UT01: INTRODUCTION TO PYTHON

25/02/2025
L2

IDE · Anaconda / Google Collab
↳ Jupyter Notebook → extentision: .ipyb

Cell - Where small codes are written in jypyter Notebook.

LIBRARY : Pandas, Numpy, matplotlib
↳ pip ← Installation command of a lib.

PYTHON : (Features)
(1) Open Source
(2) OOP
(3) Interpreted Language

26.02.25 L3

```
import pandas
pandas.__Version__
```
← Example for checking version.

To check type

```
type(a)
```

Literal : Const. value, contain digits, decimals floating. No , b/w it. → Sequence of any char or int.

1024 ← Literal

```
a = 5
type(a)
```
≥: int

26 / 2 / 2025
L3

1, 124	0.1124
≥: Error	≥: 0.1124

Range & Overflow:

int	2 Bytes	-32767 to 32768
		0 to
char	1 Bytes	-127 to 128
		0 to 255

Bin. of 20

10100

- No limit the size of an int
- Floating have both limit & limited precision
- Double precision standard format (IEEE 754). 10^{-308} to 10^{308} for floating range. w/ 16 to 17 digits of precision.
- Multiplication of two values can create a overflow situation.

1.5e200 * 2.0e210 #overflow situation
≥: inf

→ To overcome the overflow situation, we have to increase/change data types.

__/__/____

Underflow: If no. is 10^{-308}, then its an underflow situation.

$$\text{Underflow} < \left(\overset{\text{Float}}{\leftarrow 10^{-308}}\right) > \text{Overflow}$$
(÷ situation) $10^{308} \rightarrow$ (* situation)

Built in Format Function:
Its used to produce a numeric string version of the value containing a specific no. of decimal places.
Syntax: format(value, 'value')

```
format(12/5, '.2f')   # '.2f' means after
>: '2.40'             # two place

format( 1/3, '.3f')
>: '0.333'

# for very large number
format( 2 ** 100, '6e')  # ** exponential
                         # symbol.

>:  '1.267651e + 30'


tax = 0.08

print('Your cost: $', (1+ tax) * 12.99)
print('Your cost: $ ', format((1+tax) * 12.99,
      '.2f'))

>: Your cost: 14.0292000000001
   Your cost : 14.02
```

__/__/____

String Literals:
A sequence of char. delimited by a matching pair of either single (') or double (" ") or triple quotes (""").

- Must be contained all on one line except when delimited by triple quotes.

Representation of charcters values:

UTF8 & ASCII (0 - 255)

Capital	:	65	← Start
Small	:	97	← Start
Numbers	:	48	← Start

```
 ord('1')              # ASCII = ord value
>: 49
 chr(97)
>: a
```

Control Chorcters: Special chars that aren't displayed on the screen. Use for controlling the display output. [escape sequenc].

\n → New/Next line

String Formating: Same format fn, discussed earlier.

Syntax:

```
format(value, format_specifier)
```

_/_/_

```
format('Priyanshu','<20') # left justified
≥: 'Priyanshu           '
format('Hello', '>20')# Right justified
≥:'                Hello'
format('Hello', '^20')# Centered Justified
≥:'       Hello        '
```

Implicit & Explicit Line Joining

↳ To Reduce line tracking, its better to use a format (Json format) etc.
' , ' is used for joining two lines.

```
_nam = "Rian"
_age = 18
_id = 110
print('Name:'_name, 'Age:'_age,
      'Identity No:' _id)

≥: Name: Rian Age: 18 Identity No: 110
```

↳ Explicit line Joining by usin '\' character.

```
yr_birth = 2008
avg_nums_yer = 31560038
month_birth = 2
NumofSec = ((C yr_birth - 1900)* avg_nums
_yer )+ \ (month_birth - 1)*) ~~avg~~
```

27/2/25
L3

Variables & Identifiers:
↳ Which can vary   ↳ sequence of one or more charcters.

Rules:-
(1) Python is a case sensitive lang., thus line is diff. from line.
(2) Identifiers may contain letters & digits, but can't begin with digits.
(3) '_' is allowed, no space, underscore shouldn't be first char.
(4) Quotes are not allowed
(5) Keyword can'ot be used.

# id = address

```
k = 50
print(id(k))
num = 50
print(id(num))
```

≥:
```
10752424
10752424
```

Efficent memory allocation.

Keyboard input

```
name = input('Whats your name?')
print("Hello", name, "\n")
```

≥:
```
Whas your name? Ps
Hello Ps
```

#keybord i/p for ints.

```
a = int(input('enter the value of a:'))
print (a:-a )

≥: Enter the value of a: 5
   a = 5
```

⇒ By default input function reads input as a string.

OPERATOR: Its a symbol Represent an operation that may be performed one or more operands.

$$\begin{tikzcd}
 & \text{Operator} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{Uranary} & \text{Binary} & \text{Arithemetic}
\end{tikzcd}$$

Uranary	Binary	Arithemetic
One operand req.	two operand with addition operator	(+ , − , * , /) (% ← Remainder)

```
y = 5
x = 10
print('x // y =', x// y ) # truncating division
                          # or floor division
≥: 2
z = 8
print ('x%z=', x%z )
≥: 2
print ('x** z', x**z ) # Power
```

1/3/25

L5

COMPUTATIONAL & PROBLEM SOLVING

```
# Fn w/out argument
def hell(): ← function defination
    print("Hello")
    print("How are you?")
```

#importing & using

```
    hell()          # function calling
  ≥: Hello
       How are you?
```

#Function with argument:

```
 p = "User"
def greet(p)
   print("Hi", p)

≥:

  greet(p)
 ≥: Hi User
```

#defing of path of a program (Goolgle Collab)

```
! python      /conter/vs.code/test.py
```