1Q. The two ways to put the comments in python is:

By keeping (hashtag) # before any word type and to make the multi-line comments need to put the comment within two, three double quotes.

2Q. a = “Hello world”

Variable is a, variable value is Hello World

3Q. Carrying out market research and segmentation, Exploring customer behavior, Personalizing recommendations, Predicting trends, Aiding decision-making, Decoding patterns.

4Q. The three steps businesses should take when transitioning to machine learning are: Build an AI and ML-based strategy, Focus on quality data, and Act quickly.

5Q. The six reasons are:

1.Python is extremely versatile, with multiple uses.

2.Python is the fastest growing programming language.

3.Python is in high demand for jobs.

4.Python is easy to read, write, and learn.

5. Python developers make great money.

6.Python has an incredibly supportive community.

According to me it has an incredibly supportive community, every industry wants to search in depth needs.

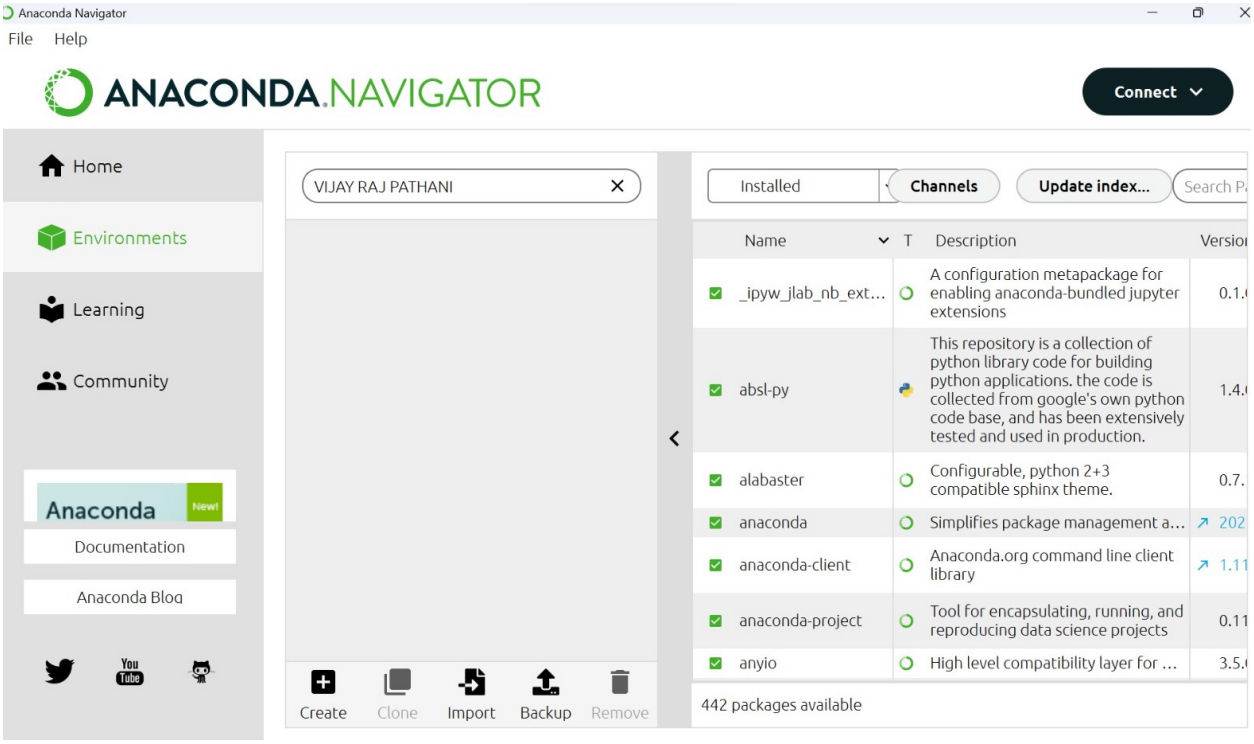
6Q. Guido Van Rossum initially developed Python in the late 1980s, and later it is modeled into English language.

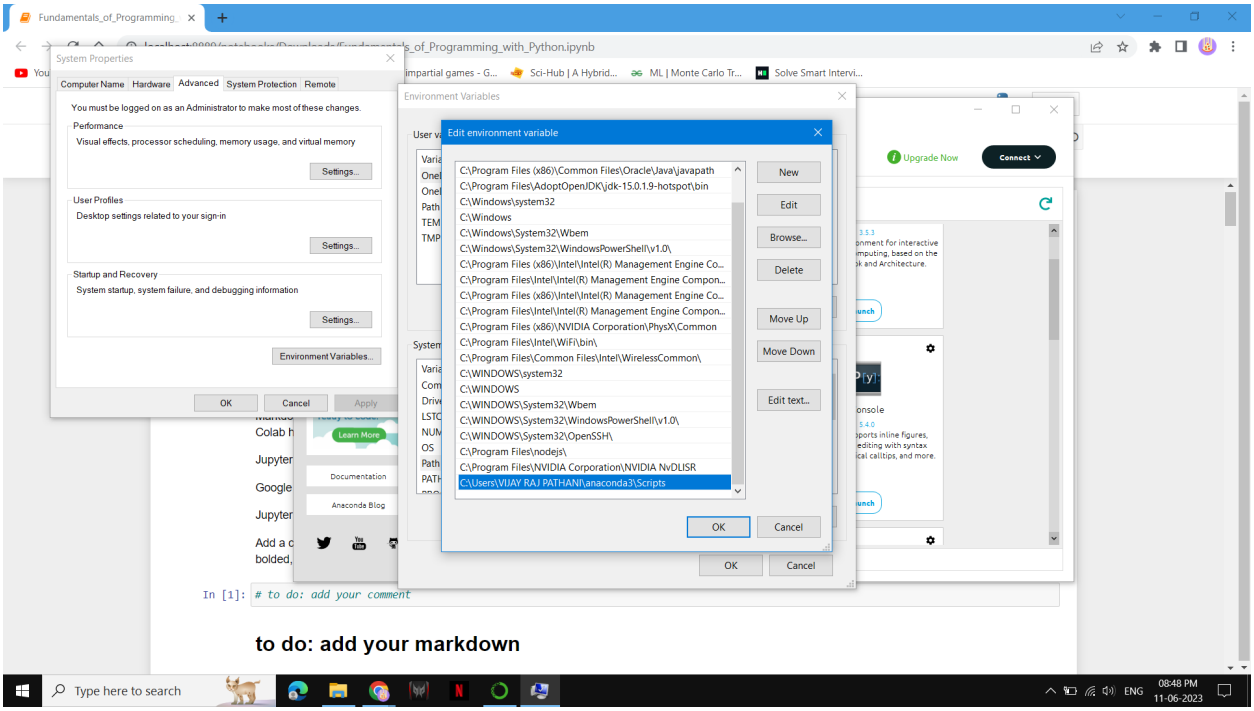
7Q. According to the reports of 2020: 1. Avg Python Developer makes $119,082.

2.Avg JavaScript Developer makes $ 117,718.

3.Avg Ruby Developer makes $121,253.

8Q.


Anaconda Navigator
File Help
ANACONDA.NAVIGATOR
Connect
Home
Environments
Learning
Community
Anaconda
New!
Documentation
Anaconda Blog
VIJAY RAJ PATHANI
Installed
Channels
Update index...
Search P
Name
T
Description
Versio
_ipyw_jlab_nb_ext...
A configuration metapackage for enabling anaconda-bundled jupyter extensions
absl-py
This repository is a collection of python library code for building python applications. the code is collected from google's own python code base, and has been extensively tested and used in production.
alabaster
Configurable, python 2+3 compatible sphinx theme.
anaconda
Simplifies package management a...
anaconda-client
Anaconda.org command line client library
anaconda-project
Tool for encapsulating, running, and reproducing data science projects
anyio
High level compatibility layer for ...
Create
Clone
Import
Backup
Remove
442 packages available


Fundamentals_of_Programming
System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
Environment Variables...
Edit environment variable
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel
Upgrade Now
Connect
In [1]:
to do: add your comment
to do: add your markdown
Type here to search
ENG
08:48 PM
11-06-2023

Fundamentals of Programming:

Programming for Data Science with Python

1. Let's Start with Comments and Markdown

In Python, comments begin with a hash mark (#), a whitespace character and continue to the end of the line.

Markdown is a text-to-HTML conversion tool. Markdown allows you to write visually appealing comments instead of plain text. Jupyter Notebook and Google Colab have Markdown Guides you can review for practicing your skills.

Jupyter: https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Working%20With%20Markdown%20Cells.html (https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Working%20With%20Markdown%20Cells.html)

Google Colab: https://colab.research.google.com/notebooks/markdown_guide.ipynb (https://colab.research.google.com/notebooks/markdown_guide.ipynb)

Jupyter Notebook and Google Colab have two types of cells: text and code. Text cells are for comments or markdown.

Add a comment below providing your **First Name, Last Name, Date** as a plain text comment. Next add markdown with the same information in large font, bolded, centered, and extra flair without using the features provided in the box.

In [1]:

```
# to do: add your comment
```

to do: add your markdown

2. Hello World

In Python, printing hello world is elementary as opposed to other programming languages.

Run the code to print ***Hello World***

Next, enter the code to print ***Hello Professor Floyd***

Run the following code:

In [2]:

```
print ("Hello World")
```

```
Hello World
```

In [13]:

```
print("hello professor floyd")
# to do: add code to print Hello Professor Floyd
```

```
hello professor floyd
```

3. Identifiers, Keywords, and Data Types

Overview

- Identifiers are the names that identify the elements of a program such as variables, functions, classes, methods, constants, etc.
- An identifier should start with a character or Underscore, then use a digit. The characters are A-Z or a-z, an UnderScore (_) , and digit (0-9). Do not use special characters (#, @, $, %, !) in identifiers.
- Identifiers are case sensitive, and they cannot start with digits or numbers.
- Keywords or reserved words cannot be used as identifiers.
- Here is the list of some keywords words in Python:

> and, assert, break, class, continue, def, del, elid, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while

- Data types are an important attribute of data. Python has several standard data types that are built into the interpreter, and the types are numerics, text, sequences, mappings, classes, instances, and exceptions. You can get the data type of any object by using the type() function. The following categories describe the main data types in Python:
- Text Type: str
- Numeric Types: int, float, complex
- Sequence Types: list, tuple, range
- Mapping Type: dict
- Set Types: set, frozenset
- Boolean Type: bool
- Binary Types: bytes, bytearray, memoryview

Let's put what we have learned to practice.

Run the following code:

In [14]:

```
# x is a name of a variable. x is an identifier.
x = 3

print ("x is a variable. It is an identifier. Its value is: ",x, "\n")
print("Data type of x: ", type (x), '\n')

#stdName is a name of a variable that represents the name of a student.
#stdName is an identifier

stdName = "John Smith"

print ("stdName is a variable. It is an identifier. Its value is: ", stdName, "\n")
print("Data type of stdName: ", type (stdName), '\n')
```

```
x is a variable. It is an identifier. Its value is:  3

Data type of x:  <class 'int'>

stdName is a variable. It is an identifier. Its value is:  John Smith

Data type of stdName:  <class 'str'>
```

3.1 Data Containers

Variables are data containers whose values are likely to be changed along the course of executing a program.

Variable/Identifier x:

- x is an identifier. It is a ***variable.***
- its value is 3: 3 is ***data***
- The variable (identifier) x is the ***data container*** that contains the piece of data "3".
- ***Data type*** of x is "int"

Here is an example

Run the following code:

In [15]:

```
# x is a name of a variable. x is an identifier.
x = 3
print ("x is a variable. It is an identifier. Its value is: ", x, "\n")
print("Data type of x: ", type(x), '\n')
```

```
x is a variable. It is an identifier. Its value is:  3

Data type of x:  <class 'int'>
```

Variable/Identifier stdName:

- ***stdName*** is an ***identifier***. It is a ***variable***.
- Its ***value*** is "john Smith": "JohN Smith" is ***data***.
- The ***variable (identifier)*** stdName is the ***data container*** that contains the piece of data "John Smith".
- ***Data*** type of stdName is "str" (or String)

IMPORTANT NOTES:

- In Python, identifiers are unlimited in length. Case is significant.
- However, the user is strongly discouraged from using a very long name to label variables. Besides, the names should be meaningful.

Here is an illustration:

Run the following code:

In [16]:

```
#stdName is a name of a variable that represents the name of a student.
#stdName is an identifier
std = "John Smith"
print ("stdName is a variable. It is an identifier. Its value is: ", stdName, "\n")
print("Data type of stdName: ", type (stdName), '\n')
```

```
stdName is a variable. It is an identifier. Its value is:  John Smith

Data type of stdName:  <class 'str'>
```

3.2 Constants

Constants are data containers that contain permanent values, i.e., these values cannot be changed, along the course of executing a program.

4. Assignments

Overview

The assignment operator, denoted by the “=” symbol, is the operator that is used to assign values to variables in Python.

We use Python assignment statements to assign objects to names. The target of an assignment statement is written on the left side of the equal sign (=), and the object on the right can be an arbitrary expression that computes an object.

Binding a variable in Python means setting a name to hold a ***reference*** to some object.

- Assignments create references, not copies.

Names in Python do not have an intrinsic type. ***Objects have types***.

- Python determines the ***type of the reference*** automatically based on the ***data object*** assigned to it.

A name is created when it appears the first time on the left side of an assignment expression:

> x = 3

A ***reference is deleted*** via garbage collection when NO variables or identifiers refer to it.

4.1. Reference Semantic in Python

Overview: The process of assigning a value

What happens when we type: x = 3?

- First, an integer 3 is created and stored in memory
- Then a name is created
- Next, a reference to the memory location storing the 3 is assigned to the name x

4.2 Assignment manipulates references

- y = x: An assignment of x to y
- y = x: The assignment does not make a copy of the value of x.
- y = x: The assignment makes y refer to the same object as x does.

> Let's say you assign x = 3

> And then you assign y = x

When you assign y = x, you now have y = 3 since x = 3

Here is an example

Run the following code:

In [17]:

```
x=3

y=x

y
```

Out[17]:

```
3
```

5. Operators and Operations

5.1 Numeric Operations

Operator	Operation	Example	Result
+	Addition	33 + 3	36

Operator	Operation	Example	Result
-	Subtraction	33 - 3	30
*	Multiplication	33 * 3	99
/	Division	11 / 3	3.666
%	Remainder	33 % 3	0
**	Exponent	33 ** 3	35937
//	Floor	11 // 3	3

5.2 Augmented Operators

Operator	Operation	Example	Result
+=	Addition assignment	i += 8	i = i + 8
-=	Subtraction assignment	i -= 8	i = i - 8
*=	Multiplication assignment	i *= 8	i = i * 8
/=	Division assignment	i /= 8	i = i / 8
%=	Remainder assignment	i %= 8	i = i % 8
**=	Exponent assignment	i **= 8	i = i ** 8
//=	Floor assignment	i //= 8	i = i // 8

5.3 Relational Operators (a.k.a. Comparison Operators)

- Relational operators, a.k.a. comparison operators, are used for comparisons.
- A comparison, e.g. a>b, is called a conditional expression or boolean expression.
- The result of a boolean expression is either True or False.

Operator	Meaning	Example	Ex. Result
(<)	Less than	radius (<) 0	False
(=<)	Less than or equal to	radius (<=) 0	True
(>)	Greater than	radius (>) 0	True
(>=)	Greater than or equal to	radius (>=) 0	False
(==)	Equal to	radius (==) 0	True

```
|<font size=4>(!=)</font> |<font size=4>Not equal to</font>       |<font size=
4>radius (!=) 0</font>    | <font size=4>False</font>  |
```

5.4 Logical Operators

Operator	Meaning	Example
not	Opposite	not (radius < 0)
and	And	(a > b) or (c < d)
or	Or	(a > b) or (c < d)

5.4.2 and Operators

a and b ***is true** only when ***both a and b are true***.

a	b	a and b
True	True	True
True	False	False
False	True	False
False	False	False

5.5 Identity Operators

In Python, ***identity operators*** are used to check if the ***operands are identical***, i.e., they refer to the same object.

Operator	Result	Example
is	True if the operands are identical, i.e., they refer to the same object;	x is y
is not	True if the operands are not identical, i.e., they do not refer to the same object	x is not y

Let's illustrate what we have learned.

Run the following 3 code blocks:

In [18]:

```
x = 5
y = 5

isTrue = "x is y"
isFalse = "x is not y"

if (x is y):
    print (isTrue)
else:
        print (isFalse)
```

```
x is y
```

In [19]:

```
x = 5
y = 8

isTrue = "x is y"
isFalse = "x is not y"

if (x is y):
    print (isTrue)

else:
    print (isFalse)
```

```
x is not y
```

In [20]:

```
x = 5
y = 5

isTrue = "x is y"
isFalse = "x is not y"

if (x is not y):
    print (isTrue)
else:
        print (isFalse)
```

```
x is not y
```

5.6 Membership Operators

In Python, many data structures have their internal structure of a sequence, e.g., String, List, Tuple, etc.

For example:

- aString = "Hello World"

- aList = [1,2,3,4,5]
- aTuple = (1, 2, 3, 4, 5)

Membership operators - in, not in - are used to test ***if a value is found*** in a sequence or not.

Operator	Result	Example
in	True if the value/variable is found in the sequence;	x in y
not in	True if the value/variable is not found in the sequence	x not in y

Run the following 2 code blocks:

In [21]:

```
x = 'Hello World'
if ('H' in x):
    print ("H in x")
else:
        print ("H not in x")
```

```
H in x
```

In [22]:

```
# You will get an error but think why you have an error.
aList = [1, 2, 3, 4, 5]
if (8 not isn aList):
    print ("8 not in aList")
else:
        print ("8 in aList")
```

```
  Cell In[22], line 3
    if (8 not isn aList):
        ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

6. Pseudo-Code

Overview

In machine learning, many developers write pseudo-code for their machine learning algorithms. Python pseudocode is a syntax-free representation of code. So, the Python pseudocode does not involve any code in it. The Python pseudocode is a very close representation of the algorithmic logic. Read through the following scenario to better explain the purpose of pseudo-code.

6.1 Scenario: A Problem

It is assumed that a software developer is asked to write a Python program that can calculate and print the diameter and the circumference of a circle. The user enters the data of the radius and its measurement unit (in, ft, cm, or m) from the console.

6.2 How to Solve the problem

- First, we need to write pseudo-code (or create a flow-chart) of steps that can solve the problem. Formally, we design an algorithm that offers a solution to the problem.
- Then, based on the steps of the pseudo-code/flowchart (or algorithm), we write the code of the program.

These two steps are the foundation of the software engineering process.

6.3 Pseudo-Code

With pseudo-code, the developer writes down the steps to solve the problem in plain English.

1. Start
2. Read the input of the radius from the console
3. Read the measurement unit of the radius (in, ft, cm, m)
4. Calculate the diameter of the circle
    - diameter = 2*radius
5. Calculate the circumference of the circle
    - Circumference = diameter*PI (3.14159)
6. Print out the diameter
7. Print out the circumference
8. End

Great Job!

You have learned the fundamentals of programming for data science with Python.

In [ ]: