**Python Revision**

_____

**Variable names** may contain upper-case and lowercase letters (A–Z, a–z), digits (0–9), and underscores (_), but they cannot begin with a digit.
* Variables come into existence when first assigned
to
* A variable can refer to an object of any type

**For example**, each of the following is a valid Python variable name:
**string1**
**_a1p4a**
**list_of_names**

The following aren't valid variable names because they start with a
digit:
**9lives**
**99_balloons**
**2beOrNot2Be**

**Note : In addition to English letters and digits, Python variable names may contain many different valid Unicode characters.**
Unicode is a standard for digitally representing characters used in most of the world's writing systems. That means variable names can contain letters from non-English alphabets, such as decorated letters like é and ü, and even Chinese, Japanese, and Arabic symbols.
However, not every system can display decorated characters, so it's a good idea to avoid them if you're going to share your code with people in different regions.

**Keywords in Python -**

_____
The following words are the keywords, which form the basis of the Python language. You are not allowed to use these words to name your variables, because these are the core commands of Python.

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

**Jargon**

_____

• Complex data types: These are structured or compound types constructed from a sequence of other types of data.
• Constant: A constant value does not change during the execution of the program.
• Constrain: Ensure that the results of a calculation fall between a specified range.
• Hash: A hash is number calculated from a dictionary key to help with storage. This number is designed to be smaller than the key to aid efficiency.
• Immutable: An immutable value that cannot be edited in place.
• Index: An index is a token of an immutable type in square brackets immediately following the variable name, used to point to a specific item in the sequence.
• Iterable: This refers to a code object that can be iterated.
• Iterate: When you loop through items in a sequence one item at a time, you iterate it.
• Iterator: This construct is designed to allow looping.
• Mapping: This refers to a sequence that maps hashable values to arbitrary objects.
• Matrix: A matrix is a multidimensional sequence.

• Method: A method is a function specifically attached to an object or class of objects.
• Mutable: A mutable value can be changed.
• Operation: This action is performed on two variables (operands), usually of a mathematical or logical nature.
• Queue: A queue is a first in, first out (FIFO) structure. You push things in at one end and pop values out of the other.
• Resultant: This value is returned as the result of a process.
• Separator: This text string is used to distinguish between items of data.
• Sequence: A sequence is the simplest sort of complex data type in Python. Lists, strings, and tuples are the types of sequences.
• Sequence packing: Sequence packing is the action of assigning a sequence of comma-separated values to a variable.
• Slice: This refers to a smaller segment of a sequence.
• Stack: This is a last in, first out (LIFO) structure, used rather like the discard pile in a card game.
• Traverse: When you go through the items in a sequence in order, you traverse them.

**Python is a high level general purpose programming language:**

● Because code is automatically compiled to byte code and executed, Python is suitable for use as a scripting language, Web application implementation language, etc.
● Because Python can be extended in C and C++, Python can provide the speed needed for even compute intensive tasks.
● Because of its strong structuring constructs (nested code blocks, functions, classes, modules, and packages) and its consistent use of objects and object oriented programming, Python enables us to write clear, logical applications for small and large tasks.

**Important features of Python:**

● Builtin high level data types: strings, lists, dictionaries, etc.
● The usual control structures: if, ifelse, ifelifelse, while, plus a powerful collection iterator (for).
● Multiple levels of organizational structure: functions, classes, modules, and packages. These assist in organizing code. An excellent and large example is the Python standard library.
● Compile on the fly to byte code Source code is compiled to byte code without a separate compile step. Source code modules can also be "pre-compiled" to byte code files.
● Object oriented Python provides a consistent way to use objects: everything is an object. And, in Python it is easy to implement new object types (called classes in object oriented programming).
● Extensions in C and C++ Extension modules and extension types can be written by hand. There are also tools that help with this, for example, SWIG, sip, Pyrex.

**Sequence types in Python : Immutable types**
Tuples : Made of anything
Strings : Made of characters

**Mutable types**
Lists : Made of anything

**Sample code in Python - Predict the output**

---

```
x = 34 - 23 # A comment.
y = “Hello” # Another one.
z = 3.45
if z == 3.45 or y == “Hello”:
x = x + 1
y = y + “ World” # String concat.
print x
print y
```

---

**Python has five standard data types-**
- Numbers
- String
- List
- Tuple
- Dictionary

**Predict the data types of the following -**

---

```
var1 = 10
var 2 = 1
```

Different numerical data types :
int (signed integers), float (floating point real values), complex (complex numbers)

---

```
str = 'Hello World!'
print (str) # Prints complete string
print (str[0]) # Prints first character of the string
print (str[2:5]) # Prints characters starting from 3rd to 5th
print (str[2:]) # Prints string starting from 3rd character
print (str * 2) # Prints string two times
print (str + "TEST") # Prints concatenated string
```

Output:

---

```
Hello World!
H
llo
llo World!
Hello World!Hello World!
Hello World!TEST
```

---

```
#!/usr/bin/python3
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']
print (list) # Prints complete list
print (list[0]) # Prints first element of the list
print (list[1:3]) # Prints elements starting from 2nd till 3rd
```

```
print (list[2:]) # Prints elements starting from 3rd element
print (tinylist * 2) # Prints list two times
print (list + tinylist) # Prints concatenated lists
```

Output

```
['abcd', 786, 2.23, 'john', 70.200000000000003]
abcd
[786, 2.23]
[2.23, 'john', 70.200000000000003]
[123, 'john', 123, 'john']
['abcd', 786, 2.23, 'john', 70.200000000000003, 123, 'john']
```

---

**Operators in Python :**

---

**Python language supports the following types of operators-**

- Arithmetic Operators (+, -, *, /, %, **, //)
- Comparison (Relational) Operators (==, !=, >, <, >=, <=)
- Assignment Operators ( =, +=, -=, *=, /=. %=, **=, //=)
- Logical Operators (and, or, not)
- Bitwise Operators (&, |, ^, ~, <<, >>)
- Membership Operators (in, not in)
- Identity Operators ( is, is not)