Advanced Certification in AIML

A Program by Datafolkz

Introduction to Python

Intro to Jupyter Notebook

- SHIFT + ENTER for EXECuting the CELL using Jupyter noteboook

Shortcuts in both modes:

- Shift + Enter run the current cell, select below
- Ctrl + Enter run selected cells Alt + Enter run the current cell, insert below Ctrl + S save and checkpoint

In [ ]:

```
 
```

In [22]:

```
print('Hello World')
```

```
Hello World
```

In [23]:

```
a=15
a
```

Out[23]:

```
15
```

- esc + m for code to markdown
- esc + y for markdown to code

Julia , Python and R can be done using JupyterR

About Python

- Python is a scripting langugae
- Its case sensitive
- Its object oriented language

Python program is first compiled and then interpreted. The compilation part is hidden from the programmer thus, many programmers believe that it is an interpreted language. The compilation part is done first when we execute our code and this will generate byte code and internally this byte code gets converted by the python virtual machine(p.v.m) according to the underlying platform(machine+operating system).

An interpreter translates high-level instructions into an intermediate form, which it then executes. In contrast, a compiler translates high-level instructions directly into machine language. Compiled programs generally run faster than interpreted programs. The advantage of an interpreter, however, is that it does not need to go through the compilation stage during which machine instructions are generated. This process can be time-consuming if the program is long. The interpreter, on the other hand, can immediately execute high-level program

Lexical Structure

The lexical structure of a programming language is the set of basic rules that govern how you write programs in that language.

It is the lowest-level syntax of the language and specifies such things as what variable names look like and which characters denote comments.

Each Python source file, like any other text file, is a sequence of characters.

You can also usefully consider it as a sequence of lines, tokens, or statements.

These different lexical views complement and reinforce each other.

Python is very particular about program layout,especially with regard to lines and indentation, so you'll want to pay attention to this information if you are coming to Python from another language.

Lines and Indentation

• A Python program is composed of a sequence of logical lines, each made up of one or more physical lines. Each physical line may end with a comment. A hash sign (#) that is not inside a string literal begins a comment.

• All characters after the # and up to the physical line end are part of the comment, and the Python interpreter ignores them.

• A line containing only whitespace, possibly with a comment, is known as a blank line, and Python totally ignores it.

• In an interactive interpreter session, you must enter an empty physical line (without any whitespace or comment) to terminate a multiline statement.

• However, Python automatically joins adjacent physical lines into one logical line if an open parenthesis ((), bracket ([), or brace ({) has not yet been closed, and taking advantage of this mechanism, generally produces more readable code instead of explicitly inserting backslashes at physical line ends.

• Triple-quoted string literals can also span physical lines. Physical lines after the first one in a logical line are known as continuation lines.

Indentation

• The indentation issues covered next do not apply to continuation lines but only to the first physical line of each logical line.

• Python uses indentation to express the block structure of a program. Unlike other languages, Python does not use braces, or other begin/end delimiters, around blocks of statements; indentation is the only way to denote such blocks.

• Each logical line in a Python program is indented by the whitespace on its left. A block is a contiguous sequence of logical lines, all indented by the same amount; a logical line with less indentation ends the block.

• All statements in a block must have the same indentation, as must all clauses in a compound statement.

• The first statement in a source file must have no indentation (i.e., must not begin with any whitespace).

• Statements that you type at the interactive interpreter primary prompt >>> (covered in Interactive Sessions) must also have no indentation.

Tokens

• Python breaks each logical line into a sequence of elementary lexical components known as tokens. Each token corresponds to a substring of the logical line.

• The normal token types are identifiers, keywords, operators, delimiters, and literals, as covered in the following sections.

• You may freely use whitespace between tokens to separate them. Some whitespace separation is necessary between logically adjacent identifiers or keywords; otherwise, Python would parse them as a single, longer identifier. For example, printx is a single identifier; to write the keyword print followed by the identifier x, you need to insert some whitespace (e.g., print x).

Identifiers

• An identifier is a name used to identify a variable, function, class, module, or other object. An identifier starts with a letter (A to Z or a to z) or an underscore (_) followed by zero or more letters, underscores, and digits (0 to 9).

• Case is significant in Python: lowercase and uppercase letters are distinct. Python does not allow punctuation characters such as @, $, and % within identifiers.

• Normal Python style is to start class names with an uppercase letter and all other identifiers with a lowercase letter.

• Starting an identifier with a single leading underscore indicates by convention that the identifier is meant to be private.

• Starting an identifier with two leading underscores indicates a strongly private identifier; if the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

• The identifier *(a single underscore) is special in interactive interpreter sessions: the interpreter binds* to the result of the last expression statement it has evaluated interactively, if any.

Keywords

Keywords are the reserved words in Python.

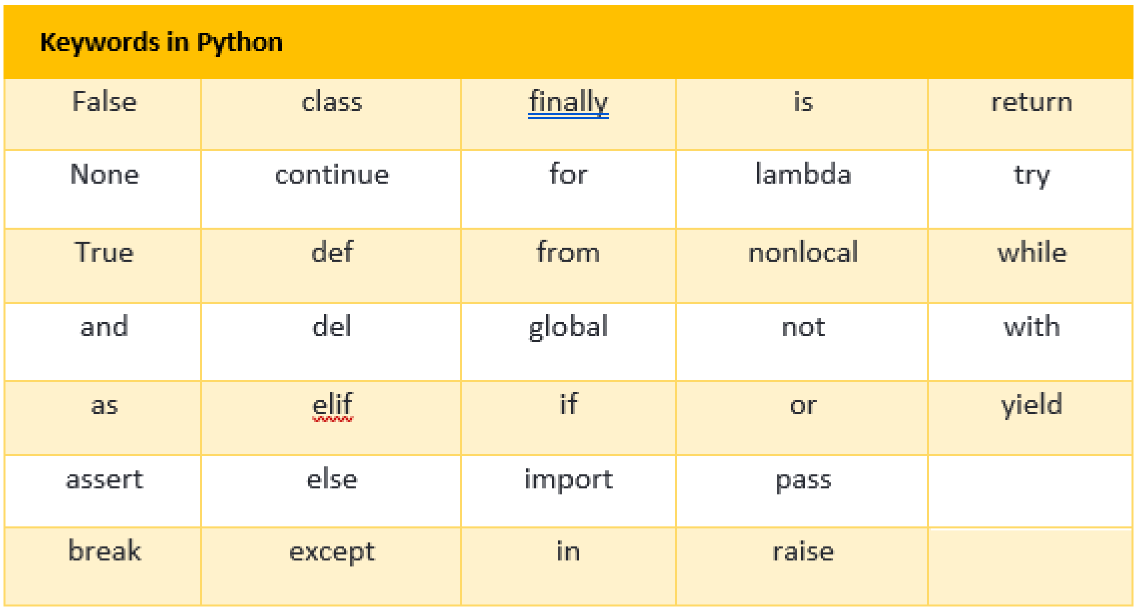
We cannot use a keyword as a variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.

In Python, keywords are case sensitive.

There are 33 keywords in Python 3.7. This number can vary slightly over the course of time.

All the keywords except True, False and None are in lowercase and they must be written as they are.

The list of all the keywords is given below.



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

Operators

Python uses non alphanumeric characters and character combinations as operators.

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Delimiter

- **Used for grouping, punctuations and assignments**
- (), [], {}
- , : . ' = ;
- += -= *= /= //= %=
- &= |= ^= >>= <<= **=

Literals

A literal is a number or string that appears directly in a program.

The following are all literals in Python:

- Integer literal 42
- Floating-point literal 3.14
- Imaginary literal 1.0j
- String literal 'hello'
- Another string literal "world"
- Triple-quoted string literal """Good night"""

Identifier/Variable

Which stores a value

- It is a sequence of characters that consists of letters, digits and underscore
- Can be of any length
- Can starts with a letter which can be either lower or upper case
- Can start with an underscore '_'
- Can start with a digit
- Cannot be a keyword.

Why Statistics is required for Analytics?

Without statistics, the interpretation of data can quickly become massively flawed. Take, for example, the estimated number of German tanks produced during World War II, also known as the “German Tank Problem.” The estimate of the number of German tanks produced per month from standard intelligence data was 1,550; however, the statistical estimate based on the number of tanks observed was 327, which was very close to the actual production number of 342

Similarly, using the wrong tests can also lead to erroneous results.

In general, statistics will help to

• Clarify the question.

• Identify the variable and the measure of that variable that will answer that question.

• Determine the required sample size.

• Describe variation.

• Make quantitative statements about estimated parameters.

• Make predictions based on your data.

In [8]:

```
a1 = 10
print(a1)
a1
```

```
10
```

Out[8]:

```
10
```

Scalar Data Types and Operations

Python has four scalar data types:

- Integer
- Float
- Boolean
- Complex

We will review these in this notebook

Integers

The usual arithmetic operators are available: + for addition, - for subtraction and * for multiplication.

In [9]:

```
print(2 + 4)
print(12878 + 1)
print(7 - 3)
print(127 - 128)
print(8 * 6)
print(17 * 12)
```

```
6
12879
4
-1
48
204
```

There are two division operators: / for everyday (float) division and // for truncating (integer) division

In [10]:

```
print(17 / 4)
print(17 // 4)
```

```
4.25
4
```

The modulo operator % and the exponentiation operator ** are also available.

In [11]:

```
print(18 % 3)
print(72 % 5)
```

```
0
2
```

The usual operator precedence rules and use of parentheses to override that are available

In [12]:

```
print(10 ** 2 * 7 - 3)
print(10 ** (2 * 7) - 3)
print(10 ** (2 * 7 - 3))
```

```
697
99999999999997
100000000000
```

In [13]:

```
a = 123456789
b = a ** 2
c = b ** 3
d = c ** 4
e = d ** 5
print(e)
```

```
958952863472220347910866056628119697853871068539013465444839536426801167
993054293929163280448039645449064125681735868389753896938894669317247873
826022090268514137186729925009454590509679737540744466828401151851734912
328949430367182708052666864754395565918183061461425415310424576068784408
691689720122009297814507919738662894350588988639808054498211481999246620
265727724263303632728183536002926776399146501636024023870460829453965042
281573516177879660268666389774283362707610397338074992295741527447084067
099991981084218755936746700177249185329319981796616794873716079218694887
796144801746541830505541482462955134309726876547981403821624463403579823
482701864443881297209415411583905385414926857107526238526227207683344304
688520972269171491716609536262544669869407652533823580013391749032963471
697200236172979943497791606970504253730471210647484214341251283087641154
070120309491472640661167311715173396372668461182435867576704179829651253
32116703787163047835949441767799201
```

In [ ]:

```
##### Assignment operators
The standard assignment operators are available. That is,  $\alpha$  $\odot=\beta
$ is a shorthand for $\alpha = \alpha \odot \beta$
where $\odot$ is any binary arithmetic operator we saw above
```

In [14]:

```
a = 12
b = 5
a += b
print(a, b)
a -= b
print(a, b)
a *= b
print(a, b)
a **=b
print(a,b)
```

```
17 5
12 5
60 5
777600000 5
```

In [ ]:

```
#### Boolean
The two constants True and False are defined.

The usual boolean operators are also available: ==, !=, >, >=, <, <=
```

In [15]:

```
a = 12
b = 13
print(a == b - 1,a == b, a != b, a < b, a >= b)
```

```
True False True True False
```

Float

Python has a float datatype (and it is the same as C's double!) and the above operations are available

In [17]:

```
a = 12.9
b = 3.6
c = a + b
d = a - b
e = a * b #Watch out for the round off error!
f = a / b
print(a, b, c, d, e, f, sep="\n")
print(a, b, c, d, e, f)
```

```
12.9
3.6
16.5
9.3
46.440000000000005
3.5833333333333335
12.9 3.6 16.5 9.3 46.440000000000005 3.5833333333333335
```

Complex

Complex numbers are built-in

In [18]:

```
z = 3 - 4j
print(z, z.real, z.imag, abs(z), sep="\n")
```

```
(3-4j)
3.0
-4.0
5.0
```

In [19]:

```
x = 5 + 12j
```

In [21]:

```
print(z + x, z - x, z * x, x / z, sep="\n")
```

```
(8+8j)
(-2-16j)
(63+16j)
(-1.32+2.24j)
```

Variables

We have been using variables already. Python variable naming rules are simple and similar to all major programming languages

- must start with a letter
- can have letters, numbers and underscore
- case sensitive

Note that we need not declare variables but using an undefined variable is an error

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: