Validation OTP for CodeTantra | CodeTantra Edu

csmssengg.codetantra.com/secure/topic-details1.jsp?cid=65969c70422c5e79de0c160e&tid=65969c71422c5e79de0c16b6&bd=AMTc3M19jdF9jaA==

Q1

100%

L2 / Comments / Understanding Python Comments

A computer `program` is a collection of instructions or statements.

A **Python** program is usually composed of multiple statements. Each statement is composed of one or a combination of the following:

1. Comments
2. Whitespace characters
3. Tokens

Comments : In a computer program, a `comment` is used to mark a section of code as non-executable.

1. Comments can be used to explain Python code, so one can understand the code better.
2. Comments can be used to prevent execution when testing code.

In **Python**, there are two types of comments:

1. **Single-line comments:** These start with a # symbol, and everything after it on the same line is treated as a comment.
2. **Docstring comment:** These are enclosed in triple quotes **(('''') or (""""))**. We'll learn more about these later.

Below code example demonstrates the usage of comments.

```
riverName1 = "Ganga" # A comment can follow after a line of code.
```

Comments1.py

Submit

```
1  # This is my first program
2  # print("I am a Python Guru")
3  print("I am a Python Guru")
4
```

Close | Reset | Submit

Type here to search

Singles Live

ENG US 3:36 PM 10/05/2024

Validation OTP for CodeTantra | CodeTantra Edu

csmssengg.codetantra.com/secure/topic-details1.jsp?cid=65969c70422c5e79de0c160e&tid=65969c71422c5e79de0c16b6&bd=AMTc3M19jdF9jaA==

Q2

100%

L2/Comments/Problem Solving in Comments

Comment and **uncomment** appropriate lines of code so that the given program prints only **odd** numbers.

Sample Test Cases

Test Case 1:
Expected Output:
1
3
5
7
9
11
13

Comments2.py

Submit

```
1  l=[1,3,5,7,9,11,13]
2
3  for i in l:
4      print(i)
```

Close Reset Submit

Type here to search

Singles Live

ENG US 3:36 PM 10/05/2024

Validation OTP for CodeTantra | CodeTantra Edu

csmssengg.codetantra.com/secure/topic-details1.jsp?cid=65969c70422c5e79de0c160e&tid=65969c71422c5e79de0c16b6&bd=AMTc3M19jdF9jaA==

Q3

100%

L2 / Comments / Understanding Docstring in Python

Docstrings:

- **Python** has a way to specify documentation comments known as `docstrings`.
- A `docstring` is a simple text, usually in English, placed in your code to explain what a specific part of the code does.
- A `docstring` is a text enclosed in triple quotes, and it's usually the first non-comment statement in a module, function, class, or method definition.
- This `docstring` becomes a special attribute called **__doc__** for that object.
- **Python** allows the use of both """*triple-double-quotes*""" or '''*triple-single-quotes*''' to create `docstrings`. However, the **Python** coding conventions specification recommends us to use """*triple-double-quotes*""" for consistency.
- The main purpose of **docstrings** in **Python** is to provide information on what a particular **Python** object does and not how it does.
- According to the Python coding conventions, the **docstring** should always begin with a **capital letter** and end with a **period** (`.`)

There are two types of docstrings:

- **One-line docstring:** Provides information what an object does in a single line.
- **Multi-line docstring:** Provides information what an object does in multiple lines.

An example of a one-line docstring is as follows :

```
def add(a, b):
    """Return the sum of given arguments."""
```

DocStringExample.py

Submit

```
1  def add(a, b):
2      """Return sum of given arguments."""
3      return a + b
4
5  def power(b, e):
6      """Return the power value.
7
8      b -- is the base
9      e -- is the exponent
10     """
11     return b ** e
12
13 print(add.__doc__)
14
15 print(power.__doc__)
16
17
```

Close | Reset | Submit

Type here to search

Singles Live

ENG US 3:37 PM 10/05/2024

Validation OTP for CodeTantra | CodeTantra Edu

csmssengg.codetantra.com/secure/topic-details1.jsp?cid=65969c70422c5e79de0c160e&tid=65969c71422c5e79de0c16b6&bd=AMTc3M19jdF9jaA==

Q1

100%

L2 / Identifiers and Keywords / Understanding Identifiers in Python

Identifiers:

- An `identifier` is a name used to identify a variable, function, class, module, or object.
- **Identifier** helps in differentiating one entity from the other.
- Python is a case-sensitive programming language. Meaning, `Age` and `age` are two different **identifiers** in Python.
- Let us consider an example:

```
Name = "codeTantra1"
name = "CodeTantra2"
```

Here the identifiers `Name` and `name` are different, because of the difference in their case.

Rules for writing `identifiers`:

1. `Identifiers` can be a combination of lowercase letters **(a to z)** or uppercase letters **(A to Z)** or digits **(0 to 9)** or an underscore **(_)**.
   Examples: myClass, var_1, print_this_to_screen, _number etc..

2. An identifier can start with an alphabet or an underscore (_), but not with a digit.
   1_variable is invalid, but **variable_1** is perfectly fine.

- [x] **Identifiers** are used for identifying entities in a program.
- [ ] We can use any special character like **@,#,$** as part of **identifiers**.
- [ ] **1st_string** is a valid **identifier**.
- [x] **string_1** is valid **identifier**.
- [x] **Identifiers** can be of any length.

Close | Reset | Submit

Type here to search

Singles Live

ENG US 3:37 PM 10/05/2024

L2 / Identifiers and Keywords / Understanding Python Keywords

Every programming language usually has a set of words known as keywords.

These are reserved words with special meaning and purpose. They are used only for the intended purpose.

> **Note**: We cannot use a keyword as a variable name, function name, or as any other identifier name.

Python also has its own set of **reserved words** called **keywords**.

The interpreter uses the **keywords** to recognize the structure of the program.

Python 2 has **32** keywords while **Python 3.5** has **33** keywords. An extra keyword called **nonlocal** was added in **Python 3.5**.

> **Note: Python 3.5** introduced async and await as context-dependent keywords, and their usage as identifiers was deprecated. In **Python 3.7** async and await were added as keywords, resulting in a total of **35** keywords in **Python 3.7**.

The **33** keywords are as follows:

```
False       class       finally     is          return
```

- [x] Python version **3.5** has **33** keywords.
- [ ] true is a valid keyword in Python.
- [x] The keyword **nonlocal** does not exist in **Python 2**.
- [x] Interpreter raises an `error` when you try to use keyword as a **name** of an entity.
- [ ] A programmer can easily modify the **keywords**.

Close Reset Submit

Validation OTP for CodeTantra | CodeTantra Edu

csmssengg.codetantra.com/secure/topic-details1.jsp?cid=65969c70422c5e79de0c160e&tid=65969c71422c5e79de0c16b6&bd=AMTc3M19jdF9jaA==

Q3

100%

L2 / Identifiers and Keywords / Fill in the missing code

Fill in the missing code in the below program to print whether the given words - **exec**, **nonlocal** and **False** are **keywords** or **not** in **Python** .

Hint: See how the word **and** is verified to be a keyword or not.

Sample Test Cases

Test Case 1:
Expected Output:
and is a keyword : True
exec is a keyword : False
nonlocal is a keyword : True
False is a keyword : True

IsKeywodsExample.py

Submit

```
1  import keyword
2  print('and is a keyword :', keyword.iskeyword('and'))
3
4  #Fill in the missing code in the below lines
5  print('exec is a keyword :', keyword.iskeyword('exec'))
6  print('nonlocal is a keyword :', keyword.iskeyword('nonlocal'))
7  print('False is a keyword :', keyword.iskeyword('False'))
```

Close Reset Submit

Type here to search

Singles Live

ENG US 3:37 PM 10/05/2024