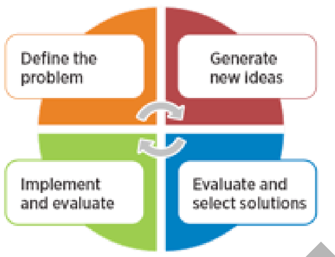
Table of Contents

1. PROBLEM-SOLVING PROCESS

Problem-solving is the act of defining a problem; determining the cause of the problem; identifying, prioritizing, and selecting alternatives for a solution; and implementing a solution.



1.1. Understanding the problem (Define the problem)

Diagnose the situation so that focus is on the problem, not just its symptoms. Helpful problem-solving techniques include using flowcharts to identify the expected steps of a process and cause-and-effect diagrams to define and analyze root causes.

The sections below help explain key problem-solving steps. These steps support the involvement of interested parties, factual information, comparison of expectations to reality, and a focus on the root causes of a problem.

It would be best if began by:

- Review and document how processes currently work (i.e., who does what, with what information, using what tools, communicating with organizations and individuals, in what time frame, using what format).
- Evaluate the possible impact of new tools and revised policies in developing the "what should be" model.

1.2. Defining the problem and boundaries

Postpone the selection of one solution until several problem-solving alternatives are identified. Considering multiple options can significantly enhance the value of your ideal solution. Once you have decided on the "what should be" model, this target standard becomes the basis for developing a road map for investigating alternatives. Brainstorming and team problem-solving techniques are both valuable tools in this stage of problem-solving.

Many alternative solutions to the problem should be generated before the final evaluation. In most cases first suggested solution is taken without considering the alternative. That would be an issue in getting results as we expected. In that case, we miss the potential to learn something new that will allow for real improvement in the problem-solving process.

3.PROGRAMMING PARADIGMS

3.1. Evolution of programming languages

Programming Language is indeed the fundamental unit of today's tech world. It is considered the set of commands and instructions that we give to the machines to perform a particular task.

For example, give some instructions to add two numbers. The machine will do it for us and tell us the correct answer accordingly. However, do you know that Programming Languages are having a long and rich history of their evolution? Moreover, with a similar concern, here in this article, we will look at the evolution of Programming Languages over the period.

In the computer world, we have about 500+ programming languages that having their syntax and features. Moreover, suppose you type who is the father of the computer. In that case, the search engine will show you the result of Charles Babbage, but the father of the computer did not write the first code. **It was Ada Lovelace who has written the first-ever computer programming language, and the year was 1883.**

1883: The Journey starts from here…!!

- In the early days, Charles Babbage had made the device. However, he was confused about how to give instructions to the machine. Then Ada Lovelace wrote the instructions for the analytical engine.
- Charles Babbage made the device, and Ada Lovelace wrote the code for computing Bernoulli's number.
- First time in history that the capability of computer devices was judged.

1949: Assembly Language

- It is a type of low-level language.
- It mainly consists of instructions (kind of symbols) that only machines could understand.
- In today's time, assembly language is also used in real-time programs such as simulation flight navigation systems and medical equipment, e.g., Fly-by-wire (FBW) systems.
- It is used to create computer viruses.

1952: Autocode

- Developed by Alick Glennie.
- The first compiled computer programming language
- COBOL and FORTRAN are the languages referred to as Autocode.

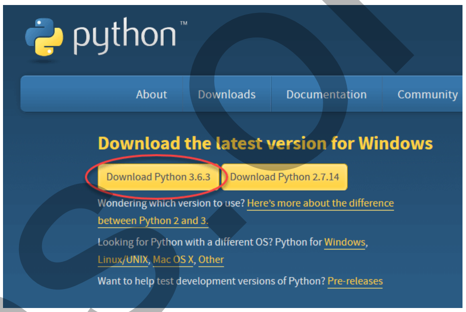
1957: FORTRAN

- Python shell with syntax highlighting,
- Multi-window text editor,
- Code autocompletion,
- Intelligent indenting,
- Program animation and stepping, which allows one line of code to run at a time helpful for debugging,
- Persistent breakpoints,
- Finally, Call stack visibility.

5.3. How to Install Python IDE

Below is a step by step process on how to download and install Python on Windows:

Step 1) To download and install Python, visit the official website of Python https://www.python.org/downloads/ and choose your version. We have chosen Python version 3.6.3



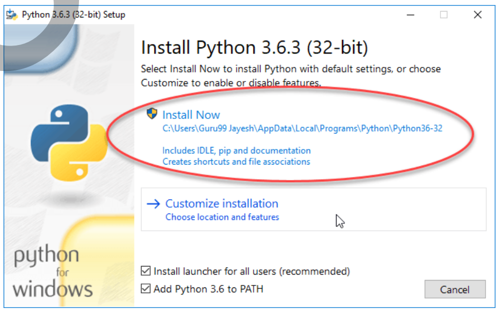
Download the latest version for Windows

Wondering which version to use? Here's more about the difference between Python 2 and 3.

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases

Step 2) Once the download is completed, run the .exe file to install Python. Now click on Install Now.



Install Python 3.6.3 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

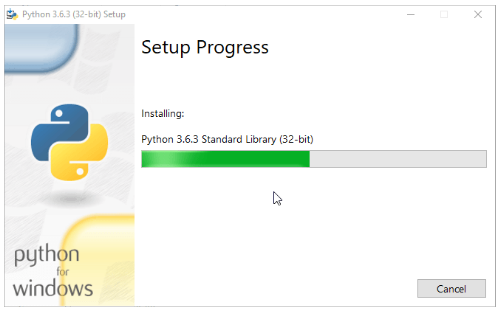
Install Now
C:\Users\Guru99 Jayesh\AppData\Local\Programs\Python\Python36-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.6 to PATH

Step 3) You can see Python installing at this point.



Setup Progress

Installing:

Python 3.6.3 Standard Library (32-bit)

/ Division	Divides left-hand operand by right hand operand	b / a = 2
% Modulus	Divides left-hand operand by right-hand operand and return the 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

Assignment operators

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 the right operand to the left operand and assigns the result to the left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts the right operand from the left operand and assigns the result to the 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

```
   print 'Current Letter :', letter

fruits = ['banana', 'apple',  'mango']
for fruit in fruits:        # Second Example
   print 'Current fruit :', fruit

print "Good bye!"
```

When the above code is executed, it produces the following result –

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current fruit : banana
Current fruit : apple
Current fruit : mango
Good bye!
```

Nested Loop

Python programming language allows to use one loop inside another loop. Following section shows few examples to illustrate the concept.

```
for iterating_var in sequence:
   for iterating_var in sequence:
      statements(s)
   statements(s)
```

The syntax for a nested while loop statement in Python programming language is as follows –

```
while expression:
   while expression:
      statement(s)
   statement(s)
```

A final note on loop nesting is that you can put any type of loop inside of any other type of loop. For example a for loop can be inside a while loop or vice versa.

The following program uses a nested for loop to find the prime numbers from 2 to 100 –

```
#!/usr/bin/python

i = 2
while(i < 100):
   j = 2
   while(j <= (i/j)):
      if not(i%j): break
```

Example: This example returns the items from the beginning to, but NOT included, "kiwi":

```
thistuple =
("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[:4])
```

By leaving out the end value, the range will go on to the end of the list:

Example: This example returns the items from "cherry" and to the end:

```
thistuple =
("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[2:])
```

Range of Negative Indexes

Specify negative indexes if you want to start the search from the end of the tuple:

Example: This example returns the items from index -4 (included) to index -1 (excluded)

```
thistuple =
("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[-4:-1])
```

Check if Item Exists

To determine if a specified item is present in a tuple use the `in` keyword:

Example; Check if "apple" is present in the tuple:

```
thistuple = ("apple", "banana", "cherry")
if "apple" in thistuple:
  print("Yes, 'apple' is in the fruits tuple")
```

Update Tuple

Tuples are unchangeable, meaning that you cannot change, add, or remove items once the tuple is created.

Change Tuple Values

Once a tuple is created, you cannot change its values. Tuples are unchangeable, or immutable as it also is called.

But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.