tty(-like) device, else False. |
| 24 | [**os.lchflags(path, flags)**](https://www.tutorialspoint.com/python/os_lchflags.htm)  Set the flags of path to the numeric flags, like chflags(), but do not follow symbolic links. |
| 25 | [**os.lchmod(path, mode)**](https://www.tutorialspoint.com/python/os_lchmod.htm)  Change the mode of path to the numeric mode. |
| 26 | [**os.lchown(path, uid, gid)**](https://www.tutorialspoint.com/python/os_lchown.htm)  Change the owner and group id of path to the numeric uid and gid. This function will not follow symbolic links. |
| 27 | [**os.link(src, dst)**](https://www.tutorialspoint.com/python/os_link.htm)  Create a hard link pointing to src named dst. |
| 28 | [**os.listdir(path)**](https://www.tutorialspoint.com/python/os_listdir.htm)  Return a list containing the names of the entries in the directory given by path. |
| 29 | [**os.lseek(fd, pos, how)**](https://www.tutorialspoint.com/python/os_lseek.htm)  Set the current position of file descriptor fd to position pos, modified by how. |
| 30 | [**os.lstat(path)**](https://www.tutorialspoint.com/python/os_lstat.htm)  Like stat(), but do not follow symbolic links. |
| 31 | [**os.major(device)**](https://www.tutorialspoint.com/python/os_major.htm)  Extract the device major number from a raw device number. |
| 32 | [**os.makedev(major, minor)**](https://www.tutorialspoint.com/python/os_makedev.htm)  Compose a raw device number from the major and minor device numbers. |
| 33 | [**os.makedirs(path[, mode])**](https://www.tutorialspoint.com/python/os_makedirs.htm)  Recursive directory creation function. |
| 34 | [**os.minor(device)**](https://www.tutorialspoint.com/python/os_minor.htm)  Extract the device minor number from a raw device number. |
| 35 | [**os.mkdir(path[, mode])**](https://www.tutorialspoint.com/python/os_mkdir.htm)  Create a directory named path with numeric mode mode. |
| 36 | [**os.mkfifo(path[, mode])**](https://www.tutorialspoint.com/python/os_mkfifo.htm)  Create a FIFO (a named pipe) named path with numeric mode mode. The default mode is 0666 (octal). |
| 37 | [**os.mknod(filename[, mode=0600, device])**](https://www.tutorialspoint.com/python/os_mknod.htm)  Create a filesystem node (file, device special file or named pipe) named filename. |
| 38 | [**os.open(file, flags[, mode])**](https://www.tutorialspoint.com/python/os_open.htm)  Open the file file and set various flags according to flags and possibly its mode according to mode. |
| 39 | [**os.openpty()**](https://www.tutorialspoint.com/python/os_openpty.htm)  Open a new pseudo-terminal pair. Return a pair of file descriptors (master, slave) for the pty and the tty, respectively. |
| 40 | [**os.pathconf(path, name)**](https://www.tutorialspoint.com/python/os_pathconf.htm)  Return system configuration information relevant to a named file. |
| 41 | [**os.pipe()**](https://www.tutorialspoint.com/python/os_pipe.htm)  Create a pipe. Return a pair of file descriptors (r, w) usable for reading and writing, respectively. |
| 42 | [**os.popen(command[, mode[, bufsize]])**](https://www.tutorialspoint.com/python/os_popen.htm)  Open a pipe to or from command. |
| 43 | [**os.read(fd, n)**](https://www.tutorialspoint.com/python/os_read.htm)  Read at most n bytes from file descriptor fd. Return a string containing the bytes read. If the end of the file referred to by fd has been reached, an empty string is returned. |
| 44 | [**os.readlink(path)**](https://www.tutorialspoint.com/python/os_readlink.htm)  Return a string representing the path to which the symbolic link points. |
| 45 | [**os.remove(path)**](https://www.tutorialspoint.com/python/os_remove.htm)  Remove the file path. |
| 46 | [**os.removedirs(path)**](https://www.tutorialspoint.com/python/os_removedirs.htm)  Remove directories recursively. |
| 47 | [**os.rename(src, dst)**](https://www.tutorialspoint.com/python/os_rename.htm)  Rename the file or directory src to dst. |
| 48 | [**os.renames(old, new)**](https://www.tutorialspoint.com/python/os_renames.htm)  Recursive directory or file renaming function. |
| 49 | [**os.rmdir(path)**](https://www.tutorialspoint.com/python/os_rmdir.htm)  Remove the directory path |
| 50 | [**os.stat(path)**](https://www.tutorialspoint.com/python/os_stat.htm)  Perform a stat system call on the given path. |
| 51 | [**os.stat\_float\_times([newvalue])**](https://www.tutorialspoint.com/python/os_stat_float_times.htm)  Determine whether stat\_result represents time stamps as float objects. |
| 52 | [**os.statvfs(path)**](https://www.tutorialspoint.com/python/os_statvfs.htm)  Perform a statvfs system call on the given path. |
| 53 | [**os.symlink(src, dst)**](https://www.tutorialspoint.com/python/os_symlink.htm)  Create a symbolic link pointing to src named dst. |
| 54 | [**os.tcgetpgrp(fd)**](https://www.tutorialspoint.com/python/os_tcgetpgrp.htm)  Return the process group associated with the terminal given by fd (an open file descriptor as returned by open()). |
| 55 | [**os.tcsetpgrp(fd, pg)**](https://www.tutorialspoint.com/python/os_tcsetpgrp.htm)  Set the process group associated with the terminal given by fd (an open file descriptor as returned by open()) to pg. |
| 56 | [**os.tempnam([dir[, prefix]])**](https://www.tutorialspoint.com/python/os_tempnam.htm)  Return a unique path name that is reasonable for creating a temporary file. |
| 57 | [**os.tmpfile()**](https://www.tutorialspoint.com/python/os_tmpfile.htm)  Return a new file object opened in update mode (w+b). |
| 58 | [**os.tmpnam()**](https://www.tutorialspoint.com/python/os_tmpnam.htm)  Return a unique path name that is reasonable for creating a temporary file. |
| 59 | [**os.ttyname(fd)**](https://www.tutorialspoint.com/python/os_ttyname.htm)  Return a string which specifies the terminal device associated with file descriptor fd. If fd is not associated with a terminal device, an exception is raised. |
| 60 | [**os.unlink(path)**](https://www.tutorialspoint.com/python/os_unlink.htm)  Remove the file path. |
| 61 | [**os.utime(path, times)**](https://www.tutorialspoint.com/python/os_utime.htm)  Set the access and modified times of the file specified by path. |
| 62 | [**os.walk(top[, topdown=True[, onerror=None[, followlinks=False]]])**](https://www.tutorialspoint.com/python/os_walk.htm)  Generate the file names in a directory tree by walking the tree either top-down or bottom-up. |
| 63 | [**os.write(fd, str)**](https://www.tutorialspoint.com/python/os_write.htm)  Write the string str to file descriptor fd. Return the number of bytes actually written. |

# មេរៀនទី ៧: ERRORs AND EXCEPTIONs

Python បានផ្តល់នូវ featureសំខាន់២សម្រាប់ដោះស្រាយនូវបញ្ហាerror ដែលអាចកើតមានឡើងដោយប្រការណាមួយ ឬមិនអាចដឹងមុនគឺ Exception Handling និង Assertions។

នៅក្នុងមេរៀននេះ នឹងបង្ហាញអោយអ្នកមានសមត្ថភាពដូចខាងក្រោមៈ

* យល់ដឹងពី ប្រភេទនៃ Error
* យល់ដឹងពី របៀបដោះស្រាយ និងការពារនូវError



## Exception Handling

**Exception** : សំដៅលើ Event ដែលកើតឡើងនៅពេលកំពុងដំណើរការរបស់codeត្រូវបានហើយវាទៅកាត់ផ្តាច់នូវលំហូរនៃ program's instructionsនោះ។ Exceptionគឺជាobjectតំណាងឲ្យerrorណាមួយ។

**Handling an exception** : ដើម្បីចាប់ឬទប់ទល់ឲ្យបាននូវerrorនៃcodeដែលអាចកើតឡើងដោយប្រការណាមួយគឺគេប្រើប្រាស់ **try** block statement។

Syntax:

try:

You do your operations here;

......................

except ExceptionI:

If there is ExceptionI, then execute this block.

except ExceptionII:

If there is ExceptionII, then execute this block.

......................

else:

If there is no exception then execute this block.

|  |  |
| --- | --- |
| **Sr.No.** | **Exception Name & Description** |
| 1 | **Exception**  Base class for all exceptions |
| 2 | **StopIteration**  Raised when the next() method of an iterator does not point to any object. |
| 3 | **SystemExit**  Raised by the sys.exit() function. |
| 4 | **StandardError**  Base class for all built-in exceptions except StopIteration and SystemExit. |
| 5 | **ArithmeticError**  Base class for all errors that occur for numeric calculation. |
| 6 | **OverflowError**  Raised when a calculation exceeds maximum limit for a numeric type. |
| 7 | **FloatingPointError**  Raised when a floating point calculation fails. |
| 8 | **ZeroDivisionError**  Raised when division or modulo by zero takes place for all numeric types. |
| 9 | **AssertionError**  Raised in case of failure of the Assert statement. |
| 10 | **AttributeError**  Raised in case of failure of attribute reference or assignment. |
| 11 | **EOFError**  Raised when there is no input from either the raw\_input() or input() function and the end of file is reached. |
| 12 | **ImportError**  Raised when an import statement fails. |
| 13 | **KeyboardInterrupt**  Raised when the user interrupts program execution, usually by pressing Ctrl+c. |
| 14 | **LookupError**  Base class for all lookup errors. |
| 15 | **IndexError**  Raised when an index is not found in a sequence. |
| 16 | **KeyError**  Raised when the specified key is not found in the dictionary. |
| 17 | **NameError**  Raised when an identifier is not found in the local or global namespace. |
| 18 | **UnboundLocalError**  Raised when trying to access a local variable in a function or method but no value has been assigned to it. |
| 19 | **EnvironmentError**  Base class for all exceptions that occur outside the Python environment. |
| 20 | **IOError**  Raised when an input/ output operation fails, such as the print statement or the open() function when trying to open a file that does not exist. |
| 21 | **IOError**  Raised for operating system-related errors. |
| 22 | **SyntaxError**  Raised when there is an error in Python syntax. |
| 23 | **IndentationError**  Raised when indentation is not specified properly. |
| 24 | **SystemError**  Raised when the interpreter finds an internal problem, but when this error is encountered the Python interpreter does not exit. |
| 25 | **SystemExit**  Raised when Python interpreter is quit by using the sys.exit() function. If not handled in the code, causes the interpreter to exit. |
| 26 | **TypeError**  Raised when an operation or function is attempted that is invalid for the specified data type. |
| 27 | **ValueError**  Raised when the built-in function for a data type has the valid type of arguments, but the arguments have invalid values specified. |
| 28 | **RuntimeError**  Raised when a generated error does not fall into any category. |
| 29 | **NotImplementedError**  Raised when an abstract method that needs to be implemented in an inherited class is not actually implemented. |

ឧទាហរណ៍

try:

fh = open("testfile", "w")

fh.write("This is my test file for exception handling!!")

except IOError:

print ("Error: can\'t find file or read data")

else:

print ("Written content in the file successfully")

fh.close()

ឬ

try:

fh = open("testfile", "r")

fh.write("This is my test file for exception handling!!")

except IOError:

print ("Error: can\'t find file or read data")

else:

print ("Written content in the file successfully")

**The except Clause with No Exceptions**

try:

You do your operations here;

......................

except:

If there is any exception, then execute this block.

......................

else:

If there is no exception then execute this block.

**The except Clause with Multiple Exceptions**

try:

You do your operations here;

......................

except(Exception1[, Exception2[,...ExceptionN]]]):

If there is any exception from the given exception list,

then execute this block.

......................

else:

If there is no exception then execute this block.

**The try-finally Clause**

try:

You do your operations here;

......................

Due to any exception, this may be skipped.

finally:

This would always be executed.

......................

ឧទាហរណ៍

try:

fh = open("testfile", "w")

fh.write("This is my test file for exception handling!!")

finally:

print ("Error: can\'t find file or read data")

ឬ

try:

fh = open("testfile", "w")

try:

fh.write("This is my test file for exception handling!!")

finally:

print ("Going to close the file")

fh.close()

except IOError:

print ("Error: can\'t find file or read data")

**Argument of an Exception**

try:

You do your operations here;

......................

except ExceptionType, Argument:

You can print value of Argument here...

ឧទាហរណ៍

# Define a function here.

def temp\_convert(var):

try:

return int(var)

except ValueError, Argument:

print ("The argument does not contain numbers\n", Argument)

# Call above function here.

temp\_convert("xyz");

**Raising an Exceptions:** សម្រាប់បង្ខំឲ្យមាន error happen

Syntax: raise [Exception [, args [, traceback]]]

def functionName( level ):

if level < 1:

raise "Invalid level!", level

# The code below to this would not be executed

# if we raise the exception

ដើម្បី test នូវ raise event ខាងលើសូមប្រើគំរូcode ខាងក្រោម

try:

Business Logic here...

except "Invalid level!":

Exception handling here...

else:

Rest of the code here...

**User-Defined Exceptions**: បង្កើតប្រភេទerror ថ្មីដោយខ្លួនឯង

ដើម្បីបង្កើតexceptionថ្មីអ្នកត្រូវតែ deriving classes from the standard built-in exceptions។

ឧទាហរណ៍៖create new exception name Networkerror that derive from RuntimeError

class Networkerror(RuntimeError):

def \_\_init\_\_(self, arg):

self.args = arg

To use this error type see form code below:

try:

raise Networkerror("Bad hostname")

except Networkerror,e:

print (e.args)

**Assertions in Python(Introduced in version 1.5)**

**Assertion** : គឺជា sanity-check ដែលយើងអាច turn on or turn off នៅពេលដែលយើងបញ្ចប់នូវការ test នូវ program code។

Programmers often place assertions at the start of a function to check for valid input, and after a function call to check for valid output.

**The assert Statement:**

Syntax: assert Expression[, Arguments]

Assertion សង្ឃឹងថា result របស់expression និង true, តើបើវា false វិញនោះ Python raises an AssertionError exception។

ឧទាហរណ៍៖

def KelvinToFahrenheit(Temperature):

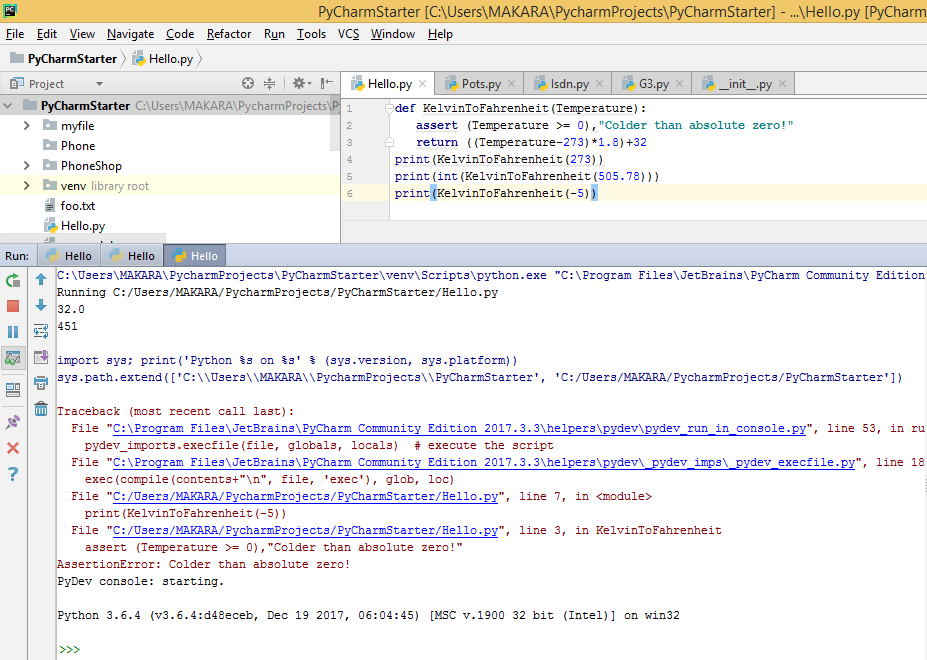
assert (Temperature >= 0),"Colder than absolute zero!"

return ((Temperature-273)\*1.8)+32

print(KelvinToFahrenheit(273))

print(int(KelvinToFahrenheit(505.78)))

print(KelvinToFahrenheit(-5))



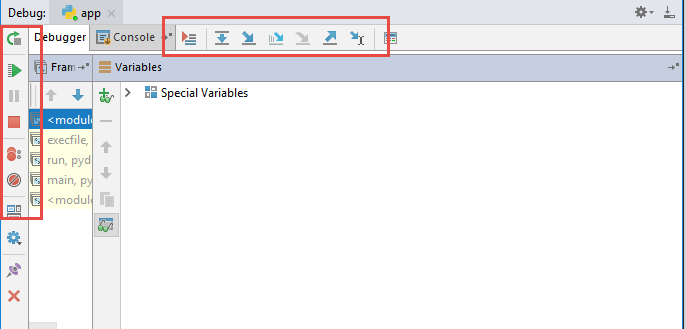
## Debugging Tools

នៅពេលដែលអ្នកជួបនូវបញ្ហាប្រភេទ logic error យើអាចប្រើប្រាស់នូវ debug tool ដើម្បីស្វែងរកកំហុសដែលបណ្តាលឲ្យលទ្ធផលចេញខុស។ វិធីនេះគឺយើងត្រូវពិនិត្យ និងស្វែងរកកំហុសម្តងមួយបន្ទាត់ៗនៅពេលអ្នកបាន run program ក្នុងកម្រិត debug mode។

**ទី១**៖ ត្រូវកំណត់ breaking point 🡪 រួចចុចលើ រូប debug ដើម្បី run



**ទី២**៖ ប្រើ tool ខាងក្រោមដើម្បី check ដំណើរការនៃ code



## Logging Module

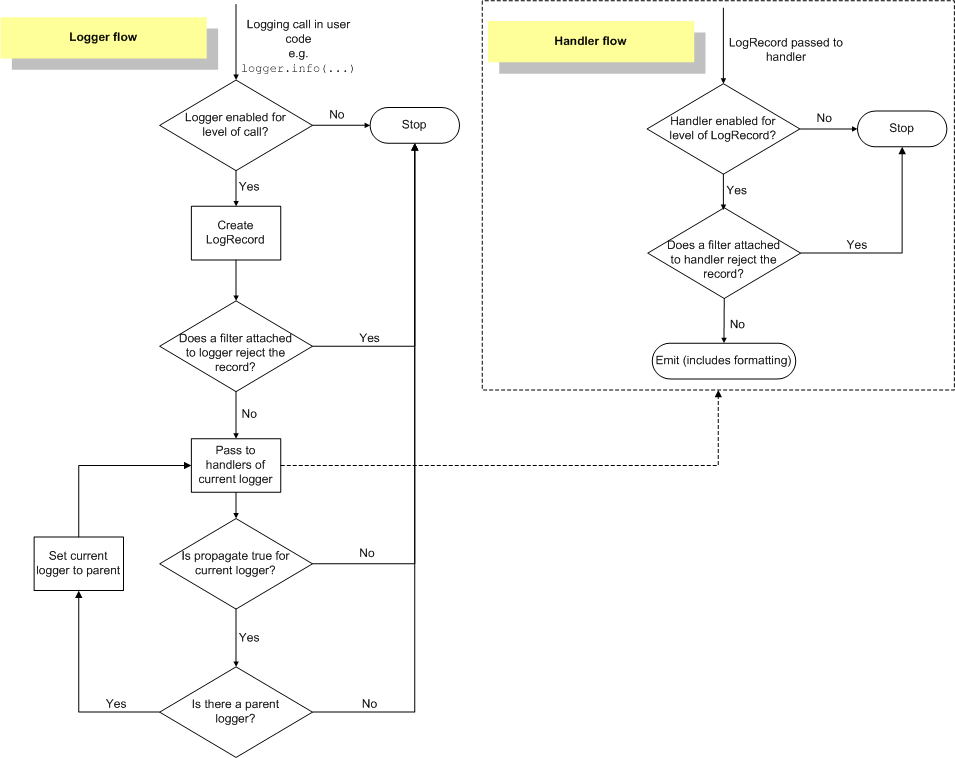
**Logging** : ជា class module ដែលយើងប្រើប្រាស់វាដើម្បីតាមដាននូវព្រឹត្តិការណ៍ផ្សេងៗដែលកើតមាននៅពេលដែលយើងធ្វើការ run software។ ជាទៅទៅ Loging បានផ្តល់ឲ្យយើងនូវការកត់ត្រា log events របស់ software នៅលើ console ក៏បាន ហើយនៅលើ file ក៏បាន។

**Logging Level** : Logging បានផ្តល់នូវ level មួយចំនួនដូចជា ៖

| **Level** | **When it’s used** | **Numeric** |
| --- | --- | --- |
| DEBUG | Detailed information, typically of interest only when diagnosing problems. | 10 |
| INFO | Confirmation that things are working as expected. | 20 |
| WARNING | An indication that something unexpected happened, or indicative of some problem in the near future (e.g. ‘disk space low’). The software is still working as expected. | 30 |
| ERROR | Due to a more serious problem, the software has not been able to perform some function. | 40 |
| CRITICAL | A serious error, indicating that the program itself may be unable to continue running. | 50 |
| NOTSET | Mix all of above | 0 |

**Logging Function** : Logging បានផ្តល់នូវ function ជាច្រើនដើម្បីshow message ៖

* logger.**debug**('debug message' [, \*args[, \*\*kwargs]])
* logger.**info**('info message' [, \*args[, \*\*kwargs]])
* logger.**warn**('warn message' [, \*args[, \*\*kwargs]])
* logger.**error**('error message' [, \*args[, \*\*kwargs]])
* logger.**critical**('critical message' [, \*args[, \*\*kwargs]])
* logger. **exception** ('exception message' [, \*args[, \*\*kwargs]])
* logger. **log** (*level*, 'critical message' [, \*args[, \*\*kwargs]])



**របៀបបង្កើត Logging Object**៖

import logging

logger = logging.getLogger(\_\_name\_\_)

logging.warning('is when this event was logged.')

**របៀប config Logging**៖

**+ config Level**: ប្រើ NOTSET បើអ្នកចង់ឲ្យ message អាចបង្ហាញបានគ្រប់ level

logging.basicConfig(level=logging.NOTSET)

**+ config Format**: ដើម្បី format រូបរាងនៃ message នៅពេលដែលអ្នកបង្ហាញ message

import logging

logging.basicConfig(format='**%(levelname)s:%(message)s**', level=logging.DEBUG)

ខាងក្រោមនេះគឺជា format keyword៖

| **Attribute name** | **Format** | **Description** |
| --- | --- | --- |
| args | You shouldn’t need to format this yourself. | The tuple of arguments merged into msgto produce message, or a dict whose values are used for the merge (when there is only one argument, and it is a dictionary). |
| asctime | %(asctime)s | Human-readable time when the[**LogRecord**](https://docs.python.org/2/library/logging.html#logging.LogRecord) was created. By default this is of the form ‘2003-07-08 16:49:45,896’ (the numbers after the comma are millisecond portion of the time). |
| created | %(created)f | Time when the **[LogRecord](https://docs.python.org/2/library/logging.html" \l "logging.LogRecord" \o "logging.LogRecord)** was created (as returned by **[time.time()](https://docs.python.org/2/library/time.html" \l "time.time" \o "time.time)**). |
| exc\_info | You shouldn’t need to format this yourself. | Exception tuple (à la sys.exc\_info) or, if no exception has occurred, None. |
| filename | %(filename)s | Filename portion of pathname. |
| funcName | %(funcName)s | Name of function containing the logging call. |
| levelname | %(levelname)s | Text logging level for the message ('DEBUG', 'INFO', 'WARNING', 'ERROR', 'CRITICAL'). |
| levelno | %(levelno)s | Numeric logging level for the message (**DEBUG**, **INFO**, **WARNING**, **ERROR**, **CRITICAL**). |
| lineno | %(lineno)d | Source line number where the logging call was issued (if available). |
| module | %(module)s | Module (name portion of filename). |
| msecs | %(msecs)d | Millisecond portion of the time when the[**LogRecord**](https://docs.python.org/2/library/logging.html#logging.LogRecord) was created. |
| message | %(message)s | The logged message, computed as msg% args. This is set when[**Formatter.format()**](https://docs.python.org/2/library/logging.html#logging.Formatter.format)is invoked. |
| msg | You shouldn’t need to format this yourself. | The format string passed in the original logging call. Merged with args to produce message, or an arbitrary object (see [Using arbitrary objects as messages](https://docs.python.org/2/howto/logging.html#arbitrary-object-messages)). |
| name | %(name)s | Name of the logger used to log the call. |
| pathname | %(pathname)s | Full pathname of the source file where the logging call was issued (if available). |
| process | %(process)d | Process ID (if available). |
| processName | %(processName)s | Process name (if available). |
| relativeCreated | %(relativeCreated)d | Time in milliseconds when the LogRecord was created, relative to the time the logging module was loaded. |
| thread | %(thread)d | Thread ID (if available). |
| threadName | %(threadName)s | Thread name (if available). |

**+ config Date Format**: សម្រាប់ format រូបរាង date/time

import logging

logging.basicConfig(format='%(asctime)s %(message)s', **datefmt**='%m/%d/%Y %I:%M:%S %p')

logging.warning('is when this event was logged.')

**+ config File Log(store on file only)**: សម្រាប់បង្កើត file log ដើម្បី store log information

import logging

logger = logging.getLogger(\_\_name\_\_)

logging.basicConfig(level=logging.NOTSET,filename='myapp.log',filemode='a')

logging.warning('is when this event was logged.')

logger.info('Start reading database')

# read database here

records = {'john': 55, 'tom': 66}

logger.debug('Records: %s', records)

logger.info('Updating records ...')

# update records here

logger.info('Finish updating records')

**+ config File Log(store on both file and console)**:

import logging

logger = logging.getLogger(\_\_name\_\_)

#1create file handler which logs even debug messages

**fh = logging.FileHandler(filename='myapp.log',mode='a+')**

**fh.setLevel(logging.NOTSET)**

#2create console handler with a higher log level

**ch = logging.StreamHandler()**

**ch.setLevel(logging.NOTSET)**

#3create formatter and add it to the handlers

formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')

fh.setFormatter(formatter)

ch.setFormatter(formatter)

#4add the handlers to the logger

**logger.addHandler(fh)**

**logger.addHandler(ch)**

logger.setLevel(logging.INFO)

logging.warning('is when this event was logged.')

logger.setLevel(logging.INFO)

logger.info('Start reading database')

# read database here

records = {'john': 55, 'tom': 66}

logger.setLevel(logging.DEBUG)

logger.debug('Records: %s', records)

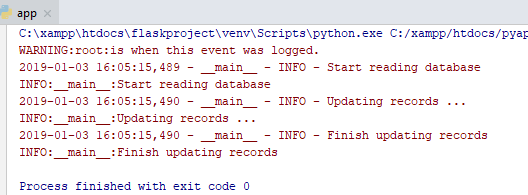
logger.setLevel(logging.INFO)

logger.info('Updating records ...')

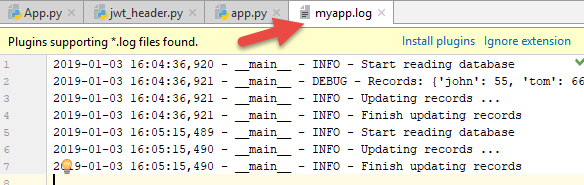
# update records here

logger.info('Finish updating records')

Console.log:



File.log



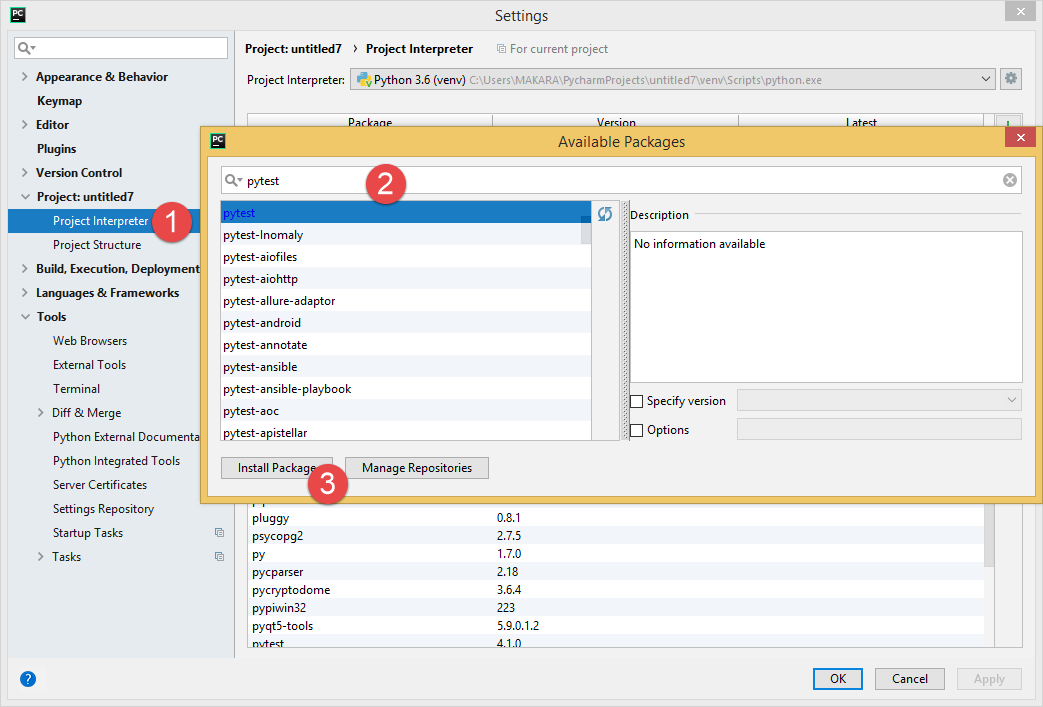
## Python Test

**Python Test**: ដូចទៅនឹងភាសាផ្សេងៗដែល python អាចឲ្យធ្វើការ test នូវគុណភាពនៃ software របស់អ្នកបាន។ **Unit Test** គឺជា python framework ដែលយើងយើងអាចយកវាមកធ្វើការ test និង វាយតម្លៃនូវគុណភាពនៃ software របស់អ្នក។ Unit Test ផ្តល់នូវមុខងារសម្រាប់ Test ដូចខាងក្រោម៖

* test fixture៖ គឺជាការ Test លើផ្នែកណាមួយនៃ software ឬ device ឬ Item ណាមួយ។ ជាទូទៅវានឹងដំណើរការជា loop ដើម្បី check មើលលទ្ធផលនៃ code(function) ដែលបានប្រើប្រាស់ក្នុង software ថាតើវាត្រឹមត្រូវឬអត់។ យើងប្រើប្រាស់វារួមជាមួយនឹង assert statement ដើម្បី check មើល initialize value ក្នុងcode(functions) នីមួយៗត្រឹមត្រូវឬអត់។
* test case៖ យើងប្រើវាដើម្បី test មើលលទ្ធផលរបស់វា(response result)លើផ្នែកណាមួយនៃ software។
* test suite៖ គឺការ test លក្ខណជាសំនុំនៃ test case ឬ test suite ឬទាំងពីរ។ វានឹង test ដោយយកសំនុំទាំងនោះទៅគណនារួមគ្នា។
* test runner៖ វាជួយសម្របសម្រួលទៅដល់ការប្រតិបត្តិការនៃការ test និងផ្តល់មកវិញនូវលទ្ធផលអ្វីមួយទៅ user។ វាអាចជា graphical interface, textual interface ឬជា special valueបញ្ជាក់ពីអ្វីមួយនៃលទ្ធផល។
* ...................................................................

**+Test Fxtures**៖ មានfunctionមួយត្រូវបានគេយកទៅcallនៅទីកន្លែងផ្សេងទៀត ហើយយើងចង់ដឹងថាតើគ្រប់កន្លែងដែលយើងបានcall នោះវា pass/fail។

ចូលទៅ project setting ដើម្បីតំឡើង pytest(**pip install pytest**)



បង្កើតfile **App.py\*\*\*\*\*\*\***

import pytest

g\_num=None

#@pytest.fixture(scope='module')#this module only

**@pytest.fixture()#all module**

def **loop\_fun**():

print('setting up: below is before test')

global g\_num

g\_num=0

**yield g\_num**# use for split into teardown method

print('teardown:after test, clear sth')

g\_num=None

def **test\_**tom\_id(**loop\_fun**):

print('\nstart test tom id')

assert 0 == g\_num

def **test\_**jonh\_id(**loop\_fun**):

print('\nstart test jonh id')

assert 0==g\_num

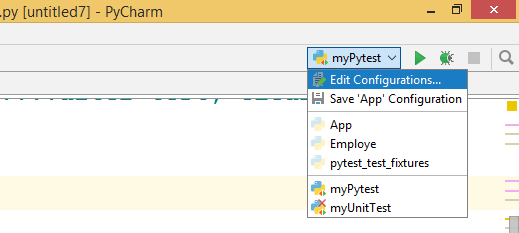
if \_\_name\_\_ == '\_\_main\_\_':

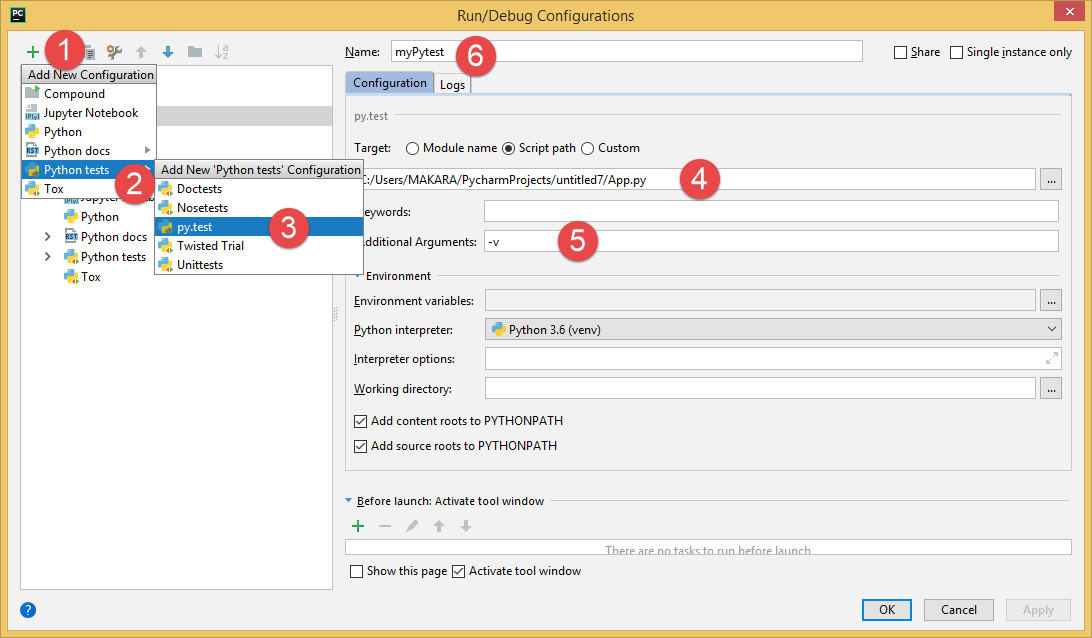
**pytest.main()**

**ចំណាំ**៖

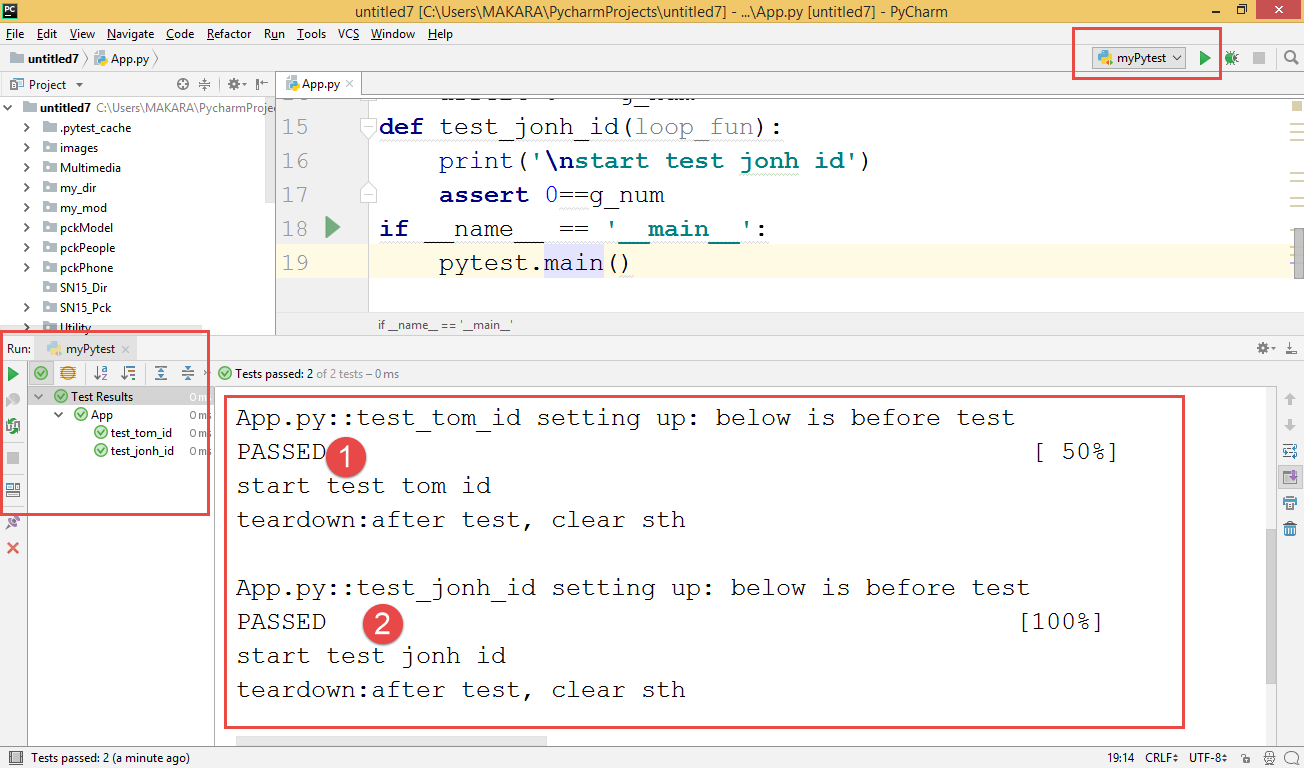
* រាល់អនុគមន៍ដែលត្រូវប្រមូលយកទៅtest ត្រូវតែសរសេរផ្តើមដោយពាក្យ **test**(អក្សរតូច)
* ហើយអនុគមន៍ដែលផ្តើមដោយ test នោះត្រូវមានparameterជាឈ្មោះរបស់ functionដែលបានប្រើប្រាស់ដោយ@python.fixture()
* រាល់ពេលrun function test ទាំងនោះគឺវាត្រូវ fixture function(lool\_fun) ជាមុនសិនមុននឹងវា run test function by fixture

ចូលទៅ Edit Configuration





បន្ទាប់មក run Pytest file ដែលទើបconfig រួចហើយ៖



**+Test Case**៖ (import unittest)

class myCalculator:

def sum(self,m,n):

return m+n;

def sub(self,m,n):

return m-n;

def mul(self,m,n):

return m\*n;

def div(self,m,n):

if(n==0):

raise ValueError('cannot divide by zero')

return m/n;

**import unittest**

class **TestCalculator**(**unittest.TestCase**):

def **test\_sum**(self):

self.assertEqual(myCalculator.sum(self,1,2),3)

self.assertEqual(myCalculator.sum(self,3, 3), 6)

self.assertEqual(myCalculator.sum(self,6, 6), 12)

def **test\_div**(self):

self.assertEqual(myCalculator.div(self,1,2),0.5)

self.assertEqual(myCalculator.div(self,3, 3), 1)

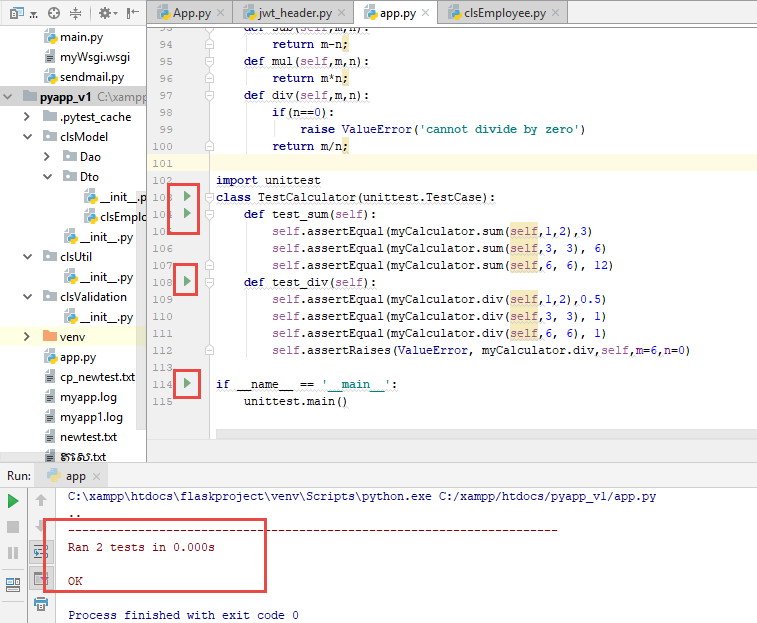
self.assertEqual(myCalculator.div(self,6, 6), 1)

self.assertRaises(ValueError, myCalculator.div,self,m=6,n=0)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

**ចំណាំ**៖ រាល់ឈ្មោះរបស់ method ដែលបង្កើតក្នុង class TestCalculator ត្រូវផ្តើមដោយ test(អក្សរតូចជានិច្ច)។



**Method ផ្សេងៗទៀតដែលប្រើក្នុង​ class unittest.TestCase**

| **Method** | **Checks that** | **New in** |
| --- | --- | --- |
| [assertEqual(a, b)](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertEqual)  assertEqual(first, second, msg=None) | a == b |  |
| [assertNotEqual(a, b)](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertNotEqual)  assertNotEqual(first, second, msg=None) | a != b |  |
| [assertTrue(x)](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertTrue)  assertTrue(expr, msg=None) | bool(x) is True |  |
| [asassertFalse(expr, msg=None)sertFalse(x)](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertFalse) | bool(x) is False |  |
| [assertIs(a, b)](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertIs)  assertIs(first, second, msg=None) | a is b | 3.1 |
| [assertIsNot(a, b)](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertIsNot)  assertIsNot(first, second, msg=None) | a is not b | 3.1 |
| [assertIsNone(x)](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertIsNone)  assertIsNone(expr, msg=None) | x is None | 3.1 |
| [assertIsNotNone(x)](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertIsNotNone)  assertIsNotNone(expr, msg=None) | x is not None | 3.1 |
| [assertIn(a, b)](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertIn)  assertIn(first, second, msg=None) | a in b | 3.1 |
| [assertNotIn(a, b)](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertNotIn)  assertNotIn(first, second, msg=None) | a not in b | 3.1 |
| [assertIsInstance(a, b)](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertIsInstance)  assertIsInstance(obj, cls, msg=None) | isinstance(a, b) | 3.2 |
| [assertNotIsInstance(a, b)](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertNotIsInstance)  assertNotIsInstance(obj, cls, msg=None) | not isinstance(a, b) | 3.2 |

យើងក៏អាចcheck នូវ production ដូចជា exceptions, warnings, និង log messages ប្រើនូវ methods ដូចខាងក្រោម៖

| **Method** | **Checks that** | **New in** |
| --- | --- | --- |
| [**assertRaises(exc, fun, \*args,\*\*kwds)**](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertRaises)  assertRaises(exception, callable, \*args, \*\*kwds)  assertRaises(exception, \*, msg=None) | fun(\*args, \*\*kwds)raises *exc* |  |
| [**assertRaisesRegex(exc, r, fun,\*args, \*\*kwds)**](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertRaisesRegex)  assertRaisesRegex(exception, regex, callable, \*args, \*\*kwds)  assertRaisesRegex(exception, regex, \*, msg=None) | fun(\*args, \*\*kwds)raises *exc* and the message matches regex *r* | 3.1 |
| [**assertWarns(warn, fun, \*args,\*\*kwds)**](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertWarns)  assertWarns(warning, callable, \*args, \*\*kwds)  assertWarns(warning, \*, msg=None) | fun(\*args, \*\*kwds)raises *warn* | 3.2 |
| [**assertWarnsRegex(warn, r, fun,\*args, \*\*kwds)**](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertWarnsRegex)  assertWarnsRegex(warning, regex, callable, \*args, \*\*kwds)  assertWarnsRegex(warning, regex, \*, msg=None) | fun(\*args, \*\*kwds)raises *warn* and the message matches regex *r* | 3.2 |
| [**assertLogs(logger, level)**](https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertLogs)  assertLogs(logger=None, level=None) | The with block logs on *logger* with minimum *level* | 3.4 |

# មេរៀនទី​ ៨: Python - Object Oriented



## សេចក្តីណែនាំ Introduction to OOP

OOP គឺជា structure មួយដែលត្រូវបានគេប្រើប្រាស់សំរាប់កសាង software ក្នុងទំរង់ជា unit ហើយដែលត្រូវបានគេស្គាល់ថាជា object។ រាល់កម្មវិធី programming language ទាំងឡាយណាដែល support ជាមួយនឹង OOP នេះត្រូវមានមូលដ្ឋានគ្រឹះសំខាន់ៗ ៤ យ៉ាងដូចខាងក្រោម៖

* Class and Object : បង្កើតdata type ថ្មី(class) និងហៅមកប្រើវិញដោយបង្កើតជា object
* Encapsulation : ជាវិធីសាស្ត្រលាក់ attribute នៅក្នុង class ណាមួយមិនឲ្យគេ access ពី class ខាងក្រៅ តែអាចឲ្យគេ access លើវាបានតាមរយៈ public method។
* Inheritance : ជាការផ្ទេរមរតក(attribute and method)ពីclass មួយទៅclass មួយទៀត។ class ដែលជាអ្នកផ្ទេរមរតក ហៅថា supper class រឺ base class រឺ parent class។ class ដែលជាអ្នកទទួលមរតកពីគេ គេហៅថា sub class រឺ dervided class រឺ child class។
* Polymorphism : សំដៅលើការធ្វើប្រតិបត្តិការផ្សេងគ្នា បើទោះបីជា object នីមួយៗត្រូវបាន call ចេញពី method តែមួយក៍ដោយ។ មានបច្ចេកទេសសំខាន់ពីរគឺ Overloading and Overiding Method

ដើម្បីកាន់តែងាយស្រួលក្នុងការសិក្សា OOP ក្នុងកម្មវិធី Python យើងគួរស្គាល់នូវពាក្យមួយចំនួនខាងក្រោមសិន៖

* **Class** − A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
* **Class** **variable** − A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
* **Data** **member** − A class variable or instance variable that holds data associated with a class and its objects.
* **Function overloading** − The assignment of more than one behavior to a particular function. The operation performed varies by the types of objects or arguments involved.
* **Instance variable** − A variable that is defined inside a method and belongs only to the current instance of a class.
* **Inheritance** − The transfer of the characteristics of a class to other classes that are derived from it.
* **Instance** − An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle.
* **Instantiation** − The creation of an instance of a class.
* Method − A special kind of function that is defined in a class definition.
* **Object** − A unique instance of a data structure that's defined by its class. An object comprises both data members (class variables and instance variables) and methods.
* **Operator overloading** − The assignment of more than one function to a particular operator.
* **Keyword self = this ក្នុងភាសា java or c#, = Me in VB and VB.Net**

## ការបង្កើត Class and Object

**Class : គឺជាពុំគំរូ template សំរាប់បង្កើត object។ class មួយអាចបង្កើតបាន ច្រើន object។**

**Object : គឺជាការ instance(attribute and method) ដែលកើតចេញពីclass ។**

**រូបមន្តបង្កើត class**

**class ClassName:**

**'Optional class documentation string'**

**class\_suite**

**The class\_suite** consists of all the component statements defining class members, data attributes and functions។

**The class** has a documentation string, which can be accessed via **ClassName.\_\_doc\_\_**

**ឧទាហរណ៍**

**class Employee:**

**'Common base class for all employees'**

**empCount = 0**

**def \_\_init\_\_(self, name, salary):**

**self.name = name**

**self.salary = salary**

**Employee.empCount += 1**

**def displayCount(self):**

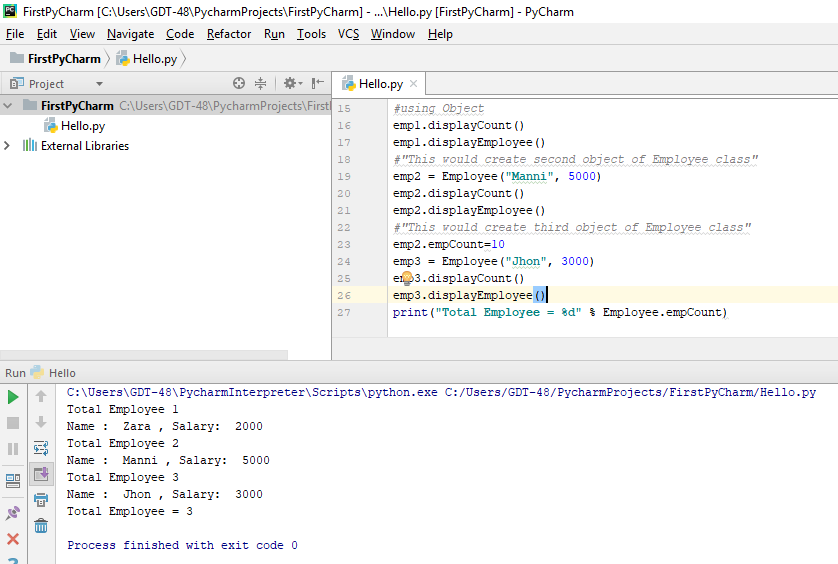
**print ("Total Employee %d" % Employee.empCount)**

**def displayEmployee(self):**

**print ("Name : ", self.name, ", Salary: ", self.salary)**

**ពន្យល់ :**

* + **empCount គឺជា class variable ដែល value របស់វាអាច shared គ្រប់ instances ទាំងអស់ដែលកើតចេញពី class មួយនេះ។ យើងអាច access វាបានដោយប្រើ Employee.empCount from នៅក្នុង class ក៏បានឬ ក្រៅ class ក៏បាន.**
  + method **\_\_init\_\_() វាគឺជា special method ដែលអាចហៅម្យ៉ាងទៀតបានថា class constructor ឬ initialization method ដែរ Python ហៅវាឲ្យដំណើរការនៅរាល់ពេលដែរអ្នកបានបង្កើត new instance ចេញពី classនេះ**។
  + **ចំណែកឯ displayCount(...) និង displayEmployee(..) គឺជា class method។**

**Creating Instance Objects :**

class Employee:

'Common base class for all employees'

empCount = 0

def \_\_init\_\_(self, name, salary):

self.name = name

self.salary = salary

Employee.empCount += 1

def displayCount(self):

print ("Total Employee %d" % Employee.empCount)

def displayEmployee(self):

print ("Name : ", self.name, ", Salary: ", self.salary)

#Below is creating the object

#"This would create first object of Employee class"

emp1 = Employee("Zara", 2000)

#using Object

emp1.displayCount()

emp1.displayEmployee()

#"This would create second object of Employee class"

emp2 = Employee("Manni", 5000)

emp2.displayCount()

emp2.displayEmployee()

#"This would create third object of Employee class"

emp2.empCount=10

emp3 = Employee("Jhon", 3000)

emp3.displayCount()

emp3.displayEmployee()

print ("Total Employee %d" % Employee.empCount)

🡺You can **add, remove, or modify** attributes of classes and objects at any time

emp1.age = 7 # Add an 'age' attribute.

emp1.age = 8 # Modify 'age' attribute.

emp1.displayEmployee()

print("Age=",emp1.age)

del emp1.age # Delete 'age' attribute.

print("Age=",emp1.age) #will be error

**យើងក៏អាចប្រើប្រាស់** function **ខាងក្រោម៖**

* The **getattr(obj, name[, default])** − to access the attribute of object.
* The **hasattr(obj,name)** − to check if an attribute exists or not.
* The **setattr(obj,name,value)** − to set an attribute. If attribute does not exist, then it would be created.
* The **delattr(obj, name)** − to delete an attribute.

hasattr(emp1, 'age') # Returns true if 'age' attribute exists

getattr(emp1, 'age') # Returns value of 'age' attribute

setattr(emp1, 'age', 8) # Set attribute 'age' at 8

delattr(empl, 'age') # Delete attribute 'age'

**Built-In Class Attributes:គ្រប់ class ទាំងអស់សុទ្ធតែមាន attribute ខាងក្រោម៖**

* **\_\_dict\_\_** : Dictionary containing the class's namespace.
* **\_\_doc\_\_** : Class documentation string or none, if undefined.
* **\_\_name\_\_** : Class name.
* **\_\_module\_\_** : Module name in which the class is defined. This attribute is "\_\_main\_\_" in interactive mode.
* **\_\_bases\_\_** : A possibly empty tuple containing the base classes, in the order of their occurrence in the base class list.

class Employee:

'Common base class for all employees'

empCount = 0

def \_\_init\_\_(self, name, salary):

self.name = name

self.salary = salary

Employee.empCount += 1

def displayCount(self):

print ("Total Employee %d" % Employee.empCount)

def displayEmployee(self):

print ("Name : ", self.name, ", Salary: ", self.salary)

print("Employee.\_\_doc\_\_:", Employee.\_\_doc\_\_)

print("Employee.\_\_name\_\_:", Employee.\_\_name\_\_)

print("Employee.\_\_module\_\_:", Employee.\_\_module\_\_)

print("Employee.\_\_bases\_\_:", Employee.\_\_bases\_\_)

print("Employee.\_\_dict\_\_:", Employee.\_\_dict\_\_)

**Destroying Objects (Garbage Collection):**

* **ជាទូទៅ Python នឹងលុប uneeded object(**built-in types or class instances**)ដោយស្វ័យប្រវត្តិដើម្បីចំនេញ memory space។**
* **Garbage Collector : វានឹង destroy object នៅពេលដែល program RUN ហើយនិងobject's reference រាប់ទៅឃើញ zero។ object's reference រាប់កើនពេលដែលវាត្រូវបានគេដាក់ឈ្មោះថ្មីឬយកវាទៅដាក់នៅក្នុងពពួកcontainer(**list, tuple, or dictionary**) ហើយវារាប់ថយនៅពេលដែលយើងលុប​​ object ដោយប្រើពាក្យ del ។**
* **Destructor​ : ជា special method ដែលដំណើរការ automatic នៅពេលដែល** **Garbage Collector វាដំណើរការ destroy object។ យើងអាចប្រើវាដើម្បីគ្រាន់តែជាការ noted ពេលdestroy object ហើយយើងអាចបង្កើតវាបានដោយឈ្មោះmethod \_\_del\_\_() ក្នុងclass។**
* Method **id(..) ជាbuild-in mehtodប្រើដើម្បីបង្ហាញលេខរៀង(address)របស់object**

a = 40 # Create object <40>

b = a # Increase ref. count of <40>

c = [b] # Increase ref. count of <40>

del a # Decrease ref. count of <40>

b = 100 # Decrease ref. count of <40>

c[0] = -1 # Decrease ref. count of <40>

**ឧទាហរណ៍**

class Point:

def \_\_init\_\_( self, x=0, y=0):

self.x = x

self.y = y

def \_\_del\_\_(self):

class\_name = self.\_\_class\_\_.\_\_name\_\_

print(class\_name, "destroyed")

pt1 = Point()

pt2 = pt1

pt3 = pt1

print(id(pt1), id(pt2), id(pt3)) # prints the ids of the obejcts

del pt1

del pt2

del pt3 #you can delete this line

**Static Variable/Function : ជាទូទៅក្នុង python class គ្រប់ public function និង variable គឺសុទ្ធតែជា static ។ មានន័យថាយើង access លើ variable/function ដោយមិនបាច់បង្កើត object។**

**File(Product.py)**

class Product:

\_\_ProductID=None

\_\_ProductName=None

\_\_FullDescription=None

\_\_Cost=None

\_\_Price=None

def getProductName(self):

return self.\_\_ProductName

def setProductName(self,pName):

self.\_\_ProductName=pName

def \_\_printProductName(self):

print(self.\_\_ProductName)

**File Main(App.py)**

from clsModel.Dto.clsProduct import Product

Product.setProductName(Product,'Milkita')

print(Product.getProductName(Product))

## ការបង្កើត Encapsulation(Data Hiding)

ជាវិធីសាស្ត្រលាក់ attribute នៅក្នុង class ណាមួយមិនឲ្យគេ access ពី class ខាងក្រៅ តែអាចឲ្យគេ access លើវាបានតាមរយៈ public method។

Other Class

Class

Private attribute

Public method

ដើម្បី hide attribute យើងត្រូវ \_ \_ ពីមុខ

ឧទាហរណ៍

class JustCounter:

\_\_secretCount = 0

def count(self):

self.\_\_secretCount += 1

print(self.\_\_secretCount)

counter = JustCounter()

counter.count()

counter.count()

print(counter.\_\_secretCount)#This line make error

-------------------------------------------------ឧទាហរណ៍គំរូ----------------------------------------------------  
**-------file model(clsProduct.py)**

tblProduct=[{'01':['ABC','ABC Des',0,0]}]

class ProductDto:

\_\_ProductID=None

\_\_ProductName=None

\_\_FullDescription=None

\_\_Cost=None

\_\_Price=None

def setProductID(self,pid):

self.\_\_ProductID=pid

def getProductID(self):

return self.\_\_ProductID

def setProductName(self,pname):

self.\_\_ProductName=pname

def getProductName(self):

return self.\_\_ProductName

def getFullDescription(self):

return self.\_\_FullDescription

def setFullDescription(self,pdesc):

self.\_\_FullDescription=pdesc

def setCost(self,cost):

self.\_\_Cost = cost

def getCost(self):

return self.\_\_Cost

def setPrice(self,price):

self.\_\_Price = price

def getPrice(self):

return self.\_\_Price

class ProductDao:

def findOne(self,pid):

global tblProduct

for itm in tblProduct:

for k,v in dict(itm).items():

if k==pid:

print(itm)

return;

print('Record not found!')

def findAll(self=None):

global tblProduct

for itm in tblProduct:

print(itm)

def addOne(self,pid,pname,pdesc,cost,price):

global tblProduct

itm={str(pid):[str(pname),str(pdesc),float(cost),float(price)]}

tblProduct.append(itm)

return True

def addMany(self,pList):

count=0;

global tblProduct

for p in pList:

pid=p.getProductID()

pname=p.getProductName()

pdesc=p.getFullDescription()

cost=p.getCost()

price=p.getPrice()

itm={str(pid):[str(pname),str(pdesc),float(cost),float(price)]}

tblProduct.append(itm)

count=count+1;

return count

def removeID(self,pid):

global tblProduct

for itm in tblProduct:

for k, v in dict(itm).items():

if k == pid:

tblProduct=list(tblProduct)

tblProduct.remove(itm)

return;

print('Remove not found!')

**-------file main(App.py)**

from clsModel.clsProduct import ProductDto as pdto,ProductDao as pdao

pdao.findOne(pdto,'02')

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

pdao.findAll()

print('------------add one-----------')

pdao.addOne(pdto,'02','XO','desc',1,2)

pdao.findAll()

print('-----------add many------------')

p1=pdto();

p1.setProductID('04'); p1.setProductName('Fanta'); p1.setCost(0); p1.setPrice(0)

p2=pdto();

p2.setProductID('05');p2.setProductName('Pepsi');p2.setCost(1); p2.setPrice(1)

pList=[]

pList.append(p1); pList.append(p2)

pdao.addMany(pdao,pList)

pdao.findAll()

print('-----------remove one------------')

pdao.removeID(pdao,'04')

pdao.findAll()

## ការបង្កើត Inheritance

ជាការផ្ទេរមរតក(attribute and method)ពីclass មួយទៅclass មួយទៀត។ class ដែលជាអ្នកផ្ទេរមរតក ហៅថា supper class រឺ base class រឺ parent class។ class ដែលជាអ្នកទទួលមរតកពីគេ គេហៅថា sub class រឺ dervided class រឺ child class។

**Syntax**

**class SubClassName (ParentClass1[, ParentClass2, ...]):**

**'Optional class documentation string'**

**class\_suite**

-----------------------------------------------------------------------------------------

class **Parent**: # define parent class

parentAttr = 100

def \_\_init\_\_(self):

print("Calling parent constructor")

def parentMethod(self):

print('Calling parent method')

def setAttr(self, attr):

Parent.parentAttr = attr

def getAttr(self):

print("Parent attribute :", Parent.parentAttr)

class **Child**(Parent): # define child class

def \_\_init\_\_(self):

**Parent.\_\_init\_\_(self)**

print("Calling child constructor")

def childMethod(self):

print('Calling child method')

c = Child() # instance of child

c.childMethod() # child calls its method

c.parentMethod() # calls parent's method

c.setAttr(200) # again call parent's method

c.getAttr() # again call parent's method

**Build-In Function:**

* The **issubclass(sub, sup)** boolean function returns true if the given subclass sub is indeed a subclass of the superclass sup.
* The **isinstance(obj, Class)** boolean function returns true if obj is an instance of class Class or is an instance of a subclass of Class

## ការបង្កើត Polymorphism

សំដៅលើការធ្វើប្រតិបត្តិការផ្សេងគ្នា បើទោះបីជា object នីមួយៗត្រូវបាន call ចេញពី method

តែមួយក៍ដោយ។

ចូរបង្កើត class ដូចរូបខាងក្រោម៖

Human

Hello()

Japanese

Chinese

Override

Hello()

Hello()

**Overiding method៖ គឺជា method ដែលត្រូវបានគេយកទៅសរសេរឡើងវិញនៅក្នុង sub class។**

**ជាទូទៅវាត្រូវ៖**

* **មានឈ្មោះ និង parameter ដូចគ្នាបេះបិទទៅនឹង method របស់ base class**
* **តែសកម្មភាពអាចធ្វើការងារខុសគ្នា**

**ឧទាហរណ៍**

class Parent: # define parent class

def **myMethod**(self):

print('Calling parent method')

class Child(Parent): # define child class

def **myMethod**(self):#overide method

print ('Calling child method')

c=[Parent(),Child()]

**c[0].myMethod()** # instance of child

**c[1].myMethod()** # child calls overridden method

ខាងក្រោមនេះគឺជា build-in function ដែលអ្នកអាចយកវាទៅ override ក្នុងclass បាន

|  |  |
| --- | --- |
| **Sr.No.** | **Method, Description & Sample Call** |
| 1 | **\_\_init\_\_ ( self [,args...] )**  Constructor (with any optional arguments)  Sample Call : *obj = className(args)* |
| 2 | **\_\_del\_\_( self )**  Destructor, deletes an object  Sample Call : *del obj* |
| 3 | **\_\_repr\_\_( self )**  Evaluable string representation  Sample Call : *repr(obj)* |
| 4 | **\_\_str\_\_( self )**  Printable string representation  Sample Call : *str(obj)* |
| 5 | **\_\_cmp\_\_ ( self, x )**  Object comparison  Sample Call : *cmp(obj, x)* |

**Overloading Function៖ គឺជា function ស្ថិតនៅក្នុងclassតែមួយហើយត្រូវ៖**

* **មានឈ្មោះ ដូចគ្នា តែរបៀបcallត្រូវមានparrameter ខុសគ្នា**
* **សកម្មភាពអាចធ្វើការងារខុសគ្នា**
* **Python ប្រើប្រាស់ optional parameter ដើម្បីបង្កើតជា overloading function**

**ឧទាហរណ៍**

class MyClass:

def myMethod(self,Name='default'):

print('Hello {} {}'.format('Mr.',Name))

obj=MyClass()

obj.myMethod()

obj.myMethod("Makara")

**Overloading Operator៖ ជាវិធីសាស្ត្រក្នុងការ customize នៅពេលដែលអ្នកប្រើប្រាស់ + - \* / %....នៅលើ object។**

**ឧទាហរណ៍**

class Vector:

def \_\_init\_\_(self, a, b):

self.a = a

self.b = b

def \_\_str\_\_(self):

return 'Vector (%d, %d)' % (self.a, self.b)

def \_\_add\_\_(self, other):#it call when you use + on object

return Vector(self.a + other.a, self.b + other.b)

v1 = Vector(2, 10)

v2 = Vector(5, -2)

print(v1 + v2)

**Python Abstract Classes**

Abstraction is the concept in object-oriented programming that is used to hide the internal functionality of the classes from the users. Abstraction is implemented using the abstract classes. An abstract class in Python is typically created to declare a set of methods that must be created in any child class built on top of this abstract class. Similarly, an abstract method is one that doesn't have any implementation.

We cannot create an abstract class in Python directly. However, Python does provide a module that allows us to define abstract classes. The module we can use to create an abstract class in Python is **abc**(abstract base class) module.

Abstract methods force the child classes to give the implementation of these methods in them and thus help us achieve abstraction as each subclass can give its own implementation. A class containing one or more than one abstract method is called an abstract class.

We can use the following syntax to create an abstract class in Python:

from abc import ABC

class <Abstract\_Class\_Name>(ABC):

# body of the class

Here we just need to inherit the ABC class from the abc module in Python.

from abc import ABC

class DemoAbstractClass(ABC):

pass

To define an abstract method we use the @abstractmethod decorator of the abc module. It tells Python that the declared method is abstract and should be overridden in the child classes.

from abc import ABC, abstractmethod

class <Abstract\_Class\_Name>(ABC):

@abstractmethod

def <abstract\_method\_name>(self,other\_parameters):

pass

We just need to put this decorator over any function we want to make abstract, and the abc module takes care of the rest.

from abc import ABC, abstractmethod

class DemoAbstractClass(ABC):

@abstractmethod

def abstract\_method\_name(self):

pass

**Example1:**

**Step1**: Create Abstract Class

from abc import ABC, abstractmethod

class Shape(ABC):

def \_\_init\_\_(self, shape\_name):

self.shape\_name = shape\_name

@abstractmethod

def draw(self):

pass

**Step2**: Impement Abstract Class

from Shape import Shape

class Circle(Shape):

def \_\_init\_\_(self):

super().\_\_init\_\_("circle")

def draw(self):

print("Drawing a Circle")

**Step3**: Using Child Class

from Circle import Circle

#create a circle object

circle = Circle()

circle.draw()

**Example2**:

from abc import ABC,abstractmethod

class abClass(ABC):

@abstractmethod

def requireMothod(self):

pass;

class baseClass():

def printMe(self):

print("Hello from base class!")

class usingAbstClass(baseClass,abClass):

def childMethod(self):

print("hello from child method!")

def requireMothod(self):

print("this method must be overwrite!")

obj=usingAbstClass();

obj.printMe()

obj.childMethod();

obj.requireMothod();

Special Method ផ្សេងៗទៀតដែលអ្នកអាចយកទៅ overloading បាន

| **OPERATOR** | **FUNCTION** | **METHOD DESCRIPTION** |
| --- | --- | --- |
| + | \_\_add\_\_(self, other) | Addition |
| \* | \_\_mul\_\_(self, other) | Multiplication |
| - | \_\_sub\_\_(self, other) | Subtraction |
| % | \_\_mod\_\_(self, other) | Remainder |
| / | \_\_truediv\_\_(self, other) | Division |
| < | \_\_lt\_\_(self, other) | Less than |
| <= | \_\_le\_\_(self, other) | Less than or equal to |
| == | \_\_eq\_\_(self, other) | Equal to |
| != | \_\_ne\_\_(self, other) | Not equal to |
| > | \_\_gt\_\_(self, other) | Greater than |
| >= | \_\_ge\_\_(self, other) | Greater than or equal to |
| [index] | \_\_getitem\_\_(self, index) | Index operator |
| in | \_\_contains\_\_(self, value) | Check membership |
| len | \_\_len\_\_(self) | The number of elements |
| str | \_\_str\_\_(self) | The string representation |

## Reqular Expressions

វាមិនមែនជាមេរៀននៃ OOP ទេ។ regular expression គឺជា special sequence of characters ដែលជួយអ្នកក្នុងការផ្គូរផ្គង(match) ឬស្វែងរក(find)ក្នុង strings ឬ sets of strings។

**The match Function**:

**Syntax**: re.match(pattern, string, flags=0)

This function attempts to match RE pattern to string with optional flags.

|  |  |
| --- | --- |
| **Sr.No.** | **Parameter & Description** |
| 1 | **pattern**  This is the regular expression to be matched. |
| 2 | **string**  This is the string, which would be searched to match the pattern at the beginning of string. |
| 3 | **flags**  You can specify different flags using bitwise OR (|). These are modifiers, which are listed in the table below. |

នៅពេលដែលអ្នកប្រើ re.match function នោះវានឹងreturn មកវិញជា match object ពេលsuccess តែបើ fail វិញវានឹងreturn​ **None**។

|  |  |
| --- | --- |
| **Sr.No.** | **Match Object Method & Description** |
| 1 | **group(num=0)**  This method returns entire match (or specific subgroup num) |
| 2 | **groups()**  This method returns all matching subgroups in a tuple (empty if there weren't any) |

ឧទាហរណ៍

import re

line = "Cats are smarter than dogs"

matchObj = re.match( r'(.\*) are (.\*?) .\*', line, re.M|re.I)

if matchObj:

print("matchObj.group() : ", matchObj.group())

print("matchObj.group(1) : ", matchObj.group(1))

print("matchObj.group(2) : ", matchObj.group(2))

else:

print("No match!!")

**The search Function:**

**Syntax** : re.search(pattern, string, flags=0)

This function searches for first occurrence of RE pattern within string with optional flags.

|  |  |
| --- | --- |
| **Sr.No.** | **Parameter & Description** |
| 1 | **pattern**  This is the regular expression to be matched. |
| 2 | **string**  This is the string, which would be searched to match the pattern anywhere in the string. |
| 3 | **flags**  You can specify different flags using bitwise OR (|). These are modifiers, which are listed in the table below. |

The research function វានឹង return match object ពេល​success និង​return None ពេលfail ដូចទៅនឹង rematch ដែល។ ខាងក្រោមជា​​​method ដែលប្រើជាមួយវា៖

|  |  |
| --- | --- |
| **Sr.No.** | **Match Object Methods & Description** |
| 1 | **group(num=0)**  This method returns entire match (or specific subgroup num) |
| 2 | **groups()**  This method returns all matching subgroups in a tuple (empty if there weren't any) |

ឧទាហរណ៍

import re

line = "Cats are smarter than dogs";

searchObj = re.search( r'(.\*) are (.\*?) .\*', line, re.M|re.I)

if searchObj:

print("searchObj.group() : ", searchObj.group())

print("searchObj.group(1) : ", searchObj.group(1))

print("searchObj.group(2) : ", searchObj.group(2))

else:

print("Nothing found!!")

**Matching Versus Searching**

Python offers two different primitive operations based on regular expressions: **match** checks for a match only at the beginning of the string, while **search** checks for a match anywhere in the string (this is what Perl does by default).

import re

line = "Cats are smarter than dogs";

matchObj = re.match( r'dogs', line, re.M|re.I)

if matchObj:

print("match --> matchObj.group() : ", matchObj.group())

else:

print("No match!!")

searchObj = re.search( r'dogs', line, re.M|re.I)

if searchObj:

print("search --> searchObj.group() : ", searchObj.group())

else:

print("Nothing found!!")

**Search and Replace:**

**Syntax**: re.sub(pattern, repl, string, max=0)

This method replaces all occurrences of the RE pattern in string with repl, substituting all occurrences unless max provided. This method returns modified string.

ឧទាហរណ៍

import re

phone = "2004-959-559 # This is Phone Number"

# Delete Python-style comments

num = re.sub(r'#.\*$', "", phone)

print("Phone Num : ", num)

# Remove anything other than digits

num = re.sub(r'\D', "", phone)

print ("Phone Num : ", num)

**Regular Expression Modifiers: Option Flags:**

|  |  |
| --- | --- |
| **Sr.No.** | **Modifier & Description** |
| 1 | **re.I**  Performs case-insensitive matching. |
| 2 | **re.L**  Interprets words according to the current locale. This interpretation affects the alphabetic group (\w and \W), as well as word boundary behavior(\b and \B). |
| 3 | **re.M**  Makes $ match the end of a line (not just the end of the string) and makes ^ match the start of any line (not just the start of the string). |
| 4 | **re.S**  Makes a period (dot) match any character, including a newline. |
| 5 | **re.U**  Interprets letters according to the Unicode character set. This flag affects the behavior of \w, \W, \b, \B. |
| 6 | **re.X**  Permits "cuter" regular expression syntax. It ignores whitespace (except inside a set [] or when escaped by a backslash) and treats unescaped # as a comment marker. |

**Regular Expression Patterns:**

|  |  |
| --- | --- |
| **Sr.No.** | **Pattern & Description** |
| 1 | **^**  Matches beginning of line. |
| 2 | **$**  Matches end of line. |
| 3 | **.**  Matches any single character except newline. Using m option allows it to match newline as well. |
| 4 | **[...]**  Matches any single character in brackets. |
| 5 | **[^...]**  Matches any single character not in brackets |
| 6 | **re\***  Matches 0 or more occurrences of preceding expression. |
| 7 | **re+**  Matches 1 or more occurrence of preceding expression. |
| 8 | **re?**  Matches 0 or 1 occurrence of preceding expression. |
| 9 | **re{ n}**  Matches exactly n number of occurrences of preceding expression. |
| 10 | **re{ n,}**  Matches n or more occurrences of preceding expression. |
| 11 | **re{ n, m}**  Matches at least n and at most m occurrences of preceding expression. |
| 12 | **a| b**  Matches either a or b. |
| 13 | **(re)**  Groups regular expressions and remembers matched text. |
| 14 | **(?imx)**  Temporarily toggles on i, m, or x options within a regular expression. If in parentheses, only that area is affected. |
| 15 | **(?-imx)**  Temporarily toggles off i, m, or x options within a regular expression. If in parentheses, only that area is affected. |
| 16 | **(?: re)**  Groups regular expressions without remembering matched text. |
| 17 | **(?imx: re)**  Temporarily toggles on i, m, or x options within parentheses. |
| 18 | **(?-imx: re)**  Temporarily toggles off i, m, or x options within parentheses. |
| 19 | **(?#...)**  Comment. |
| 20 | **(?= re)**  Specifies position using a pattern. Doesn't have a range. |
| 21 | **(?! re)**  Specifies position using pattern negation. Doesn't have a range. |
| 22 | **(?> re)**  Matches independent pattern without backtracking. |
| 23 | **\w**  Matches word characters. |
| 24 | **\W**  Matches nonword characters. |
| 25 | **\s**  Matches whitespace. Equivalent to [\t\n\r\f]. |
| 26 | **\S**  Matches nonwhitespace. |
| 27 | **\d**  Matches digits. Equivalent to [0-9]. |
| 28 | **\D**  Matches nondigits. |
| 29 | **\A**  Matches beginning of string. |
| 30 | **\Z**  Matches end of string. If a newline exists, it matches just before newline. |
| 31 | **\z**  Matches end of string. |
| 32 | **\G**  Matches point where last match finished. |
| 33 | **\b**  Matches word boundaries when outside brackets. Matches backspace (0x08) when inside brackets. |
| 34 | **\B**  Matches nonword boundaries. |
| 35 | **\n, \t, etc.**  Matches newlines, carriage returns, tabs, etc. |
| 36 | **\1...\9**  Matches nth grouped subexpression. |
| 37 | **\10**  Matches nth grouped subexpression if it matched already. Otherwise refers to the octal representation of a character code. |

**Regular Expression Examples:**



## Literal characters

|  |  |
| --- | --- |
| **Sr.No.** | **Example & Description** |
| 1 | **python**  Match "python". |

## Character classes

|  |  |
| --- | --- |
| **Sr.No.** | **Example & Description** |
| 1 | **[Pp]ython**  Match "Python" or "python" |
| 2 | **rub[ye]**  Match "ruby" or "rube" |
| 3 | **[aeiou]**  Match any one lowercase vowel |
| 4 | **[0-9]**  Match any digit; same as [0123456789] |
| 5 | **[a-z]**  Match any lowercase ASCII letter |
| 6 | **[A-Z]**  Match any uppercase ASCII letter |
| 7 | **[a-zA-Z0-9]**  Match any of the above |
| 8 | **[^aeiou]**  Match anything other than a lowercase vowel |
| 9 | **[^0-9]**  Match anything other than a digit |

## 

## Special Character Classes

|  |  |
| --- | --- |
| **Sr.No.** | **Example & Description** |
| 1 | **.**  Match any character except newline |
| 2 | **\d**  Match a digit: [0-9] |
| 3 | **\D**  Match a nondigit: [^0-9] |
| 4 | **\s**  Match a whitespace character: [ \t\r\n\f] |
| 5 | **\S**  Match nonwhitespace: [^ \t\r\n\f] |
| 6 | **\w**  Match a single word character: [A-Za-z0-9\_] |
| 7 | **\W**  Match a nonword character: [^A-Za-z0-9\_] |

## Repetition Cases

|  |  |
| --- | --- |
| **Sr.No.** | **Example & Description** |
| 1 | **ruby?**  Match "rub" or "ruby": the y is optional |
| 2 | **ruby\***  Match "rub" plus 0 or more ys |
| 3 | **ruby+**  Match "rub" plus 1 or more ys |
| 4 | **\d{3}**  Match exactly 3 digits |
| 5 | **\d{3,}**  Match 3 or more digits |
| 6 | **\d{3,5}**  Match 3, 4, or 5 digits |

## Nongreedy repetition

This matches the smallest number of repetitions −

|  |  |
| --- | --- |
| **Sr.No.** | **Example & Description** |
| 1 | **<.\*>**  Greedy repetition: matches "<python>perl>" |
| 2 | **<.\*?>**  Nongreedy: matches "<python>" in "<python>perl>" |

## Grouping with Parentheses

|  |  |
| --- | --- |
| **Sr.No.** | **Example & Description** |
| 1 | **\D\d+**  No group: + repeats \d |
| 2 | **(\D\d)+**  Grouped: + repeats \D\d pair |
| 3 | **([Pp]ython(, )?)+**  Match "Python", "Python, python, python", etc. |

## Backreferences

This matches a previously matched group again −

|  |  |
| --- | --- |
| **Sr.No.** | **Example & Description** |
| 1 | **([Pp])ython&\1ails**  Match python&pails or Python&Pails |
| 2 | **(['"])[^\1]\*\1**  Single or double-quoted string. \1 matches whatever the 1st group matched. \2 matches whatever the 2nd group matched, etc. |

## Alternatives

|  |  |
| --- | --- |
| **Sr.No.** | **Example & Description** |
| 1 | **python|perl**  Match "python" or "perl" |
| 2 | **rub(y|le))**  Match "ruby" or "ruble" |
| 3 | **Python(!+|\?)**  "Python" followed by one or more ! or one ? |

## Anchors

This needs to specify match position.

|  |  |
| --- | --- |
| **Sr.No.** | **Example & Description** |
| 1 | **^Python**  Match "Python" at the start of a string or internal line |
| 2 | **Python$**  Match "Python" at the end of a string or line |
| 3 | **\APython**  Match "Python" at the start of a string |
| 4 | **Python\Z**  Match "Python" at the end of a string |
| 5 | **\bPython\b**  Match "Python" at a word boundary |
| 6 | **\brub\B**  \B is nonword boundary: match "rub" in "rube" and "ruby" but not alone |
| 7 | **Python(?=!)**  Match "Python", if followed by an exclamation point. |
| 8 | **Python(?!!)**  Match "Python", if not followed by an exclamation point. |

## Special Syntax with Parentheses

|  |  |
| --- | --- |
| **Sr.No.** | **Example & Description** |
| 1 | **R(?#comment)**  Matches "R". All the rest is a comment |
| 2 | **R(?i)uby**  Case-insensitive while matching "uby" |
| 3 | **R(?i:uby)**  Same as above |
| 4 | **rub(?:y|le))**  Group only without creating \1 backreference |

# មេរៀនទី​ ៩: Python – GUI Programming



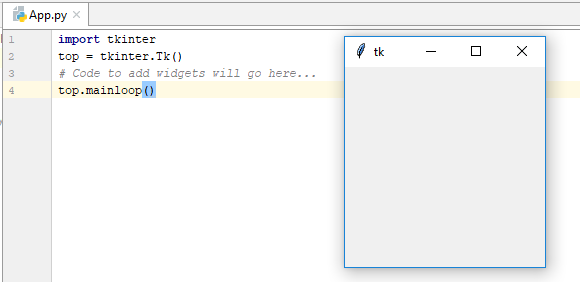
## សេចក្តីណែនាំ Introduction to GUI

Python បានផ្តល់នូវជម្រើសជាច្រើនក្នុងការ developing កម្មវិធីជាមួយនឹង graphical user interface។ GUIs ដែលគេនិយមប្រើមានដូចខាងក្រោម៖

* **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
* **PyQt --** Mainly, you can develop desktop applications with PyQT. PyQt is a binding for QT framework. Qt is written in C++, So with PyQt you can use all QT classes and methods with Python languages without knowing C++. It is aslo open source.
* **wxPython** − This is an open-source Python interface for wxWindows <http://wxpython.org>.
* **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine <http://www.jython.org>.

## GUI-Tkinter Programming

TKinter គឺជា standard GUI library ដែលមានមកស្រាប់ជាមួយភាសា Python ដែលអាចបង្កើតជា GUI application បានលឿននិងងាយស្រួល។ ដើម្បីប្រើប្រាស់ TKinter ចូរធ្វើដូចខាងក្រោម៖

* Import the Tkinter module.
* Create the GUI application main window
* Add one or more of the above-mentioned widgets to the GUI application
* Enter the main event loop to take action against each event triggered by the user.

import tkinter

top = tkinter.Tk()

# Code to add widgets will go here...

top.mainloop()

**Tkinter Widgets**

Widgets វាprovideឲ្យយើងនូវ control ជាច្រើន(buttons, labels and text boxes...)សម្រាប់បង្កើតជា GUI application។ Widgets រួមមានcontrolដូចខាងក្រោម៖

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| 1 | [**Button**](https://www.tutorialspoint.com/python/tk_button.htm)  The Button widget is used to display buttons in your application. |
| 2 | [**Canvas**](https://www.tutorialspoint.com/python/tk_canvas.htm)  The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application. |
| 3 | [**Checkbutton**](https://www.tutorialspoint.com/python/tk_checkbutton.htm)  The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time. |
| 4 | [**Entry**](https://www.tutorialspoint.com/python/tk_entry.htm)  The Entry widget is used to display a single-line text field for accepting values from a user. |
| 5 | [**Frame**](https://www.tutorialspoint.com/python/tk_frame.htm)  The Frame widget is used as a container widget to organize other widgets. |
| 6 | [**Label**](https://www.tutorialspoint.com/python/tk_label.htm)  The Label widget is used to provide a single-line caption for other widgets. It can also contain images. |
| 7 | [**Listbox**](https://www.tutorialspoint.com/python/tk_listbox.htm)  The Listbox widget is used to provide a list of options to a user. |
| 8 | [**Menubutton**](https://www.tutorialspoint.com/python/tk_menubutton.htm)  The Menubutton widget is used to display menus in your application. |
| 9 | [**Menu**](https://www.tutorialspoint.com/python/tk_menu.htm)  The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton. |
| 10 | [**Message**](https://www.tutorialspoint.com/python/tk_message.htm)  The Message widget is used to display multiline text fields for accepting values from a user. |
| 11 | [**Radiobutton**](https://www.tutorialspoint.com/python/tk_radiobutton.htm)  The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time. |
| 12 | [**Scale**](https://www.tutorialspoint.com/python/tk_scale.htm)  The Scale widget is used to provide a slider widget. |
| 13 | [**Scrollbar**](https://www.tutorialspoint.com/python/tk_scrollbar.htm)  The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes. |
| 14 | [**Text**](https://www.tutorialspoint.com/python/tk_text.htm)  The Text widget is used to display text in multiple lines. |
| 15 | [**Toplevel**](https://www.tutorialspoint.com/python/tk_toplevel.htm)  The Toplevel widget is used to provide a separate window container. |
| 16 | [**Spinbox**](https://www.tutorialspoint.com/python/tk_spinbox.htm)  The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values. |
| 17 | [**PanedWindow**](https://www.tutorialspoint.com/python/tk_panedwindow.htm)  A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically. |
| 18 | [**LabelFrame**](https://www.tutorialspoint.com/python/tk_labelframe.htm)  A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts. |
| 19 | [**tkMessageBox**](https://www.tutorialspoint.com/python/tk_messagebox.htm)  This module is used to display message boxes in your applications. |

ខាងក្រោមនេះគឺជា attribute ដែលគេតែងតែឧស្សាហ៍ប្រើប្រាស់៖

* Dimensions: ជាខ្នាតប្រវែងរបស់widget(width,height,borderwidht,padX,padY, highlightthickness, selectborderwidth, wraplength, underline) ដែរជាទូទៅវាត្រូវបានគេគិតជាខ្នាត Pixel។ តែយើងក៏អាចកំណត់ជា Centimeters, inches, Millimetersបានផងដែរ។
* Colors: យើងអាចប្តូរcolor បានពីរបៀបគឺ hexadecimal digits("#fff", "#000fff000") ឬ Color Name("white", "black", "red", "green", "blue", "cyan", "yellow", and "magenta")។ ដែរជាទូទៅគេប្រើប្រាស់វាលើ(activebackground, activeforeground, background, disabledforeground, foreground, highlightbackground, highlightcolor, selectbackground, selectforeground)
* Fonts: សម្រាប់ប្តូរពុម្ភអក្សរ(family{font name}, size{int pixcel}, weight{“bold” ,”normal”}, slant{ "italic" , "roman" }, underline{0,1}, overstrike{0,1})
* Anchors: define where text is positioned relative to a reference point(CENTER, NW,N,NE,W,E,SW,S,SE)
* Relief styles: ជាstyle របស់widget(FLAT, RAISED, SUNEN,GROOVE,RIDGE)
* Bitmaps: សម្រាប់រូបភាពbitmapមានស្រាប់(error, wearning,…)
* Cursors: to change mouse course on control

ខាងក្រោមនេះគឺជា Method ដែលគេតែងតែឧស្សាហ៍ប្រើប្រាស់(តាង W ជា widget ផ្សេងៗ)៖

* **W.get() :** សម្រាប់ return នូវ keyword/option ទាំងឡាយដែលអាចប្រើប្រាស់បានជាមួយនឹង widget។
* **W.cget(“option”) = W["option"]:** Return the current value of an option. Both the option name, and the returned value, are strings
* **W.configure(option=value, …)=** W.**config(option=value, …) = W["option"] :** Set one or more options (given as keyword arguments)។ Ex: W.configure(bg=’red’, fg=’green’)។
* **W.bind(‘<event\_name>’,functionName) :** សម្រាប់ bind event ឲ្យ widget ជាមួយនឹង function។ ដែល event\_name រួមមាន៖ <Button-1>, <Double-1>, <Motion>…

**Events Name: use with bind() method**

|  |  |
| --- | --- |
| **Event** | **Description** |
| <Button> | A mouse button is pressed with the mouse pointer over the widget. The detail part specifies which button, e.g. The left mouse button is defined by the event <Button-1>, the middle button by <Button-2>, and the rightmost mouse button by <Button-3>. <Button-4> defines the scroll up event on mice with wheel support and and <Button-5> the scroll down. If you press down a mouse button over a widget and keep it pressed, Tkinter will automatically "grab" the mouse pointer. Further mouse events like Motion and Release events will be sent to the current widget, even if the mouse is moved outside the current widget. The current position, relative to the widget, of the mouse pointer is provided in the x and y members of the event object passed to the callback. You can use ButtonPress instead of Button, or even leave it out completely: , , and <1> are all synonyms. |
| <Motion> | The mouse is moved with a mouse button being held down. To specify the left, middle or right mouse button use <B1-Motion>, <B2-Motion> and <B3-Motion> respectively. The current position of the mouse pointer is provided in the x and y members of the event object passed to the callback, i.e. event.x, event.y |
| <ButtonRelease> | Event, if a button is released. To specify the left, middle or right mouse button use <ButtonRelease-1>, <ButtonRelease-2>, and <ButtonRelease-3> respectively. The current position of the mouse pointer is provided in the x and y members of the event object passed to the callback, i.e. event.x, event.y |
| <Double-Button> | Similar to the Button event, see above, but the button is double clicked instead of a single click. To specify the left, middle or right mouse button use <Double-Button-1>, <Double-Button-2>, and <Double-Button-3> respectively. You can use Double or Triple as prefixes. Note that if you bind to both a single click (<Button-1>) and a double click (<Double-Button-1>), both bindings will be called. |
| <Enter> | The mouse pointer entered the widget. Attention: This doesn't mean that the user pressed the Enter key!. <Return> is used for this purpose. |
| <Leave> | The mouse pointer left the widget. |
| <FocusIn> | Keyboard focus was moved to this widget, or to a child of this widget. |
| <FocusOut> | Keyboard focus was moved from this widget to another widget. |
| <Return> | The user pressed the Enter key. You can bind to virtually all keys on the keyboard: The special keys are Cancel (the Break key), BackSpace, Tab, Return(the Enter key), Shift\_L (any Shift key), Control\_L (any Control key), Alt\_L (any Alt key), Pause, Caps\_Lock, Escape, Prior (Page Up), Next (Page Down), End, Home, Left, Up, Right, Down, Print, Insert, Delete, F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, Num\_Lock, and Scroll\_Lock. |
| <Key> | The user pressed any key. The key is provided in the char member of the event object passed to the callback (this is an empty string for special keys). |
| a | The user typed an "a" key. Most printable characters can be used as is. The exceptions are space (<space>) and less than (<less>). Note that 1 is a keyboard binding, while <1> is a button binding. |
| <Shift-Up> | The user pressed the Up arrow, while holding the Shift key pressed. You can use prefixes like Alt, Shift, and Control. |
| <Configure> | The size of the widget changed. The new size is provided in the width and height attributes of the event object passed to the callback. On some platforms, it can mean that the location changed. |

**មាន២វិធីក្នុងការបង្កើត event ឲ្យ Widget**

**ឧទាហរណ៍ទី១៖ ប្រើ bind() method**

from tkinter import \*

def hello(event,name):

print("Single Click, Button-l, welcome ", name) ;

def quit(event):

print("Double Click, so let's stop")

import sys; sys.exit()

widget = Button(None, text='Mouse Clicks')

widget.pack()

widget.bind('<Button-1>', hello(None, "Makara"))

widget.bind('<Double-1>', quit)

widget.mainloop()

**ឧទាហរណ៍ទី២៖ ប្រើ command attribute**

from tkinter import \*

root=Tk();

def hello():

print("Single Click with no Parrameter")

w1 = Button(root, text='Mouse Clicks',command=hello)

w1.pack()

def hello2(par):

print("Single Click with Parrameter=" + str(par))

w2 = Button(root, text='Mouse Clicks',command=**lambda :hello2(1)**)

w2.pack()

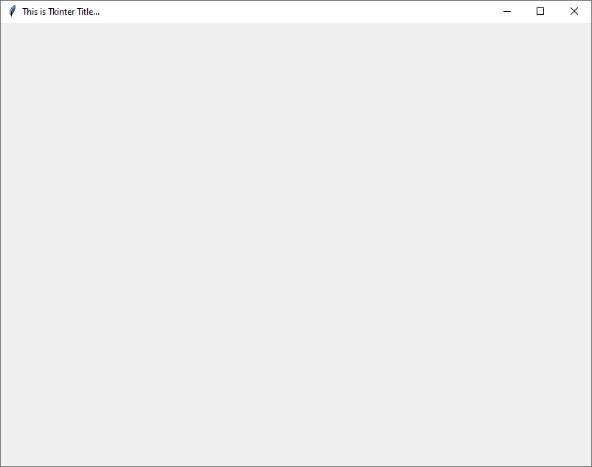
root.mainloop()

**Geometry Manager:** ជាពពួកmethod សម្រាប់រៀបចំទៅតាំងរបស់controlក្នុងparent widget។ រួមមាន៖

* **pack()**: This geometry manager organizes widgets in blocks before placing them in the parent widget.
* **grid()**: This geometry manager organizes widgets in a table-like structure in the parent widget.
* **place()**: This geometry manager organizes widgets by placing them in a specific position in the parent widget.

**ចំណាំ៖** pack() និង grid() គឺមិនអាចប្រើប្រាស់រួមគ្នាក្នុង master window តែមួយបានទេ។

**របៀបធ្វើឲ្យ Window Form នៅ Center Screen**:

from tkinter import \*;

root =Tk();

root.title("This is Tkinter Title...");

root.iconbitmap("myico.ico")

#===============make form center screen====================

form\_width=800; form\_high=600;

x\_cor=root.winfo\_screenwidth()/2-form\_width/2;

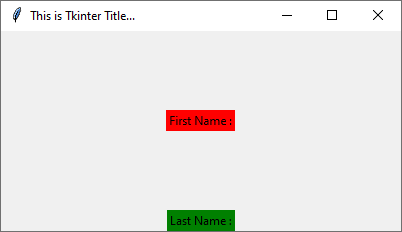
y\_cor=root.winfo\_screenheight()/2-form\_high/2;

root.geometry("%dx%d+%d+%d"%(form\_width,form\_high,x\_cor,y\_cor));

root.mainloop();

**The options of Pack() method**:

* **expand** : When set to **True**, widget expands to fill any space not otherwise used in widget's parent.
* **fill** : Determines whether widget fills any extra space allocated to it by the packer, or keeps **its** own minimal dimensions: NONE (default), X (fill only horizontally), Y (fill only vertically), or BOTH (fill both horizontally and vertically).
* **side** : Determines which side of the parent widget packs against: TOP (default), BOTTOM, LEFT, or RIGHT.
* **Padding :** pady(External padding, vertically), padx=( External padding, horizontally), ipadx(Internal padding, horizontally), ipady(Internal padding, vertically). The default of padding is 0(zero).

ឧទាហរណ៍១៖ using **expand** option

from tkinter import \*;

root =Tk();

root.title("This is Tkinter Title...");

#design control====================

lbl1=Label(root,text='First Name :',bg="red")

lbl1.pack(expand=True);

lbl2=Label(root,text='Last Name :',bg="green")

lbl2.pack();

#===============make form center screen====================

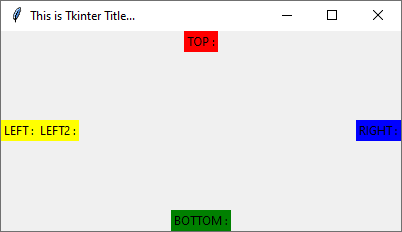
form\_width=400; form\_high=200;

x\_cor=root.winfo\_screenwidth()/2-form\_width/2;

y\_cor=root.winfo\_screenheight()/2-form\_high/2;

root.geometry("%dx%d+%d+%d"%(form\_width,form\_high,x\_cor,y\_cor));

root.mainloop();

ឧទាហរណ៍២៖ using **side** option

import tkinter as tk;

root =tk.Tk();

root.title("This is Tkinter Title...");

#design control====================

lbl1=tk.Label(root,text='TOP :',bg="red")

lbl1.pack(side=tk.TOP);

lbl2=tk.Label(root,text='BOTTOM :',bg="green")

lbl2.pack(side=tk.BOTTOM);

lbl3=tk.Label(root,text='LEFT :',bg="yellow")

lbl3.pack(side=tk.LEFT);

lbl4=tk.Label(root,text='RIGHT :',bg="blue")

lbl4.pack(side=tk.RIGHT);

lbl5=tk.Label(root,text='LEFT2 :',bg="yellow")

lbl5.pack(side=tk.LEFT);

#===============make form center screen====================

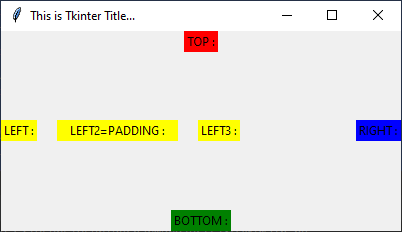
form\_width=400; form\_high=200;

x\_cor=root.winfo\_screenwidth()/2-form\_width/2;

y\_cor=root.winfo\_screenheight()/2-form\_high/2;

root.geometry("%dx%d+%d+%d"%(form\_width,form\_high,x\_cor,y\_cor));

root.mainloop();

ឧទាហរណ៍៣៖ using **padding** option

import tkinter as tk;

root =tk.Tk();

root.title("This is Tkinter Title...");

#design control====================

lbl1=tk.Label(root,text='TOP :',bg="red")

lbl1.pack(side=tk.TOP);

lbl2=tk.Label(root,text='BOTTOM :',bg="green")

lbl2.pack(side=tk.BOTTOM);

lbl3=tk.Label(root,text='LEFT :',bg="yellow")

lbl3.pack(side=tk.LEFT);

lbl4=tk.Label(root,text='RIGHT :',bg="blue")

lbl4.pack(side=tk.RIGHT);

lbl5=tk.Label(root,text='LEFT2=PADDING :',bg="yellow")

**lbl5.pack(side=tk.LEFT,padx=20,ipadx=10);**

lbl5=tk.Label(root,text='LEFT3 :',bg="yellow")

lbl5.pack(side=tk.LEFT);

#===============make form center screen====================

form\_width=400; form\_high=200;

x\_cor=root.winfo\_screenwidth()/2-form\_width/2;

y\_cor=root.winfo\_screenheight()/2-form\_high/2;

root.geometry("%dx%d+%d+%d"%(form\_width,form\_high,x\_cor,y\_cor));

root.mainloop();

**The options of Grid() method**: តំរៀប form ក្នុងទម្រង់ជា table(row, column)

* **column** : The column to put widget in; default 0 (leftmost column).
* **columnspan** : How many columns widget occupies; default 1.
* **ipadx, ipady** : How many pixels to pad widget, horizontally and vertically, inside widget's borders.
* **padx, pady** : How many pixels to pad widget, horizontally and vertically, outside v's borders.
* **row** : The row to put widget in; default the first row that is still empty.
* **rowspan** : How many rowswidget occupies; default 1.
* **sticky** : What to do if the cell is larger than widget. By default, with sticky='', widget is centered in its cell. sticky may be the string concatenation of zero or more of N, E, S, W, NE, NW, SE, and SW, compass directions indicating the sides and corners of the cell to which widget sticks.

ឧទាហរណ៍៖ ការតំរៀប widget ដោយប្រើ rol, column and columnspan

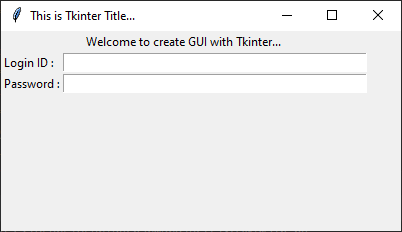
import tkinter as tk;

root =tk.Tk();

root.title("This is Tkinter Title...");

#design control====================

lbl1=tk.Label(root,text='Welcome to create GUI with Tkinter...')



lbl1.grid(columnspan=2,sticky='');

lbl2=tk.Label(root,text='Login ID :')

lbl2.grid(row=1,column=0,sticky='W');

txt1=tk.Entry(root,width=50)

txt1.grid(row=1,column=1);

lbl3=tk.Label(root,text='Password :')

lbl3.grid(row=2,column=0,sticky='W');

txt2=tk.Entry(root,width=50)

txt2.grid(row=2,column=1);

#===============make form center screen====================

form\_width=400; form\_high=200;

x\_cor=root.winfo\_screenwidth()/2-form\_width/2;

y\_cor=root.winfo\_screenheight()/2-form\_high/2;

root.geometry("%dx%d+%d+%d"%(form\_width,form\_high,x\_cor,y\_cor));

root.mainloop();

**The options of Place() method**: តំរៀប control(widget) លើទីតាំងច្បាស់លាស់

* **anchor** : The exact spot of widget other options refer to: may be N, E, S, W, NE, NW, SE, or SW, compass directions indicating the corners and sides of widget; default is NW (the upper left corner of widget)
* **bordermode** : INSIDE (the default) to indicate that other options refer to the parent's inside (ignoring the parent's border); OUTSIDE otherwise.
* **height, width** : Height and width in pixels.
* **relheight, relwidth** : Height and width as a float between 0.0 and 1.0, as a fraction of the height and width of the parent widget.
* **relx, rely** : Horizontal and vertical offset as a float between 0.0 and 1.0, as a fraction of the height and width of the parent widget.
* **x, y** : Horizontal and vertical offset in pixels.

ឧទាហរណ៍៖ ការតំរៀបទីតាំងដោយប្រើ width, hight, x , y

import tkinter as tk;

root =tk.Tk();

root.title("This is Tkinter Title...");

#===============make form center screen====================

form\_width=400; form\_high=200;

x\_cor=root.winfo\_screenwidth()/2-form\_width/2;

y\_cor=root.winfo\_screenheight()/2-form\_high/2;

root.geometry("%dx%d+%d+%d"%(form\_width,form\_high,x\_cor,y\_cor));

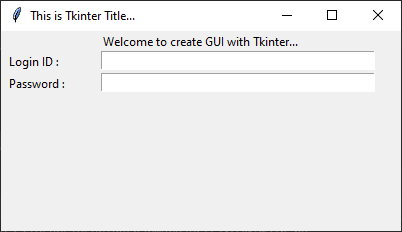
root.maxsize(form\_width,form\_high)

root.minsize(form\_width,form\_high)

#design control====================

lbl1=tk.Label(root,text='Welcome to create GUI with Tkinter...')

lbl1.pack();



lbl2=tk.Label(root,text='Login ID :')

lbl2.place(x=5,y=20);

txt1=tk.Entry(root,width=45)

txt1.place(x=100,y=20);

lbl3=tk.Label(root,text='Password :')

lbl3.place(x=5,y=42);

txt2=tk.Entry(root,width=45)

txt2.place(x=100,y=42);

#==========show form===============

root.mainloop();

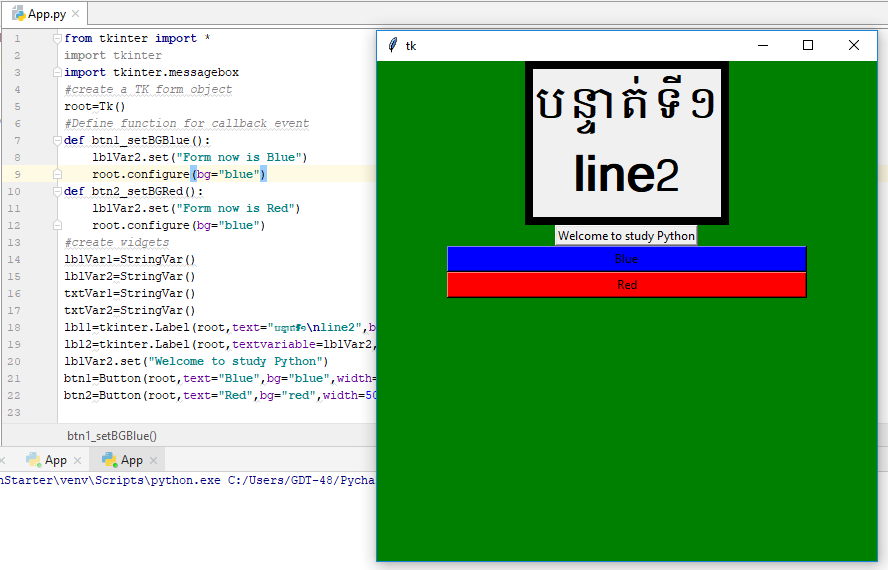
**របៀបប្រើប្រាស់ Label:**

Syntax : w = Label ( master, option, ... )

Attribute សំខាន់ៗ:

* text : set default text to to label, cannot use with textvariable
* textvariable: ប្រភេទជា StringVar ដែលបម្រើឲ្យការ get,set លើlabel
* font: ប្តូរពុម្ភអក្សរ
* bd = borderwidth
* relief : style of label

from tkinter import \*

import tkinter

import tkinter.messagebox

#create a TK form object

root=Tk()

#Define function for callback event

def btn1\_setBGBlue():

lblVar2.set("Form now is Blue")

root.configure(bg="blue")

def btn2\_setBGRed():

lblVar2.set("Form now is Red")

root.configure(bg="blue")

#create widgets

lblVar1=StringVar()

lblVar2=StringVar()

txtVar1=StringVar()

txtVar2=StringVar()

lbl1=tkinter.Label(root,text="បន្ទាត់ទី១\nline2",borderwidth=8,relief="solid",font="Khmer 32 bold").pack()

lbl2=tkinter.Label(root,textvariable=lblVar2,relief=RAISED).pack()

lblVar2.set("Welcome to study Python")

btn1=Button(root,text="Blue",bg="blue",width=50,command=btn1\_setBGBlue).pack()

btn2=Button(root,text="Red",bg="red",width=50,command=btn2\_setBGRed).pack()

#Before Open Form Event

root.title="GUI-TKinter Programming-Checkutton"

root.geometry("500x500")

root.configure(background="green")

#Open Form

root.mainloop()

**របៀបប្រើប្រាស់ Entry**: accept single-line text strings from a user

Syntax : w = Entry( master, option, ... )

Attribute សំខាន់ៗ៖

* command: callback event when
* exportselection: By default, if you select text within an Entry widget, it is automatically exported to the clipboard. To avoid this exportation, use exportselection=0.
* highlightcolor: The color of the focus highlight when the checkbutton has the focus.
* selectbackground:
* selectborderwidth:
* selectforeground:
* show: for encrypt the text, Ex: show=”\*”
* textvariable : use to control set, get the text from this widgets(StringVal)
* xscrollcommand:
* delete ( first, last=None ): index start from 1
* get() : get all text
* index ( index ): Shift the contents of the entry so that the character at the given index is the leftmost visible character. Has no effect if the text fits entirely within the entry.
* insert ( index, s ): Inserts string s before the character at the given index.
* select\_adjust ( index )
* icursor ( index )
* select\_clear() : clear the selected text
* select\_from ( index )
* select\_to ( index )
* select\_present():If there is a selection, returns true, else returns false.
* select\_range ( start, end )
* xview ( index )
* xview\_scroll ( number, what )

from tkinter import \*

import tkinter

#create main window

root=Tk()

#function to callback event

def getText():

L2.config(text="You have entry text : " + txt.get())

#design widgets

txt=StringVar()

L1 = Label(root, text="User Name")

L1.pack( side = LEFT)

E1 = Entry(root, bd =5,textvariable=txt)

E1.pack(side = RIGHT)

B1=Button(root,text="Click me",command=getText)

B1.pack(side=BOTTOM)

L2=Label(root)

L2.pack()

#do sth before load window form

#open window form

root.mainloop()

**របៀបប្រើប្រាស់ Text**: for multi-line text entry

Syntax: w = Text ( master, option, ... )

Attribute សំខាន់ៗរួមមាន៖

* exportselection
* font
* height : សំដៅលើចំនួន line
* highlightbackground
* highlightcolor
* insertbackground: The color of the insertion cursor. Default is black.
* insertborderwidth: Size of the 3-D border around the insertion cursor. Default is 0.
* insertofftime
* insertontime
* insertwidth
* selectbackground
* selectborderwidth
* spacing1: increase verticle line space….., default is 0
* spacing2: incease verticle line space…., default is 0
* spacing3: incease verticle line space…., default is 0
* tabs: កំណត់ចំនួន tab character
* width: គិតជាចំនួនតួអក្សរ
* wrap: បង្កើត wrapទិន្នន័យ(wrap=WORD, default is wrap=CHAR)
* xscrollcommand: To make the text widget horizontally scrollable, set this option to the set() method of the horizontal scrollbar.
* yscrollcommand: To make the text widget vertically scrollable, set this option to the set() method of the vertical scrollbar.
* delete(startindex [,endindex])
* get(startindex [,endindex])
* index(index): Returns the absolute value of an index based on the given index.
* insert(index [,string]...):
* see(index): This method returns true if the text located at the index position is visible.

Text widgets support three distinct helper structures: Marks, Tabs, and Indexes −

Marks are used to bookmark positions between two characters within a given text. We have the following methods available when handling marks

* index(mark): Returns the line and column location of a specific mark.
* mark\_gravity(mark [,gravity]):
* mark\_names():Returns all marks from the Text widget.
* mark\_set(mark, index): Informs a new position to the given mark.
* mark\_unset(mark): Removes the given mark from the Text widget.

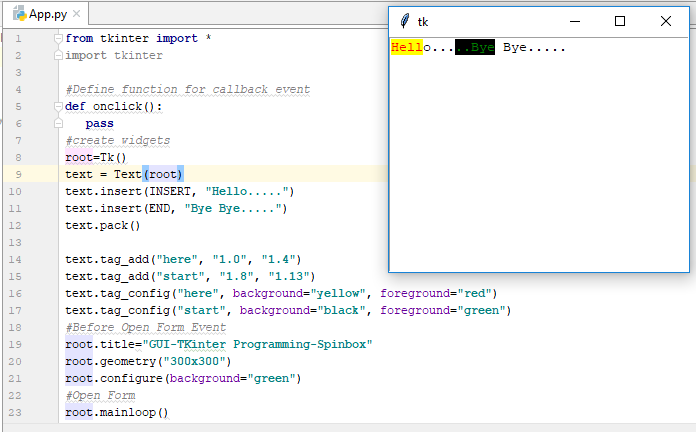
Tags are used to associate names to regions of text which makes easy the task of modifying the display settings of specific text areas.

* tag\_add(tagname, startindex[,endindex] ...)
* tag\_config
* tag\_delete(tagname)
* tag\_remove(tagname [,startindex[.endindex]] ...)

from tkinter import \*

import tkinter

#Define function for callback event

def onclick():

pass

#create widgets

root=Tk()

text = Text(root)

text.insert(INSERT, "Hello.....")

text.insert(END, "Bye Bye.....")

text.pack()

text.tag\_add("here", "1.0", "1.4")

text.tag\_add("start", "1.8", "1.13")

text.tag\_config("here", background="yellow", foreground="red")

text.tag\_config("start", background="black", foreground="green")

#Before Open Form Event

root.title("GUI-TKinter Programming-Spinbox")

root.geometry("300x300")

root.configure(background="green")

#Open Form

root.mainloop()

**របៀបប្រើប្រាស់ Button**:

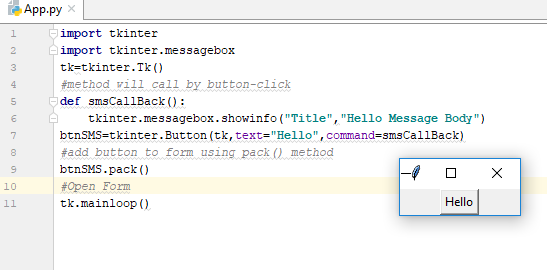
Syntax : w = Button ( master, option=value, ... )

**master** − This represents the parent window.

**options** − Here is the list of most commonly used options for this widget

Option and Method សំខាន់៖

* command :ជាoption សម្រាប់call method or function ពេល​​user clicked
* flash() : delay time between active and normal colors
* invoke(): បញ្ជាឲ្យ click

import tkinter

import tkinter.messagebox

tk=tkinter.Tk()

#method will call by button-click

def smsCallBack():

tkinter.messagebox.showinfo("Title","Hello Message Body")

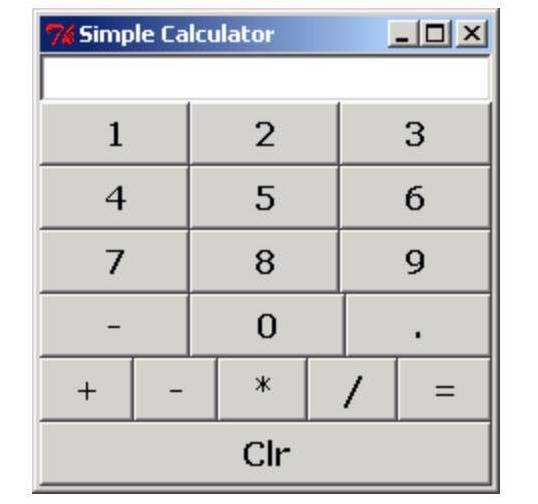
btnSMS=tkinter.Button(tk,text="Hello",command=smsCallBack)

#add button to form using pack() method

btnSMS.pack()

#Open Form

tk.mainloop()

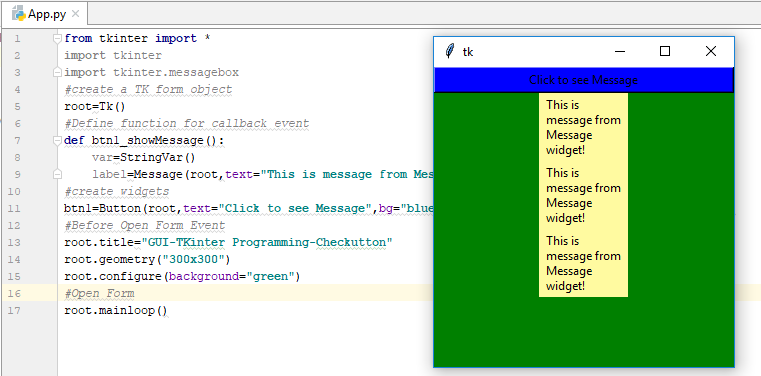
**លំហាត់**៖

**របៀបប្រើប្រាស់ Message:**

Syntax : w = Message ( master, option, ... )

Attribute សំខាន់ៗ៖

* text
* textvariable: for control on widget and return with type of StringVar
* bg
* fg
* image
* relief…..

from tkinter import \*

import tkinter

import tkinter.messagebox

#create a TK form object

root=Tk()

#Define function for callback event

def btn1\_showMessage():

var=StringVar()

label=Message(root,text="This is message from Message widget!",bg="#fffaa0").pack()

#create widgets

btn1=Button(root,text="Click to see Message",bg="blue",width=50,command=btn1\_showMessage).pack()

#Before Open Form Event

root.title="GUI-TKinter Programming-Checkutton"

root.geometry("300x300")

root.configure(background="green")

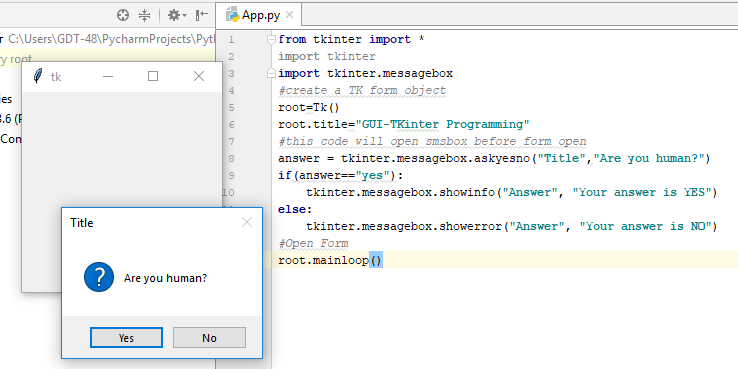
#Open Form

root.mainloop()

**របៀបប្រើប្រាស់ MessageBox:**

Syntax : messagebox.methodName (title, message [, options])

Method សំខាន់ៗសម្រាប់បង្ហាញmessage:

* showinfo()
* showwarning()
* showerror ()
* askquestion()
* askokcancel()
* askyesno ()
* askretrycancel ()

from tkinter import \*

import tkinter

import tkinter.messagebox

#create a TK form object

root=Tk()

root.title("GUI-TKinter Programming")

#this code will open smsbox before form open

answer = tkinter.messagebox.askquestion("Title","Are you human?")

if(answer=="yes"):

tkinter.messagebox.showinfo("Answer", "Your answer is YES")

else:

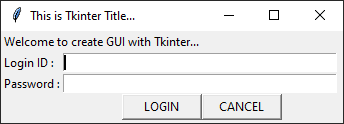
tkinter.messagebox.showerror("Answer", "Your answer is NO")

#Open Form

root.mainloop()

**លំហាត់៖**

ចូរបង្កើត UI ខាងក្រោម៖



* ពេលចុច [LOGIN] 🡪 show message (You click on LOGIN!)
* ពេលចុច [CANCEL] 🡪 show message (Pls click 'YES' or 'NO'!),
  + ពេលចុច button [YES]🡪 show message (You clicked on YES)
  + ពេលចុច button [NO]🡪 show message (You clicked on NO)

**ចម្លើយ**៖

១**) Login\_frm.py**

class Longin\_frm:

import tkinter as tk;

import tkinter.messagebox as tksms;

root = tk.Tk();

#=============create EVENT===============

def btnLogin\_click(self):

self.tksms.showinfo("LOGIN", "You click on LOGIN!")

def btnCancel\_click(self):

result=self.tksms.askquestion("LOGIN", "Pls click 'YES' or 'NO'!")

if(result=='yes'):

self.tksms.showinfo("EXIT", "System will exit!")

self.root.quit();

else:

self.tksms.showinfo("BACK", "System will back!")

#=============end EVENT===================

def \_\_init\_\_(self):

self.root.title("This is Tkinter Title...");

# ===============make form center screen====================

form\_width = 350;

form\_high = 100;

x\_cor = self.root.winfo\_screenwidth() / 2 - form\_width / 2;

y\_cor = self.root.winfo\_screenheight() / 2 - form\_high / 2;

self.root.geometry("%dx%d+%d+%d" % (form\_width, form\_high, x\_cor, y\_cor));

self.root.maxsize(form\_width, form\_high)

self.root.minsize(form\_width, form\_high)

# design control interface UI====================

lblTitle = self.tk.Label(self.root, text='Welcome to create GUI with Tkinter...')

lblTitle.grid(columnspan=3);

lblLoginId = self.tk.Label(self.root, text='Login ID :')

lblLoginId.grid(row=1, column=0, sticky='W');

txtLoginId = self.tk.Entry(self.root, width=45)

txtLoginId.grid(row=1, column=2, sticky='W', columnspan=2);

lblPwd = self.tk.Label(self.root, text='Password :')

lblPwd.grid(row=2, column=0, sticky='W');

txtPwd = self.tk.Entry(self.root, width=45 ,show='\*')

txtPwd.grid(row=2, column=2, sticky='W', columnspan=2);

# buton login + cancel

btnLogin = self.tk.Button(self.root, text="LOGIN", width=10, command=self.btnLogin\_click)

btnLogin.grid(row=3, column=2, sticky='E');

btnCancel = self.tk.Button(self.root, text="CANCEL", width=10, command=self.btnCancel\_click)

btnCancel.grid(row=3, column=3, sticky='W');

# set focus

txtLoginId.focus();

# ==========end design of interface UI=================

def show(self):

#==========show form===============

self.root.mainloop();

២) **main.py**

from Login\_frm import Longin\_frm

login=Longin\_frm();

login.show();

**របៀបប្រើប្រាស់ CheckButton:** checkbutton គេប្រើសម្រាប់ multiple-choice,

Syntax: Checkbutton ( master, option, ... )

Method និង Attribute សំខាន់ៗ:

* deselect() : clear selection(unchecked) on widget
* flash() : Redraws the button several times, alternating between active and normal appearance.
* invoke()​ : Calls the command associated with the button
* select() : set selection(checked) on widget
* toggle() : Clears the checkbutton if set, sets it if cleared.
* command : bind to method name on event check state changed
* cursor : change cursor
* font : change font
* text: set text to box
* variable: state of the checkbutton, 0 means cleared and 1 means set(link to onvlaue and offvalue) type របស់វាគឺ IntVar
* textvariable : Associates a Tkinter variable (usually a StringVar) with the button. If the variable is changed, the button text is updated. Also see the variable option. (textVariable/Variable)
* onvalue: សម្រាប់កំណត់តំលៃពេលcheck(default set value 1 when box is on)
* offvalue: សម្រាប់កណត់តលៃពេលun-check (default set value 0 when box is off)
* state : default is state=NORMAL, state=DISABLED for unresponsive

from tkinter import \*

import tkinter

import tkinter.messagebox

#create a TK form object

root=Tk()

root.title("GUI-TKinter Programming-Checkutton")

#Define function for callback event

def chk1\_checkChange():

tkinter.messagebox.showinfo("Checkbutton","Your check state on Chk1 is " + str(chkVar1.get()))

def chk2\_checkChange():

tkinter.messagebox.showinfo("Checkbutton","Your check state on Chk2 is " + str(chkVar2.get()))

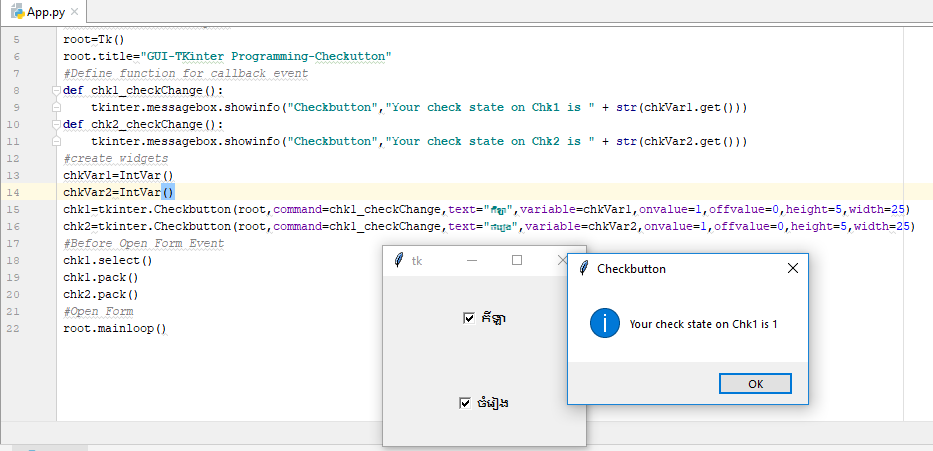
#create widgets

chkVar1=IntVar()

chkVar2=IntVar()

chk1=tkinter.Checkbutton(root,command=chk1\_checkChange,text="កីឡា",variable=chkVar1,onvalue=1,offvalue=0,height=5,width=25)

chk2=tkinter.Checkbutton(root,command=chk1\_checkChange,text="ចំរៀង",variable=chkVar2,onvalue=1,offvalue=0,height=5,width=25)

#Before Open Form Event

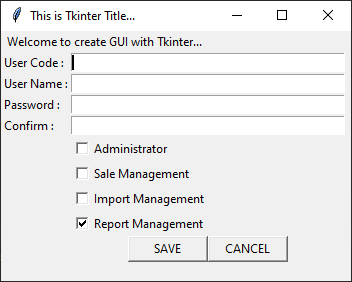
chk1.select()

chk1.pack()

chk2.pack()

#Open Form

root.mainloop()

លំហាត់៖

* ចូរ Message បង្ហាញនូវ(value both Text and CheckedSate) ពេលuser checked change លើ Checkbutton ខាងលើ

ចម្លើយ៖

១) **main.py**

from User\_frm import \*;

user\_frm=User\_frm();

user\_frm.show()

២) **User\_frm.py**

from tkinter import \*;

class User\_frm:

import tkinter as tk;

import tkinter.messagebox as tksms;

root = tk.Tk();

#=============create EVENT===============

def chkAdmin\_Checked(self):

self.tksms.showinfo("ADMINISTRATOR", "The checked value =" + str(self.chkAdminVar.get()))

def chkSale\_Checked(self):

self.tksms.showinfo("SALE", "The checked value =" + str(self.chkSaleVar.get()))

def chkImp\_Checked(self):

self.tksms.showinfo("IMPORT", "The checked value =" + str(self.chkImpVar.get()))

def chkRpt\_Checked(self):

self.tksms.showinfo("REPORT", "The checked value =" + str(self.chkRptVar.get()))

def btnSave\_click(self):

self.tksms.showinfo("SAVE", "You click on Save!")

def btnCancel\_click(self):

result=self.tksms.askquestion("LOGIN", "Pls click 'YES' or 'NO'!")

if(result=='yes'):

self.tksms.showinfo("EXIT", "System will exit!")

self.root.quit();

else:

self.tksms.showinfo("BACK", "System will back!")

#=============end EVENT===================

#==========gloable varible===========

chkAdminVar=IntVar()

chkSaleVar=IntVar()

chkImpVar=IntVar()

chkRptVar=IntVar()

#=========end gloable variable=============

def \_\_init\_\_(self):

self.root.title("This is Tkinter Title...");

# ===============make form center screen====================

form\_width = 350;

form\_high = 250;

x\_cor = self.root.winfo\_screenwidth() / 2 - form\_width / 2;

y\_cor = self.root.winfo\_screenheight() / 2 - form\_high / 2;

self.root.geometry("%dx%d+%d+%d" % (form\_width, form\_high, x\_cor, y\_cor));

self.root.maxsize(form\_width, form\_high)

self.root.minsize(form\_width, form\_high)

# design control interface UI====================

lblTitle = self.tk.Label(self.root, text='Welcome to create GUI with Tkinter...')

lblTitle.grid(columnspan=3,sticky='');

lblUserCode = self.tk.Label(self.root, text='User Code :')

lblUserCode.grid(row=1, column=0, sticky='W');

txtUserCode = self.tk.Entry(self.root, width=45)

txtUserCode.grid(row=1, column=2, sticky='W', columnspan=2);

lblUserName = self.tk.Label(self.root, text='User Name :')

lblUserName.grid(row=2, column=0, sticky='W');

txtUserName = self.tk.Entry(self.root, width=45)

txtUserName.grid(row=2, column=2, sticky='W', columnspan=2);

lblPwd = self.tk.Label(self.root, text='Password :')

lblPwd.grid(row=3, column=0, sticky='W');

txtPwd = self.tk.Entry(self.root, width=45)

txtPwd.grid(row=3, column=2, sticky='W', columnspan=2);

lblConfirm = self.tk.Label(self.root, text='Confirm :')

lblConfirm.grid(row=4, column=0, sticky='W');

txtConfirm = self.tk.Entry(self.root, width=45)

txtConfirm.grid(row=4, column=2, sticky='W', columnspan=2);

chkAdmin=self.tk.Checkbutton(self.root,text='Administrator', variable=self.chkAdminVar,command=self.chkAdmin\_Checked)

chkAdmin.grid(row=5, column=2, sticky='W', columnspan=2);

chkSale = self.tk.Checkbutton(self.root, text='Sale Management', variable=self.chkSaleVar,command=self.chkSale\_Checked)

chkSale.grid(row=6, column=2, sticky='W', columnspan=2);

chkImp = self.tk.Checkbutton(self.root, text='Import Management', variable=self.chkImpVar,command=self.chkImp\_Checked)

chkImp.grid(row=7, column=2, sticky='W', columnspan=2);

chkRpt = self.tk.Checkbutton(self.root, text='Report Management', variable=self.chkRptVar,command=self.chkRpt\_Checked)

chkRpt.grid(row=8, column=2, sticky='W', columnspan=2);

chkRpt.select();

# buton save + cancel

btnSave = self.tk.Button(self.root, text="SAVE", width=10, command=self.btnSave\_click)

btnSave.grid(row=9, column=2, sticky='E');

btnCancel = self.tk.Button(self.root, text="CANCEL", width=10, command=self.btnCancel\_click)

btnCancel.grid(row=9, column=3, sticky='W');

# set focus

txtUserCode.focus();

# ==========end design of interface UI=================

def show(self):

#==========show form===============

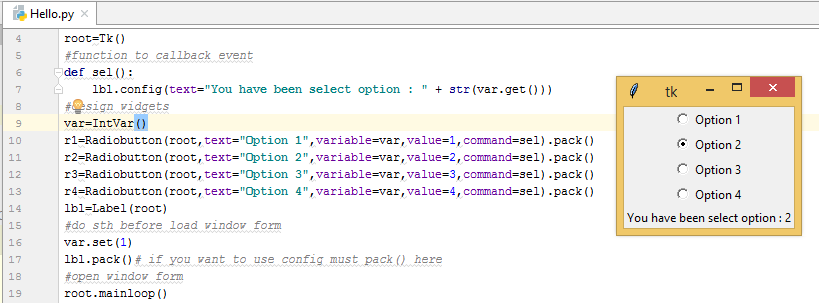
self.root.mainloop();

**របៀបប្រើប្រាស់ RadioButton:** radiobutton គឺប្រើសម្រាប់ single-choice

**Syntax**: w = Radiobutton ( master, option, ... )

Method និង Attribute សំខាន់ៗ:

* deselect()
* flash()
* invoke()
* select()
* command : the function to be call when check state changed
* cursor : change cursor
* font : change font
* text: set text to box label
* variable: The control variable that this radiobutton shares with the other radiobuttons in the group. This can be either an IntVar or a StringVar.
* textvariable: To slave the text displayed in a label widget to a control variable of class StringVar, set this option to that variable.
* value: When a radiobutton is turned on by the user, its control variable is set to its current value option. If the control variable is an IntVar, give each radiobutton in the group a different integer value option. If the control variable is a StringVar, give each radiobutton a different string value option.
* state : default is state=NORMAL, state=DISABLED for unresponsive



from tkinter import \*

import tkinter

#create main window

root=Tk()

#function to callback event

def sel():

lbl.config(text="You have been select option : " + str(var.get()))

#design widgets

var=IntVar()

r1=Radiobutton(root,text="Option 1",variable=var,value=1,command=sel).pack()

r2=Radiobutton(root,text="Option 2",variable=var,value=2,command=sel).pack()

r3=Radiobutton(root,text="Option 3",variable=var,value=3,command=sel).pack()

r4=Radiobutton(root,text="Option 4",variable=var,value=4,command=sel).pack()

lbl=Label(root)

#do sth before load window form

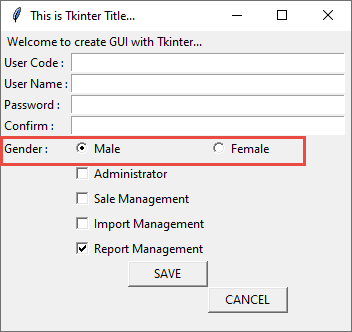
var.set(1)

lbl.pack()# if you want to use config must pack() here

#open window form

root.mainloop()

root.quit()

លំហាត់៖

**របៀបប្រើប្រាស់ Spinbox:**សម្រាប់numberមានup-down button

Syntax : w = Spinbox( master, option, ... )

Attribute សំខាន់ៗ៖

* text
* command: function callback when scrollbar is move up-down
* format : string format
* from\_ : min value
* to: max value
* textvariable: for control on widget
* validate : Validation mode. Default is NONE.
* validatecommand: Validation callback. No default value.
* values : A tuple containing valid values for this widget. Overrides from/to/increment.
* vcmd: Same as validatecommand.
* xscrollcommand:
* bg
* fg
* relief
* delete(startindex [,endindex]): This method deletes a specific character or a range of text.
* get(startindex [,endindex]): This method returns a specific character or a range of text.
* identify(x, y): Identifies the widget element at the given location.
* index(index): Returns the absolute value of an index based on the given index.
* insert(index [,string]...): This method inserts strings at the specified index location.
* invoke(element) : Invokes a spinbox button.

from tkinter import \*

import tkinter

import tkinter.messagebox

#create a TK form object

root=Tk()

#Define function for callback event

def spinb\_Scroll():

print("spinbox was scroll")

#create widgets

spin=Spinbox(root,from\_=0,to=11,command=spinb\_Scroll).pack()

#Before Open Form Event

root.title("GUI-TKinter Programming-Spinbox")

root.geometry("300x300")

root.configure(background="green")

#Open Form

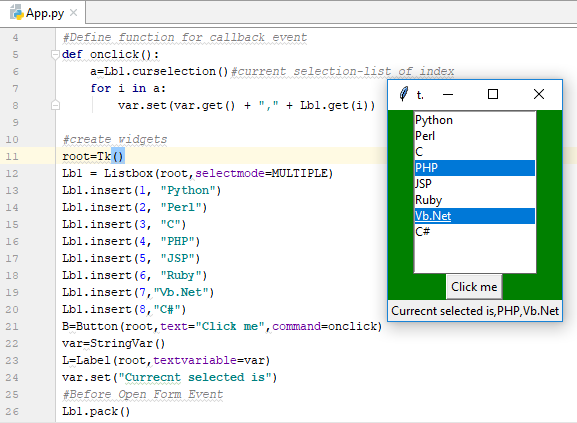
root.mainloop()

**របៀបប្រើប្រាស់ Listbox**: display a list of item

Syntax : w = Listbox ( master, option, ... )

Attribute សំខាន់ៗ៖

* bg
* fg
* bd
* font
* height : number of line
* highlightcolor
* highlightthickness
* relief: default is SUNKEN
* selectbackground
* selectmode: Determines how many items can be selected, and how mouse drags affect the selection(BROWSE, SINGLE, MULTIPLE, EXTENDED)
* width: number of character in width, default=20.
* xscrollcommand: scroll the listbox horizontally
* yscrollcommand: scroll the listbox vertically
* activate ( index ): select item by index
* curselection():Returns a tuple containing the line numbers of the selected element or elements, counting from 0. If nothing is selected, returns an empty tuple.
* delete ( first, last=None ): delete item by index
* get ( first, last=None ): Returns a tuple containing the text of the lines with indices from first to last, inclusive. If the second argument is omitted, returns the text of the line closest to first.
* index ( i )
* insert ( index, \*elements ): Insert one or more new lines into the listbox before the line specified by index. Use **END** as the first argument if you want to add new lines to the end of the listbox.
* nearest ( y )
* see ( index )
* size(): return number of items
* xview():To make the listbox horizontally scrollable. set the command option of the associated horizontal scrollbar to this method
* xview\_moveto ( fraction ): Scroll the listbox so that the leftmost fraction of the width of its longest line is outside the left side of the listbox. Fraction is in the range [0,1].
* xview\_scroll ( number, what ): Scrolls the listbox horizontally. For the what argument, use either UNITS to scroll by characters, or PAGES to scroll by pages, that is, by the width of the listbox. The number argument tells how many to scroll.
* yview():
* yview\_moveto ( fraction )
* yview\_scroll ( number, what )

from tkinter import \*

import tkinter

#Define function for callback event

def onclick():

a=Lb1.curselection()#current selection-list of index

for i in a:

var.set(var.get() + "," + Lb1.get(i))

#create widgets

root=Tk()

Lb1 = Listbox(root,selectmode=MULTIPLE)

Lb1.insert(1, "Python")

Lb1.insert(2, "Perl")

Lb1.insert(3, "C")

Lb1.insert(4, "PHP")

Lb1.insert(5, "JSP")

Lb1.insert(6, "Ruby")

Lb1.insert(7,"Vb.Net")

Lb1.insert(8,"C#")

B=Button(root,text="Click me",command=onclick)

var=StringVar()

L=Label(root,textvariable=var)

var.set("Currecnt selected is")

#Before Open Form Event

Lb1.pack()

B.pack()

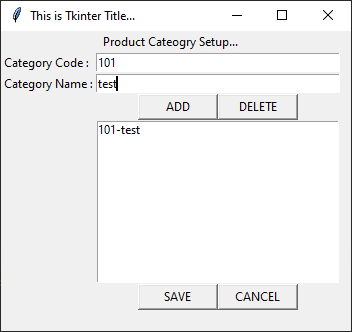
L.pack()

root.title("GUI-TKinter Programming-Spinbox")

root.configure(background="green")

#Open Form

root.mainloop()

លំហាត់៖

ចម្លើយ៖

១) Category\_frm.py

from tkinter import \*;

class Category\_frm:

import tkinter as tk;

import tkinter.messagebox as tksms;

root = tk.Tk();

#=============create EVENT===============

def btnAdd\_click(self):

self.listbox.insert(self.tk.END,self.txtCatCodeVar.get() + "-" + self.txtCatNameVar.get())

self.tksms.showinfo("ADD", "The Item was Added")

def btnDelete\_click(self):

self.listbox.delete(self.listbox.index(self.listbox.curselection()), self.listbox.index(self.listbox.curselection()))

print(self.listbox.curselection())

self.tksms.showinfo("DELETE", "Item is deleted!")

def btnSave\_click(self):

self.tksms.showinfo("SAVE", "You click on Save!")

def btnCancel\_click(self):

result=self.tksms.askquestion("LOGIN", "Pls click 'YES' or 'NO'!")

if(result=='yes'):

self.tksms.showinfo("EXIT", "System will exit!")

self.root.quit();

else:

self.tksms.showinfo("BACK", "System will back!")

#=============end EVENT===================

#==========gloable varible===========

txtCatCodeVar=StringVar()

txtCatNameVar=StringVar()

listbox=None;

#=========end gloable variable=============

def \_\_init\_\_(self):

self.root.title("This is Tkinter Title...");

# ===============make form center screen====================

form\_width = 350;

form\_high = 300;

x\_cor = self.root.winfo\_screenwidth() / 2 - form\_width / 2;

y\_cor = self.root.winfo\_screenheight() / 2 - form\_high / 2;

self.root.geometry("%dx%d+%d+%d" % (form\_width, form\_high, x\_cor, y\_cor));

self.root.maxsize(form\_width, form\_high)

self.root.minsize(form\_width, form\_high)

# design control interface UI====================

lblTitle = self.tk.Label(self.root, text='Product Cateogry Setup...')

lblTitle.grid(columnspan=3,sticky='');

lblCatCode = self.tk.Label(self.root, text='Category Code :')

lblCatCode.grid(row=1, column=0, sticky='W');

txtCatCode = self.tk.Entry(self.root, width=40,textvariable=self.txtCatCodeVar)

txtCatCode.grid(row=1, column=1, sticky='W', columnspan=2);

lblCatName = self.tk.Label(self.root, text='Category Name :')

lblCatName.grid(row=2, column=0, sticky='W');

txtCatName = self.tk.Entry(self.root, width=40,textvariable=self.txtCatNameVar)

txtCatName.grid(row=2, column=1, sticky='W', columnspan=2);

# buton ADD & DELETE

btnADD = self.tk.Button(self.root, text="ADD", width=10, command=self.btnAdd\_click)

btnADD.grid(row=3, column=1, sticky='E');

# buton ADD

btnDELETE = self.tk.Button(self.root, text="DELETE", width=10, command=self.btnDelete\_click)

btnDELETE.grid(row=3, column=2, sticky='W');

lBox=self.tk.Listbox(self.root,selectmode=self.tk.SINGLE, width=40)

lBox.grid(row=4, column=1, sticky='W', columnspan=2)

self.listbox=lBox;

# buton save + cancel

btnSave = self.tk.Button(self.root, text="SAVE", width=10, command=self.btnSave\_click)

btnSave.grid(row=11, column=1, sticky='E');

btnCancel = self.tk.Button(self.root, text="CANCEL", width=10, command=self.btnCancel\_click)

btnCancel.grid(row=11, column=2, sticky='W');

# ==========end design of interface UI=================

# before show form

txtCatCode.focus();

def show(self):

#==========show form===============

self.root.mainloop();

**របៀបប្រើប្រាស់ Create PhotoImage**:

**PhotoImage** គឺជា class ដែលត្រូវបានប្រើប្រាស់ដើម្បីបង្ហាញជា icon / images (PPM, PNG, JPEG, GIF, TIFF, BMP…..) នៅក្នុង labels, buttons, canvases, and text widgets។

វាមាន method សំខាន់ៗមួយចំនួនដូចជា៖

-Var= renderImage.**subsample**(x=int,y=int) : សម្រាប់ resize/stretch រូបដោយប្រើខ្នាត x,y

-Var= renderImage.**zoom**(x=int,y=int) : សម្រាប់ zoom រូបភាបដោយប្រើខ្នាត x,y

មាន២រូបមន្ត៖

ទី១៖ renderImage= PhotoImage(file="file\_path\_string")

ទី២៖ renderImage= PhotoImage(data="base64\_string")

ដើម្បី convert Image ទៅជា base64\_string មានវិធីជាច្រើនដូចជា៖

ទី១៖ using Base64 and open()

import base64

with open("imgs/1.png", "rb") as img\_file:

base64\_string = base64.b64encode(img\_file.read())

print(base64\_string)

renderImg=PhotoImage(data=base64\_string)

img=Label(root,width=40,height=20,bg='red',image=renderImg).pack();

ទី២៖ using PIL class(pip install pillow)

from PIL import Image, ImageTk

base64\_string = Image.open("file\_path\_string")

renderImage = ImageTk.PhotoImage(base64\_string)

img=Label(root,width=40,height=20,bg='red',image=renderImg).pack();

Image with Label / Button:

**Syntax**: Label(root, image=?, justify=?, text=?, compound = ?).pack()

* justify : សម្រាប់តំរឹមtext មានដូចជា LEFT, RIGHT or CENTER
* compound​ : សម្រាប់តំរៀប image ធៀបនឹង text មានដូចជា CENTER,BOTTOM, LEFT, RIGHT, or TOP

**Syntax**: Button(root, image=?, text=?, compound = ?).pack()

* compound​ : សម្រាប់តំរៀប image ធៀបនឹង text មានដូចជា CENTER,BOTTOM, LEFT, RIGHT, or TOP

ឧទាហរណ៍ image with Label

from tkinter import \*

import tkinter

#Define function for callback event

def setImg1():

L1['image']=img1

def setImg2():

L1['image']=img2

#create widgets

root=Tk()

img1=PhotoImage(file="02.png")

img2=PhotoImage(file="1002.png")

L1=Label(root,image=img1,width=40,height=40)

B1=Button(root,text="Image 1",command=setImg1)

B2=Button(root,text="Image 2",command=setImg2)

#Before Open Form Event

L1.pack()

B1.pack()

B2.pack()

root.configure(background="green")

root.geometry("400x400+500+200")

#Open Form

root.mainloop()

ឧទាហរណ៍ image with Button

import base64

with open("imgs/1.png", "rb") as img\_file:

base64\_string = base64.b64encode(img\_file.read())

print(base64\_string)

from tkinter import \*;

root=Tk();

renderImg=PhotoImage(data=base64\_string)

Button(root,height=30, text = 'Click Me !', image = renderImg, compound = LEFT).pack()

root.mainloop()

ឧទាហរណ៍ image with Canvas

from tkinter import \*

import tkinter

#Define function for callback event

#create widgets

root=Tk()

img=PhotoImage(file="1002.png")

canvas=Canvas(root,width=500,height=500)

canvas.create\_image(0,0,anchor=NW,image=img)

#Before Open Form Event

canvas.pack()

#Open Form

root.mainloop()

**របៀបប្រើប្រាស់ Canvas Widgets**:

**Canvas** អាចឲ្យយើងយកទៅបង្កើតជារូបភាព graphic ផ្សេងៗដូចជា line, image, circle,Oval, Polygons,rectangle,arc,…។

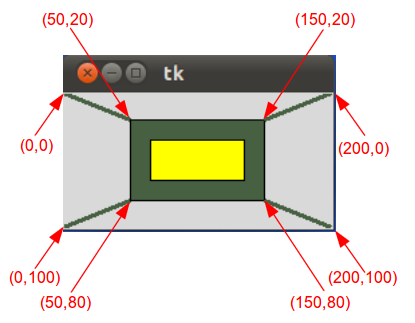
**Syntax**: Canvas(master, width=?, height=?)

Attribute សំខាន់ៗ៖

* confine : default is (true) cannot scroll outside of the scrollregion
* scrollregion: A tuple (w, n, e, s) that defines over how large an area the canvas can be scrolled, where w is the left side, n the top, e the right side, and s the bottom.
* height : Y dimension
* width: X dimension
* xscrollincrement
* xscrollcommand
* yscrollincrement
* yscrollcommand

Method សម្រាប់ draw ជាobject ផ្សេងៗរួមមាន៖

* create\_rectangle(x,y,width,height,option): គូរាងចតុកោណកែង  
  option រួមមាន៖ fill=’color’, outline=’color’, dash=(int,int)……
* create\_line(x0,y0,x1,y1,option): គូជាបន្ទាត់  
  option រួមមាន៖ fill=’color’, outline=’color’, dash=(int,int)……
* create\_oval ( x0, y0, x1, y1, option, ... ) : គូជារៀងរង្វង់
* create\_polygon( [x0,y0, x1,y1, x2,y2, ...],option,…): គូរូបពហុកោណ  
  យ៉ាងហោចណាស់ត្រូវមាន coord ត្រឹម x0,y0, x1,y1, x2,y2
* create\_image(x0,y0,option…): ដាក់រូបភាព  
  option រួមមាន anchor=NW, image=img….
* create\_text(x0,y0,text=’text’): គូរ Text
* canvas.create\_arc(coord, option…)  
  coord = x0, y0, x1, y1  
  option : start=int, extent=int, fill="color"

**ឧទាហរណ៍ Canvas with Line and Rectangle**

from tkinter import \*

master = Tk()

w = Canvas(master, width=200, height=100)

w.pack()

w.create\_rectangle(50, 20, 150, 80, fill="#476042")

w.create\_rectangle(65, 35, 135, 65, fill="yellow")

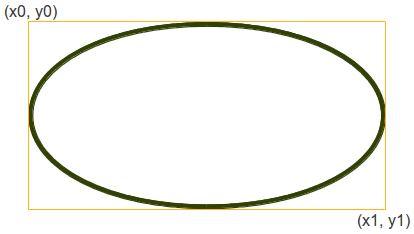
w.create\_line(0, 0, 50, 20, fill="#476042", width=3)

w.create\_line(0, 100, 50, 80, fill="#476042", width=3)

w.create\_line(150,20, 200, 0, fill="#476042", width=3)

w.create\_line(150, 80, 200, 100, fill="#476042", width=3)

mainloop()



**ឧទាហរណ៍ Canvas with Oval**

from tkinter import \*

canvas\_width = 190

canvas\_height =150

master = Tk()

w = Canvas(master,

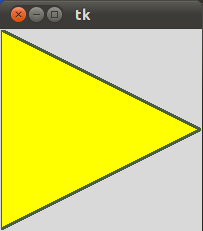
width=canvas\_width,

height=canvas\_height)

w.pack()

w.create\_oval(50,50,100,100)

mainloop()

**ឧទាហរណ៍** **Canvas with Oval**

from tkinter import \*

canvas\_width = 200

canvas\_height =200

python\_green = "#476042"

master = Tk()

w = Canvas(master,

width=canvas\_width,

height=canvas\_height)

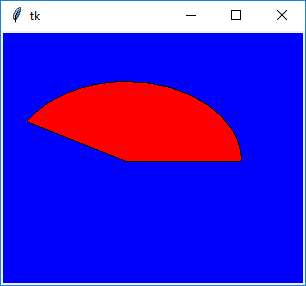
w.pack()

points = [0,0,canvas\_width,canvas\_height/2, 0, canvas\_height]

w.create\_polygon(points, outline=python\_green,

fill='yellow', width=3)

mainloop()

**ឧទាហរណ៍ Canvas with Arc**

from tkinter import \*

#Define function for callback event

#create widgets

root=Tk()

C = Canvas(root, bg="blue", height=250, width=300)

coord = 10, 50, 240, 210

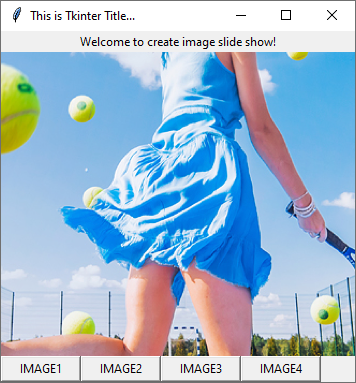
arc = C.create\_arc(coord, start=0, extent=150, fill="red")

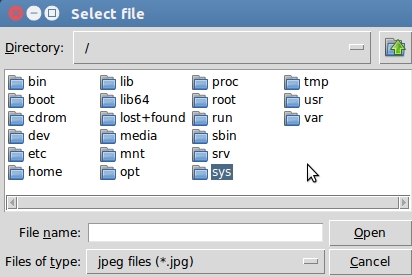
C.pack()

#Before Open Form Event

#Open Form

root.mainloop()

**លំហាត់**៖

**របៀបប្រើប្រាស់ FileDialog**: សម្រាប់brows ទៅរកទីតាំងfile ឬ folder(save file,open file, brow directory)។

**Options ខាងក្រោមគឺសម្រាប់ customize Dialog**៖

* + **parent** - the window to place the dialog on top of
  + **title** - the title of the window
  + **initialdir** - the directory that the dialog starts in
  + **initialfile** - the file selected upon opening of the dialog
  + **filetypes** - a sequence of (label, pattern) tuples, ‘\*’ wildcard is allowed
  + **defaultextension** - default extension to append to file (save dialogs)
  + **multiple** - when true, selection of multiple items is allowed

**Method ខាងក្រោមគឺសម្រាប់ customize Dialog**៖

* tkinter.filedialog.**askopenfile**(mode="r", \*\*options)
* tkinter.filedialog.**askopenfiles**(mode="r", \*\*options)  
  The above two functions create an Open dialog and return the opened file object(s) in read-only mode.
* tkinter.filedialog.**asksaveasfile**(mode="w", \*\*options)  
  Create a SaveAs dialog and return a file object opened in write-only mode.
* tkinter.filedialog.**askopenfilename**(\*\*options)
* tkinter.filedialog.**askopenfilenames**(\*\*options)  
  The above two functions create an Open dialog and return the selected filename(s) that correspond to existing file(s).
* tkinter.filedialog.**asksaveasfilename**(\*\*options)  
  Create a SaveAs dialog and return the selected filename.
* tkinter.filedialog.**askdirectory**(\*\*options)  
  Prompt user to select a directory.  
  Additional keyword option: mustexist - determines if selection must be an existing directory.

**Syntax** : root.varableName=filedialog.methodName(option,…)

from tkinter import \*

from tkinter import filedialog

#Define function for callback event

#create widgets

root=Tk()

root.fileName=filedialog.askopenfilename(initialdir="/",title="Open Image",filetypes=(("PNG file","\*.png"),("All files","\*.\*")))

img=PhotoImage(file=root.fileName)

canvas=Canvas(root,width=500,height=500)

canvas.create\_image(0,0,anchor=NW,image=img)

#Before Open Form Event

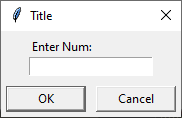
canvas.pack()

#Open Form

root.mainloop()

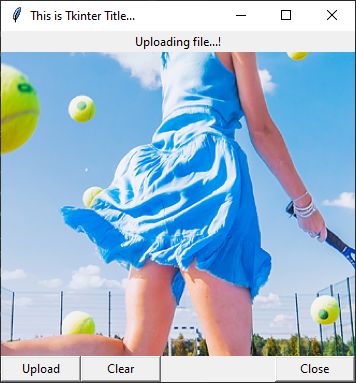
**របៀបប្រើប្រាស់ SimpleDialog**: ជា​ dialog សម្រាប់ឲ្យ user បញ្ចូលទិន្នន័យ(លេខ, អក្សរ)។

មាន​ **method** ៣សំខាន់ដូចខាងក្រោម៖

* tkinter.simpledialog.**askfloat**(title, prompt, \*\*kw)
* tkinter.simpledialog.**askinteger**(title, prompt, \*\*kw)
* ****tkinter.simpledialog.**askstring**(title, prompt, \*\*kw)

**Syntax** : root.varableName= simpledialog.methodName(title, prompt, \*\*kw)

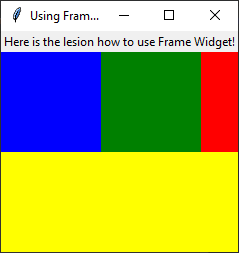
ដែល title និង prompt គឺជា string។

**លំហាត់**៖

**របៀបប្រើប្រាស់ Frame**: គឺជា container ប្រើសម្រាប់ hold/grouping នូវ widgets។

Syntax : w = Frame ( master, option, ... )

* master − This represents the parent window
* options – are key-value pairs separated by commas មានដូចជា bg,bd, cursor, height, highlightbackground, highlightcolor, highlightthickness, width, relief

from tkinter import \*

root=Tk();

root.title("Using Frame-Container");

#event/method

#widgets

frame=Frame(root);

frame.pack(side=TOP,fill=X);

lbl1=Label(frame,text="Here is the lesion how to use Frame Widget!");

lbl1.pack();

frameTop=Frame(root,bg='red',height=100);

frameTop.pack(fill=X);

frameLeft=Frame(frameTop,bg='blue',height=100,width=100);

frameLeft.grid(row=0,column=0);

frameRight=Frame(frameTop,bg='green',height=100,width=100);

frameRight.grid(row=0,column=1);

frameBottom=Frame(root,bg='yellow',height=100);

frameBottom.pack(fill=X,expand=1);

root.mainloop()

**របៀបប្រើប្រាស់ PanedWindow**: ជា​ប្រភេទមួយទៀតនៃ container ដែលប្រើសម្រាប់តំរៀប widgets តាមពីរទិសដៅគឺ horizontally ឬ vertically។

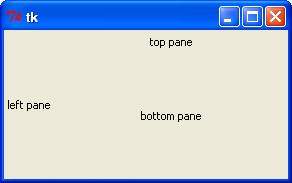
Syntax : w = PanedWindow( master, option, ... )

* master − This represents the parent window
* options − រួមមាន bg, bd, borderwidth, cursor, height, width, relief, orient(Default is HORIZONTAL, VERTICAL)………….

មាន​ **method** សំខាន់ៗដូចខាងក្រោម៖

* add(child, options)​ : Adds a child window to the paned window.
* get(startindex [,endindex]) : This method returns a specific character or a range of text.
* config(options) : Modifies one or more widget options. If no options are given, the method returns a dictionary containing all current option values.

from tkinter import \*

m1 = PanedWindow()

m1.pack(fill=BOTH, expand=1)

left = Label(m1, text="left pane")

m1.add(left)

m2 = PanedWindow(m1, orient=VERTICAL)

m1.add(m2)

top = Label(m2, text="top pane")

m2.add(top)

bottom = Label(m2, text="bottom pane")

m2.add(bottom)

mainloop()

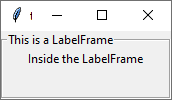
**របៀបប្រើប្រាស់ LabelFrame**: ជាប្រភេទមួយទៀតនៃ container ដែលជាទូទៅវាមានស៊ុំ និង text title របស់វានៅពីលើ។

Syntax : w = LabelFrame( master, option, ... )

* master − This represents the parent window
* options − ជាគូនៃ key-value រួមមាន text, font, labelAnchor , highlightbackground, highlightcolor, highlightthickness, bg, bd, cursor, height, width, relief, ….

from tkinter import \*

root = Tk()

labelframe = LabelFrame(root, text="This is a LabelFrame")

labelframe.pack(fill="both", expand="yes")

left = Label(labelframe, text="Inside the LabelFrame")

left.pack()

root.mainloop()

**របៀបប្រើប្រាស់ Scrollbar**: ប្រើសម្រាប់ដាក់ vertical/horizontal scrollbar ក្នុង​widget ផ្សេងៗទៀតដូចជា Text, Listbox,Canvas,Entry,….។

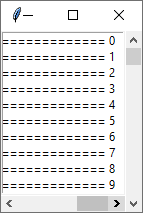
Syntax: w = Scrollbar ( master, option, ... )

* master − This represents the parent window
* options − រួមមាន៖

|  |  |
| --- | --- |
| **Sr.No.** | **Option & Description** |
| 1 | **activebackground**  The color of the slider and arrowheads when the mouse is over them. |
| 2 | **bg**  The color of the slider and arrowheads when the mouse is not over them. |
| 3 | **bd**  The width of the 3-d borders around the entire perimeter of the trough, and also the width of the 3-d effects on the arrowheads and slider. Default is no border around the trough, and a 2-pixel border around the arrowheads and slider. |
| 4 | **command**  A procedure to be called whenever the scrollbar is moved. |
| 5 | **cursor**  The cursor that appears when the mouse is over the scrollbar. |
| 6 | **elementborderwidth**  The width of the borders around the arrowheads and slider. The default is elementborderwidth=-1, which means to use the value of the borderwidth option. |
| 7 | **highlightbackground**  The color of the focus highlight when the scrollbar does not have focus. |
| 8 | **highlightcolor**  The color of the focus highlight when the scrollbar has the focus. |
| 9 | **highlightthickness**  The thickness of the focus highlight. Default is 1. Set to 0 to suppress display of the focus highlight. |
| 10 | **jump**  This option controls what happens when a user drags the slider. Normally (jump=0), every small drag of the slider causes the command callback to be called. If you set this option to 1, the callback isn't called until the user releases the mouse button. |
| 11 | **orient**  Set orient=HORIZONTAL for a horizontal scrollbar, orient=VERTICAL for a vertical one. |
| 12 | **repeatdelay**  This option controls how long button 1 has to be held down in the trough before the slider starts moving in that direction repeatedly. Default is repeatdelay=300, and the units are milliseconds. |
| 13 | **repeatinterval**  repeatinterval |
| 14 | **takefocus**  Normally, you can tab the focus through a scrollbar widget. Set takefocus=0 if you don't want this behavior. |
| 15 | **troughcolor**  The color of the trough. |
| 16 | **width**  Width of the scrollbar (its y dimension if horizontal, and its x dimension if vertical). Default is 16. |

មាន​ **method** សំខាន់ៗដូចខាងក្រោម៖

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **get()**  Returns two numbers (a, b) describing the current position of the slider. The a value gives the position of the left or top edge of the slider, for horizontal and vertical scrollbars respectively; the b value gives the position of the right or bottom edge. |
| 2 | **set ( first, last )**  To connect a scrollbar to another widget w, set w's xscrollcommand or yscrollcommand to the scrollbar's set() method. The arguments have the same meaning as the values returned by the get() method. |

ឧទាហរណ៍៖

from tkinter import \*

root = Tk()

frame=Frame(root)

frame.pack()

scrollbarY=Scrollbar(frame)

scrollbarY.pack(side=RIGHT,fill=Y);

scrollbarX=Scrollbar(frame,orient=HORIZONTAL)

scrollbarX.pack(side=BOTTOM,fill=X);

mylist=Listbox(frame,yscrollcommand=scrollbarY.set,xscrollcommand=scrollbarX.set)

mylist.pack(side=LEFT,fill=BOTH)

for line in range(100):

mylist.insert(END, "This is line number============================= " + str(line))

scrollbarY.configure(command=mylist.yview)

scrollbarX.configure(command=mylist.xview)

root.mainloop()

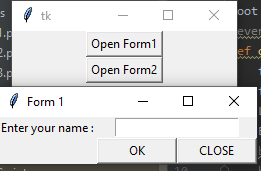
**របៀបប្រើប្រាស់ Toplevel**: is used to provide a separate window container។

Syntax : w = Toplevel ( option, ... )

* master − This represents the parent window
* options − រួមមាន bg, bd, fg, cursor, height, width, font, relief, relief, class\_,………….

មាន​ **method** សំខាន់ៗដូចខាងក្រោម៖

|  |  |
| --- | --- |
| **Sr.No.** | **Methods & Description** |
| 1 | **deiconify()**  Displays the window, after using either the iconify or the withdraw methods. |
| 2 | **frame()**  Returns a system-specific window identifier. |
| 3 | **group(window)**  Adds the window to the window group administered by the given window. |
| 4 | **iconify()**  Turns the window into an icon, without destroying it. |
| 5 | **protocol(name, function)**  Registers a function as a callback which will be called for the given protocol. |
| 6 | **iconify()**  Turns the window into an icon, without destroying it. |
| 7 | **state()**  Returns the current state of the window. Possible values are normal, iconic, withdrawn and icon. |
| 8 | **transient([master])**  Turns the window into a temporary(transient) window for the given master or to the window's parent, when no argument is given. |
| 9 | **withdraw()**  Removes the window from the screen, without destroying it. |
| 10 | **maxsize(width, height)**  Defines the maximum size for this window. |
| 11 | **minsize(width, height)**  Defines the minimum size for this window. |
| 12 | **positionfrom(who)**  Defines the position controller. |
| 13 | **resizable(width, height)**  Defines the resize flags, which control whether the window can be resized. |
| 14 | **sizefrom(who)**  Defines the size controller. |
| 15 | **title(string)**  Defines the window title. |

**ឧទាហរណ៍១**៖

from tkinter import \*

root = Tk()

#event

def openForm1():

form1=Toplevel()

form1.title("Form 1")

Label(form1,text='Enter your name :').grid(row=0,column=0);

Entry(form1,width=20).grid(row=0,column=1,columnspan=2);

btnOK=Button(form1,text='OK',width=10).grid(row=1,column=1)

btnClose = Button(form1, text='CLOSE',width=10).grid(row=1,column=2)

def openForm2():

form2 = Toplevel()

form2.title("Form 1")

form2.geometry("100x100");

lbl1 = Label(form2, text='Here is the text in form2').pack();

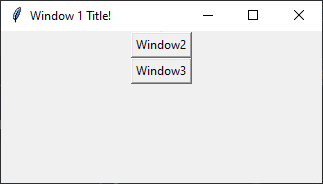
#end event

#design widget

btn1=Button(root,text='Open Form1',command=openForm1).pack()

btn2=Button(root,text='Open Form2',command=openForm2).pack()

root.mainloop()



**ឧទាហរណ៍២**៖

from tkinter import \*

class **Window1**:

def \_\_init\_\_(self,root):

self.master=root;

self.master.title("Window 1 Title!")

self.master.geometry('800x600+0+0');

self.frame=Frame(self.master,bg='powder blue');

self.frame.pack();

self.btnWin2=Button(self.master,text='Window2',command=self.new\_window2).pack()

self.btnWin3 = Button(self.master, text='Window3', command=self.new\_window3).pack()

self.master.mainloop()

def new\_window2(self):

self.newWindow=Toplevel(self.master)

self.app=Window2(self.newWindow)

def new\_window3(self):

self.newWindow=Toplevel(self.master)

self.app=Window2(self.newWindow)

class **Window2**:

def \_\_init\_\_(self,root):

self.master=root;

self.master.title("Window 2 Title!")

self.master.geometry('600x300+0+0');

self.frame=Frame(self.master,bg='powder blue');

self.lbl1 = Label(self.frame, text='Label 1 in Window 2').pack()

self.frame.pack();

class **Window3**:

def \_\_init\_\_(self, root):

self.master = root;

self.master.title("Window 3 Title!")

self.master.geometry('600x300+0+0');

self.frame = Frame(self.master, bg='powder blue');

self.lbl1=Label(self.frame,text='Label 1 in Window 3').pack()

self.frame.pack();

def new\_window(self):

self.newWin=Toplevel(self.master)

root = Tk()

app=Window1(root);

**របៀបប្រើប្រាស់ Menu**: ជា​ widget ប្រើសម្រាប់បង្កើតជា menu ផ្សេងៗក្នុង app។

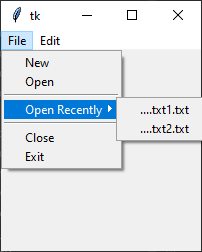
Syntax: w = Menu ( master, option, ... )

មាន​ **attribute** សំខាន់ៗដូចខាងក្រោម៖

|  |  |
| --- | --- |
| **Sr.No.** | **Description** |
| 1 | **activebackground**  The background color that will appear on a choice when it is under the mouse. |
| 2 | **activeborderwidth**  Specifies the width of a border drawn around a choice when it is under the mouse. Default is 1 pixel. |
| 3 | **activeforeground**  The foreground color that will appear on a choice when it is under the mouse. |
| 4 | **bg**  The background color for choices not under the mouse. |
| 5 | **bd**  The width of the border around all the choices. Default is 1. |
| 6 | **cursor**  The cursor that appears when the mouse is over the choices, but only when the menu has been torn off. |
| 7 | **disabledforeground**  The color of the text for items whose state is DISABLED. |
| 8 | **font**  The default font for textual choices. |
| 9 | **fg**  The foreground color used for choices not under the mouse. |
| 10 | **postcommand**  You can set this option to a procedure, and that procedure will be called every time someone brings up this menu. |
| 11 | **relief**  The default 3-D effect for menus is relief=RAISED. |
| 12 | **image**  To display an image on this menubutton. |
| 13 | **selectcolor**  Specifies the color displayed in checkbuttons and radiobuttons when they are selected. |
| 14 | **tearoff**  Normally, a menu can be torn off, the first position (position 0) in the list of choices is occupied by the tear-off element, and the additional choices are added starting at position 1. If you set tearoff=0, the menu will not have a tear-off feature, and choices will be added starting at position 0. |
| 15 | **title**  Normally, the title of a tear-off menu window will be the same as the text of the menubutton or cascade that lead to this menu. If you want to change the title of that window, set the title option to that string. |

មាន​ **method** សំខាន់ៗដូចខាងក្រោម៖

|  |  |
| --- | --- |
| **Sr.No.** | **Option & Description** |
| 1 | **add\_command (options)**  Adds a menu item to the menu. |
| 2 | **add\_radiobutton( options )**  Creates a radio button menu item. |
| 3 | **add\_checkbutton( options )**  Creates a check button menu item. |
| 4 | **add\_cascade(options)**  Creates a new hierarchical menu by associating a given menu to a parent menu |
| 5 | **add\_separator()**  Adds a separator line to the menu. |
| 6 | **add( type, options )**  Adds a specific type of menu item to the menu. |
| 7 | **delete( startindex [, endindex ])**  Deletes the menu items ranging from startindex to endindex. |
| 8 | **entryconfig( index, options )**  Allows you to modify a menu item, which is identified by the index, and change its options. |
| 9 | **index(item)**  Returns the index number of the given menu item label. |
| 10 | **insert\_separator ( index )**  Insert a new separator at the position specified by index. |
| 11 | **invoke ( index )**  Calls the command callback associated with the choice at position index. If a checkbutton, its state is toggled between set and cleared; if a radiobutton, that choice is set. |
| 12 | **type ( index )**  Returns the type of the choice specified by index: either "cascade", "checkbutton", "command", "radiobutton", "separator", or "tearoff". |



from tkinter import \*

def donothing():

print("Do nothing................!")

root = Tk()

menubar=Menu(root)

filemenu=Menu(menubar,tearoff=0)

filemenu.add\_command(label='New',command=donothing)

filemenu.add\_command(label='Open',command=donothing)

filemenu.add\_separator()

opensubmenu=Menu(root,tearoff=0)

opensubmenu.add\_command(label='....txt1.txt')

opensubmenu.add\_command(label='....txt2.txt')

filemenu.add\_cascade(label='Open Recently',menu=opensubmenu)

filemenu.add\_separator()

filemenu.add\_command(label='Close',command=donothing)

filemenu.add\_command(label='Exit',command=donothing)

menubar.add\_cascade(label='File',menu=filemenu)

editbar=Menu(root)

editbar=Menu(menubar,tearoff=0)

editbar.add\_command(label='Cut',command=donothing)

editbar.add\_command(label='Copy',command=donothing)

editbar.add\_command(label='Paste',command=donothing)

menubar.add\_cascade(label='Edit',menu=editbar)

root.configure(menu=menubar)

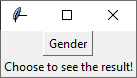
root.mainloop()

**របៀបប្រើប្រាស់ Menubutton**: ជា​ drop-down menu សម្រាប់ប្រើជាមួយ menu។

Syntax: w = Menubutton ( master, option, ... )

មាន​ **attribute** សំខាន់ៗដូចខាងក្រោម៖

|  |  |
| --- | --- |
| **Sr.No.** | **Option & Description** |
| 1 | **activebackground**  The background color when the mouse is over the menubutton. |
| 2 | **activeforeground**  The foreground color when the mouse is over the menubutton. |
| 3 | **anchor**  This options controls where the text is positioned if the widget has more space than the text needs. The default is anchor=CENTER, which centers the text. |
| 4 | **bg**  The normal background color displayed behind the label and indicator. |
| 5 | **bitmap**  To display a bitmap on the menubutton, set this option to a bitmap name. |
| 6 | **bd**  The size of the border around the indicator. Default is 2 pixels. |
| 7 | **cursor**  The cursor that appears when the mouse is over this menubutton. |
| 8 | **direction**  Set direction=LEFT to display the menu to the left of the button; use direction=RIGHT to display the menu to the right of the button; or use direction='above' to place the menu above the button. |
| 9 | **disabledforeground**  The foreground color shown on this menubutton when it is disabled. |
| 10 | **fg**  The foreground color when the mouse is not over the menubutton. |
| 11 | **height**  The height of the menubutton in lines of text (not pixels!). The default is to fit the menubutton's size to its contents. |
| 12 | **highlightcolor**  Color shown in the focus highlight when the widget has the focus. |
| 13 | **image**  To display an image on this menubutton, |
| 14 | **justify**  This option controls where the text is located when the text doesn't fill the menubutton: use justify=LEFT to left-justify the text (this is the default); use justify=CENTER to center it, or justify=RIGHT to right-justify. |
| 15 | **menu**  To associate the menubutton with a set of choices, set this option to the Menu object containing those choices. That menu object must have been created by passing the associated menubutton to the constructor as its first argument. |
| 16 | **padx**  How much space to leave to the left and right of the text of the menubutton. Default is 1. |
| 17 | **pady**  How much space to leave above and below the text of the menubutton. Default is 1. |
| 18 | **relief**  Selects three-dimensional border shading effects. The default is RAISED. |
| 19 | **state**  Normally, menubuttons respond to the mouse. Set state=DISABLED to gray out the menubutton and make it unresponsive. |
| 20 | **text**  To display text on the menubutton, set this option to the string containing the desired text. Newlines ("\n") within the string will cause line breaks. |
| 21 | **textvariable**  You can associate a control variable of class StringVar with this menubutton. Setting that control variable will change the displayed text. |
| 22 | **underline**  Normally, no underline appears under the text on the menubutton. To underline one of the characters, set this option to the index of that character. |
| 23 | **width**  The width of the widget in characters. The default is 20. |
| 24 | **wraplength**  Normally, lines are not wrapped. You can set this option to a number of characters and all lines will be broken into pieces no longer than that number. |

from tkinter import \*

root = Tk()

def checkValue():

if(maleVar.get()==1):

lbl1Var.set("You click on Male");

if (femaleVar.get() == 1):

lbl1Var.set("You click on Female");

mb=Menubutton(root, text="Gender", relief=RAISED)

mb.menu = Menu(mb,tearoff=0)

mb["menu"] =mb.menu

maleVar = IntVar()

femaleVar = IntVar()

mb.menu.add\_checkbutton(label="ប្រុស",variable=maleVar,command=checkValue)

mb.menu.add\_checkbutton( label="ស្រី",variable=femaleVar,command=checkValue)

mb.pack()

lbl1Var=StringVar();

lbl1=Label(root,textvariable=lbl1Var).pack()

lbl1Var.set('Choose to see the result!');

root.mainloop()

**របៀបប្រើប្រាស់ Treeview**: ជា​ widget ប្រើសម្រាប់បង្ហាញ data ជាទម្រង់ hierarchical collection of items ។

Syntax : tree=ttk.Treeview(master,option)

Option៖

|  |  |
| --- | --- |
| **Option** | **Description** |
| columns | A list of column identifiers, specifying the number of columns and their names. |
| displaycolumns | A list of column identifiers (either symbolic or integer indices) specifying which data columns are displayed and the order in which they appear, or the string “#all”. |
| height | Specifies the number of rows which should be visible. Note: the requested width is determined from the sum of the column widths. |
| padding | Specifies the internal padding for the widget. The padding is a list of up to four length specifications. |
| selectmode | Controls how the built-in class bindings manage the selection. One of “extended”, “browse” or “none”. If set to “extended” (the default), multiple items may be selected. If “browse”, only a single item will be selected at a time. If “none”, the selection will not be changed.  Note that the application code and tag bindings can set the selection however they wish, regardless of the value of this option. |
| show | A list containing zero or more of the following values, specifying which elements of the tree to display.   * tree: display tree labels in column #0. * headings: display the heading row.   The default is “tree headings”, i.e., show all elements.  **Note**: Column #0 always refers to the tree column, even if show=”tree” is not specified. |

Item/Row Options៖

|  |  |
| --- | --- |
| **Option** | **Description** |
| text | The textual label to display for the item. |
| image | A Tk Image, displayed to the left of the label. |
| values | The list of values associated with the item.  Each item should have the same number of values as the widget option columns. If there are fewer values than columns, the remaining values are assumed empty. If there are more values than columns, the extra values are ignored. |
| open | True/False value indicating whether the item’s children should be displayed or hidden. |
| tags | A list of tags associated with this item. |

Column Identifiers៖ របៀបកំណត់អត្តសញ្ញាណរបស់ column

* A symbolic name from the list of columns option.
* An integer n, specifying the nth data column.
* A string of the form #n, where n is an integer, specifying the nth display column.

Virtual Events៖

|  |  |
| --- | --- |
| **Event** | **Description** |
| <<TreeviewSelect>> | Generated whenever the selection changes. |
| <<TreeviewOpen>> | Generated just before settings the focus item to open=True. |
| <<TreeviewClose>> | Generated just after setting the focus item to open=False. |

មាន​ **method** សំខាន់ៗដូចខាងក្រោម៖

**+ bbox(item, column=None)**

Returns the bounding box (relative to the treeview widget’s window) of the specified item in the form (x, y, width, height).

If column is specified, returns the bounding box of that cell. If the item is not visible (i.e., if it is a descendant of a closed item or is scrolled offscreen), returns an empty string.

**+ get\_children(item=None)**

Returns the list of children belonging to item.

If item is not specified, returns root children.

**+ set\_children(item, \*newchildren)**

Replaces item’s child with newchildren.

Children present in item that are not present in newchildren are detached from the tree. No items in newchildren may be an ancestor of item. Note that not specifying newchildren results in detaching item’s children.

**+ column(column, option=None, \*\*kw)**

Query or modify the options for the specified column.

If kw is not given, returns a dict of the column option values. If option is specified then the value for that option is returned. Otherwise, sets the options to the corresponding values.

The valid options/values are:

* Id: Returns the column name. This is a read-only option.
* anchor: One of the standard Tk anchor values. Specifies how the text in this column should be aligned with respect to the cell.
* minwidth: width ,The minimum width of the column in pixels. The treeview widget will not make the column any smaller than specified by this option when the widget is resized or the user drags a column.
* stretch: True/False, Specifies whether the column’s width should be adjusted when the widget is resized.
* width: width, The width of the column in pixels.

**+ delete(\*items)**

Delete all specified items and all their descendants.

The root item may not be deleted.

**+ detach(\*items)**

Unlinks all of the specified items from the tree.

The items and all of their descendants are still present, and may be reinserted at another point in the tree, but will not be displayed.

The root item may not be detached.

**+ exists(item)**

Returns True if the specified item is present in the tree.

**+ focus(item=None)**

If item is specified, sets the focus item to item. Otherwise, returns the current focus item, or ‘’ if there is none.

**+ heading(column, option=None, \*\*kw)**

Query or modify the heading options for the specified column.

If kw is not given, returns a dict of the heading option values. If option is specified then the value for that option is returned. Otherwise, sets the options to the corresponding values.

The valid options/values are:

* text: text,The text to display in the column heading.
* image: imageName,Specifies an image to display to the right of the column heading.
* anchor: anchor,Specifies how the heading text should be aligned. One of the standard Tk anchor values.
* command: callback,A callback to be invoked when the heading label is pressed.

**+ identify(component, x, y)**

Returns a description of the specified component under the point given by x and y, or the empty string if no such component is present at that position.

**+ identify\_row(y)**

Returns the item ID of the item at position y.

**+ identify\_column(x)**

Returns the data column identifier of the cell at position x.

The tree column has ID #0.

**+ identify\_region(x, y)**

Returns one of:

|  |  |
| --- | --- |
| **region** | **meaning** |
| heading | Tree heading area. |
| separator | Space between two columns headings. |
| tree | The tree area. |
| cell | A data cell. |

**+ identify\_element(x, y)**

Returns the element at position x, y.

**+ index(item)**

Returns the integer index of item within its parent’s list of children.

**+ insert(parent, index, iid=None, \*\*kw)**

Creates a new item and returns the item identifier of the newly created item.

parent is the item ID of the parent item, or the empty string to create a new top-level item. index is an integer, or the value “end”, specifying where in the list of parent’s children to insert the new item. If index is less than or equal to zero, the new node is inserted at the beginning; if index is greater than or equal to the current number of children, it is inserted at the end. If iid is specified, it is used as the item identifier; iid must not already exist in the tree. Otherwise, a new unique identifier is generated.

See Item Options for the list of available points.

**+ item(item, option=None, \*\*kw)**

Query or modify the options for the specified item.

If no options are given, a dict with options/values for the item is returned. If option is specified then the value for that option is returned. Otherwise, sets the options to the corresponding values as given by kw.

**+ move(item, parent, index)**

Moves item to position index in parent’s list of children.

It is illegal to move an item under one of its descendants. If index is less than or equal to zero, item is moved to the beginning; if greater than or equal to the number of children, it is moved to the end. If item was detached it is reattached.

**+ next(item)**

Returns the identifier of item’s next sibling, or ‘’ if item is the last child of its parent.

**+ parent(item)**

Returns the ID of the parent of item, or ‘’ if item is at the top level of the hierarchy.

**+ prev(item)**

Returns the identifier of item’s previous sibling, or ‘’ if item is the first child of its parent.

**+ reattach(item, parent, index)**

An alias for Treeview.move().

**+ see(item)**

Ensure that item is visible.

Sets all of item’s ancestors open option to True, and scrolls the widget if necessary so that item is within the visible portion of the tree.

**+ selection()**

Returns a tuple of selected items.

Changed in version 3.8: selection() no longer takes arguments. For changing the selection state use the following selection methods.

**+ selection\_set(\*items)**

items becomes the new selection.

Changed in version 3.6: items can be passed as separate arguments, not just as a single tuple.

**+ selection\_add(\*items)**

Add items to the selection.

Changed in version 3.6: items can be passed as separate arguments, not just as a single tuple.

**+ selection\_remove(\*items)**

Remove items from the selection.

Changed in version 3.6: items can be passed as separate arguments, not just as a single tuple.

**+ selection\_toggle(\*items)**

Toggle the selection state of each item in items.

Changed in version 3.6: items can be passed as separate arguments, not just as a single tuple.

**+ set(item, column=None, value=None)**

With one argument, returns a dictionary of column/value pairs for the specified item. With two arguments, returns the current value of the specified column. With three arguments, sets the value of given column in given item to the specified value.

**+ tag\_bind(tagname, sequence=None, callback=None)**

Bind a callback for the given event sequence to the tag tagname. When an event is delivered to an item, the callbacks for each of the item’s tags option are called.

**+ tag\_configure(tagname, option=None, \*\*kw)**

Query or modify the options for the specified tagname.

If kw is not given, returns a dict of the option settings for tagname. If option is specified, returns the value for that option for the specified tagname. Otherwise, sets the options to the corresponding values for the given tagname.

**+ tag\_has(tagname, item=None)**

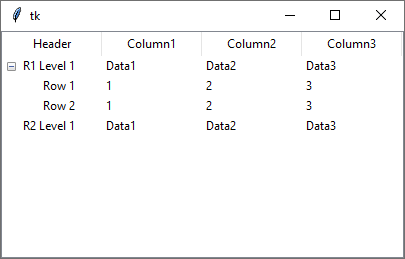
If item is specified, returns 1 or 0 depending on whether the specified item has the given tagname. Otherwise, returns a list of all items that have the specified tag.

**+ xview(\*args)**

Query or modify horizontal position of the treeview.

**+ yview(\*args)**

Query or modify vertical position of the treeview.

**ឧទាហរណ៍ទី១**៖

from tkinter import \*

from tkinter import ttk;

root = Tk()

tree=ttk.Treeview(root)

#==========prepare column============

tree["column"]=("One","Two","Three")

tree.column("#0",width=100, minwidth=100, stretch=NO)

tree.column("#1",width=100, minwidth=100, stretch=NO)

tree.column("Two",width=100, minwidth=100, stretch=NO)

tree.column("Three",width=100, minwidth=100, stretch=NO)

tree.heading("#0",text="Header",anchor=CENTER)

tree.heading("#1",text="Column1",anchor=CENTER)

tree.heading("#2",text="Column2",anchor=CENTER)

tree.heading("#3",text="Column3",anchor=CENTER)

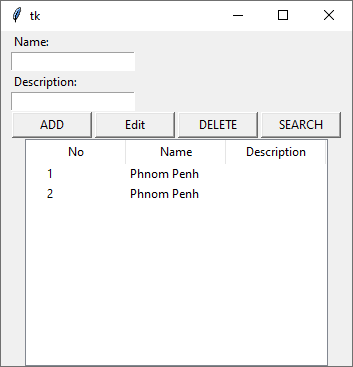
#==========add rows=============

row1\_level1=tree.insert("",1,text="R1 Level 1",value=("Data1","Data2","Data3"),open=True)

tree.insert(row1\_level1,END,text='Row 1',value=("1","2","3"))

tree.insert(row1\_level1,END,text='Row 2',value=("1","2","3"))

row2\_level1=tree.insert("",2,text="R2 Level 1",value=("Data1","Data2","Data3"))

tree.pack()

root.mainloop()

**ឧទាហរណ៍ទី២**៖

from tkinter import \*

from tkinter import ttk;

root = Tk()

#design master=======

def btnSearch\_click():

itm=tree.get\_children()

if(len(itm)==0):

return;

for row in itm:

values = tuple(tree.item(row)['values'])

name=values[0]

if(name==nameVar.get()):

descVar.set(values[1])

def btnEdit\_click():

itm = tree.selection()

if (len(itm) == 0):

return;

row = itm[0]

values = tuple(tree.item(row)['values'])

text = str(tree.item(row)['text'])

if(btnaddVar.get()=="Edit"):

btnaddVar.set("Change")

nameVar.set(values[0])

descVar.set(values[1])

else:

btnaddVar.set("Edit")

tree.item(row,text=text,values=(nameVar.get(),descVar.get()))

def btnAdd\_click():

tree.insert("",END,text=str(len(tree.get\_children())+1),value=(nameVar.get(),descVar.get()))

def btnDelete\_click():

itm=tree.selection()

if(len(itm)==0):

return;

tree.delete(itm[0])

Label(root,text="Name:").grid(sticky=W,padx=10)

nameVar=StringVar();

Entry(root,textvariable=nameVar,width=20).grid(sticky=W,padx=10)

Label(root,text="Description:").grid(sticky=W,padx=10)

descVar=StringVar();

Entry(root,textvariable=descVar,width=20).grid(sticky=W,padx=10)

f=PanedWindow(orient=HORIZONTAL)

f.grid(padx=10)

btnAdd=Button(f,text="ADD",width=10,command=btnAdd\_click)

f.add(btnAdd)

btnaddVar=StringVar();

btnEdit=Button(f,text="EDIT",width=10,textvariable=btnaddVar,command=btnEdit\_click)

btnaddVar.set("Edit")

f.add(btnEdit)

btnDelete=Button(f,text="DELETE",width=10,command=btnDelete\_click)

f.add(btnDelete)

btnSearch=Button(f,text="SEARCH",width=10,command=btnSearch\_click)

f.add(btnSearch)

#==========design detail============

tree=ttk.Treeview(root)

tree["column"]=("Name","Description")

tree.column("#0",width=100, minwidth=100, stretch=NO)

tree.column("#1",width=100, minwidth=100, stretch=NO)

tree.column("#2",width=100, minwidth=100, stretch=NO)

tree.heading("#0",text="No",anchor=CENTER)

tree.heading("#1",text="Name",anchor=CENTER)

tree.heading("#2",text="Description",anchor=CENTER)

#==========add rows=============

tree.insert("",END,text="1",value=("Phnom Penh",""))

tree.insert("",END,text="2",value=("Phnom Penh",""))

tree.grid()

root.mainloop()

\*\*\*\*\*\*នៅមាន widget(control) ជាច្រើនទៀតដែលអ្នកត្រូវរៀនជាមួយលោកគ្រូអ្នកគ្រូដូចជា ttk.OptionMenu, ttk.Notebook, ttk.LabeledScale, ttk.Separator, ttk.Progressbar, ttk.Style, ttk.Widget.........។

## GUI-PyQt Programming

PyQt គឺជា standard GUI library ដែលបានបង្កើតឡើងដោយក្រុមហ៊ុន QT ដែលអាចRunបាន

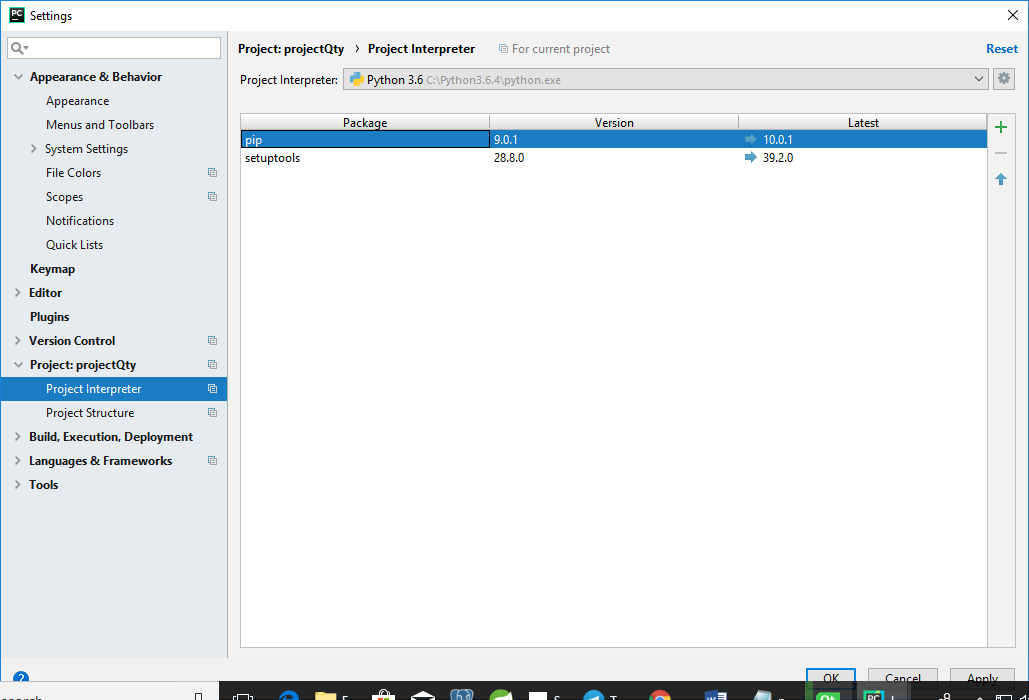
ជាមួយភាសា Python ក្នុងការបង្កើត GUI application។ PyQt ជា framework ដែលអាចដំណើរការបានសឹងគ្រប់ប្រភេទ platform ដូចជា Windows, OS X, Linux, iOS and Android។ វាជា open source ដែល PyQt5 វាsupport ជាមួយ Qt framework version 5 ហើយ PyQt4 supports ជាមួយ Qt v4។

ក្នុងការ develop GUI application ជាមួយនឹង PyQt កាន់តែពិសេសនោះគឺថា វាមានកម្មវិធី IDE ឈ្មោះ QT Designer ដែលអាចជួយសំរួលដល់លោកអ្នកក្នុងការ design inferface។

ដើម្បីប្រើ PyQt អ្នកត្រូវទៅតំឡើង software ពីរនេះបន្ថែមទៀតគឺ៖

* PyQt for window(use google = “PyQt5-5.6-gpl-Py3.5-Qt5.6.0-x32-2.exe”)

**ចំណាំ៖**

* **បើទីតាំងរបស់ Pythonគឺ៖ C:\Python3.6.4**
* **នោះទីតាំងរបស់ PyQt5គឺ៖ C:\Python3.6.4\python.exe 🡪 បន្ទាប់មកវានឹង install soft ចូលទៅទីតាំង C:\Python3.6.4\Lib\site-packages\PyQt5**
* **បន្ទាប់ពីដំឡើងចប់ទីតាំងរបស់ QT Designer គឺ៖ C:\Python3.6.4\Lib\site-packages\PyQt5\designer.exe**
* **បើអ្នកមិនអាច import PyQt5 បានសូមចូលទៅកាន់ File🡪Setting**

**របៀបបង្កើត Window Form-PyQt:**

import sys

from PyQt5 import QtWidgets

def window():

app=QtWidgets.QApplication(sys.argv)

w=QtWidgets.QWidget()

w=w.setWindowTitle("PyQt5 lession 1")

w.show()

sys.exit(app.exec\_())

window()

**របៀបប្រើប្រាស់ Qt-Desinger**:

បើកកម្មវិធី C:\Python3.6.4\Lib\site-packages\PyQt5\designer.exe

