S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

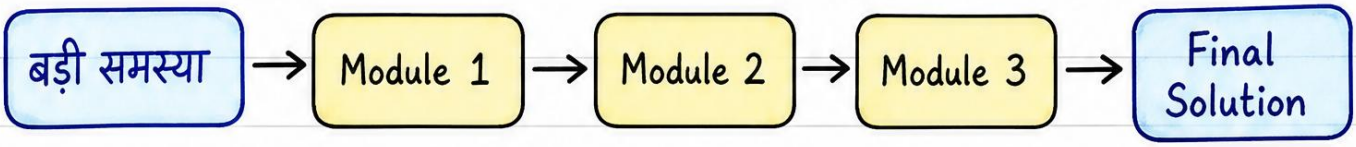
Unit 6: Functions

Top-down Approach, Modular Programming और Functions

★ Top-down approach में बड़ी समस्या को छोटी-छोटी समस्याओं में बाँटकर क्रम से हल किया जाता है.

★ Modulor programming में program को छोटे independent modules में बाँटा जाता है.

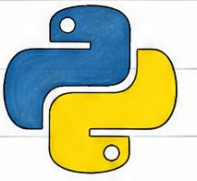
★ Function एक reusable code block है जो specific task करता है.



Function का उदाहरण

```
def greet():
    print("Hello")

greet()
```



python

Without Function	VS	With Function

`print("Hello")`
`print("Hello")`
`print("Hello")`
⋮
एक ही काम बार-बार लिखा गया है.
		`def greet():`
		`  print("Hello")`
		
		`greet()`
		`greet()`
		`greet()`
		⋮
		Function को जहाँ ज़रूरत हो, वहाँ बार-बार call कर सकते हैं.

★ Reusability (पुन: उपयोगिता): Function को एक बार लिखकर हम उसे कई जगह पर बार-बार उपयोग कर सकते हैं. इससे समय की बचत होती है, कोड छोटा, साफ और आसान हो जाता है तथा त्रुटियाँ कम होती हैं.

S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

Unit 6: Functions

Function Parameters, Local Variables, Return Statement, Doc Strings और global Statement

1 Parameter

→ Parameter function को input देता है।

उदाहरण (Example)

```
def add(a, b):
        ↑  ↑  ↑
    return a + b
```

Parameter

`a` और `b` Parameter हैं।

2 Argument

→ Argument वह value होती है जो function को call करते समय दी जाती है।

उदाहरण (Example)

```
result = add(3, 4)
             ↑  ↑
          Argument
```

यहाँ 3 और 4 Argument हैं जो `a` और `b` Parameters को भेजे जा रहे हैं।

3 Local Variable

→ Local variable केवल function के अंदर उपलब्ध होता है और function के बाहर उपयोग नहीं किया जा सकता।

उदाहरण (Example)

```
def add(a, b):
    total = a + b   # total एक local variable है
    return total
result = add(3, 4)
# total यहाँ उपलब्ध नहीं है (NameError आएगा)
```

4 Return Statement

→ return statement function से value वापस भेजता है।

उदाहरण (Example)

```
def add(a, b):
    return a + b
result = add(3, 4)   # result में 7 store होगा
```

5 Doc String और global statement

→ Doc String function का description है और triple quotes में लिखा जाता है।

→ global statement function के अंदर global variable बदलने देता है।

Doc String का उदाहरण (Example)

```
def square(n):
    """यह number का square लौटाता है।"""
    return n * n
```

---

global statement का उदाहरण (Example)

```
count = 0          # global variable
def increase():
    global count   # यह लाइन बताती है कि हम global
    count += 1     # variable को बदलना चाहते हैं
```

सारांश (Summary)

- Parameter function को input देता है और Argument वह value है जो call करते समय दी जाती है।
- Local variable केवल function के अंदर उपलब्ध होता है।
- return statement function से value वापस भेजता है।
- Doc String function का description है और triple quotes में लिखा जाता है।
- global statement function के अंदर global variable बदलने देता है।

S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

Unit 6: Functions

Default Argument Values, Keyword Arguments और VarArgs Parameters

- Default argument का default value होता है।
- Keyword argument में parameter name के साथ value दी जाती है।
- VarArgs अनेक positional arguments स्वीकार करता है और *args tuple के रूप में मिलता है।

1) Default Argument

```
def greet(name, message="Good Morning"):
    print(message, name)
greet("Ravi")
```

2) Keyword Argument

```
def student(name, age):
    print(name, age)
student(age=20, name="Asha")
```

3) VarArgs Parameter

```
def total(*args):
    return sum(args)
print(total(2, 4, 6))  gives 12
```

Syntax	Input	Example

def function(param, param2=default_value):
  statements
	कम arguments दिए जा सकते हैं। बचे हुए के लिए default value उपयोग होता है।	def greet(name, message="Good Morning"):
		  print(message, name)
		greet("Ravi")

def function(param1, param2):
  statements
	arguments को parameter name के साथ किसी भी क्रम में दिया जा सकता है।	def student(name, age):
		  print(name, age)
		student(age=20, name="Asha")

def function(*args):
  statements
	कोई भी संख्या में positional arguments दिए जा सकते हैं; *args tuple के रूप में मिलता है।	def total(*args):
		  return sum(args)
		print(total(2, 4, 6)) gives 12

S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

Unit 6: Functions

Library Functions: input(), eval() और print()

★ input() यूज़र से data लेता है और सामान्यत: string देता है।

★ eval() string expression को Python expression के रूप में evaluate करता है।

★ print() output दिखाता है।

Function	कार्य	Example
input()	यूज़र से input लेता है और string के रूप में देता है।	name = input("Name: ") print(name)
eval()	string expression को Python expression के रूप में evaluate करता है।	x = eval("10 + 5") print(x)
print()	output स्क्रीन पर दिखाता है।	print("Sum =", 3 + 4)

①
```
name = input("Name: ")
print(name)
```
⇒
```
Input: Name: Rohan
Output: Rohan
```

②
```
x = eval("10 + 5")
print(x)
```
⇒
```
Output: 15
```

③
```
print("Sum =", 3 + 4)
```
⇒
```
Output: Sum = 7
```

⚠ eval() का उपयोग केवल trusted input के साथ करें।

S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

Unit 6: Functions

String Functions – Part 1

Python में String पर कई उपयोगी functions होते हैं जो String को खोजने, बदलने, या उसका रूप बदलने में मदद करते हैं। नीचे कुछ महत्वपूर्ण String functions और उनके उदाहरण दिए गए हैं।

Function	Example
count() occurrences गिनता है:	"banana".count("a") = 3
find() पहला index देता है:	"Python".find("th") = 2
rfind() आखिरी index देता है:	"banana".rfind("a") = 5
capitalize() पहला अक्षर capital करता है:	"hello world".capitalize() = "Hello world" "Python Programming".capitalize() = "Python programming"
title() हर word का पहला अक्षर capital करता है:	"hello world".title() = "Hello World" "Python Programming".title() = "Python Programming"
lower() lowercase बनाता है:	"hello world".lower() = "hello world" "Python Programming".lower() = "python programming"
upper() uppercase बनाता है:	"hello world".upper() = "HELLO WORLD" "Python Programming".upper() = "PYTHON PROGRAMMING"
swapcase() case बदलता है:	"hello world".swapcase() = "HELLO WORLD" "Python Programming".swapcase() = "pYTHON pROGRAMMING"

Note:
String immutable होता है; functions नया String लौटाते हैं।

S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

Unit 6: Functions

String Functions – Part 2, Slicing, Membership और Pattern Matching

फंकशन	परिभाषा / उदाहरण
islower()	चेक करता है कि सभी अक्षर लोअरकेस हैं या नहीं। उदाहरण: "Hello".islower() = False
isupper()	चेक करता है कि सभी अक्षर अपरकेस हैं या नहीं। उदाहरण: "hello".upper()
istitle()	चेक करता है कि हर शब्द का पहला अक्षर कैपिटल है या नहीं। उदाहरण: "Python Is Fun".istitle() = True
replace()	सब्स्ट्रिंग को दुसरी सब्स्ट्रिंग से बदलता है। उदाहरण: "banana".replace("a", "o") = "bonono"
strip()	स्ट्रिंग के दोनों ओर से व्हाइटस्पेस हटाता है। उदाहरण: " Python ".strip() = "Python"
lstrip()	स्ट्रिंग के बाएँ ओर से व्हाइटस्पेस हटाता है। उदाहरण: " Python".lstrip() = "Python"
rstrip()	स्ट्रिंग के दाएँ ओर से व्हाइटस्पेस हटाता है। उदाहरण: "Python ".rstrip() = "Python"
split()	डिलीमीटर के आधार पर स्ट्रिंग को लिस्ट में तोड़ता है। उदाहरण: "a,b,c".split(",") = ["a", "b", "c"]
partition()	स्ट्रिंग को दिए गए सेपरेटर पर 3 भागों में बाँटता है (पहले, सेपरेटर, बाद में)। उदाहरण: "key=value".partition("=") = ("key", "=", "value")
join()	किसी इटरेबल की सभी स्ट्रिंग्स को जोड़कर एक स्ट्रिंग बनाता है। उदाहरण: "-".join(["2026", "08", "26"]) = "2026-08-26"
isspace()	चेक करता है कि स्ट्रिंग केवल व्हाइटस्पेस से बनी है या नहीं। उदाहरण: " ".isspace() = True
isalpha()	चेक करता है कि स्ट्रिंग में केवल अक्षर हैं या नहीं। उदाहरण: "Python".isalpha() = True
isdigit()	चेक करता है कि स्ट्रिंग में केवल डिजिट हैं या नहीं। उदाहरण: "12345".isdigit() = True
isalnum()	चेक करता है कि स्ट्रिंग में केवल अक्षर और डिजिट हैं या नहीं। उदाहरण: "Python2026".isalnum() = True
startswith()	चेक करता है कि स्ट्रिंग किसी दिए गए पैटर्न से शुरू होती है या नहीं। उदाहरण: "Python".startswith("Py") = True
endswith()	चेक करता है कि स्ट्रिंग किसी दिए गए पैटर्न पर समाप्त होती है या नहीं। उदाहरण: "Python".endswith("on") = True
encode()	स्ट्रिंग को बाइट्स में एन्कोड करता है। उदाहरण: "Python".encode("utf-8") = b'Python'
decode()	बाइटस को स्ट्रिंग में डिकोड करता है। उदाहरण: b'Python'.decode("utf-8") = "Python"

String Slicing

स्ट्रिंग के हिस्से (सब्स्ट्रिंग) निकालने के लिए स्लाइसिंग का उपयोग किया जाता है।

```
s = "Python"
s[1:4] = "yth"
s[0:6] = "Pytho"
s[2:]  = "thon"
s[:3]  = "Pyt"
s[-3:] = "hon"
```

Membership

in और not in का उपयोग सब्स्ट्रिंग की उपस्थिति जाँचने के लिए किया जाता है।

```
"th" in "Python" = True
"z" in "Python" = False
"Py" not in "Python" = False
"on" in "Python" = True
```

Pattern Matching

startswith() और endswith() का उपयोग पैटर्न मिलान के लिए किया जाता है।

```
"Python".startswith("Py") = True
"Python".startswith("py") = False
"Python".endswith("on") = True
"Python".endswith("th") = False
```

S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

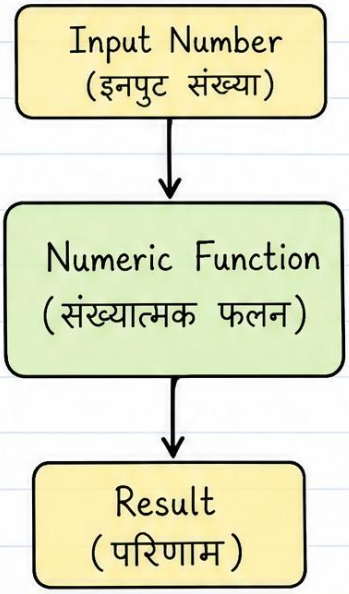
Unit 6: Functions

Numeric Functions और Date & Time Functions

1. Numeric Functions

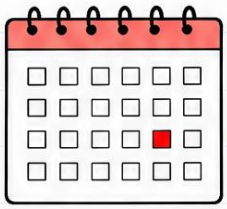
Numeric Functions का उपयोग संख्याओं पर विभिन्न प्रकार की गणनाएँ करने के लिए किया जाता है। ये फलन गणितीय कार्यों को सरल और तेज़ बनाते हैं।

Function (उदाहरण)	Result (परिणाम)
eval("2 + 3")	= 5
max(4, 9, 2)	= 9
min(4, 9, 2)	= 2
pow(2, 3)	= 8
round(3.1416, 2)	= 3.14
int(5.9)	= 5
random.randint(1, 6)	gives a random integer
ceil(3.2)	= 4
floor(3.8)	= 3
sqrt(25)	= 5



2. Date & Time Functions

Date & Time Functions का उपयोग तिथि (date), समय (time), दिन (day), माह (month), वर्ष (year) और current timestamp प्राप्त करने के लिए किया जाता हैं।




```
import datetime
today = datetime.date.today()
now = datetime.datetime.now()
print(today)
print(now)
```

इससे date, time, day, month, year और current timestamp प्राप्त किए जा सकते हैं।

★ इन फलनों का उपयोग प्रोग्राम में समय और संख्या से संबंधित कार्यों को आसान बनाता है।

S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

Unit 6: Functions

Recursion

Recursion में function स्वयं को call करता हैं.
हर recursive function में base case होना आवश्यक है,
वरना infinite recursion हो सकती है.

```
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)

print(factorial(5))  # gives 120
```

Base Case	Recursive Case
जब n == 0 होता है, तब function 1 return करता है. यही recursion रुकने की स्थिति है.	जब n > 0 होता है, तब function n * factorial(n - 1) के रूप में स्वयं को call करता है.

Step-by-Step (Call Flow):

factorial(5) → 5 × factorial(4) → 4 × factorial(3) → 3 × factorial(2) → 2 × factorial(1) → 1

Return Flow (Values Returning):

1 → 2 → 6 → 24 → 120

factorial(0) returns 1 | 2 × 1 = 2 | 3 × 2 = 6 | 4 × 6 = 24 | 5 × 24 = 120 (Final Result)

Advantages (लाभ)

- problem को छोटे भागों में हल करना आसान.

Limitation (सीमा)

- अधिक calls से memory बढ़ सकती है.