Online Workshop on ‘Applications of Python programming in Data analytics and Machine Learning – Research Perspective’

15.6.20 Day 1 session 1

Dr. Sm. Badhusha

Professor/ CSE department

Alva’s Institute Engineering and Technology

Mijar, Moodbidri, Mangalore

Basics of Python Programming

Objectives of the Day 1 session 1

To acquire knowledge in basic programming constructs in Python

To comprehend the concept of functions in Python

To practice the simple problems in programming constructs and functions in Python

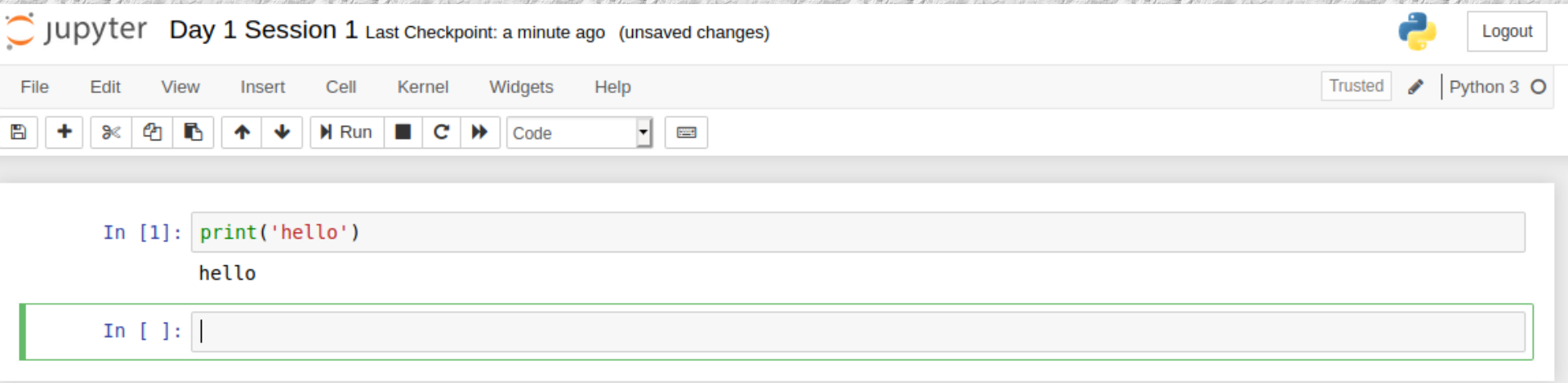
Introduction to Python

- **Python - a general-purpose,Interpreted, interactive, object-oriented and high-level programming language.**
- **Fastest growing open source Programming language**
- **Dynamically typed**
- **Versatile and can be adapted in DA, ML,GUI,Software &Web development**
- **It was created by Guido van Rossum during 1985-1990.**

Python IDEs

- IDLE
- Pycharm
- Spyder
- Thonny
- Atom
- Anaconda -Jupyter Notebook, Ipython for larger project in different domains.
- Google colab

Anaconda activated Jupyter notebook/google colab



Comment lines

- **Single comment line is # comment line**
- **Multiple comment lines triple single quotes or triple double quotes ‘’’ or “””**
- **‘’’ multiple comment lines**

```
        ……
        …. ‘’’
  “““  This is the Program for blah
    blah blah.- multiple comment line”””

    # This is a program for adding 2 nos
```

Multiple Assignment

- **You can also assign to multiple names at the same time.**

```
>>> x, y = 2, 3
>>> x
2
>>> y
3
```

Swapping assignment in Python

```
x,y=y,x
```

Reserved Words

(these words can't be used as varibles)

and	exec	Not
as	finally	or
assert	for	pass
break	from	print
class	global	raise
continue	if	return
def	import	try
del	in	while
elif	is	with
else	lambda	yield

Indentation and Blocks

- Python doesn't use braces({}) to indicate blocks of code for class and function definitions or flow control.

- Blocks of code are denoted by line indentation, which is rigidly enforced.

- All statements within the block must be indented the same level

- **Python uses white space and indents to denote blocks of code**
- **Lines of code that begin a block end in a colon:**
- **Lines within the code block are indented at the same level**
- **To end a code block, remove the indentation**

Python data types

Dynamically Typed: Python determines the data types of variable bindings in a program automatically.

But Python's not casual about types, it enforces the types of objects.

"Strong Typed"

So, for example, you can't just append an integer to a string. You must first convert the integer to a string itself.

```
x = "the answer is " # Decides x is bound to a string.
y = 23       # Decides y is bound to an integer.
print x + y  # Python will complain about this.
```

Conditional Execution

- **if and else**

```
if v == c:
        #do something based on the condition
else:
        #do something based on v != c
```

- **elif allows for additional branching**

```
if condition:
elif another condition:
…
else: #none of the above
```

```
# python program for finding greater of two numbers
a=int(input('Enter the first number'))
b=int(input('Enter the first number'))

if a>b:
   print("The greater number is",a)
else:
   print("The greater number is",b)

# for satisfying equality condition
if a>b:
   print("The greater number is",a)
elif a==b:
   print("both numbers are equal",a)
else:
    print("The greater number is",b)
```

Nested conditionals

One conditional can also be nested within another. We could have written the three-branch example like this:

```
a=int(input('Enter the first number'))
b=int(input('Enter the first number'))

if a==b:
   print("Both the numbers are equal",a)
else:
   if a>b:
         print("The greater number is",a)
   else:
        print("The greater number is",b)
```

Variables, expressions, and statements

```
python
>>> print(4)
4
```

If you are not sure what type a value has, the interpreter can tell you.

```
>>> type('Hello, World!')
<class 'str'>
>>> type(17)
<class 'int'>
>>> type(3.2)
<class 'float'>
>>> type('17')
<class 'str'>
>>> type('3.2')
<class 'str'>
```

If you give a variable an illegal name, you get a syntax error:

```
>>> 76trombones = 'big parade'
SyntaxError: invalid syntax

>>> more@ = 1000000
SyntaxError: invalid syntax

>>> class = 'Advanced Theoretical Zymurgy'
SyntaxError: invalid syntax
```

Operators and operands

a=20 b=10

+ Addition Adds values on either side of the operator.
a + b = 30
- Subtraction Subtracts right hand operand from left hand operand.
a – b = -10
*** Multiplication Multiplies values on either side of the operator**
a * b = 200
/ Division Divides left hand operand by right hand operand
b / a = 2.0

// Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.

9//2 = 4 and 9.0//2.0 = 4.0

% Modulus Divides left hand operand by right hand operand and returns remainder
b % a = 0

**** Exponent Performs exponential power calculation on operators**

ab =10 to the power 20**

Relational Operators

== equal to

!= or <> not equal to

> greater than

>=greater than or equal to

< less than

<= less than or equal to

Python Assignment Operators

= Assigns values from right side operands to left side operand c = a + b assigns value of a + b into c

+= Add AND It adds right operand to the left operand and assign the result to left operand c += a is equivalent to c = c + a

-= Subtract AND It subtracts right operand from the left operand and assign the result to left operand c -= a is equivalent to c = c – a

***= Multiply AND It multiplies right operand with the left operand and assign the result to left operand c *= a is equivalent to c = c * a**

/= Divide AND It divides left operand with the right operand and assign the result to left operand c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a

%= Modulus AND It takes modulus using two operands and assign the result to left operand c %= a is equivalent to c = c % a

****= Exponent Performs exponential power calculation on operators and assign value to the left c **= a is equivalent to c = c ** a**

The + operator works with strings, but it is not addition in the mathematical sense.

Instead it performs concatenation, which means joining the strings by linking them end to end. For example:

```
>>> first = 10
>>> second = 15
>>> print(first+second)
25

>>> first = '100'
>>> second = '150'
>>> print(first + second)
100150
```

Functions

A function is a block of organized, reusable code that is used to perform a single, related action.

Functions provide better modularity for your application and a high degree of code reusing.

Syntax for function definition

```
def functionname( parameters ):
    function_suite
    return [expression]
```

Example :

```
def great2(x,y) :
    if  x > y :
        return x
    else:
        return y
```

Special feature of function in Python is that it can return more than one value

Calling the Function

```
def great2(x,y) :
    if  x > y :
        return x
    else:
        return y


a=int(input('Enter a'))
b=int(input('Enter b'))

print('The greater number is', great2(a,b))
```

Catching exceptions using try and except

```
inp = input('Enter Fahrenheit Temperature:')

try:
    fahr = float(inp)
    cel = (fahr - 32.0) * 5.0 / 9.0
    print(cel)
except:
    print('Please enter a valid number')
```

Concluding Tips

Interpreted ,Object oriented and open sourced Programming language

Developed by Guido van Rossum during 1985-90

Dynamically typed but strongly typed language

Indented language which has no block level symbols {}

No ; is necessary . Block beginning starts with :

function starts with def key word followed by function name and :

#- single comment line ''' - multiple comment line

if...else if...elif ...elif No endif

Multiple assignment x,y,z=2,4,5 is possible

/ - divide with precision //- floor division (no precision)