

មាតិកា

មេរៀនទី ១:	Introduction to Python	1
1.1.	Overview.....	1
1.2	Environment Setup.....	3
មេរៀនទី ២:	Python-Basic Syntax.....	7
2.1.	Basic Syntax	7
2.2.	Variable Types	9
2.3.	Basic Operators	38
មេរៀនទី ៣	Python-Control Flow Statement	44
3.1.	Decision Making or Condition	44
3.2.	Loop Statement	46
មេរៀនទី ៤	Functions.....	49
4.1.	Basic Functions	49
4.2.	Scope of Variable.....	53
មេរៀនទី ៥:	Modules	56
5.1.	Build-In Module	56
5.2.	Define Module	56
មេរៀនទី ៦:	FILE I/O.....	60
6.1.	Opening and Closing File	60
6.2.	Reading and Writing Files.....	64
6.3.	Creating, Renaming and Deleting Files.....	66
6.4.	Directory in Python.....	67
6.5.	File and Directory Related Methods.....	67
មេរៀនទី ៧:	ERRORS AND EXCEPTIONS.....	74
7.1.	Exception Handling.....	74
7.2.	Debugging Tools	81
7.3.	Logging Module	82
7.4.	Python Test.....	86
មេរៀនទី ៨:	Python - Object Oriented.....	94
8.1.	សេចក្តីណែនាំ Introduction to OOP	94
8.2	ការបង្កើត Class and Object	95
8.3.	ការបង្កើត Encapsulation(Data Hiding).....	99
8.4.	ការបង្កើត Inheritance	102
8.5.	ការបង្កើត Polymorphism	103
8.6.	Regular Expressions.....	106
មេរៀនទី ៩:	Python – GUI Programming.....	119

9.1. សេចក្តីណែនាំ Introduction to GUI.....	119
9.2. GUI-Tkinter Programming.....	119
9.3. GUI-PyQt Programming	136

មេរៀនទី ១: Introduction to Python

នៅក្នុងមេរៀននេះ យើងនឹងពន្យល់ពីការចាប់ផ្តើមជាមួយភាសា Python ដោយមានការណែនាំជាមួយ កម្មវិធីដ៏សាមញ្ញជាច្រើន។

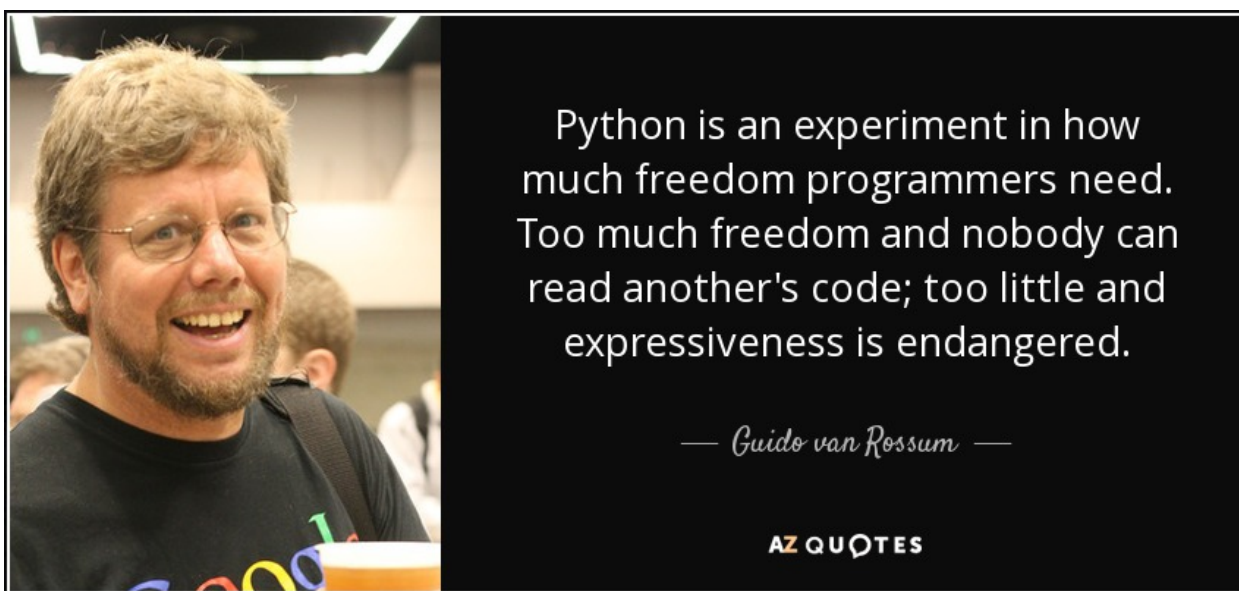
នៅក្នុងមេរៀននេះនឹងបង្ហាញអោយអ្នកមានសមត្ថភាពដូចខាងក្រោម:

- ប្រើប្រាស់ទំរង់ទូទៅ និងរចនាសម្ព័ន្ធនៃភាសា Python
- កំណត់អំពី console input/output (I/O)
- របៀបកំណត់កម្មវិធី Python
- OOP Concept, File IO, GUI Programming

1.1.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 នៅ National 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.

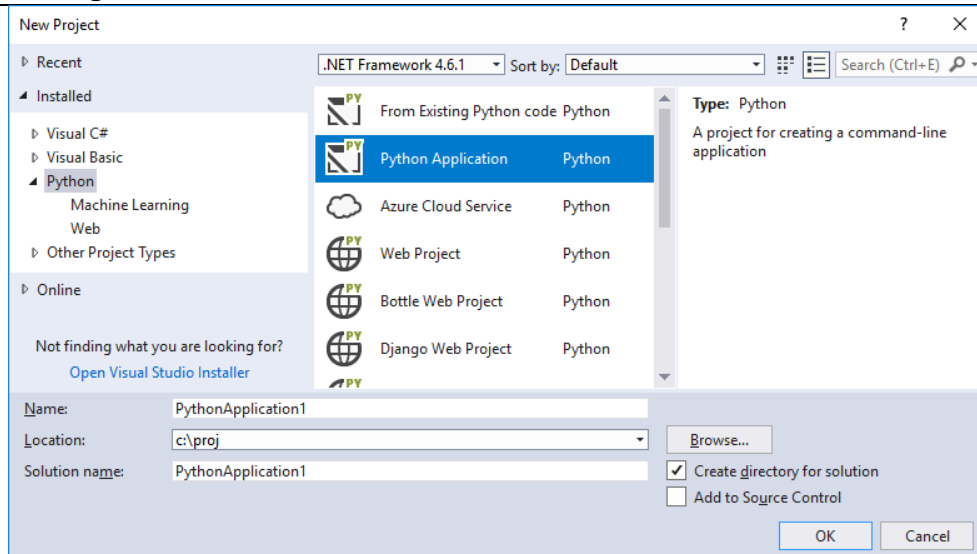
Released Version របស់ Python៖

- 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

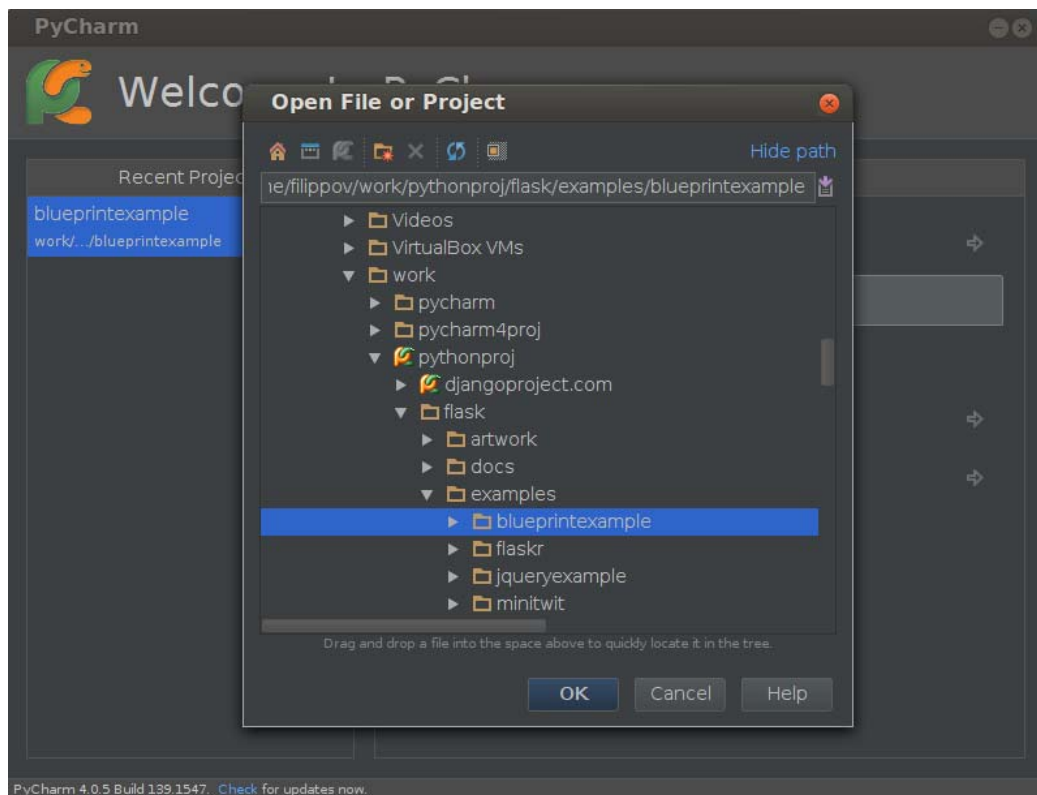
- Python 0.9.8 - January 9, 1993
- Python 0.9.9 - July 29, 1993
- Python 1.0 - January 1994
- Python 1.2 - April 10, 1995
- Python 1.3 - October 12, 1995
- Python 1.4 - October 25, 1996
- Python 1.5 - December 31, 1997
- Python 1.6 - September 5, 2000
- Python 2.0 - October 16, 2000
- Python 2.1 - April 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

1.2.Environment Setup

យើងអាចតម្លើងកម្មវិធី Python លើ platform ផ្សេងៗដូចជា Window, Linux, Mac OS ជាដើម។ បច្ចុប្បន្ននេះលោកអ្នកអាចសរសេរ code Python ជាមួយនឹង Microsoft Visual Studio IDE ក៏បាន ឬក៏ Pycharm IDE(professional is license, community is free) ឬ Netbean IDE ឬ Qt Creator IDE ក៏បាន...។



Visual Studio IDE



PyCharm IDE

លោកអ្នកអាច Download Python & PyCharm តាម link ខាងក្រោម៖

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

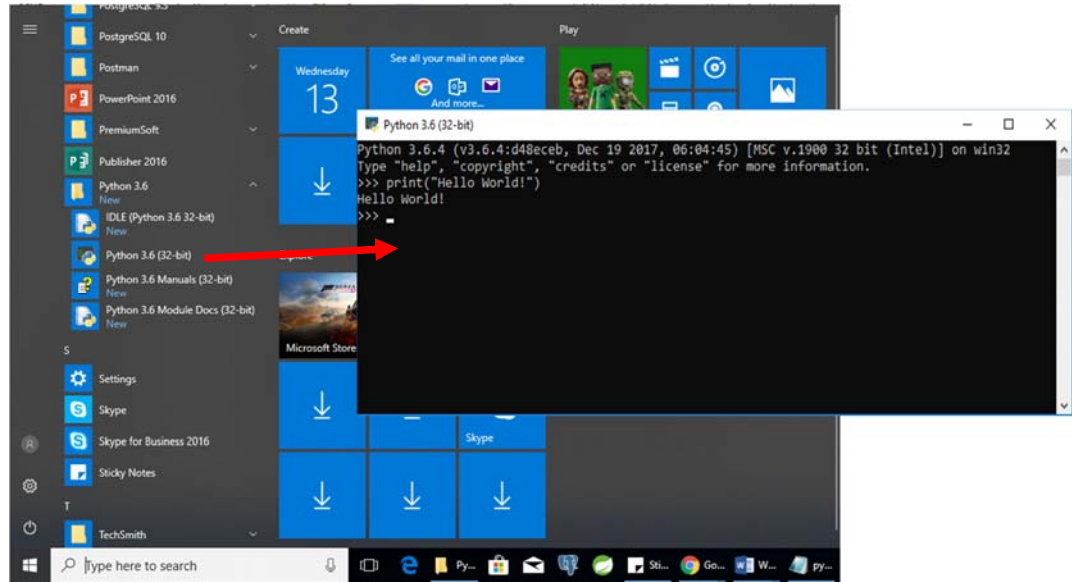
PyCharm : <https://www.jetbrains.com/pycharm/download/#section=windows>

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

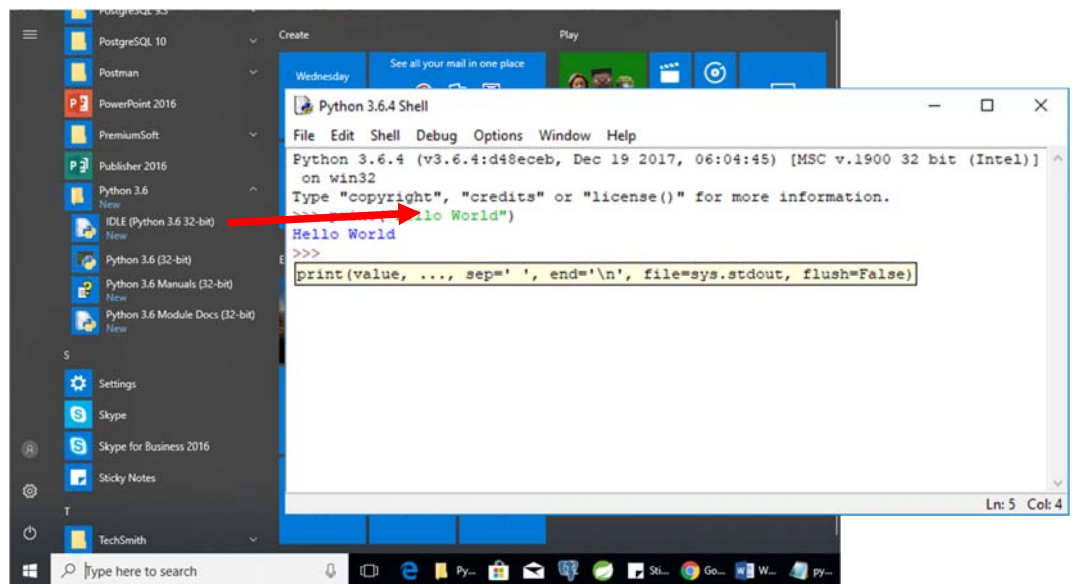
Installing Python & PyCharm :

<https://www.youtube.com/watch?v=puBXxzcWIIQ>

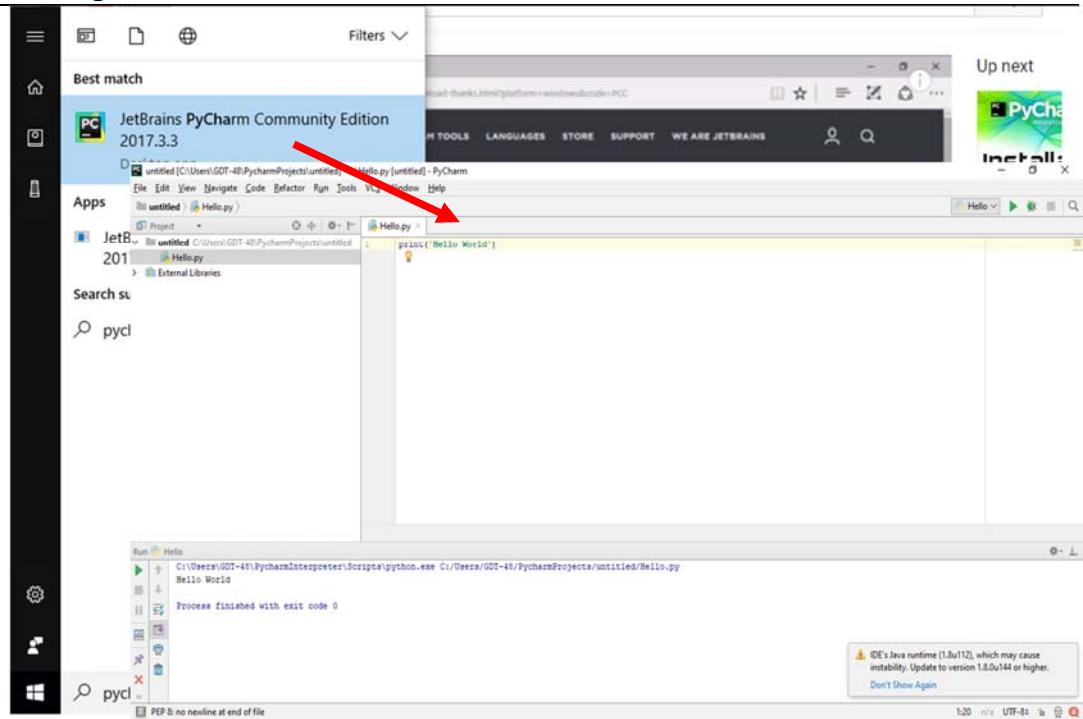
របៀបដំណើរការកម្មវិធី Python



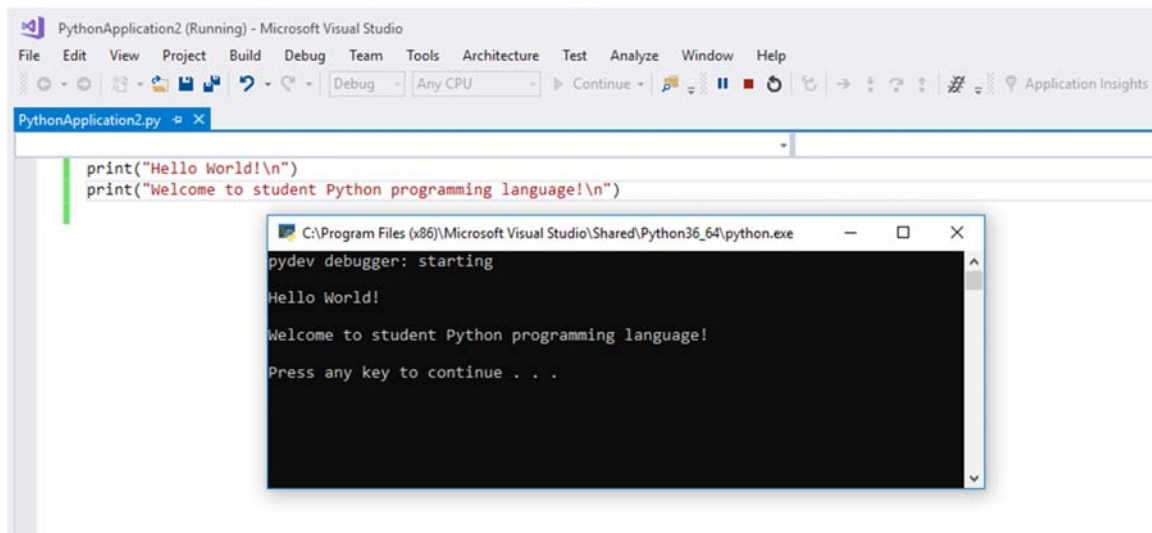
៩- Python-Console



២- Python-Shell



ទី ៣- PyCharm IDE



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

មេរៀនទី ២: Python-Basic Syntax

នៅក្នុងមេរៀននេះ នឹងបង្ហាញអោយអ្នកមានសមត្ថភាពដូចខាងក្រោម:

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

2.1. Basic Syntax

The screenshot shows the PyCharm IDE with a file named 'Hello.py'. The code in the editor is as follows:

```

1 #Comment of Header File
2 import random
3 #Comment of Variables
4 guesses_made = 0
5 name = 'Bopha'
6 number = random.randint(1, 20)
7 print('Well, (0), I am thinking of a number between 1 and 20.'.format(name))
8
9 while guesses_made < 6:
10     guess = 1
11     guesses_made += 1
12     if guess < number:
13         print('Your guess is too low.')
14     if guess > number:
15         print('Your guess is too high.')
16     if guess == number:
17         break
18 if guess == number:
19     print('Good job, (0)! You guessed my number in (1) guesses!'.format(name, guesses_made))
20 else:
21     print('Nope. The number I was thinking of was (0)'.format(number))

```

The Run window at the bottom shows the output of the program:

```

Well, Bopha, I am thinking of a number between 1 and 20.
Your guess is too low.
Your guess is too low.
Your guess is too low.
Your guess is too low.
Your guess is too low.
Your guess is too low.
Nope. The number I was thinking of was 2
Process finished with exit code 0

```

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

#Single Line Comment

```
#####
```

Multi-line

Comment in

Python Programming Language.

```
#####
```

Using Quotation : 'single quote' ឬ "double-quote" ត្រូវបានគេប្រើប្រាស់ដើម្បីផ្ទុកនូវតួអក្សរ។ បង្ហាញ single quote ឯ "" សម្រាប់បង្ហាញ double-quote។

Using Statement : គឺជារបៀបនៃការសរសេរបញ្ជានៅក្នុង 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 : ជាពាក្យ keyword Python ប្រើប្រាស់ដើម្បីបញ្ជា 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);
```

Conversion	Meaning
d	Signed integer decimal.
i	Signed integer decimal.
o	Unsigned octal.
u	Unsigned decimal.
x	Unsigned hexadecimal (lowercase).
X	Unsigned hexadecimal (uppercase).
e	Floating point exponential format (lowercase).
E	Floating point exponential format (uppercase).
f	Floating point decimal format.
F	Floating point decimal format.
g	Same as "e" if exponent is greater than -4 or less than precision, "f" otherwise.
G	Same as "E" if exponent is greater than -4 or less than precision, "F" otherwise.
c	Single character (accepts integer or single character string).
r	String (converts any python object using repr()).
s	String (converts any python object using str()).
%	No argument is converted, results in a "%" character in the result.

Exit App : ដើម្បីបិទប្រព័ន្ធចេញពីកម្មវិធីគេប្រើ function exit() ។

2.2.Variable Types

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

ច្បាប់ក្នុងការដាក់ឈ្មោះ variable

- ហាមផ្ដើមដោយលេខ
- ហាមផ្ដើមដកឃ្លា
- ហាមប្រើជាមួយនឹងសញ្ញាពិសេសផ្សេងៗ ដូចជា៖ # ? ! & | % @.....
- ហាមផ្ដើមជាន់ keyword

ទី ១ :

```
guesses_made = 0
```

ទី ២ :

```
age=int(input())
```

ទី ៣ :

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

Note: function **type(varName)** ប្រើសម្រាប់បង្ហាញប្រភេទនៃ object ។

Python Casting : ប្រើដើម្បីបំប្លែងពីប្រភេទទិន្នន័យមួយទៅជាប្រភេទទិន្នន័យមួយផ្សេងទៀត។

int(x) : បំប្លែង x ទៅជា integer

long(x) : បំប្លែង x ទៅជា long integer

float(x) : បំប្លែង x ទៅជា float number

str(x) : បំប្លែង 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	<u>abs(x)</u> The absolute value of x: the (positive) distance between x and zero.
2	<u>ceil(x)</u> The ceiling of x: the smallest integer not less than x
3	<u>cmp(x, y)</u> -1 if x < y, 0 if x == y, or 1 if x > y : compare function
4	<u>exp(x)</u> The exponential of x: e^x
5	<u>fabs(x)</u> The absolute value of x. return as float
6	<u>floor(x)</u> The floor of x: the largest integer not greater than x
7	<u>log(x)</u> The natural logarithm of x, for $x > 0$
8	<u>log10(x)</u> The base-10 logarithm of x for $x > 0$.
9	<u>max(x1, x2,...)</u> The largest of its arguments: the value closest to positive infinity
10	<u>min(x1, x2,...)</u>

	The smallest of its arguments: the value closest to negative infinity
11	<u>modf(x)</u> 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	<u>pow(x, y)</u> The value of x**y.
13	<u>round(x [,n])</u> 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	<u>sqrt(x)</u> 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	<u>choice(seq)</u> A random item from a list, tuple, or string. print("Radom : ",random.choice((1,2,3)))
2	<u>randrange ([start,] stop [,step])</u> A randomly selected element from range(start, stop, step)
3	<u>random()</u> A random float r, such that 0 is less than or equal to r and r is less than 1
4	<u>seed([x])</u> Sets the integer starting value used in generating random numbers. Call this function before calling any other random module function. Returns None.
5	<u>shuffle(lst)</u> Randomizes the items of a list in place. Returns None.
6	<u>uniform(x, y)</u>

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	<u>acos(x)</u> Return the arc cosine of x, in radians.
2	<u>asin(x)</u> Return the arc sine of x, in radians.
3	<u>atan(x)</u> Return the arc tangent of x, in radians.
4	<u>atan2(y, x)</u> Return atan(y / x), in radians.
5	<u>cos(x)</u> Return the cosine of x radians.
6	<u>hypot(x, y)</u> Return the Euclidean norm, sqrt(x*x + y*y).
7	<u>sin(x)</u> Return the sine of x radians.
8	<u>tan(x)</u> Return the tangent of x radians.
9	<u>degrees(x)</u> Converts angle x from radians to degrees.
10	<u>radians(x)</u> 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"ស្នំស្នំលោកគ្រូ")
```

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 integer
%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	<u>capitalize()</u> Capitalizes first letter of string
2	<u>center(width, fillchar)</u> Returns a space-padded string with the original string centered to a total of width columns.
3	<u>count(str, beg= 0,end=len(string))</u> 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	<u>decode(encoding='UTF-8',errors='strict')</u> Decodes the string using the codec registered for encoding. encoding defaults to the default string encoding.
5	<u>encode(encoding='UTF-8',errors='strict')</u> Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'.
6	<u>endswith(suffix, beg=0, end=len(string))</u> 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	<u>expandtabs(tabsize=8)</u> Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided.
8	<u>find(str, beg=0 end=len(string))</u> 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	<u>index(str, beg=0, end=len(string))</u> Same as find(), but raises an exception if str not found.
10	<u>isalnum()</u>

	Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.
11	<u>isalpha()</u> Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.
12	<u>isdigit()</u> Returns true if string contains only digits and false otherwise.
13	<u>islower()</u> Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.
14	<u>isnumeric()</u> Returns true if a unicode string contains only numeric characters and false otherwise.
15	<u>isspace()</u> Returns true if string contains only whitespace characters and false otherwise.
16	<u>istitle()</u> Returns true if string is properly "titlecased" and false otherwise.
17	<u>isupper()</u> Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise.
18	<u>join(seq)</u> Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string.
19	<u>len(string)</u> Returns the length of the string
20	<u>ljust(width[, fillchar])</u> Returns a space-padded string with the original string left-justified to a total of width columns.
21	<u>lower()</u> Converts all uppercase letters in string to lowercase.
22	<u>lstrip()</u> Removes all leading whitespace in string.

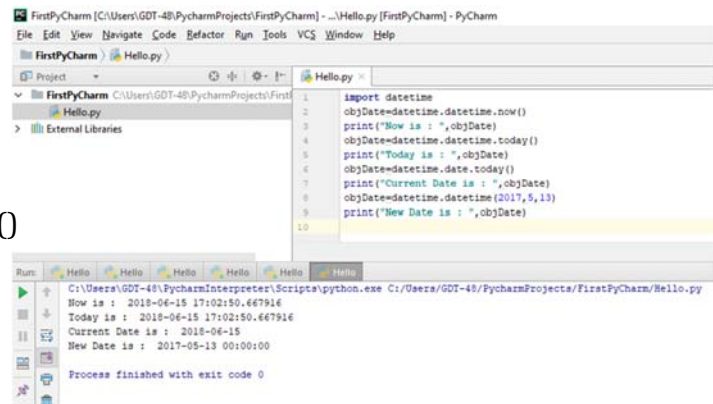
23	<u>maketrans()</u> Returns a translation table to be used in translate function.
24	<u>max(str)</u> Returns the max alphabetical character from the string str.
25	<u>min(str)</u> Returns the min alphabetical character from the string str.
26	<u>replace(old, new [, max])</u> Replaces all occurrences of old in string with new or at most max occurrences if max given.
27	<u>rfind(str, beg=0, end=len(string))</u> Same as find(), but search backwards in string.
28	<u>rindex(str, beg=0, end=len(string))</u> Same as index(), but search backwards in string.
29	<u>rjust(width[, fillchar])</u> Returns a space-padded string with the original string right-justified to a total of width columns.
30	<u>rstrip()</u> Removes all trailing whitespace of string.
31	<u>split(str="", num=string.count(str))</u> Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given.
32	<u>splitlines(num=string.count('\n'))</u> Splits string at all (or num) NEWLINES and returns a list of each line with NEWLINES removed.
33	<u>startswith(str, beg=0, end=len(string))</u> 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	<u>strip([chars])</u> Performs both lstrip() and rstrip() on string.
35	<u>swapcase()</u> Inverts case for all letters in string.
36	<u>title()</u>

	Returns "titlecased" version of string, that is, all words begin with uppercase and the rest are lowercase.
37	<u>translate(table, deletechars="")</u> Translates string according to translation table str(256 chars), removing those in the del string.
38	<u>upper()</u> Converts lowercase letters in string to uppercase.
39	<u>zfill (width)</u> Returns original string leftpadded with zeros to a total of width characters; intended for numbers, zfill() retains any sign given (less one zero).
40	<u>isdecimal()</u> Returns true if a unicode string contains only decimal characters and false otherwise.

Python Dates & Time: ក្នុងភាសាបស់ Python Date & Time គឺមិនមែនជា data type នោះទេ យើងអាចប្រើប្រាស់វាបានតាមមធ្យោបាយច្រើនផ្សេងៗគ្នា។

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))
```

```

1 import time
2 objTime=time.time()
3 print("Now in unit of second from 01-01-1970 12:00am is :",objTime)
4 objTime=time.localtime(time.time())
5 print("Current structure time is :",objTime)
6 print("Accending formatting time is : ",time.asctime(objTime))

```

Run: C:\Users\GDT-48\PycharmProjects\FirstPyCharm\Hello.py
 Now in unit of second from 01-01-1970 12:00am is : 1525058289.230614
 Current structure time is : time.struct_time(tm_year=2018, tm_mon=6, tm_mday=15, tm_hour=17, tm_min=24, tm_sec=49, tm_wday=4, tm_yday=166, tm_isdst=0)
 Accending formatting time is : Fri Jun 15 17:24:49 2018
 Process finished with exit code 0

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"))
```

```
-----Date Time Formating-----  
2018-08-08  
08-08-2018  
01-08-2019 11:10  
2018-08-08 09:22:04.404153  
2018-08-08 09:22  
09:22
```

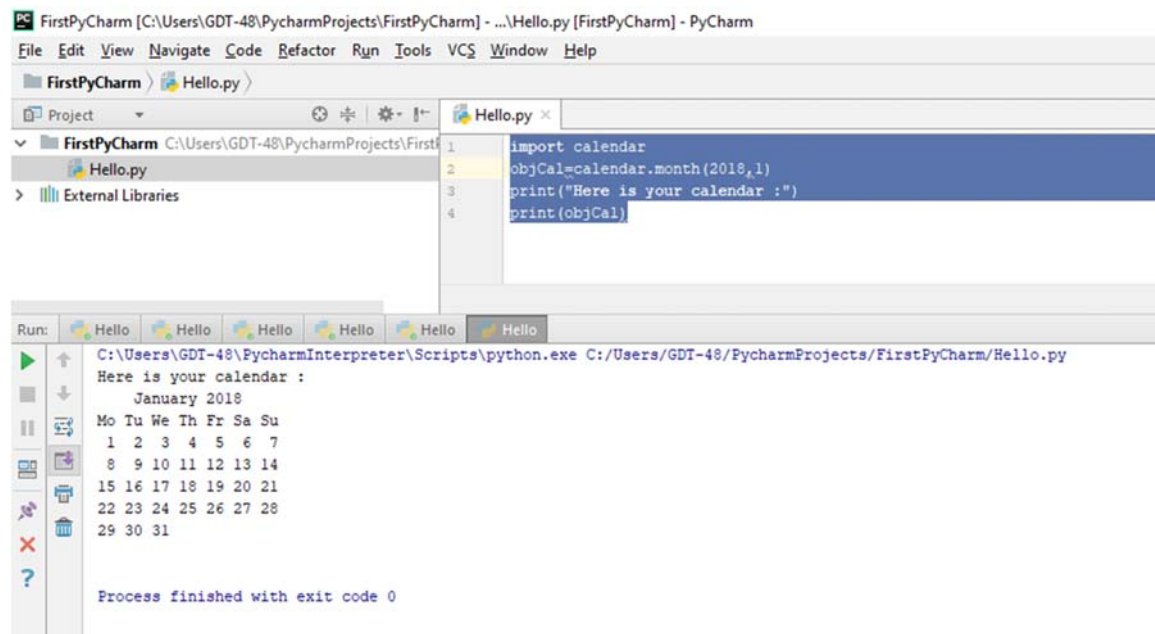
Using calendar class:

```
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	<p><u>time.altzone</u></p> <p>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.</p>
2	<p><u>time.asctime([tupletime])</u></p> <p>Accepts a time-tuple and returns a readable 24-character string such as 'Tue Dec 11 18:07:14 2008'.</p>
3	<p><u>time.clock()</u></p> <p>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().</p>
4	<p><u>time.ctime([secs])</u></p> <p>Like asctime(localtime(secs)) and without arguments is like asctime()</p>
5	<p><u>time.gmtime([secs])</u></p> <p>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</p>
6	<p><u>time.localtime([secs])</u></p> <p>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).</p>
7	<p><u>time.mktime(tupletime)</u></p> <p>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.</p>
8	<p><u>time.sleep(secs)</u></p> <p>Suspends the calling thread for secs seconds.</p>
9	<p><u>time.strftime(fmt[,tupletime])</u></p> <p>Accepts an instant expressed as a time-tuple in local time and returns a string representing the instant as specified by string fmt.</p>
10	<p><u>time.strptime(str,fmt='%a %b %d %H:%M:%S %Y')</u></p>

	Parses str according to format string fmt and returns the instant in time-tuple format.
11	<u>time.time()</u> Returns the current time instant, a floating-point number of seconds since the epoch.
12	<u>time.tzset()</u> 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.

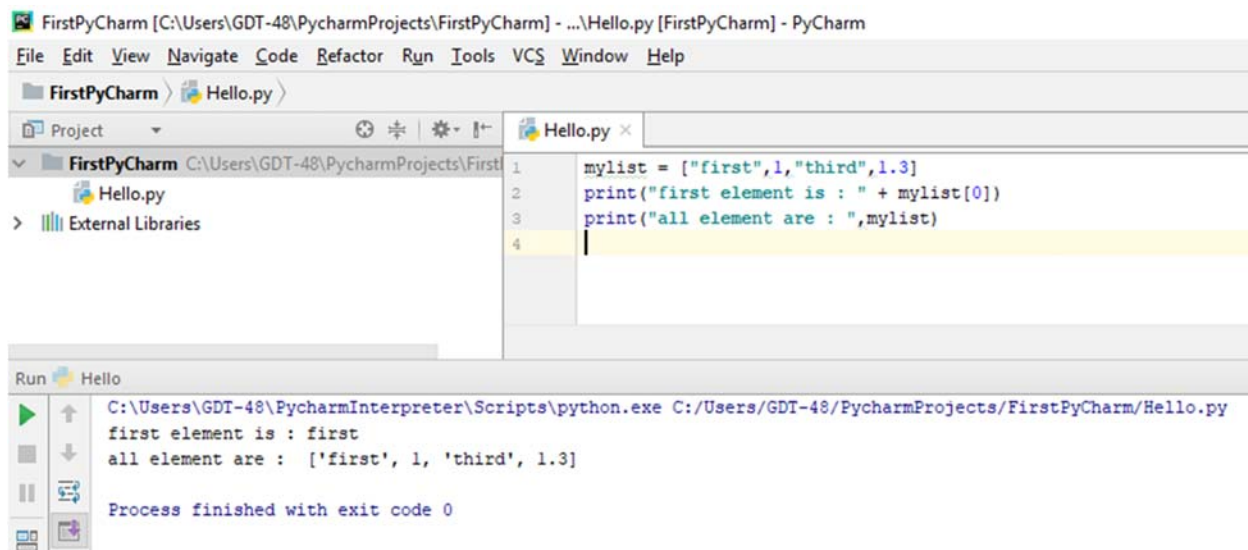
4	calendar.leapdays(y1,y2) Returns the total number of leap days in the years within range(y1,y2).
5	calendar.month(year,month,w=2,l=1) Returns a multiline string with a calendar for month month of year year, one line per week plus two header lines. w is the width in characters of each date; each line 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 ដែលមានលំដាប់អាចផ្លាស់ប្តូរ 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	<u>cmp(list1, list2)</u> Compares elements of both lists.
2	<u>len(list)</u> Gives the total length of the list.
3	<u>max(list)</u> Returns item from the list with max value.
4	<u>min(list)</u> Returns item from the list with min value.
5	<u>list(seq)</u> Converts a tuple into list.
Sr.No.	Methods with Description
1	<u>list.append(obj)</u> Appends object obj to list
2	<u>list.count(obj)</u> Returns count of how many times obj occurs in list
3	<u>list.extend(seq)</u> Appends the contents of seq to list
4	<u>list.index(obj)</u> Returns the lowest index in list that obj appears
5	<u>list.insert(index, obj)</u> Inserts object obj into list at offset index
6	<u>list.pop(obj=list[-1])</u> Removes and returns last object or obj from list
7	<u>list.remove(obj)</u> Removes object obj from list
8	<u>list.reverse()</u> Reverses objects of list in place
9	<u>list.sort([func])</u> 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	<u>cmp(tuple1, tuple2)</u> Compares elements of both tuples.
2	<u>len(tuple)</u> Gives the total length of the tuple.
3	<u>max(tuple)</u> Returns item from the tuple with max value.
4	<u>min(tuple)</u> Returns item from the tuple with min value.
5	<u>tuple(seq)</u> Converts a list into tuple.

Python Dictionaries: គឺជាប្រភេទមួយនៃ collection ដែល unordered, changeable and indexed។ វាត្រូវបានគេសរសេរនៅក្នុងសញ្ញា { } ហើយធាតុនីមួយៗរបស់វាត្រូវបានកាត់ផ្តាច់ដោយសញ្ញា , ដែលធាតុនីមួយៗនោះត្រូវមានតម្លៃជាគូគឺ key & value ដែល key អាចជា strings, numbers, ឬ tuples តែត្រូវតែ unique រីឯ value អាចជាប្រភេទផ្សេងៗ។

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,"=",v)
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	<u>cmp(dict1, dict2)</u> Compares elements of both dict.
2	<u>len(dict)</u> Gives the total length of the dictionary. This would be equal to the number of items in the dictionary.
3	<u>str(dict)</u> Produces a printable string representation of a dictionary
4	<u>type(variable)</u> 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	<u>dict.clear()</u> Removes all elements of dictionary <i>dict</i>
2	<u>dict.copy()</u> Returns a shallow copy of dictionary <i>dict</i>
3	<u>dict.fromkeys()</u> Create a new dictionary with keys from seq and values <i>set</i> to <i>value</i> .
4	<u>dict.get(key, default=None)</u> For <i>key</i> key, returns value or default if key not in dictionary
5	<u>dict.has_key(key)</u> Returns <i>true</i> if key in dictionary <i>dict</i> , <i>false</i> otherwise
6	<u>dict.items()</u> Returns a list of <i>dict</i> 's (key, value) tuple pairs
7	<u>dict.keys()</u> Returns list of dictionary <i>dict</i> 's keys
8	<u>dict.setdefault(key, default=None)</u> Similar to <i>get()</i> , but will set <i>dict[key]=default</i> if <i>key</i> is not already in <i>dict</i>
9	<u>dict.update(dict2)</u> Adds dictionary <i>dict2</i> 's key-values pairs to <i>dict</i>
10	<u>dict.values()</u> Returns list of dictionary <i>dict</i> 's values

Python Sets: គឺជាប្រភេទមួយនៃ collection ដែល 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)
```

2.3.Basic Operators

Operators : គឺជាពណ្ញក 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 / a /= 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

-----2-----

$a \& b = 0000\ 1100$

$a | b = 0011\ 1101$

$a \wedge b = 0011\ 0001$

$\sim a = 1100\ 0011$

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) :គេប្រើវាដើម្បីចងក្លាប់conditionបញ្ចូលគ្នា

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 នៅក្នុង sequence(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 find 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 : សញ្ញាគណនាអតិភាព(ខាងលើអតិភាពមុន)

Sr.No.	Operator & Description
1	<p>**</p> <p>Exponentiation (raise to the power) ស្វ័យគុណ</p>
2	<p>~ + -</p> <p>Complement, unary plus and minus (method names for the last two are +@ and -@)</p>
3	<p>* / % //</p>

	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) និងលក្ខខណ្ឌមិនពិត (FALSE) ។

នៅក្នុងមេរៀននេះ នឹងបង្ហាញអោយអ្នកមានសមត្ថភាពដូចខាងក្រោម៖

- ➔ ការប្រើប្រាស់ Operator ផ្សេងៗដើម្បីធ្វើការគណនាលើ content
- ➔ កំណត់ការប្រើប្រាស់លក្ខខណ្ឌ if else Statements
- ➔ កំណត់ការប្រើប្រាស់ continue, break, exit()
- ➔ កំណត់ការប្រើប្រាស់ Loop (while, for)

3.1.Decision Making or Condition

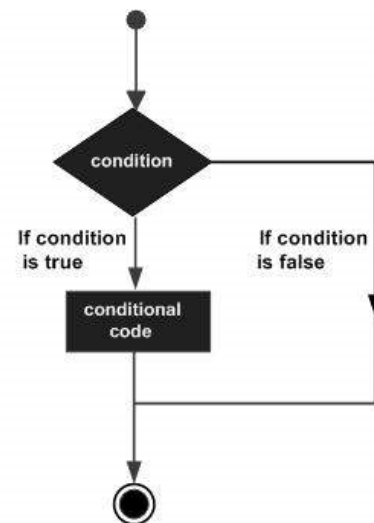
Condition Statement គឺជា structure ដែលត្រូវបានគេប្រើប្រាស់សំរាប់បញ្ជា រឺប្រាប់ទៅ system ឲ្យធ្វើអ្វីមួយដោយមានលក្ខណៈ។

If នឹងត្រូវត្រួតពិនិត្យលក្ខខណ្ឌរបស់ condition

បើ condition ពិតនោះវាអនុវត្ត Statement(s)

របស់វា តែបើមិនពិតទេនោះ គឺវានឹងមិនអនុវត្ត

Statement(s) នោះទេ ។



Sr.No.	Statement & Description
1	<p><u>if statements</u></p> <p>An if statement consists of a boolean expression followed by one or more statements.</p>
2	<p><u>if...else statements</u></p> <p>An if statement can be followed by an optional else statement, which executes when the boolean expression is FALSE.</p>
3	<p><u>nested if statements</u></p> <p>You can use one if or else if statement inside another if or else if statement(s).</p>

ឧទាហរណ៍ទី១- Single Statement with Single 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 python's 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")
```

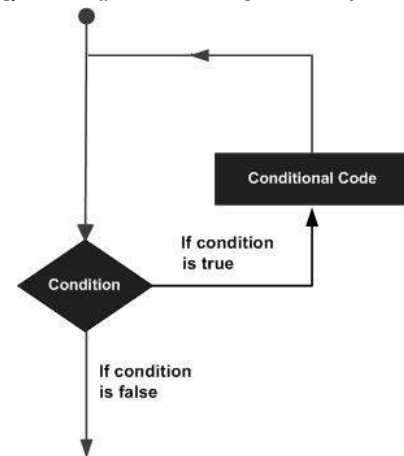
3.2. Loop Statement

នៅក្នុងការសរសេរកម្មវិធីដើម្បីដោះស្រាយបញ្ហាអ្វីមួយ ជួនកាលគេជួបប្រទះលក្ខខណ្ឌ មួយតែមួយ ដង ជួនកាលគេជួបប្រទះលក្ខខណ្ឌដដែលច្រើនដង ។ ហើយលក្ខខណ្ឌដដែលៗ ច្រើនដងនេះត្រូវបានគេ ហៅថា Loops ។ Loops គឺជាការធ្វើសកម្មភាពដដែលរហូតជួបលក្ខខណ្ឌ ណាមួយទើបវាបញ្ចប់សកម្មភាព របស់វា ។ នៅក្នុង Python Program មាន៖

- while loop
- for loop

Loop Control Statements

Sr.No.	Control Statement & Description
1	<p><u>break statement</u></p> <p>Terminates the loop statement and transfers execution to the statement immediately following the loop.</p>
2	<p><u>continue statement</u></p> <p>Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.</p>
3	<p><u>pass statement(do nothing)</u></p> <p>The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.</p>



The while Loop : វានៅតែ loop ប្រសិនបើលក្ខខណ្ឌរបស់វានៅតែពិត

```
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
```

```
␣
```

```
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)
```

```
␣
```

```
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)
```

```
␣
```

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

```
    print(x)
```

U

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

```
    print(x)
```

U

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** ។

4.1. 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)
```

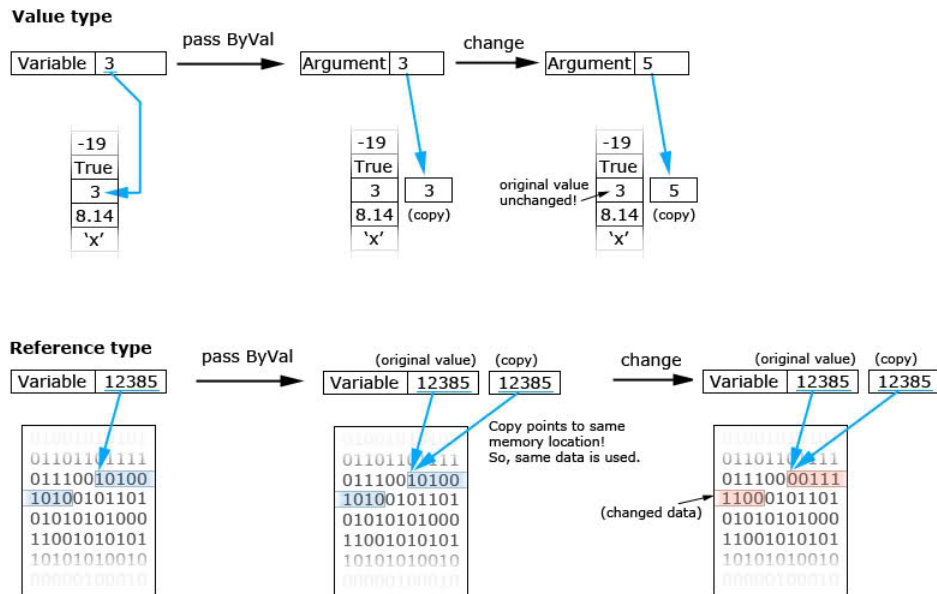
```
return;
```

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 ចំណែក ឯ List, Dictionary...គឺ Pass by reference(មានន័យថានៅពេលដែលគ្រប់ parametersរបស់functionមាន កែប្រែប្រួលនោះវានឹងផ្លាស់ប្តូរតម្លៃរបស់variable 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)
```

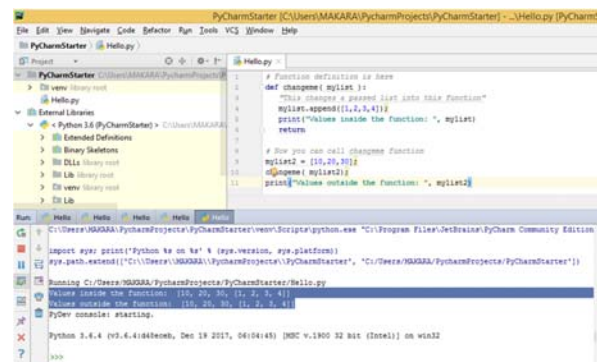
```
return
```

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 assign 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: ត្រូវផ្តល់តម្លៃ

```
# 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: គេប្រើសម្រាប់បោះពុម្ពចូលទៅកាន់ function លក្ខណជា tuple collection ។ វាជាឈ្មោះនៃ parameter ដែលផ្ដើមដោយសញ្ញា * ។

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)
```

4.2.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 ហើយត្រូវបានគេប្រើប្រាស់ក្នុង 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 ហើយត្រូវបានគេប្រើប្រាស់ក្នុង 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 ថាជាដូចជា code library ឬជាfile ដែលរួមមាននូវសំនុំ function ជាច្រើនដែលយើងអាចយកមកប្រើប្រាស់ក្នុង applicationរបស់យើងបាន។

នៅក្នុងមេរៀននេះ នឹងបង្ហាញអោយអ្នកមានសមត្ថភាពដូចខាងក្រោម:

- របៀបបង្កើត module
- របៀបប្រើប្រាស់ module

5.1.Build-In Module

គឺជា Module ដែលមានស្រាប់ លោកអ្នកអាចហៅមកប្រើប្រាស់បានតាមរយៈ import keyword។

```
import platform
import sys
x = platform.system()
print(x)
print(platform._sys_version())
sys.exit(0)
```

5.2.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 (វាមាននូវ 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.

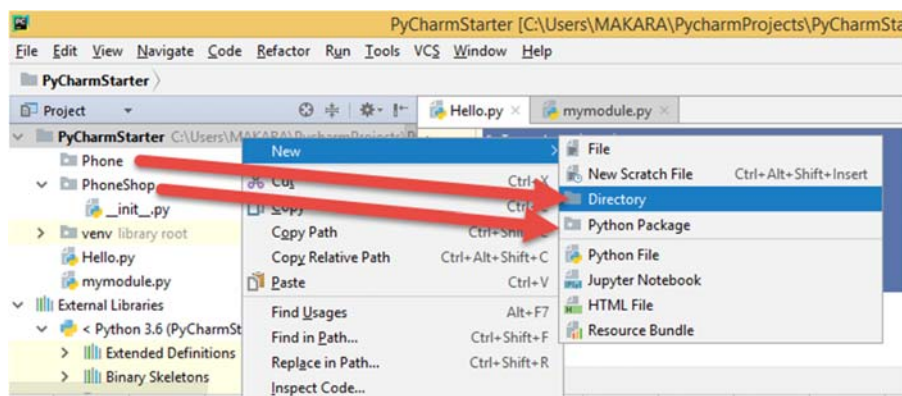
```
# Import built-in module math
import math
```

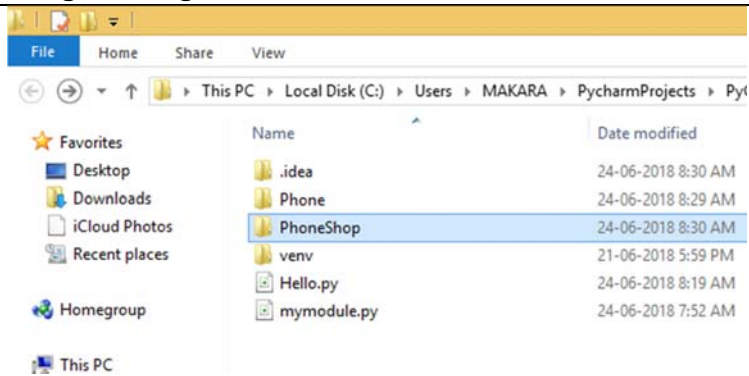
```
globalsDic=globals()
localsDic=locals()
print("Gloable")
print(globalsDic)
print("Local")
print(localsDic)
```

The reload() Function: ជាទូទៅរាល់ modules ដែលបាន import នៅខាងលើគឺវា execute បានតែម្តងទេ តែយើងអាចបញ្ជាឱ្យវា 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)

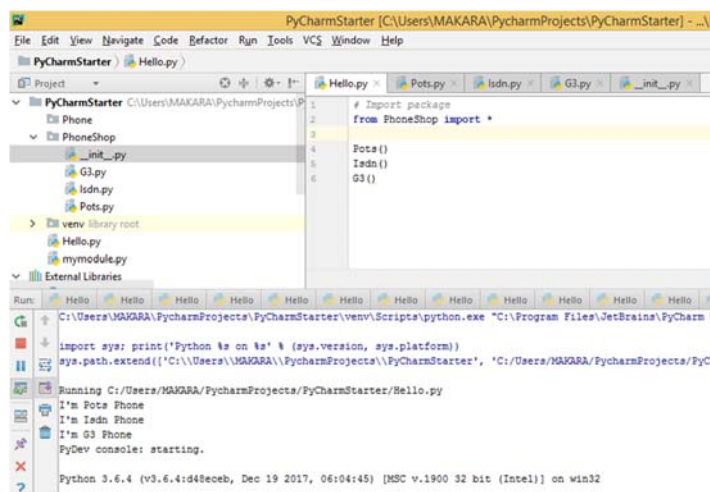
Import package

from PhoneShop import *

Pots()

Isdn()

G3()



មេរៀនទី ៦:**FILE I/O**

នៅក្នុងមេរៀននេះ នឹងបង្ហាញអោយអ្នកមានសមត្ថភាពដូចខាងក្រោម:

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

6.1.Opening and Closing File

ជាតំបូងអ្នកត្រូវស្វែងយល់ពីរបៀបក្នុងការបំប្លែងទិន្នន័យរវាង stringនិងarray of bytesជាមុនសិន៖

+ 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("afsfsfសសសសសសសសស")
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[, keepsends]])

- readlines([sizehint[, keepends]])
- reset()
- codecs.StreamReaderWriter(stream, Reader, Writer, errors='strict')
-

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

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 ផ្លាស់ 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 opened file
fo.close()
```

6.2. 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 opened 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 ពីចំណុចចាប់ផ្តើមដល់ចប់ តែបើមាននោះវានឹង read ពីកន្លែងចាប់ផ្តើមហើយយកគ្រប់ចំនួនbytesដែលបានកំណត់។

```
# Open a file
fo = open("foo.txt", "r+")
str = fo.read(10);

print ("Read String is : ", str)
# Close opened 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 opened file
fo.close()

```

The readline() method: read 1 line in file

```

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

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

f = open("demofile.txt", "r")
for x in f:
    print(x)

```

6.3.Creating, Renaming and Deleting Files

Create a New File: ប្រើ 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")
f = open("myfile.txt", "w")

```

The rename() Method: សម្រាប់ rename ឯកសារ: 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: ဝှံ့ file

Syntax: os.remove(file_name)

```
import os
```

```
# Delete file test2.txt
```

```
os.remove("text2.txt")
```

၁

```
import os
```

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

```
    os.remove("demofile.txt")
```

```
else:
```

```
    print("The file does not exists")
```

6.4.Directory in Python**Create Directory:** ဝှံ့ mkdir() Method

```
import os
```

```
# Create a directory "test"
```

```
os.mkdir("test")
```

Delete Folder: ဝှံ့ os.rmdir() method

```
import os
```

```
os.rmdir("myfolder")
```

၁

```
import os
```

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

```
os.rmdir( "/tmp/test" )
```

The getcwd() Method : ဝှံ့ 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")
```

6.5.File and Directory Related Methods

More on FILE object:

Sr.No.	Methods with Description
1	<u>file.close()</u> Close the file. A closed file cannot be read or written any more.
2	<u>file.flush()</u> Flush the internal buffer, like stdio's fflush. This may be a no-op on some file-like objects.
3	<u>file.fileno()</u> Returns the integer file descriptor that is used by the underlying implementation to request I/O operations from the operating system.
4	<u>file.isatty()</u> Returns True if the file is connected to a tty(-like) device, else False.
5	<u>file.next()</u> Returns the next line from the file each time it is being called.
6	<u>file.read([size])</u> Reads at most size bytes from the file (less if the read hits EOF before obtaining size bytes).
7	<u>file.readline([size])</u> Reads one entire line from the file. A trailing newline character is kept in the string.
8	<u>file.readlines([sizehint])</u> 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	<u>file.seek(offset[, whence])</u> Sets the file's current position
10	<u>file.tell()</u> Returns the file's current position
11	<u>file.truncate([size])</u> Truncates the file's size. If the optional size argument is present, the file is truncated to (at most) that size.
12	<u>file.write(str)</u> Writes a string to the file. There is no return value.

13	<p><u>file.writelines(sequence)</u></p> <p>Writes a sequence of strings to the file. The sequence can be any iterable object producing strings, typically a list of strings.</p>
----	---

More on OS object

Sr.No.	Methods with Description
1	<p><u>os.access(path, mode)</u></p> <p>Use the real uid/gid to test for access to path.</p>
2	<p><u>os.chdir(path)</u></p> <p>Change the current working directory to path</p>
3	<p><u>os.chflags(path, flags)</u></p> <p>Set the flags of path to the numeric flags.</p>
4	<p><u>os.chmod(path, mode)</u></p> <p>Change the mode of path to the numeric mode.</p>
5	<p><u>os.chown(path, uid, gid)</u></p> <p>Change the owner and group id of path to the numeric uid and gid.</p>
6	<p><u>os.chroot(path)</u></p> <p>Change the root directory of the current process to path.</p>
7	<p><u>os.close(fd)</u></p> <p>Close file descriptor fd.</p>
8	<p><u>os.closerange(fd_low, fd_high)</u></p> <p>Close all file descriptors from fd_low (inclusive) to fd_high (exclusive), ignoring errors.</p>
9	<p><u>os.dup(fd)</u></p> <p>Return a duplicate of file descriptor fd.</p>
10	<p><u>os.dup2(fd, fd2)</u></p> <p>Duplicate file descriptor fd to fd2, closing the latter first if necessary.</p>
11	<p><u>os.fchdir(fd)</u></p> <p>Change the current working directory to the directory represented by the file descriptor fd.</p>
12	<p><u>os.fchmod(fd, mode)</u></p>

	Change the mode of the file given by fd to the numeric mode.
13	<u>os.fchown(fd, uid, gid)</u> Change the owner and group id of the file given by fd to the numeric uid and gid.
14	<u>os.fdatasync(fd)</u> Force write of file with filedescriptor fd to disk.
15	<u>os.fdopen(fd[, mode[, bufsize]])</u> Return an open file object connected to the file descriptor fd.
16	<u>os.fpathconf(fd, name)</u> Return system configuration information relevant to an open file. name specifies the configuration value to retrieve.
17	<u>os.fstat(fd)</u> Return status for file descriptor fd, like stat().
18	<u>os.fstatvfs(fd)</u> Return information about the filesystem containing the file associated with file descriptor fd, like statvfs().
19	<u>os.fsync(fd)</u> Force write of file with filedescriptor fd to disk.
20	<u>os.ftruncate(fd, length)</u> Truncate the file corresponding to file descriptor fd, so that it is at most length bytes in size.
21	<u>os.getcwd()</u> Return a string representing the current working directory.
22	<u>os.getcwdu()</u> Return a Unicode object representing the current working directory.
23	<u>os.isatty(fd)</u> Return True if the file descriptor fd is open and connected to a tty(-like) device, else False.
24	<u>os.lchflags(path, flags)</u> Set the flags of path to the numeric flags, like chflags(), but do not follow symbolic links.
25	<u>os.lchmod(path, mode)</u> Change the mode of path to the numeric mode.
26	<u>os.lchown(path, uid, gid)</u> Change the owner and group id of path to the numeric uid and gid. This function will not follow symbolic

	links.
27	<u>os.link(src, dst)</u> Create a hard link pointing to src named dst.
28	<u>os.listdir(path)</u> Return a list containing the names of the entries in the directory given by path.
29	<u>os.lseek(fd, pos, how)</u> Set the current position of file descriptor fd to position pos, modified by how.
30	<u>os.lstat(path)</u> Like stat(), but do not follow symbolic links.
31	<u>os.major(device)</u> Extract the device major number from a raw device number.
32	<u>os.makedev(major, minor)</u> Compose a raw device number from the major and minor device numbers.
33	<u>os.makedirs(path[, mode])</u> Recursive directory creation function.
34	<u>os.minor(device)</u> Extract the device minor number from a raw device number.
35	<u>os.mkdir(path[, mode])</u> Create a directory named path with numeric mode mode.
36	<u>os.mkfifo(path[, mode])</u> Create a FIFO (a named pipe) named path with numeric mode mode. The default mode is 0666 (octal).
37	<u>os.mknod(filename[, mode=0600, device])</u> Create a filesystem node (file, device special file or named pipe) named filename.
38	<u>os.open(file, flags[, mode])</u> Open the file file and set various flags according to flags and possibly its mode according to mode.
39	<u>os.openpty()</u> Open a new pseudo-terminal pair. Return a pair of file descriptors (master, slave) for the pty and the tty, respectively.
40	<u>os.pathconf(path, name)</u>

	Return system configuration information relevant to a named file.
41	<u>os.pipe()</u> Create a pipe. Return a pair of file descriptors (r, w) usable for reading and writing, respectively.
42	<u>os.popen(command[, mode[, bufsize]])</u> Open a pipe to or from command.
43	<u>os.read(fd, n)</u> 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	<u>os.readlink(path)</u> Return a string representing the path to which the symbolic link points.
45	<u>os.remove(path)</u> Remove the file path.
46	<u>os.removedirs(path)</u> Remove directories recursively.
47	<u>os.rename(src, dst)</u> Rename the file or directory src to dst.
48	<u>os.renames(old, new)</u> Recursive directory or file renaming function.
49	<u>os.rmdir(path)</u> Remove the directory path
50	<u>os.stat(path)</u> Perform a stat system call on the given path.
51	<u>os.stat float times([newvalue])</u> Determine whether stat_result represents time stamps as float objects.
52	<u>os.statvfs(path)</u> Perform a statvfs system call on the given path.
53	<u>os.symlink(src, dst)</u> Create a symbolic link pointing to src named dst.
54	<u>os.tcgetpgrp(fd)</u> Return the process group associated with the terminal given by fd (an open file descriptor as returned by

	<code>open()</code> .
55	<u><code>os.tcsetpgrp(fd, pg)</code></u> Set the process group associated with the terminal given by fd (an open file descriptor as returned by <code>open()</code>) to pg.
56	<u><code>os.tempnam([dir[, prefix]])</code></u> Return a unique path name that is reasonable for creating a temporary file.
57	<u><code>os.tmpfile()</code></u> Return a new file object opened in update mode (w+b).
58	<u><code>os.tmpnam()</code></u> Return a unique path name that is reasonable for creating a temporary file.
59	<u><code>os.ttyname(fd)</code></u> 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	<u><code>os.unlink(path)</code></u> Remove the file path.
61	<u><code>os.utime(path, times)</code></u> Set the access and modified times of the file specified by path.
62	<u><code>os.walk(top[, topdown=True[, onerror=None[, followlinks=False]])</code></u> Generate the file names in a directory tree by walking the tree either top-down or bottom-up.
63	<u><code>os.write(fd, str)</code></u> Write the string str to file descriptor fd. Return the number of bytes actually written.

មេរៀនទី ៧: **ERRORs AND EXCEPTIONs**

Python បានផ្តល់នូវ feature សំខាន់២សម្រាប់ដោះស្រាយនូវបញ្ហា error ដែលអាចកើតមានឡើងដោយប្រការណាមួយ ឬមិនអាចដឹងមុនគឺ Exception Handling និង Assertions។

នៅក្នុងមេរៀននេះ នឹងបង្ហាញអោយអ្នកមានសមត្ថភាពដូចខាងក្រោម:

- ➔ យល់ដឹងពី ប្រភេទនៃ Error
- ➔ យល់ដឹងពី របៀបដោះស្រាយ និងការពារនូវ Error

7.1.Exception Handling

Exception : សំដៅលើ Event ដែលកើតឡើងនៅពេលកំពុងដំណើរការរបស់ code ត្រូវបានហើយវាទៅកាត់ផ្តាច់នូវលំហូរនៃ program's instructions នោះ។ Exception គឺជា object តំណាងឲ្យ error ណាមួយ។

Handling an exception : ដើម្បីចាប់ប្តូរទម្រង់នូវ error នៃ code ដែលអាចកើតឡើងដោយប្រការណាមួយគឺគេប្រើប្រាស់ 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")

្ក
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: សំប្រាប់បង្ខំឱ្យមាន 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 ខាងលើសូមប្រើកូដ ខាងក្រោម

try:
    Business Logic here...
except "Invalid level!":
    Exception handling here...
else:
    Rest of the code here...

```

User-Defined Exceptions: បង្កើតប្រភេទ error ថ្មីដោយខ្លួនឯង

ដើម្បីបង្កើត exception ថ្មីអ្នកត្រូវតែ 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.

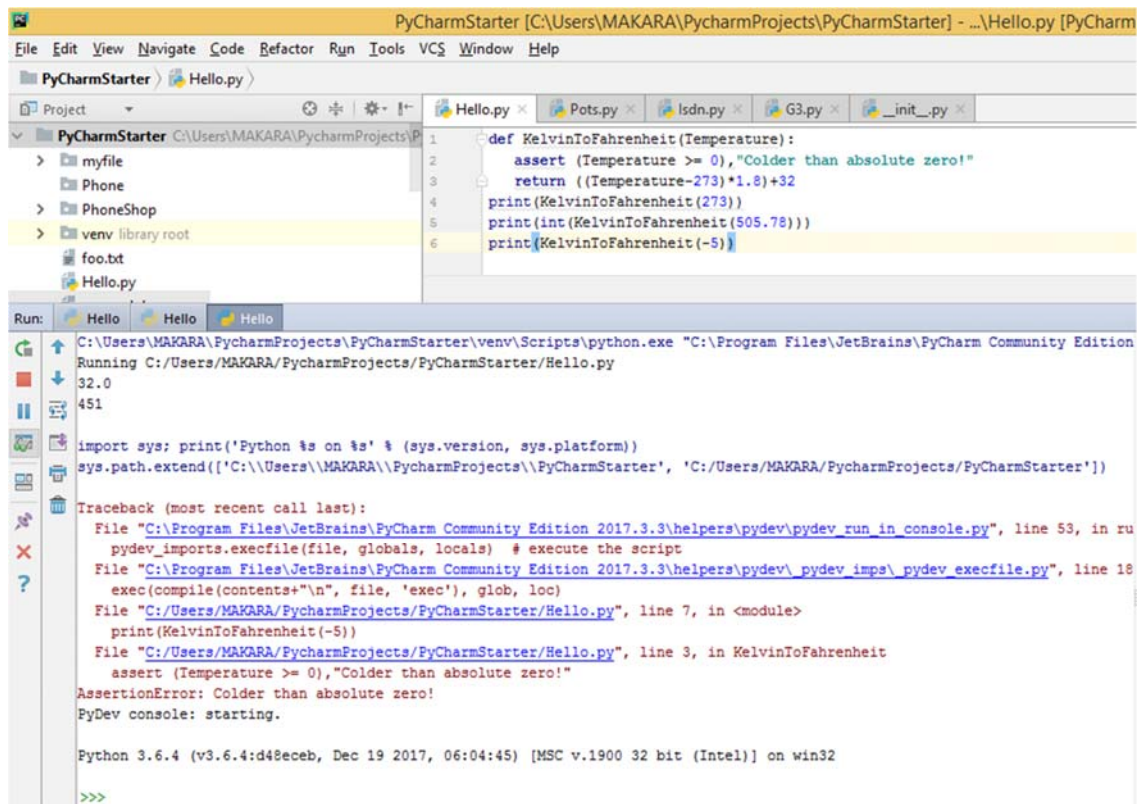
The assert Statement:

Syntax: assert Expression[, Arguments]

Assertion សង្ឃឹមថា result របស់ expression នឹង 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))
```



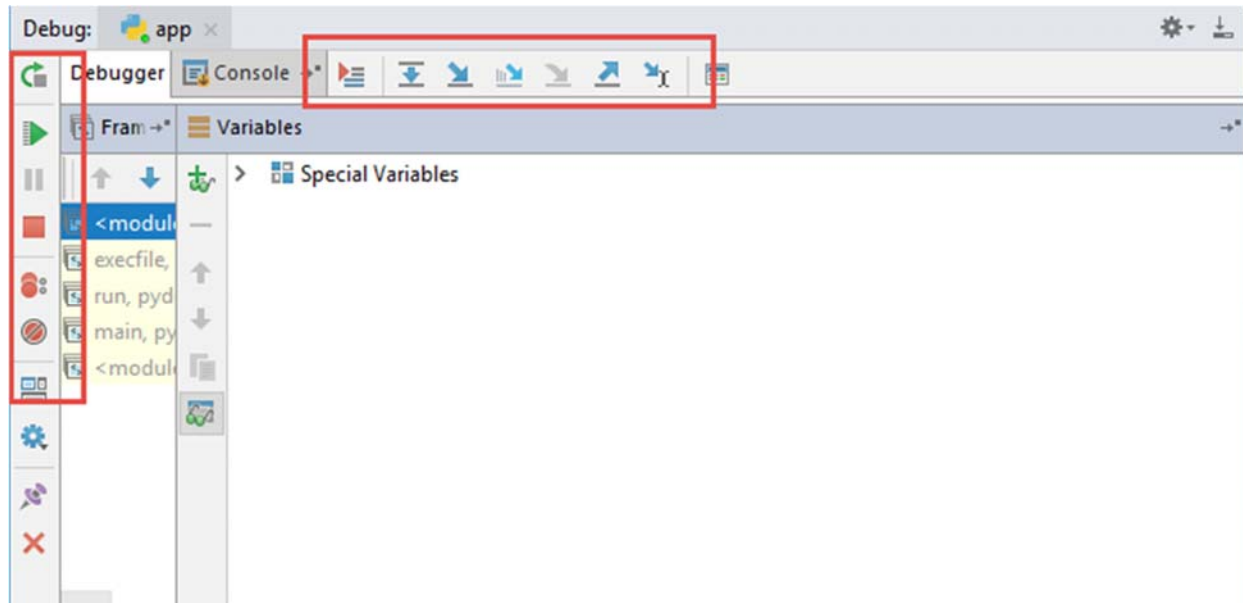
7.2. Debugging Tools

នៅពេលដែលអ្នកជួបនូវបញ្ហាប្រភេទ logic error យើងអាចប្រើប្រាស់នូវ debug tool ដើម្បីស្វែងរកកំហុសដែលបណ្តាលឲ្យលទ្ធផលចេញខុស។ វិធីនេះគឺយើងត្រូវពិនិត្យ និងស្វែងរកកំហុសម្តងម្កាលបន្ទាត់ៗ នៅពេលអ្នកបាន run program ក្នុងកម្រិត debug mode។

ទី១៖ ត្រូវកំណត់ breaking point → រួចចុចលើ រូប debug ដើម្បី run



ទី២៖ ប្រើ tool ខាងក្រោមដើម្បី check ដំណើរការនៃ code



7.3.Logging Module

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

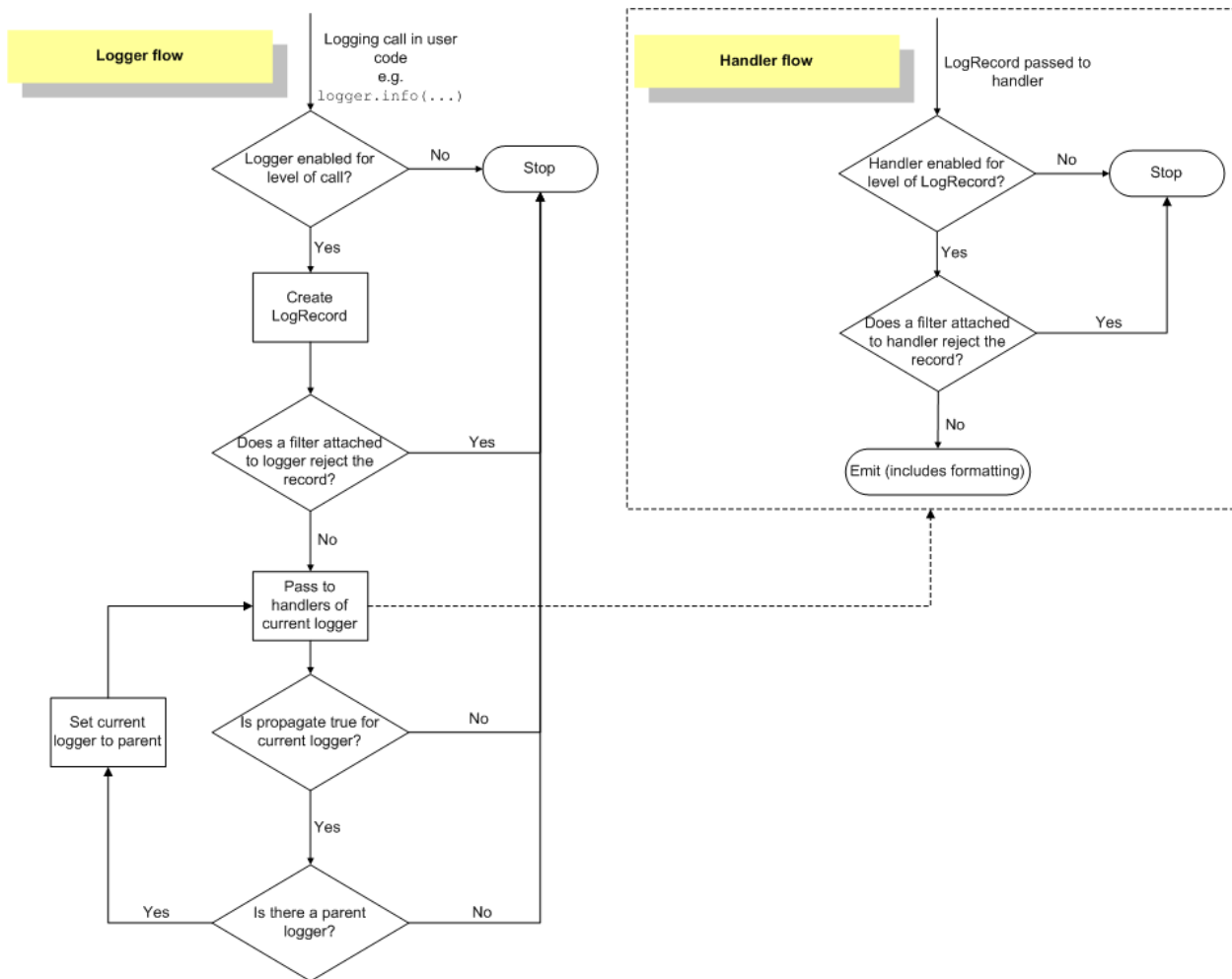
Logging Level : Logging បានផ្តល់នូវ level មួយចំនួនដូចជា ៖

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 ជាច្រើនដើម្បី show 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 msg to 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 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 was created (as returned by 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 was created.
message	%(message)s	The logged message, computed as msg% args. This is set when 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).
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:



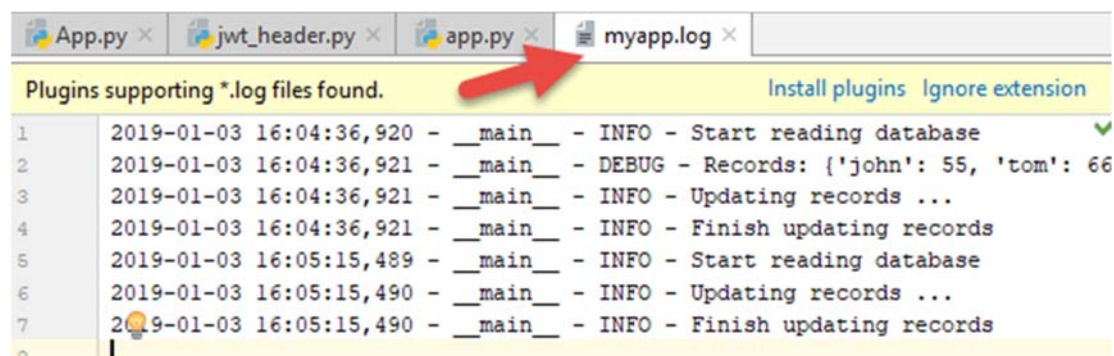
```

app x
C:\xampp\htdocs\flaskproject\venv\Scripts\python.exe C:/xampp/htdocs/pyar
WARNING:root:is when this event was logged.
2019-01-03 16:05:15,489 - __main__ - INFO - Start reading database
INFO: __main__:Start reading database
2019-01-03 16:05:15,490 - __main__ - INFO - Updating records ...
INFO: __main__:Updating records ...
2019-01-03 16:05:15,490 - __main__ - INFO - Finish updating records
INFO: __main__:Finish updating records

Process finished with exit code 0

```

File.log



```

App.py x jwt_header.py x app.py x myapp.log x
Plugins supporting *.log files found. Install plugins Ignore extension
1 2019-01-03 16:04:36,920 - __main__ - INFO - Start reading database ✓
2 2019-01-03 16:04:36,921 - __main__ - DEBUG - Records: {'john': 55, 'tom': 66}
3 2019-01-03 16:04:36,921 - __main__ - INFO - Updating records ...
4 2019-01-03 16:04:36,921 - __main__ - INFO - Finish updating records
5 2019-01-03 16:05:15,489 - __main__ - INFO - Start reading database
6 2019-01-03 16:05:15,490 - __main__ - INFO - Updating records ...
7 2019-01-03 16:05:15,490 - __main__ - INFO - Finish updating records

```

7.4.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៖

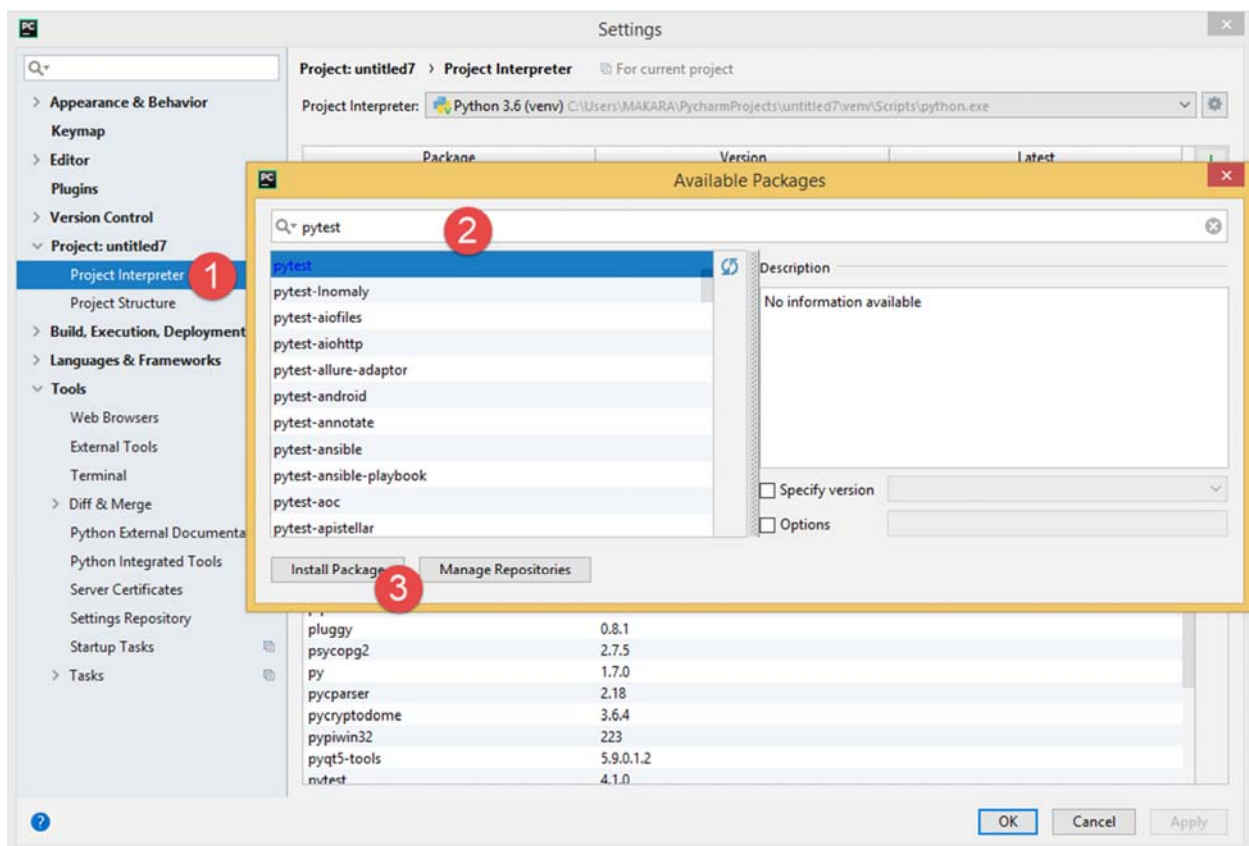
- test fixture៖ គឺជាការ Test លើផ្នែកណាមួយនៃ software ឬ device ឬ Item ណាមួយ។ ជាទូទៅវានឹងដំណើរការជា loop ដើម្បី check មើលលទ្ធផលនៃ code(function) ដែលបានប្រើ

ប្រាស់ក្នុង software ថាតើវាត្រឹមត្រូវឬអត់។ យើងប្រើប្រាស់វារួមជាមួយនឹង assert statement ដើម្បី check មើល initialize value ក្នុង code(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**)



បង្កើត file **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
```

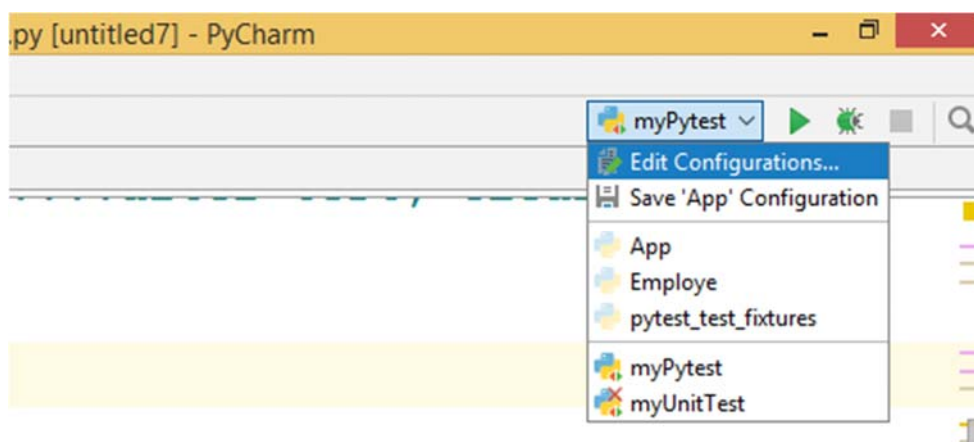
```
if __name__ == '__main__':
```

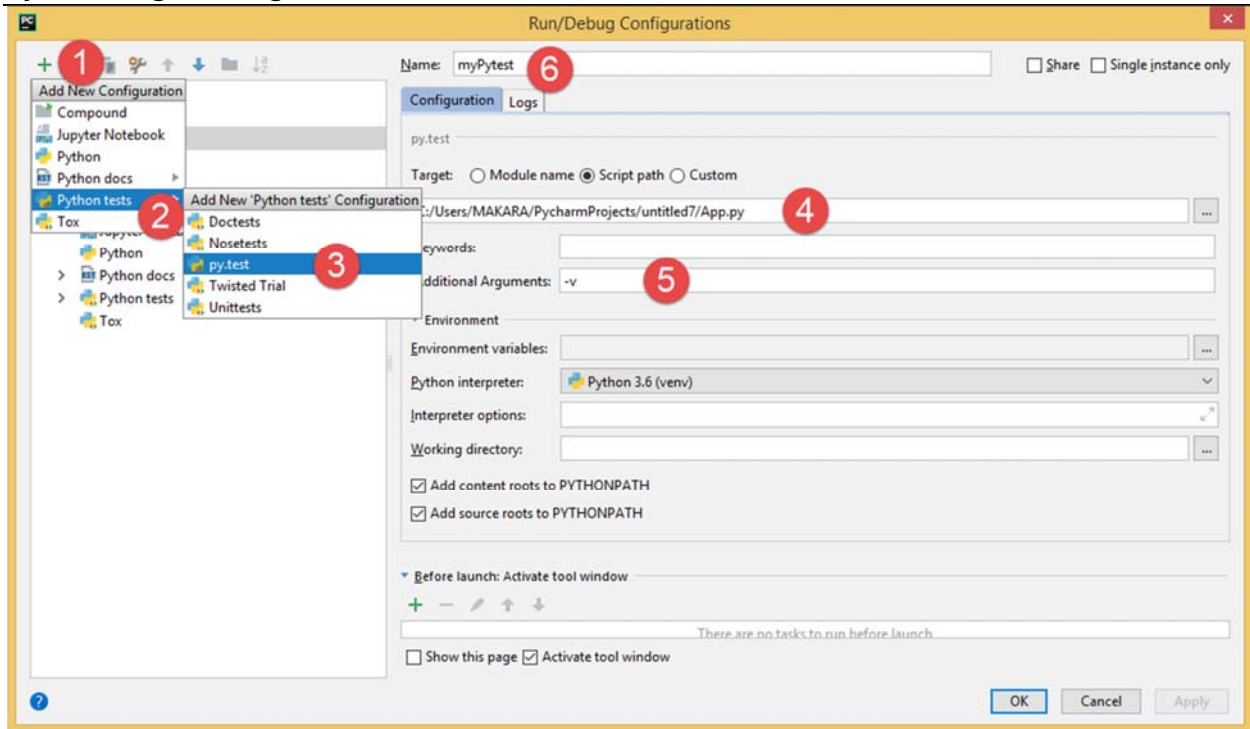
```
    pytest.main()
```

ចំណាំ៖

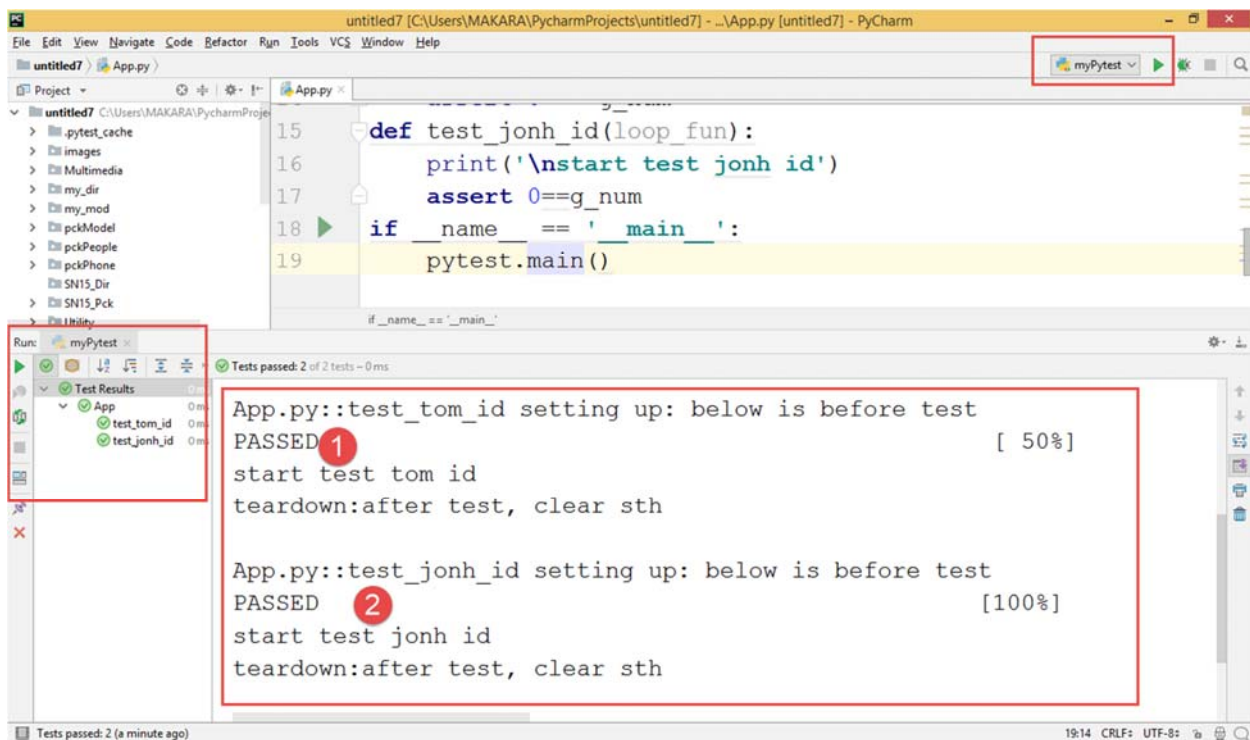
- រាល់អនុគមន៍ដែលត្រូវប្រមូលយកទៅtest ត្រូវតែសរសេរផ្ដើមដោយពាក្យ test(អក្សរតូច)
- ហើយអនុគមន៍ដែលផ្ដើមដោយ test នោះត្រូវមានparameterជាឈ្មោះរបស់ functionដែលបានប្រើប្រាស់ដោយ@pytest.fixture()
- រាល់ពេលrun function test ទាំងនោះគឺវាត្រូវ fixture function(loop_fun) ជាមុនសិនមុននឹងវា run test function by fixture

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)

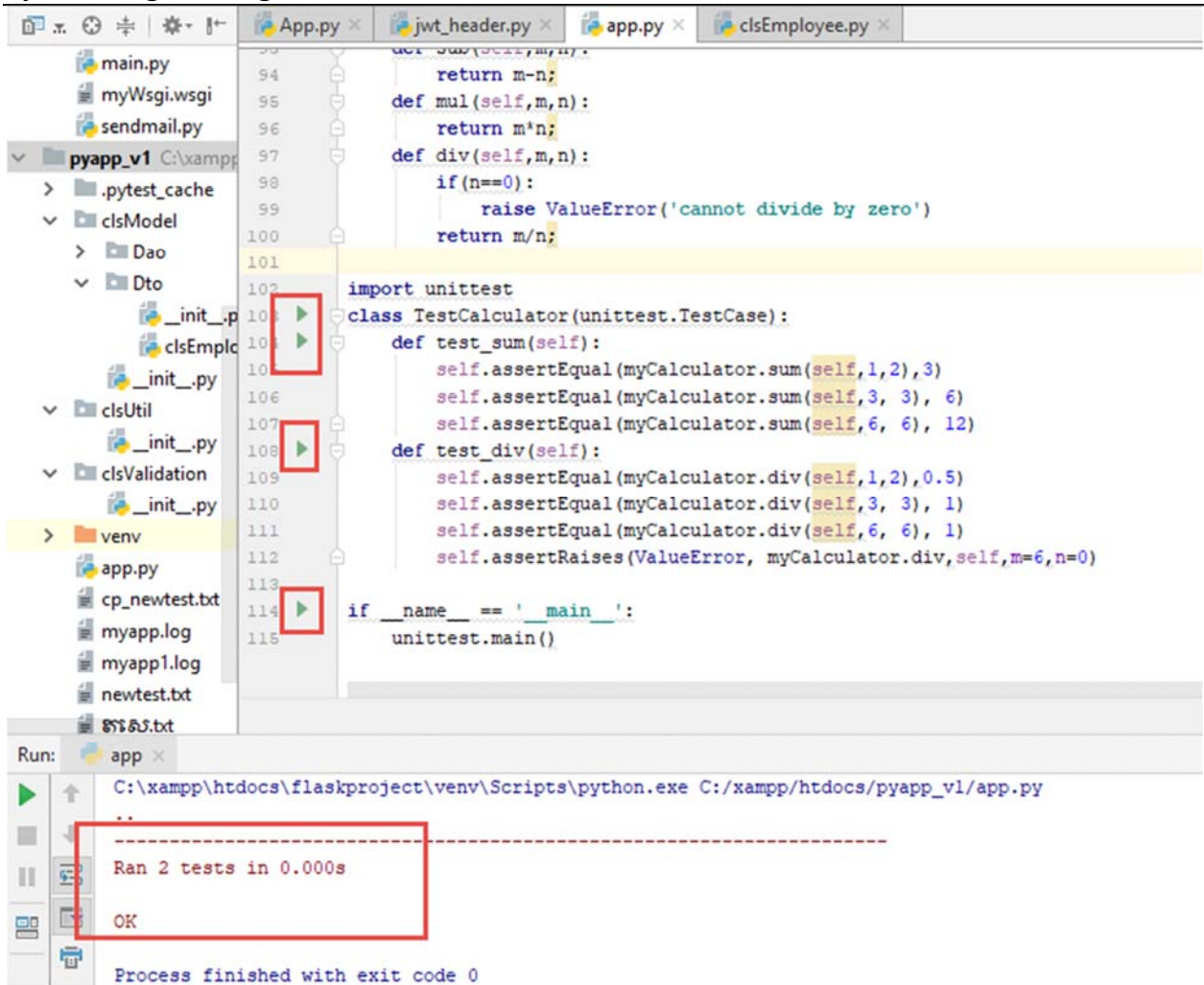
```

```

if __name__ == '__main__':
    unittest.main()

```

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



Method ផ្សេងៗទៀតដែលប្រើក្នុង class unittest.TestCase

Method	Checks that	New in
assertEqual(a, b) assertEqual(first, second, msg=None)	a == b	
assertNotEqual(a, b) assertNotEqual(first, second, msg=None)	a != b	
assertTrue(x) assertTrue(expr, msg=None)	bool(x) is True	
assertFalse(expr, msg=None) assertFalse(x)	bool(x) is False	

Method	Checks that	New in
<code>assertIs(a, b)</code> <code>assertIs(first, second, msg=None)</code>	<code>a is b</code>	3.1
<code>assertIsNot(a, b)</code> <code>assertIsNot(first, second, msg=None)</code>	<code>a is not b</code>	3.1
<code>assertIsNone(x)</code> <code>assertIsNone(expr, msg=None)</code>	<code>x is None</code>	3.1
<code>assertIsNotNone(x)</code> <code>assertIsNotNone(expr, msg=None)</code>	<code>x is not None</code>	3.1
<code>assertIn(a, b)</code> <code>assertIn(first, second, msg=None)</code>	<code>a in b</code>	3.1
<code>assertNotIn(a, b)</code> <code>assertNotIn(first, second, msg=None)</code>	<code>a not in b</code>	3.1
<code>assertIsInstance(a, b)</code> <code>assertIsInstance(obj, cls, msg=None)</code>	<code>isinstance(a, b)</code>	3.2
<code>assertNotIsInstance(a, b)</code> <code>assertNotIsInstance(obj, cls, msg=None)</code>	<code>not isinstance(a, b)</code>	3.2

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

Method	Checks that	New in
<code>assertRaises(exc, fun, *args, **kwds)</code> <code>assertRaises(exception, callable, *args, **kwds)</code> <code>assertRaises(exception, *, msg=None)</code>	<code>fun(*args, **kwds) raises exc</code>	

Method	Checks that	New in
assertRaisesRegex(exc, r, fun,*args, **kwds) assertRaisesRegex(exception, regex, callable, *args, **kwds) assertRaisesRegex(exception, regex, *, msg=None)	fun(*args, **kwds)raises <i>exc</i> and the message matches regex <i>r</i>	3.1
assertWarns(warn, fun, *args,**kwds) assertWarns(warning, callable, *args, **kwds) assertWarns(warning, *, msg=None)	fun(*args, **kwds)raises <i>warn</i>	3.2
assertWarnsRegex(warn, r, fun,*args, **kwds) assertWarnsRegex(warning, regex, callable, *args, **kwds) assertWarnsRegex(warning, regex, *, msg=None)	fun(*args, **kwds)raises <i>warn</i> and the message matches regex <i>r</i>	3.2
assertLogs(logger, level) assertLogs(logger=None, level=None)	The with block logs on <i>logger</i> with minimum <i>level</i>	3.4

មេរៀនទី ៨:

Python - Object Oriented

8.1. សេចក្តីណែនាំ Introduction to OOP

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

- Class and Object : បង្កើត data 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) ពី class មួយទៅ class មួយទៀត។ class ដែលជាអ្នកផ្ទេរមរតក ហៅថា supper class ឬ base class ឬ child class។ class ដែលជាអ្នកទទួលមរតកពីគេ គេហៅថា sub class ឬ derived class ឬ parent class។
- Polymorphism: សំដៅលើការធ្វើប្រតិបត្តិការផ្សេងគ្នា បើទោះបីជា object នីមួយៗត្រូវបាន call ចេញពី method តែមួយក៏ដោយ។ មានបច្ចេកទេសសំខាន់ពីរគឺ Overloading and Overriding 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** ក្នុងកាស៊ី java or c#, = Me in VB and VB.Net

8.2. ការបង្កើត Class and Object

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

Object : គឺជាការ **instance(attribute and method)** ដែលកើតចេញពី **class** ។

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** ហៅវាឲ្យដំណើរការនៅពេលដែលអ្នកបានបង្កើត **new instance** ចេញពី **class**នេះ។
- 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"
```

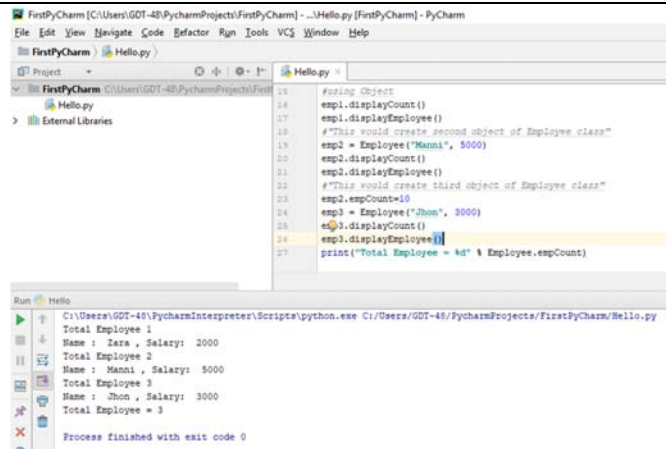
```
emp2.empCount=10
```

```
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 **unneeded object**(built-in types or class instances)ដោយស្វ័យប្រវត្តិដើម្បីចំណេញ **memory space**។
- **Garbage Collector** : វានឹង **destroy object** នៅពេលដែល **program RUN** ហើយនិង **object's reference** រាប់ទៅឃើញ **zero**។ **object's reference** រាប់កើនពេលដែលវាត្រូវបានគេដាក់ឈ្មោះថ្មីឬយកវាទៅដាក់នៅក្នុងពួក **container**(list, tuple, or dictionary) ហើយវារាប់ថយនៅពេលដែលយើងលុប **object** ដោយប្រើពាក្យ **del** ។
- **Destructor** : ជា **special method** ដែលដំណើរការ **automatic** នៅពេលដែល **Garbage Collector** វាដំណើរការ **destroy object**។ យើងអាចប្រើវាដើម្បីគ្រាន់តែជាការ **noted** ពេល **destroy object** ហើយយើងអាចបង្កើតវាបានដោយឈ្មោះ **method _del_()** ក្នុង **class**។
- **Method id(..)** ជា **build-in method** ប្រើដើម្បីបង្ហាញលេខរៀង(**address**)របស់ **object**

```
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 objects
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))

```

8.3. ការបង្កើត Encapsulation(Data Hiding)

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

ដើម្បី hide attribute យើងត្រូវ __ ពីមុខ

ឧទាហរណ៍

```

class JustCounter:
    __secretCount = 0

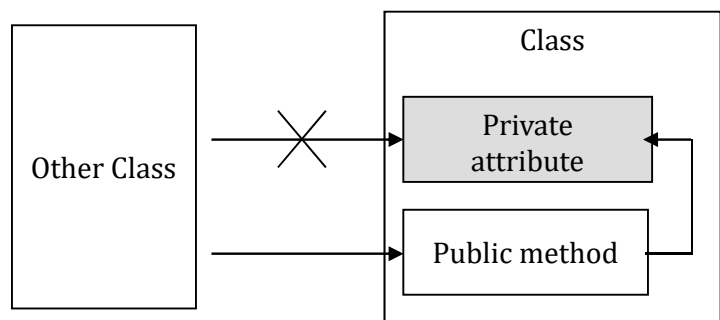
    def count(self):
        self.__secretCount += 1
        print(self.__secretCount)

```

```

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','X0','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()

```

8.4. ការបង្កើត Inheritance

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

```

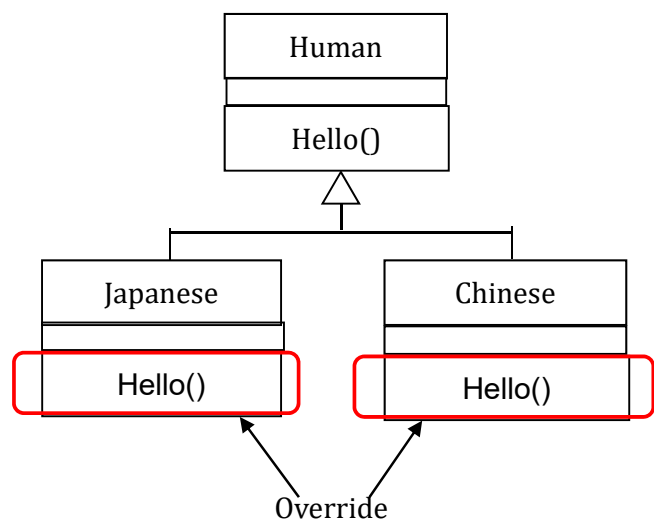
Build-In Function:

- 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

8.5. ការបង្កើត Polymorphism

សំដៅលើការធ្វើប្រតិបត្តិការផ្សេងគ្នា បើទោះបីជា object នីមួយៗត្រូវបាន call ចេញពី method តែមួយក៏ដោយ។

ចូរបង្កើត class ដូចរូបខាងក្រោម៖



Overriding 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):#override 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 ក្នុង class បាន

Sr.No.	Method, Description & Sample Call
1	<code>__init__(self [,args...])</code> Constructor (with any optional arguments) Sample Call : <code>obj = className(args)</code>
2	<code>__del__(self)</code> Destructor, deletes an object Sample Call : <code>del obj</code>
3	<code>__repr__(self)</code> Evaluable string representation Sample Call : <code>repr(obj)</code>
4	<code>__str__(self)</code> Printable string representation

	Sample Call : <code>str(obj)</code>
5	<code>__cmp__ (self, x)</code> Object comparison Sample Call : <code>cmp(obj, x)</code>

Overloading Function៖ គឺជា **function** ស្ថិតនៅក្នុង **class** តែមួយហើយត្រូវ៖

- មានឈ្មោះ ដូចគ្នា តែរបៀប **call** ត្រូវមាន **parameter** 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)
```

Special Method ផ្សេងៗទៀតដែលអ្នកអាចយកទៅ **overloading** បាន

OPERATOR	FUNCTION	METHOD DESCRIPTION
+	<code>__add__(self, other)</code>	Addition

OPERATOR	FUNCTION	METHOD DESCRIPTION
*	<code>__mul__(self, other)</code>	Multiplication
-	<code>__sub__(self, other)</code>	Subtraction
%	<code>__mod__(self, other)</code>	Remainder
/	<code>__truediv__(self, other)</code>	Division
<	<code>__lt__(self, other)</code>	Less than
<=	<code>__le__(self, other)</code>	Less than or equal to
==	<code>__eq__(self, other)</code>	Equal to
!=	<code>__ne__(self, other)</code>	Not equal to
>	<code>__gt__(self, other)</code>	Greater than
>=	<code>__ge__(self, other)</code>	Greater than or equal to
[index]	<code>__getitem__(self, index)</code>	Index operator
in	<code>__contains__(self, value)</code>	Check membership
len	<code>__len__(self)</code>	The number of elements
str	<code>__str__(self)</code>	The string representation

8.6. Regular Expressions

វាមិនមែនជាមេរៀននៃ OOP ទេ។ regular expression គឺជា special sequence of characters ដែលជួយអ្នកក្នុងការផ្ទៀងផ្ទាត់ (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	<p>pattern</p> <p>This is the regular expression to be matched.</p>
2	<p>string</p> <p>This is the string, which would be searched to match the pattern at the beginning of string.</p>
3	<p>flags</p> <p>You can specify different flags using bitwise OR (). These are modifiers, which are listed in the table</p>

below.

នៅពេលដែលអ្នកប្រើ re.match function នោះវានឹង return មកវិញជា match object ពេល success តែបើ fail វិញវានឹង return **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 `re.match()` function will return a match object if the search is successful, and `None` if it fails. You can use `re.match()` to check if a string matches a pattern at the beginning of the string. It returns a match object if the search is successful, and `None` if it fails.

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 "cutter" 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 <code>[\t\n\r\f]</code> .
26	\S Matches nonwhitespace.
27	\d Matches digits. Equivalent to <code>[0-9]</code> .
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	<code>((Pp))ython&\1ails</code> Match python&pails or Python&Pails
2	<code>(["'])[^\1]*\1</code> 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	<code>python perl</code> Match "python" or "perl"
2	<code>rub(y le)</code> Match "ruby" or "ruble"
3	<code>Python(!+ \?)</code> "Python" followed by one or more ! or one ?

Anchors

This needs to specify match position.

Sr.No.	Example & Description
1	<code>^Python</code> Match "Python" at the start of a string or internal line
2	<code>Python\$</code> Match "Python" at the end of a string or line
3	<code>\APython</code>

	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

9.1. សេចក្តីណែនាំ Introduction to GUI

Python បានផ្តល់នូវជម្រើសជាច្រើនក្នុងការ developing កម្មវិធីជាមួយនឹង graphical user interface។ GUIs ដែលគេនិយមប្រើមានដូចខាងក្រោម៖

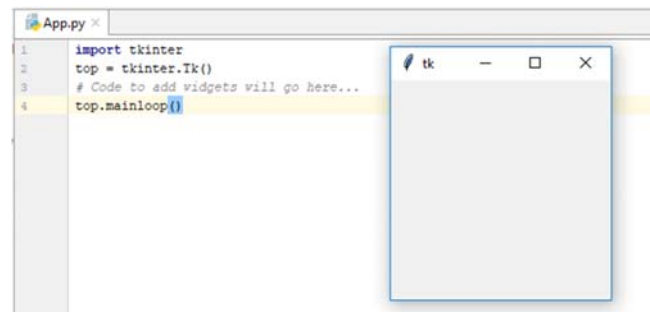
- **Tkinter** – Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
- **PyQy** -- Mainly, you can develop desktop applications with PyQt. PyQt is a binding for QT framework. Qt is written in C++, So with PyQt you can use all QT classes 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>.

9.2. 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 វា provide ឲ្យយើងនូវ control ជាច្រើន (buttons, labels and text boxes...) សម្រាប់បង្កើតជា GUI application។ Widgets រួមមាន control ដូចខាងក្រោម៖

Sr.No.	Operator & Description
1	<u>Button</u>

	The Button widget is used to display buttons in your application.
2	<u>Canvas</u> The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
3	<u>Checkbutton</u> The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
4	<u>Entry</u> The Entry widget is used to display a single-line text field for accepting values from a user.
5	<u>Frame</u> The Frame widget is used as a container widget to organize other widgets.
6	<u>Label</u> The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
7	<u>Listbox</u> The Listbox widget is used to provide a list of options to a user.
8	<u>Menubutton</u> The Menubutton widget is used to display menus in your application.
9	<u>Menu</u> The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
10	<u>Message</u> The Message widget is used to display multiline text fields for accepting values from a user.
11	<u>Radiobutton</u> 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	<u>Scale</u> The Scale widget is used to provide a slider widget.
13	<u>Scrollbar</u> The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
14	<u>Text</u>

	The Text widget is used to display text in multiple lines.
15	<u>Toplevel</u> The Toplevel widget is used to provide a separate window container.
16	<u>Spinbox</u> 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	<u>PanedWindow</u> A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
18	<u>LabelFrame</u> A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
19	<u>tkMessageBox</u> This module is used to display message boxes in your applications.

ខាងក្រោមនេះគឺជា attribute ដែលគេតែងតែឧស្សាហ៍ប្រើប្រាស់៖

- Dimensions: ជាខ្នាតប្រវែងរបស់ widget(width,height,borderwidth,padx,pady, highlightthickness, selectborderwidth, wraplength, underline) ដែលជាទូទៅវាត្រូវបានគេគិតជាខ្នាត Pixel។ តែយើងក៏អាចកំណត់ជា Centimeters, inches, Millimetersបានផងដែរ។
- Colors: យើងអាចប្តូរ color បានពីរបៀបគឺ hexadecimal digits("#fff", "#000fff000") ឬ Color Name("white", "black", "red", "green", "blue", "cyan", "yellow", and "magenta")។ ដែលជាទូទៅគេប្រើប្រាស់វាលើ(activebackground, activeforeground, background, disabledforeground, foreground, highlightbackground, highlightcolor, selectbackground, selectforeground)
- Fonts: សម្រាប់ប្តូរព័ត៌មានអក្សរ(font{font name}, size{int pixel}, 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: ជា style របស់ widget(FLAT, RAISED, SUNEN,GROOVE,RIDGE)
- Bitmaps: សម្រាប់រូបភាព bitmap មានស្រាប់(error, warning,...)
- Cursors: to change mouse course on control

Geometry Management: ជាពណ៌នាពី method សម្រាប់រៀបចំទៅតាំងរបស់ control ក្នុង parent 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.

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** : ជាការ option សម្រាប់ call method or function ពេល user clicked
- **flash()** : delay time between active and normal colors
- **invoke()** : បញ្ជាក់ឱ្យ 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()`



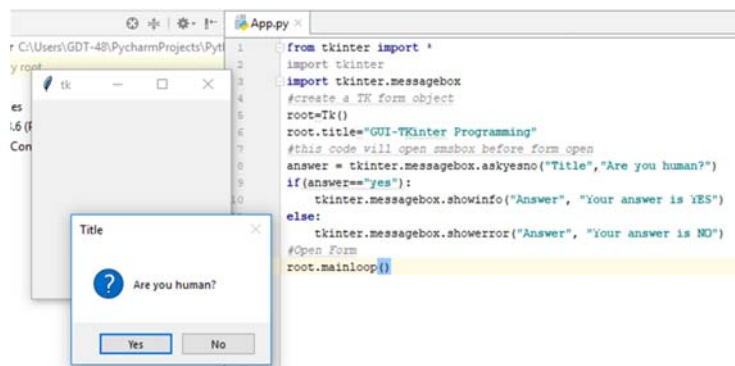
របៀបប្រើប្រាស់ MessageBox:

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

Method សំខាន់ៗសម្រាប់បង្ហាញ message:

- `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.askyesno("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()

```

របៀបប្រើប្រាស់ CheckButton:

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

Attribute សំខាន់ៗ:

- deselect()
- flash()
- invoke()
- select()
- 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
- onvalue: សម្រាប់កំណត់តម្លៃពេលcheck(default set value 1 when box is on)
- offvalue: សម្រាប់កំណត់តម្លៃពេលun-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

```

```

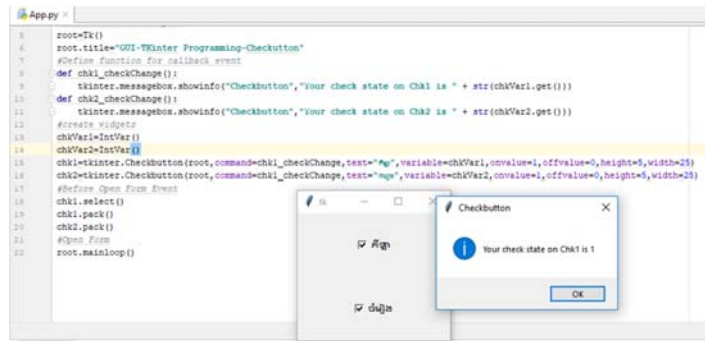
def chk1_checkChange():
    tkinter.messagebox.showinfo("Checkbutton","Your check state on Chk1 is " +
        str(chkVar1.get()))
def chk2_checkChange():
    tkinter.messagebox.showinfo("Checkbutton","Your check state on Chk2 is " +
        str(chkVar2.get()))

#create widgets
chkVar1=IntVar()
chkVar2=IntVar()
chk1=tkinter.Checkbutton(root,command=chk1_checkChange,text="ក៏ឡា",variable
    =chkVar1,onvalue=1,offvalue=0,height=5,width=25)
chk2=tkinter.Checkbutton(root,command=chk1_checkChange,text="ចំ
    រៀង",variable=chkVar2,onvalue=1,offvalue=0,height=5,width=25)

#Before Open Form Event
chk1.select()
chk1.pack()
chk2.pack()

#Open 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: ប្តូរព័ត៌មានអក្សរ
- bd = borderwidth
- relief : style of label

from tkinter import *

import tkinter

import tkinter.messagebox

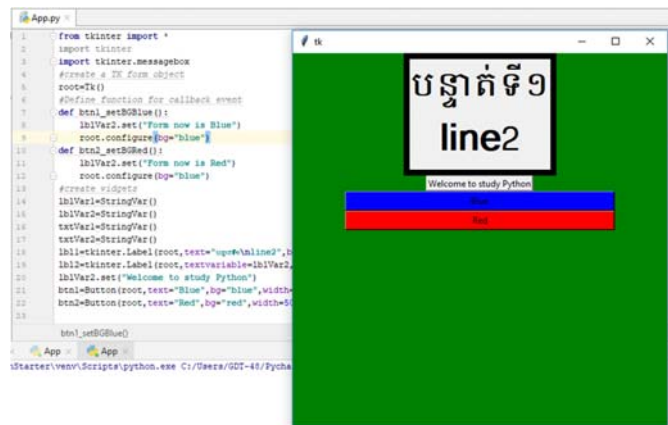
#create a TK form object

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 Message:

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

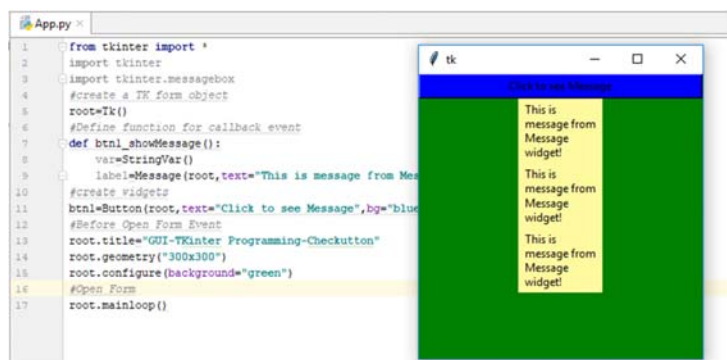
Attribute សំខាន់ៗ៖

- 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 Spinbox: សម្រាប់ number មាន up-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 RadioButton: `checkboxbutton` គេប្រើសម្រាប់ `multiple-choice`, `radiobutton` គឺប្រើសម្រាប់ `single-choice`

Syntax: `w = Radiobutton (master, option, ...)`

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()



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 checkbox 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 wrap ទិន្នន័យ (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

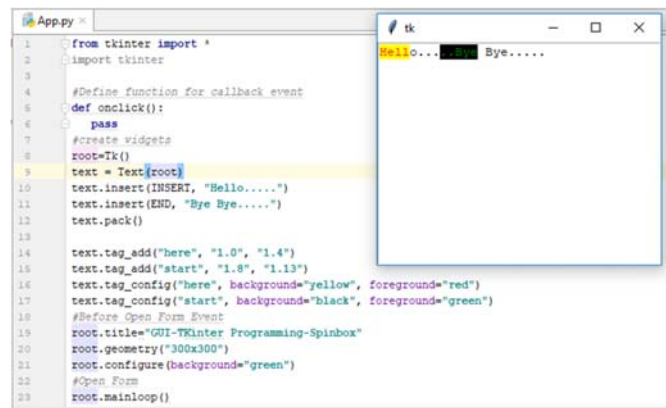
```
root=Tk()
```

```
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 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
```

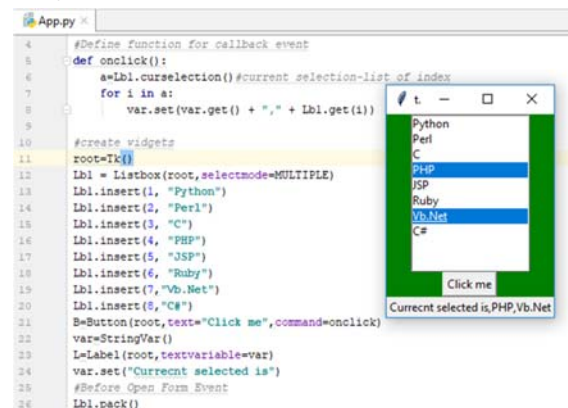
```
Lb1.pack()
```

```
B.pack()
```

```
L.pack()
```

```
root.title="GUI-TKinter Programming-Spinbox"
```

```
root.configure(background="green")
```



```
#Open Form
root.mainloop()
```

របៀបប្រើប្រាស់ Create Image:

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 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 FileDialog: សម្រាប់brows ទៅរកទីតាំងfile ឬ folder(save,open, brow)

Syntax : root.fileName=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 Canvas: សម្រាប់បង្កើតរូបភាពជាភាពផ្សេងៗ

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

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

យើងអាចប្រើ Canvas ដើម្បីបង្កើតជា

Arc: ខ្សែរកោង

```
coord = 10, 50, 240, 210
```

```
arc = canvas.create_arc(coord, start=0, extent=150, fill="blue")
```

Image: រូបភាព

```
filename = PhotoImage(file = "sunshine.gif")
```

```
image = canvas.create_image(50, 50, anchor=NE, image=filename)
```

Line: បន្ទាត់

```
line = canvas.create_line(x0, y0, x1, y1, ..., xn, yn, options)
```

Oval: រាងមូល(រង្វង់ ឬពងក្រពើ)

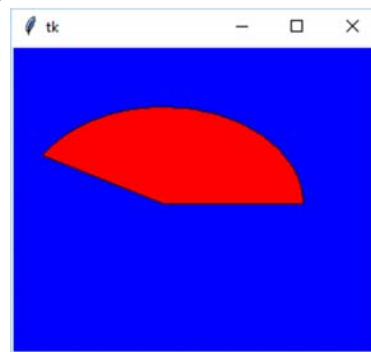
```
oval = canvas.create_oval(x0, y0, x1, y1, options)
```

Polygon: ពហុកោណ

```
oval = canvas.create_polygon(x0, y0, x1, y1,...xn, yn, options)
```

ឧទាហរណ៍

```
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()
```



*****នៅមាន widget(control) ជាច្រើនទៀតដែលអ្នកត្រូវរៀនជាមួយលោកគ្រូអ្នកគ្រូដូចជា Frame, Menubutton, Menu, Scale, Scrollbar, Toplevel, Paned Window, LabelFrame.....។

9.3. GUI-PyQt Programming

PyQt គឺជា standard GUI library ដែលបានបង្កើតឡើងដោយក្រុមហ៊ុន QT ដែលអាចRunបានជាមួយភាសា 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 interface។

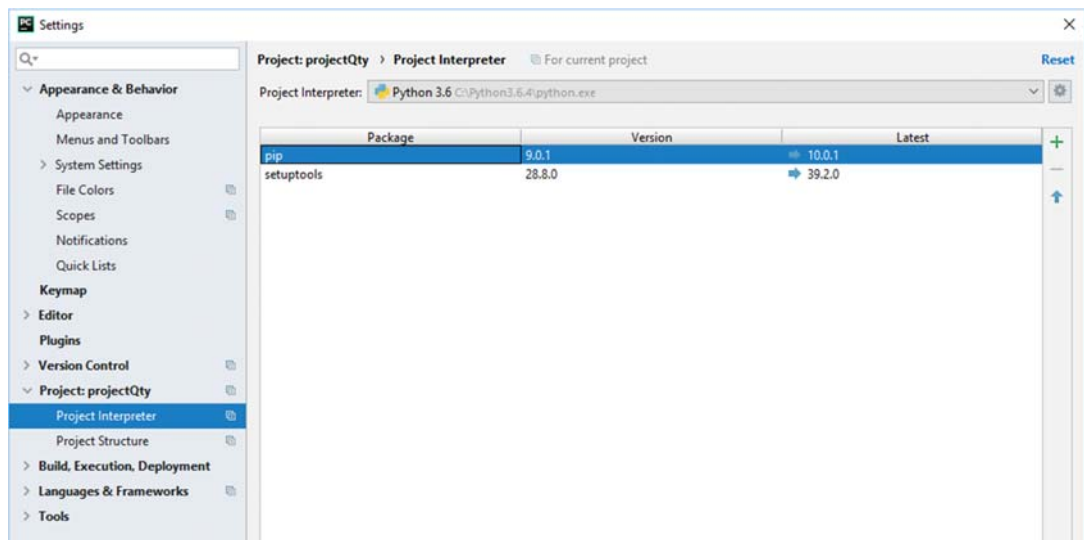
ដើម្បីប្រើ PyQy អ្នកត្រូវទៅតំឡើង software ពីរនេះបន្ថែមទៀតគឺ៖

- PyQt for window(use google = "PyQt5-5.6-gpl-Py3.5-Qt5.6.0-x32-2.exe")

ចំណាំ៖

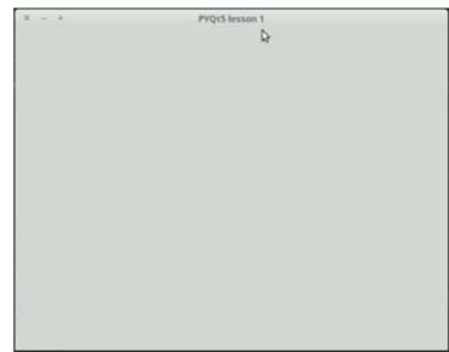
- បើទីតាំងរបស់ Pythonគឺ៖ C:\Python3.6.4
- នោះទីតាំងរបស់ PyQt5គឺ៖ C:\Python3.6.4\python.exe → បន្ទាប់មកវានឹង install soft ចូលទៅទីតាំង C:\Python3.6.4\Lib\site-packages\PyQt5
- បន្ទាប់ពីដំឡើងចប់ទីតាំងរបស់ QT Designer គឺ៖ C:\Python3.6.4\Lib\site-packages\PyQt5\designer.exe

- បើកកម្មវិធី `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 lesson 1")
    w.show()
    sys.exit(app.exec_())
window()
```



របៀបប្រើប្រាស់ Qt-Designer:

បើកកម្មវិធី C:\Python3.6.4\Lib\site-packages\PyQt5\designer.exe

