# 

[**Python - Tech Beamers Tutorial**](#_y9ovcv9ot250) **1**

[History of python](#_47y6ovvxnpbp) 1

[Silent Features - Python](#_oaab6sa26lau) 1

[Interest Calculator Program](#_7m7ya6rrjrnw) 1

[Python keywords, variables & identifiers](#_l1f0jldrldya) 2

[Python Keywords](#_2f74cz2y6oya) 2

[Python identifiers](#_2ychmgecqff) 3

# Python - Tech Beamers Tutorial

## History of python

* Originally written by Dutch programmer Guido Van Rossum in the 1980’s
* Currently one of the most polished computer programming languages in the world

## Silent Features - Python

* Python code is highly readable, this makes it reusable and maintainable
* It has broad support for advanced software engineering paradigms such as functional programming
* Clean & elegant coding style
* English-like syntax
  + You never declare a variable

## Interest Calculator Program

|  |
| --- |
| print('Interest Calculator:') amount = float(input('Principal amount ?')) roi = float(input('Rate of Interest ?')) years = int(input('Duration (no. of years) ?')) total = (amount \* pow(1 + (roi/100), years)) interest = total - amount print('\nInterest = %0.2f' %interest) |

**Dissecting this program**

* In the first line, we’re using the print function, which means that the output will be a line of text that reads Interest Calculator
* The next 3 lines use variables to store the input provided by the user
  + The variable labelled amount represents the amount borrowed and is converted to a float
  + The variable labelled roi represents the rate of interest and is converted to a float
  + The years variable represents the duration of the loan and is converted to an integer
* The total represents the amount that will be owed to the bank/loan giver after the loan period has come to an end
* The next line calculates the amount of interest owed by subtracting the amount from the total
* The last line prints the interest amount with up to 2 decimal places

## Python keywords, variables & identifiers

* Basic building blocks of Python programming
* Python keyword -
  + A unique programming term intended to perform an action
  + There are 33 such keywords in python, and build the vocabulary of the python language
  + They represent the syntax and structure of a python program
  + You can’t use their names for defining variables, classes or functions because they are reserved by the programming language

### Python Keywords

* Special words which are reserved and have a special meaning
* Case sensitive
* List of keywords:

|  |  |  |
| --- | --- | --- |
| False | continue | from |
| None | def | global |
| True | del | if |
| and | elif | import |
| as | else | in |
| assert | except | is |
| break | finally | lambda |
| class | for | nonlocal |
| not | raise | while |
| or | return | with |
| pass | try | yield |

* To see the current list of keywords (in your python version) type help> keywords or import keywords, keyword.kwlist

### Python identifiers

* User-defined names to represent a variable, function, class, module or any other object
* To form an identifier, use a sequence of letters either in lowercase or uppercase
  + You can also use numbers and underscores
  + You can’t use numbers or underscores at the start of the identifier
  + No special characters and no keyword names
  + No more than 79 characters
* To test whether the identifier is valid, type in

|  |
| --- |
| >>> import keyword >>> keyword.iskeyword("techbeamers") False >>> keyword.iskeyword("try") True |

* You can also use the str.isidentifier() function
  + This is only available in Python 3 and onwards

|  |
| --- |
| >>> 'techbeamers'.isidentifier() True >>> '1techbeamers'.isidentifier() False >>> 'techbeamers.com'.isidentifier() False >>> 'techbemaers\_com'.isidentifier() True |

### Tips for naming identifiers

* Class names should start with uppercase letters. All other identifiers should begin with a lowercase letter
* Declare private identifiers by using the underscore as the first character
* Avoid using identifiers with only one character

### Python variables

* A variable in Python represents an entity whose value can change as and when required
* It is a memory location which holds the actual value
* And we can retrieve the value from our code by querying the entity
  + However, this requires assigning a label to that memory location so we can reference it
  + We call it as a variable in programming terms
* Variables don’t require declaration
  + However, you must initialize them before using them

For example:

|  |
| --- |
| Vari = 10 |

* This will lead to the following actions:
  + Creation of an object to represent the value 10
  + If the variable “vari” doesn’t exist, it will get created
  + Association of the variable to the value/object so that it can refer to the value
* Once the expression changes, Python assigns a new object to the variable to reference the new value
* An object is just a region of memory which can hold the following:
  + The actual object values
  + A type designator to reflect the object type.
  + The reference counter which determines when it’s OK to reclaim the object