



پانٹھن میں کنٹرول اسٹرکچرز

حاصلِ تعلیم: اس باب کے اختتام تک، آپ اس قابل ہو جائیں گے کہ:



- پانٹھن میں کنٹرول اسٹرکچرز کو نافذ کر سکیں، جیسے کہ فیصلہ سازی کی سٹیٹمنٹس اور لوپس۔
- پانٹھن کے ماڈیولز، فنکشنز، اور بلاکس ان ڈیٹا اسٹرکچرز جیسے کہ لسٹس (Lists) کے ساتھ کام کر سکیں۔
- لسٹس (Lists) پر مختلف آپریشنز پر فارم کر سکیں۔

تعارف

پروگرامنگ میں، ہمیں اکثر مختلف شرائط (conditions) کی بنیاد پر اپنے پروگرام کے فلو کو کنٹرول کرنے یا کسی خاص عمل کو کئی بار دہرانے کی ضرورت ہوتی ہے۔ یہی وہ مقام ہے جہاں کنٹرول اسٹرکچرز اپنا کردار ادا کرتے ہیں۔

کنٹرول اسٹرکچرز بہت ضروری ہیں کیونکہ یہ ہمارے پروگراموں میں فیصلہ سازی کے عمل (decision-making process) اور کاموں کی تکرار (repetition of tasks) کو منظم کرنے میں ہماری مدد کرتے ہیں۔

کنٹرول اسٹرکچرز کی دو اہم اقسام ہیں:

1. فیصلہ سازی (Decision making)
2. تکرار (Looping)

4.1 فیصلہ سازی

پروگرامنگ میں فیصلہ سازی پروگرام کو شرائط (Conditions) کی بنیاد پر مختلف اقدامات کا انتخاب کرنے کی اجازت دیتا ہے۔ یہ بالکل ویسا ہی ہے جیسے ہم حقیقی زندگی میں فیصلے کرتے ہیں۔ مثال کے طور پر، موسم کی صورت حال کی بنیاد پر چھتری لینے یا نہ لینے کا فیصلہ کرنا۔

پانٹھن فیصلہ سازی کو نافذ کرنے کے لیے کئی طرح کے کنڈیشنل سٹیٹمنٹس فراہم کرتا ہے۔

if سٹیٹمنٹ

if سٹیٹمنٹ ہمیں شرائط کی بنیاد پر فیصلے کرنے دیتا ہے۔

اگر شرط درست (True) ہو، تو یہ کوڈ کے ایک بلاک (Block) کو چلاتا ہے۔

پائنتھن میں، انڈینٹیشن (Indentation) ہی آپ کے کوڈ کے ڈھانچے / بلاک یا دائرہ کار (Scope) کا تعین کرتی ہے۔
پائنتھن میں مناسب انڈینٹیشن لازمی ہے۔ غلط انڈینٹیشن کی وجہ سے آپ کے کوڈ میں "Indentation Error" آتا ہے۔

if سٹیٹمنٹ کا سنٹیکس

Syntax of if statement

if condition:

code to run if the condition is true

اگر شرط درست ہو:

یہ کوڈ اس وقت چلے گا جب شرط درست (True) ہو

مثال: اگر درجہ حرارت 30 ڈگری سے زیادہ ہو، تو ہم ایک پیغام (مثلاً: "آج بہت گرمی ہے!") پرنٹ کرتے ہیں۔

```
temperature = 35
if temperature > 30:
    print("It's a hot day")
# Output: It's a hot day
```

if-else (Statement)

if-else ہمیں اجازت دیتا ہے کہ اگر شرط درست (True) ہو تو کوڈ کا ایک بلاک چلائیں اور اگر شرط غلط (False) ہو تو کوڈ کا دوسرا بلاک چلائیں۔

if - else سٹیٹمنٹ کا سنٹیکس

if-else کا ڈھانچہ۔

if کنڈیشن: شرط درست ہونے پر عمل درآ مد ہوتا ہے۔

else بلاک: شرط درست نہ ہونے (یعنی غلط ہونے) پر عمل درآ مد ہوتا ہے۔

مثال:

if-else سٹیٹمنٹ کا استعمال کرتے ہوئے درجہ حرارت چیک کرنے کے لیے ایک پائنتھن پروگرام لکھیں۔

اگر درجہ حرارت 30 ڈگری سے زیادہ ہو تو پروگرام یہ دکھائے 'It's a hot day'

اور اگر درجہ حرارت 30 ڈگری سے کم یا برابر ہو تو پروگرام یہ ('It's not a hot day') دکھائے

```
Temperature = 15
if temperature > 30:
    print("It's a hot day ")
else :
    print("It's not a hot day ")
# It's not a hot day
```

کبھی کبھی ہمیں ایک شرط کے اندر دوسری شرط چیک کرنے کی ضرورت ہوتی ہے۔ اسے نیسٹنگ کہا جاتا ہے۔

Syntax of nested if statement

if condition1:

if condition 2:

code to run if both condition1 and condition2 are true

else:

code to run if condition 1 is true but condition2 is false

else:

code to run if condition 1 is false

مثال: اگر موسم بارش والا ہو اور درجہ حرارت 15 ڈگری سے کم ہو، تو ہم رین کوٹ پہنتے ہیں۔ اگر صرف بارش ہو رہی ہو تو، ہم چھتری لیتے ہیں۔ اگر موسم بارش والا نہ ہو تو ہم بس دن کا مزہ لیتے ہیں۔

```
weather = "rainy"
temperature = 10
if weather == "rainy":
    if temperature < 15:
        print("Wear a raincoat")
    else :
        print("Take an umbrella")
else:
    print("Enjoy your day!")
# Output: Wear a raincoat
```

وضاحت

اس مثال میں، کوڈ سب سے پہلے یہ چیک کرتا ہے کہ آیا موسم بارش والا ہے یا نہیں۔

اگر موسم بارش والا ہو تو یہ پھر یہ چیک کرتا ہے کہ آیا درجہ حرارت 15 ڈگری سے کم ہے یا نہیں۔ اگر دونوں شرائط درست ہوں تو یہ پیغام دکھاتا ہے: "رین کوٹ پہنیں"۔

اگر موسم بارش والا ہو لیکن درجہ حرارت 15 ڈگری سے کم نہ ہو تو یہ پیغام دکھاتا ہے: "چھتری لیں"۔

اگر موسم بارش والا نہ ہو تو یہ پیغام دکھاتا ہے: "دن کا مزہ لیں"۔

لوپنگ کنسٹرکٹ (Looping Constructs)

لوپس ہمیں کسی کام کو بار بار دہرانے میں مدد دیتے ہیں، جس سے ہمارا کوڈ زیادہ مؤثر اور پڑھنے میں آسان ہو جاتا ہے۔

for

while

پائتھن میں لوپس کی دو اہم اقسام ہیں :

while لوپ //

ایک while لوپ تب تک چلتا ہے جب تک دی گئی شرط درست ہو۔ یہ ہر تکرار (iteration) سے پہلے شرط کو چیک کرنا ہے اور جب شرط درست نہ رہے تو لوپ رک جاتا ہے۔

while لوپ کا سنٹیکس

Syntax of while loop

while condition:

code to run while the condition is true

یہ کوڈ تب تک چلے گا جب تک شرط درست

مثال: ایک نمبر میں 1 جمع کر کے جائیں جب تک وہ 10 تک نہ پہنچ جائے۔

<pre>number = 1 while number < 10: print(number) number += 1</pre>	<pre>Output: 1 2 3 4 5 6 7 8 9 ...</pre>
---	--

وضاحت:

اس مثال میں، کوڈ ایک نمبر سے شروع ہوتا ہے جس کی قیمت 1 رکھی گئی ہے۔ 'while' لوپ چیک کرتا ہے کہ کیا نمبر 10 سے کم ہے۔

اگر نمبر 10 سے کم ہے تو پروگرام موجودہ نمبر پر پرنٹ کرتا ہے اور پھر اس میں 1 جمع کر دیتا ہے۔

یہ لوپ اسی طرح چلتا رہتا ہے جب تک نمبر 10 تک نہ پہنچ جائے، جس کے بعد لوپ رک جاتا ہے۔

for لوپ //

for لوپ ایک کوڈ کے ایک بلاک کو ایک مخصوص تعداد میں دہراتا ہے۔ یہ عام طور پر ایک ترتیب جیسے کہ ایک لسٹ، ٹوپل، یا سٹرنگ پر تکرار کرنے کے لیے استعمال ہوتا ہے۔

for لوپ کا سنٹیکس

Syntax of for loop

for variable in sequence:

code to run for each element in the sequence

کسی ترتیب میں شامل ہر قدر کے لیے

ترتیب میں موجود ہر ایلیمنٹ کے لیے کوڈ چلے گا
مثال: دو ستوں کی فہرست میں موجود ہر دوست کو "ویکم" لکھیں۔

```

friends = ["Sami", "Raza", "Moosa"]
for friend in friends:
    print("Welcome to ", friend)
'''

```

Output:

```

Welcome to: Sami
Welcome to: Raza
Welcome to: Moosa
'''

```

range() فنکشن

ہم range() فنکشن کا استعمال نمبروں کی ایک ترتیب (sequence) بنانے کے لیے کر سکتے ہیں، جو اکثر for لوپس میں استعمال ہوتا

ریج فنکشن کا سینٹکس

range(stop)

range(start, stop)

range(start, stop, step)

مثال: پہلے 5 مکمل اعداد (Whole Numbers) پر پرنٹ کریں

```

for i in range(5):
    print(i)
'''

```

Output:

```

0
1
2
3
4
'''

```

end (end statement) شیٹ

پانٹھن میں، end شیٹ کوئی الگ کمانڈ نہیں ہے، بلکہ یہ print() فنکشن کا ایک آپشنل پیرامیٹر (optional parameter) ہے۔ بطور ڈیفالٹ، پانٹھن میں ہر print() شیٹ ایک نئی لائن پر ختم ہوتی ہے، لیکن آپ end پیرامیٹر کا استعمال کرتے ہوئے یہ تبدیل کر سکتے ہیں کہ آخر میں کیا پرنٹ کیا جائے۔

مثال: "end" شیٹ کا استعمال کرتے ہوئے دوسرے ٹکڑوں کو ایک ہی لائن پر پرنٹ کریں۔

```
print("Arshman", end=" ")
print("Ashbil")
# Output: Arshman Ashbil
print("Ayyan", end="****")
print("Naveed")
# Output: Ayyan***Naveed
```



پانچھن میں سٹرنگز یا تو سنگل کوٹیشن کی علامات (Single quotation marks) یا ڈبل کوٹیشن کی علامات (double quotation marks) سے گھیرے جاتے ہیں۔ مثال کے طور پر 'Ayyan' اور "Ayyan" ایک جیسے ہیں۔

مثال: "end" سینٹمنٹ کا استعمال کرتے ہوئے 1 سے 5 تک کے اعداد کو ایک ہی لائن پر پرنٹ کریں۔

```
for i in range(1, 6):
    # if you want to print Numbers on same line use following line
    print(i, end=" ")
# Output: 1 2 3 4 5
```

سرگرمی:

1. range () فنکشن کا استعمال کرتے ہوئے ایک for لوپ لکھیں تاکہ 2 سے 10 تک کے جفت اعداد (even numbers) کو ایک ہی لائن پر پرنٹ کیا جاسکے۔
2. ایک پانچھن پروگرام لکھیں جو for لوپ اور range () فنکشن کا استعمال کرتے ہوئے 3 کے پہلے multiples 10 پر پرنٹ کرے۔

نیسٹیڈ لوپس

نیسٹیڈ لوپس وہ لوپس ہیں جو دوسرے لوپس کے اندر ہوتے ہیں۔ بیرونی لوپ (Outer loop) پہلے چلتا ہے۔ بیرونی لوپ کے ہر ایک ٹکرا (iteration) کے لیے، اندرونی لوپ (Inner loop) مکمل طور پر چلتا ہے۔ ہم نیسٹیڈ لوپس کا استعمال اس وقت کرتے ہیں جب ہمیں کسی کام کے اندر بار بار دہرائے جانے والے کام کو انجام دینے کی ضرورت ہوتی ہے۔
مثال: بنیادی نمبر پیٹرن کے لیے نیسٹیڈ لوپس

```

print("=== Basic Number Pattern ===")
for i in range(3):      # Outer loop
    for j in range(4):  # Inner loop
        print(f"i={i}, j={j}")
    print("----")      # Separator after each outer loop iteration
'''

# Output
=== Basic Number Pattern ===
i=0, j=0
i=0, j=1
i=0, j=2
i=0, j=3
----
i=1, j=0
i=1, j=1
i=1, j=2
i=1, j=3
----
i=2, j=0
i=2, j=1
i=2, j=2
i=2, j=3
- '''

```

مثال: ستاروں کو تکنونی پیٹرن میں پرنٹ کرنے کے لیے نیسٹڈ لوپ

```

print("=== Right-Angled Triangle ===")
n = 5
for i in range(1, n+1):
    for j in range(i):      # Inner loop runs 'i' times
        print("*", end=" ")
    print() # New line after each row
'''

# Output
*
* *
* * *
* * * *
* * * * *
'''

```

4.2 پائنتھن میں لائبریریز

پائنتھن ایک وسیع معیاری لائبریری پیش کرتا ہے جس میں بلٹ ان ماڈیولز (built-in modules) اور ڈیٹا سٹرکچرز شامل ہیں۔ ڈیٹا سٹرکچر سے مراد ڈیٹا کو منظم کرنے اور ذخیرہ کرنے کا ایک مخصوص فارمیٹ یا طریقہ ہے۔ اس سیکشن میں، ہم پائنتھن میں مختلف لائبریریز کے استعمال کا جائزہ لیں گے۔

پائنتھن میں لائبریریز کا استعمال

پائنتھن میں، لائبریریز کسی ٹول یا کس کی طرح ہیں جو مفید ٹولز سے بھری ہوئی ہیں اور جو آپ کو ہر چیز کو از سر نو (from scratch) بنائے بغیر مختلف مسائل کو حل کرنے میں مدد کرتی ہیں۔ لائبریریز پہلے سے بنے ہوئے ٹول کٹس کی طرح ہیں جنہیں آپ تمام کوڈ خود لکھے بغیر استعمال کر سکتے ہیں۔ آئیے کچھ سادہ مثالوں کے ساتھ دیکھتے ہیں کہ مختلف لائبریریز کو کیسے امپورٹ (import) اور استعمال کیا جائے۔

مثال: رینڈم لائبریری کو رینڈم اعداد (random numbers) بنانے کے لیے امپورٹ کریں۔

```
import random
# Generate a random number between 1 and 10
number = random.randint(1, 10)
print("The random number is:", number)
#Output:
# The random number is: 3
```

مثال: سٹیٹسٹکس (statistics) لائبریری کو شماریاتی کیلکولیشن انجام دینے کے لیے امپورٹ کریں۔

```
import statistics
# Calculate the mean of a list of numbers
data = [23, 45, 67, 89, 12, 44, 56]
mean_value = statistics.mean(data)
print("The mean value is:", mean_value)
# Output:
# The mean value is: 48
```

وضاحت: statistics لائبریری بنیادی شماریاتی کیلکولیشن انجام دینے کے لیے ایک بہترین ٹول ہے، جیسے کہ ڈیٹا کے ایک سیٹ کا اوسط (mean)، درمیانی قیمت (median)، اور عام ویلو (mode) معلوم کرنا۔ یہ خاص طور پر ڈیٹا اینالیسٹ کے کاموں میں مفید ہو سکتا ہے۔

مختصر آ، ان لائبریریز کا استعمال کر کے، آپ وقت اور محنت کی بچت کر سکتے ہیں، جس سے آپ پہلے دوبارہ ایجاد (reinventing the wheel) کرنے کے بجائے زیر بحث مخصوص مسئلے کو حل کرنے پر توجہ مرکوز کر سکتے ہیں۔

SMART TIPS



لاہریز کو اچھوٹ کرتے وقت، یقینی بنائیں کہ آپ صرف وہی چیزیں اچھوٹ کر رہے ہیں جن کی آپ کو ضرورت ہے۔ آپ مزید لاہریز

پر دیکھ سکتے ہیں۔ <https://docs.python.org/3>

4.3 پائٹھن میں لسٹیں

پائٹھن میں، ایک لسٹ ایک کثیر المقاصد (versatile) ڈیٹا سٹرکچر ہے جو آئٹمز کا ایک مجموعہ رکھ سکتی ہے۔ آپ آسانی سے لسٹیں بنا،

سکتے ہیں۔ ان تک رسائی حاصل کر سکتے ہیں، اور ان میں ترمیم کر سکتے ہیں۔

لسٹیں بنانا

ایک لسٹ مربع قوسین [] کے اندر آئٹمز کو رکھ کر بنائی جاتی ہے، جنہیں کاما (comma) سے الگ کیا جاتا ہے۔ لسٹوں میں مختلف اقسام کی اشیاء شامل ہو سکتی ہیں، جیسے کہ نمبرز، سٹرنگز یا یہاں تک کہ دوسری لسٹیں بھی۔ مثال: اپنے پسندیدہ پھلوں کی ایک لسٹ بنائیں۔

```
fruits = ["Mango", "Apple", "Banana"]
print(fruits)
# Output: ['Mango' 'Apple' 'Banana']
```

وضاحت: یہ کوڈ fruits نامی ایک لسٹ بناتا ہے جس میں تین عناصر (elements) شامل ہیں اور پھر اس لسٹ کو پرنٹ کرتا ہے۔

لسٹ آئٹمز تک رسائی

آپ ایک لسٹ میں موجود آئٹمز تک ان کے انڈیکس کا حوالہ دے کر رسائی حاصل کر سکتے ہیں، جو 0 سے شروع ہوتا ہے۔ مثال: پھلوں کی لسٹ سے دوسری آئٹم تک رسائی حاصل کریں اور اسے پرنٹ کریں۔

```
fruits = ["Mango", "Apple", "Banana"]
print(fruits[1])
# Output: Apple
```

وضاحت: یہ کوڈ fruits نامی ایک لسٹ شروع کرتا ہے، جس میں "Mango"، "Apple" اور "Banana" شامل ہیں، اور پھر انڈیکس [1] کا استعمال کرتے ہوئے دوسری آئٹم 'Apple' کو پرنٹ کرتا ہے۔

سرگرمی:

ایک پانچھن پروگرام لکھیں جو ریڈم اور statistics لائبریریوں کو ایپورٹ کرے۔ ریڈم کا استعمال کرتے ہوئے 1 سے 50 کے درمیان 10 ریڈم اعداد کی ایک لسٹ بنائیں۔ پھر، statistics کا استعمال کرتے ہوئے ان اعداد کا اوسط (mean) نکالیں اور پرنٹ کریں۔

لسٹ میں ترمیم کرنا

آپ لسٹ آئٹمز ان کے انڈیکس کے ذریعے رسائی حاصل کر کے اور ایک نئی قدر تفویض (assign) کر کے ترمیم کر سکتے ہیں۔ مثال: لسٹ میں پہلی آئٹم کو "Orange" میں تبدیل کریں اور ایک نیا پھل "Pineapple" شامل کریں۔

```
fruits = ["Mango", "Apple", "Banana"]
fruits[0] = "Orange"
fruits.append("Pineapple")
print(fruits)
# Output: ['Orange', 'Apple', 'Banana', 'Pineapple']
```

وضاحت: یہ کوڈ fruits لسٹ کے پہلے عنصر کو تبدیل کر کے 'Orange' کرتا ہے، آخر میں 'Pineapple' کا اضافہ (appends) کرتا ہے، اور پھر اپ ڈیٹ شدہ لسٹ کو پرنٹ کرتا ہے۔

لسٹ آپریشنز

پانچھن لسٹوں کے ساتھ کام کرنے کے لیے کئی بلٹ ان میٹھڈز فراہم کرتا ہے۔ یہاں کچھ مفید میٹھڈز درج ذیل ہیں:

- append(item): لسٹ کے آخر میں ایک آئٹم کا اضافہ کرتا ہے۔
 - remove(item): لسٹ میں سے آئٹم کے پہلے وقوع (first occurrence) کو ہٹاتا ہے۔
 - sort(): لسٹ کو صعودی ترتیب (ascending order) میں ترتیب دیتا ہے۔
 - reverse(): لسٹ کی ترتیب کو الٹ (reverse) دیتا ہے۔
- مثال: طلباء کی لسٹ میں ایک نیا طالب علم شامل کریں اور لسٹ پرنٹ کریں۔

```
students = ["Ahmed", "Sara", "Ali"]
students.append("Hina")
print(students)
#Output : ['Ahmed', 'Ali', 'Sara', 'Hina']
```

وضاحت: یہ کوڈ طلباء کی ایک لسٹ بناتا ہے، اس میں 'Hina' کو شامل (adds) کرتا ہے۔

مثال: طلباء کی لسٹ میں سے ایک طالب علم کو ہٹا دیں (Remove) اور لسٹ پرنٹ کریں۔

```
# Remove a student from the list of students
students = ["Ahmed", "Sara", "Ali", "Hina", "Sara"]
students.remove("Sara")
print(students)
# Output: ['Ahmed', 'Ali', 'Hina', 'Sara']
```

وضاحت: یہ کوڈ طلباء کی ایک لسٹ بناتا ہے، لسٹ میں سے "Sara" کے پہلے وقوع (first occurrence) کو ہٹاتا ہے، اور پھر اپ ڈیٹ شدہ لسٹ کو پرنٹ کرتا ہے۔ نوٹ کریں کہ صرف پہلی مماثل آئٹم ہی ہٹائی جاتی ہے۔
مثال: اعداد کی ایک لسٹ کو صعودی ترتیب (ascending order) میں ترتیب دیں۔

```
# Sort a list of numbers in ascending order
numbers = [34, 12, 89, 5, 23]
numbers.sort()
print(numbers)
# Output: [5, 12, 23, 34, 89]
```

وضاحت: یہ کوڈ اعداد کی ایک لسٹ بناتا ہے، sort() میٹھکا کا استعمال کرتے ہوئے انہیں صعودی ترتیب (ascending order) میں ترتیب دیتا ہے، اور پھر ترتیب شدہ لسٹ کو پرنٹ کرتا ہے۔
مثال: لسٹ میں موجود پھلوں کی ترتیب کو الٹ (Reverse) دیں۔

```
# Reverse the order of fruits in a list
fruits = ["Apple", "Banana", "Orange", "Mango"]
fruits.reverse()
print(fruits)
# Output: ['Mango', 'Orange', 'Banana', 'Apple']
```

وضاحت: یہ کوڈ پھلوں کی ایک لسٹ بناتا ہے، reverse() میٹھکا کا استعمال کرتے ہوئے عناصر کی ترتیب کو الٹ (reverses) دیتا ہے، اور پھر الٹی ہوئی لسٹ کو پرنٹ کرتا ہے۔

4.5 ٹیسٹنگ اور ڈیبگنگ

پائنتھن پروگرامنگ میں کوڈ کو ٹیسٹ کرنا اور اس میں سے غلطیاں نکالنا ایک ضروری عمل ہے۔

ٹیسٹنگ

کوڈ کو بار بار مختلف ویلیوز کے ساتھ ملا کر دیکھنا اور ممکنہ غلطیوں کی نشاندہی کرنا ٹیسٹنگ کہلاتا ہے۔ اس کا مقصد کوڈ کو استعمال میں لانے سے پہلے غلطیوں کو ختم کرنا ہوتا ہے۔

/// ٹیسٹنگ کی اقسام

- یونٹ ٹیسٹنگ: کوڈ کے انفرادی حصوں (جیسے فنکشنز یا کلاسز) کو الگ تھلگ کر کے جانچتا ہے۔ پائتھن کا یونٹ ٹیسٹ ماڈیول عام طور پر اس کے لیے استعمال ہوتا ہے۔
- انٹیگریشن ٹیسٹنگ: جانچتا ہے کہ کوڈ کے مختلف حصے ایک ساتھ کیسے کام کرتے ہیں۔
- فنکشنل ٹیسٹنگ: متوثیق کرتا ہے کہ سافٹ ویئر صارف کے نقطہ نظر سے توقع کے مطابق کام کرتا ہے۔
- ریگریژن ٹیسٹنگ: یقینی بناتا ہے کہ نئی تبدیلیاں موجودہ فعالیت کو خراب نہیں کرتی ہیں۔

/// ڈیبگنگ (Debugging)

ڈیبگنگ آپ کے کوڈ میں موجود غلطیوں (errors) یا باگز (bugs) کو تلاش کرنے اور ٹھیک کرنے کا عمل ہے۔ اس میں مسائل کی بنیادی وجہ کی نشاندہی کرنا اور ضروری تبدیلیاں کرنا شامل ہے۔

عام ڈیبگنگ تکنیکیں

- پرنٹ سٹیٹمنٹس: کوڈ کے مختلف مراحل پر تغیرات کی ویلیو کو چیک کرنے کے لیے print سٹیٹمنٹس کا اضافہ کرنا۔
- ڈیبگنگ ٹولز: کوڈ سے گزرنے، تغیرات کا معائنہ کر لے، اور عمل دلاؤ کے بہاؤ (flow of execution) کو سمجھنے کے لیے (Python Debugger) pdb جیسے ٹولز کا استعمال کرنا۔
- ایرر میسجز: مسئلے کا ماخذ تلاش کرنے کے لیے ایرر میسجز کو پڑھنا اور ان کی تشریح کرنا۔

خلاصہ (Summary)

- پروگرامنگ میں فیصلہ سازی پروگرام کو اس قابل بناتی ہے کہ وہ شرائط کی بنیاد پر مختلف اقدامات کا انتخاب کر سکے۔
- if سٹیٹمنٹ ہمیں شرائط کی بنیاد پر فیصلے کرنے دیتی ہے۔ اگر شرط درست (true) ہو، تو یہ کوڈ کا ایک مخصوص حصہ (block) چلاتی ہے۔
- if-else سٹیٹمنٹ ہمیں اس بات کی اجازت دیتی ہے کہ اگر شرط درست ہو تو کوڈ کا ایک حصہ چلے، اور اگر شرط غلط (false) ہو تو دوسرا حصہ چلے۔
- کبھی کبھار ہمیں ایک شرط کے اندر مزید کئی شرائط کو چیک کرنے کی ضرورت ہوتی ہے، اسے نیسٹنگ (Nesting) کہا جاتا ہے۔

لوپس (Loops) ہمیں کاموں کو دہرانے میں مدد دیتے ہیں، جس سے ہمارا کوڈ زیادہ موثر اور پڑھنے میں آسان ہو جاتا ہے۔

ایک while لوپ اس وقت تک چلتا ہے جب تک شرط درست رہے۔ یہ ہر بار چلنے سے پہلے شرط کو چیک کرتا ہے اور شرط غلط ہونے کی صورت میں رک جاتا ہے۔

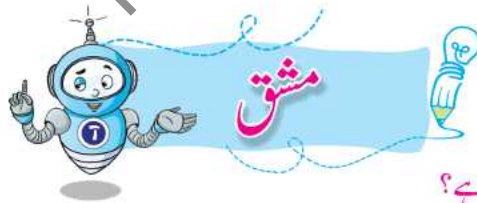
ایک for لوپ کوڈ کے ایک حصے کو مخصوص تعداد میں دہراتا ہے۔ یہ عام طور پر کسی ترتیب (sequence) پر کام کرنے کے لیے استعمال ہوتا ہے۔

ہم نمبروں کی ترتیب پیدا کرنے کے لیے range() فنکشن استعمال کر سکتے ہیں، جو اکثر for لوپس میں استعمال ہوتا ہے۔ پایتھن میں end سینٹ کوئی الگ کمانڈ نہیں ہے، بلکہ یہ print() فنکشن کا ایک اختیاری پیرامیٹر ہے۔

نیسٹڈ لوپس (Nested loops) ایسے لوپس ہوتے ہیں جو دوسرے لوپس کے اندر موجود ہوں۔ بیرونی لوپ (outer loop) پہلے چلتا ہے، اور بیرونی لوپ کے ہر ایک چکر کے لیے اندرونی لوپ (inner loop) مکمل طور پر چلتا ہے۔

پایتھن میں لسٹ ایک ہمہ گیر ڈیٹا اسٹرکچر ہے جو مختلف اشیاء کا مجموعہ رکھ سکتی ہے۔ آپ آسانی سے لسٹ بنا سکتے ہیں، اسے استعمال کر سکتے ہیں اور اس میں تبدیلی کر سکتے ہیں۔

ٹیسٹنگ کا مطلب کوڈ کو مختلف ان پٹس کے ساتھ چلا کر یہ دیکھنا ہے کہ آیا یہ توقع کے مطابق کام کر رہا ہے یا نہیں۔ ڈی بگنگ کوڈ میں غلطیوں (bugs) کو تلاش کرنے اور انہیں ٹھیک کرنے کا عمل ہے۔ اس میں مسئلے کی اصل وجہ پہچاننا اور ضروری تبدیلیاں کرنا شامل ہے۔



کثیر انتخابی سوالات

1 پایتھن میں range() فنکشن کیا کرتا ہے؟

- ب اعداد کی ایک ترتیب بناتا ہے
- د اعداد کی ایک رینج پرنٹ کرتا ہے

- الف اعداد کی ایک لسٹ بناتا ہے
- ن اعداد کا حاصل جمع نکالتا ہے

ii درج ذیل کوڈ کا نتیجہ کیا ہوگا؟

```
count = 0
for i in range(2):
    for j in range(3):
        count += 1
print(count)
```

- 5 ب
- 7 د

- 4 الف
- 6 ن

iii پانچھن میں لسٹ کے آخر میں ایک آئٹم شامل کرنے کے لیے کون سا میٹھڈ استعمال ہوتا ہے؟

append() ب

insert() الف

extend() د

add() ن

iv پروگرامنگ میں کنٹرول سٹرکچرز کا مقصد کیا ہے؟

ب فیصلہ سازی اور کاموں کی ریپیٹیشن کو منظم کرنا

الف صرف کوڈ کو ملایا کرنا

د لائبریریز امپورٹ کرنا

ن متغیرات بنانا

v کون سی سٹیٹمنٹ استعمال ہوتی ہے جب کوئی شرط غلط (false) ہو تو کوڈ کو چلایا جائے؟

elif ب

if الف

while د

else ن

vi آپ print() کوئی لائن شامل کرنے سے کیسے روکتے ہیں؟

ب newline=False استعمال کریں

الف end="" استعمال کریں

د Continue استعمال کریں

ن Break استعمال کریں

vii ریئنڈم اعداد (random numbers) بنانے کے لیے کون سی لائبریری استعمال ہوتی ہے؟

ب statistics

Datetime الف

د math

Random ن

viii list.append() کیا کرتا ہے؟

ب آخر میں ایک آئٹم شامل کرتا ہے

الف پہلی آئٹم کو ہٹاتا ہے

د لسٹ کو الٹ (reverse) دیتا ہے

ن لسٹ کو ترتیب دیتا ہے

ix range() فنکشن کا مقصد کیا ہے؟

ب اعداد کی ایک ترتیب پیدا کرنا

الف اعداد کی ایک لسٹ بنانا

د اعداد کو ترتیب دینا

ن قدروں کی رینج کا حساب لگانا

❌ کون سی ڈیبلنگ تکنیک آؤٹ پٹ شیڈیو کو شامل کرنے پر مشتمل ہے؟

ب پرٹ شیڈیو

الف یونٹ شیڈیو

د انگریزیشن شیڈیو

ن انگریزیشن شیڈیو

مختصر سوالات

1 for لوپ میں range() فنکشن کے استعمال کی وضاحت کریں۔

2 پانچھن لسٹوں میں append اور remove میٹھڈز کے درمیان کیا فرق ہے؟

3 ڈی بلنگ کی تعریف کریں۔

4 range() فنکشن کے تین پیرامیٹرز کیا ہیں؟

5 لسٹ کے تین میٹھڈز اور ان کے مقاصد کے نام بتائیں۔

6 آپ پانچھن میں ایک لائبریری کو کیسے اپورٹ کرتے ہیں؟

7 یونٹ شیڈیو کیا ہے اور یہ کیوں اہم ہے؟

8 آپ ایک لسٹ میں تیسرے عنصر تک کیسے رسائی حاصل کرتے ہیں؟

9 print() میں end پیرامیٹر کا مقصد کیا ہے؟

10 ہم لسٹ میں سے ایک عنصر تک رسائی کیسے حاصل کر سکتے ہیں؟

11 آپ ایک لسٹ میں موجود موجودہ آئٹم میں کیسے ترمیم کرتے ہیں؟

تفصیلی سوالات

1 پانچھن میں فیصلہ سازی شیڈیو کی مختلف اقسام کی مثالوں کے ساتھ وضاحت کریں۔

2 while لوپس اور for لوپس کا موازنہ اور تضاد کریں۔

3 پانچھن لسٹوں کے ساتھ کام کرنے کے طریقے کو بیان کریں، بشمول ان کو بنانے اور ان میں ترمیم کرنے کا طریقہ۔ آئٹمز شامل کرنے،

ہٹانے اور عناصر تک رسائی حاصل کرنے کے لیے مثالیں فراہم کریں۔

4 پروگرامنگ میں شیڈیو اور ڈیبلنگ کی اہمیت کی وضاحت کریں۔ مختلف ڈیبلنگ تکنیکس پر بحث کریں۔

5 پانچھن لائبریری پروگرامنگ کی کارکردگی (efficiency) کو کیسے بڑھاتی ہیں؟ ایک مثال فراہم کریں۔

6 ایک پانچھن پروگرام لکھیں جو ایک لوپ کا استعمال کرتے ہوئے 1 اور 20 کے درمیان تمام طاق اعداد (odd numbers) کو پرنٹ کرے۔

7 ایک پانچھن پروگرام لکھیں جو درج ذیل کام انجام دے:

(a) 1 اور 20 کے درمیان 5 ریٹڈم اعداد کی ایک لسٹ بنائے۔

(b) پیدا کردہ اعداد کا اوسط (mean) نکالنے اور پرنٹ کرنے کے لیے statistics لا بیریری کا استعمال کرے۔

8 ایک پانچھن پروگرام لکھیں جو درج ذیل آپریشنز انجام دے:

(a) مشہور پاکستانی کھانوں کی ایک لسٹ بنائیں: ["Biryani", "Nihari", "Karahi", "Seekh Kebabs"]۔

(b) لسٹ میں ایک نیا کھانا "Quorma" شامل کریں۔

(c) "Karahi" کو "Chicken Karahi" میں تبدیل کریں۔

(d) لسٹ میں سے "Nihari" کو ہٹا دیں۔

9 نیسٹڈ لوپس کا استعمال کرتے ہوئے تین قطاروں (rows) اور پانچ کالموں (columns) میں ستاروں (*) کو مستطیل پیٹرن میں پرنٹ کرنے کے لیے کوڈ لکھیں۔

کثیر انتخابی سوالات کے جوابات

1 ب — اعداد کی ایک ترتیب بناتا ہے

2 ج — 6

3 ب — append()

4 ب — فیصلہ سازی اور کاموں کی ریٹیشن کو منظم کرنا

5 ج — else

6 الف — end="" استعمال کریں

7 ج — Random

8 ب — آخر میں ایک آئٹم شامل کرتا ہے

9 ب — اعداد کی ایک ترتیب پیدا کرنا

10 ب — ایرر میسجز