باب-5

پائتھن کے ساتھ ابتدا
(GETTING STARTED WITH PYTHON)

5196CH05

کمپیوٹر پروگرامنگ ایک فن ہے کیوں کہ یہ معلومات کے مجموعہ کا اطلاق ساری دنیا پر کرتا ہے۔ کیوں کہ اس کے لیے مہارت اور اختراع پسندی کی ضرورت ہوتی ہے اور خاص طور پر اس لیے بھی کیوں کہ یہ حسن کے مظاہر کی بازیافت کرتا ہے۔ ایک پروگرامر جو اپنے تحت الشعور میں خود کو ایک فن کار تصور کرتا ہے وہ اپنے بالفعل اور بالقوہ شاہ کاروں سے لطف اندوز ہوتا ہے۔

— ڈونالڈ کنُتھ
(Donald Knuth)

اس باب میں

- پائتھن کا تعارف
- پائتھن کے کلیدی الفاظ
- شناخت کنندہ
- تبصرے (Comments)
- ڈیٹا ٹائپ
- آپریٹر
- عبارتیں (Expressions)
- بیانات
- ان پُٹ اور آؤٹ پُٹ
- ڈیٹا ٹائپ کی تبدیلی
- ڈی بگنگ

5.1 تعارف

ہم نے باب 4 میں مختلف مسائل کے لیے الگورتھم تحریر کیے ہیں۔ آیئے اب ایک قدم آگے بڑھیں اور پائتھن 3 کے کسی بھی ورژن کا استعمال کر کے ایک پروگرام تشکیل دیتے ہیں۔ لیکن پائتھن پروگرامنگ لینگویج کی آموزش سے پہلے آیئے یہ سمجھنے کی کوشش کریں کہ پروگرامنگ لینگویج کیا ہے اور یہ کس طرح کام کرتی ہے؟

ہدایات کا ایسا مرتب سیٹ جن پر کوئی کمپیوٹر کسی مخصوص کام کو انجام دینے کے لیے عمل درآمد کرتا ہے پروگرام کہلاتا ہے اور کمپیوٹر کے لیے ہدایات کے اس سیٹ کو متعین کرنے کے لیے استعمال کی جانے والی زبان پروگرامنگ لینگویج کہلاتی ہے۔

جیسا کہ ہم جانتے ہیں کمپیوٹر صرف 0s اور 1s کی زبان کو سمجھتے ہیں جسے مشینی زبان یا ادنیٰ سطحی زبان (LLL) کہتے ہیں۔ حالاں کہ انسانوں کے لیے 0s اور 1s کا استعمال کر کے ہدایات کو لکھنا اور سمجھنا بہت مشکل ہے۔ اسی سبب سے پائتھن، C++، ویژول بیسک، PHP، جاوا جیسی اعلیٰ سطحی لینگویجز کا ظہور ہوا جنھیں انسانوں کے ذریعے استعمال کرنا آسان ہے لیکن کمپیوٹر انھیں براہ راست نہیں سمجھ سکتے ہیں۔

اعلیٰ سطحی لینگویج (HLL) میں تحریر کیا گیا پروگرام سورس کوڈ کہلاتا ہے۔ باب 1 کے حوالے سے یاد کیجیے کہ سورس کوڈ کو مشینی لینگویج میں تبدیل کرنے کے لیے کمپائلر (Compiler) اور انٹرپریٹر (Interpreter) جیسے لسانی مترجم کی ضرورت ہوتی ہے۔ پائتھن اپنی ہدایات کو مشینی زبان میں تبدیل کرنے کے لیے ایک انٹرپریٹر کا استعمال کرتا ہے تاکہ کمپیوٹر اسے سمجھ سکے۔ انٹرپریٹر، پروگرام کے بیانات کی یکے بعد دیگرے پروسیسنگ کرتا ہے۔ پہلے ترجمہ (Translate) کرتا ہے اور پھر اس پر عمل درآمد (Execute) کرتا ہے۔ یہ عمل اس وقت تک جاری رہتا ہے جب تک کوئی غلطی نہ آجائے یا پورے پروگرام پر کامیابی کے ساتھ عمل درآمد نہیں ہو جاتا۔ دونوں ہی معاملوں میں پروگرام پر عمل درآمد کا سلسلہ رک جائے گا۔ اس کے برعکس ایک کمپائلر تمام سورس کوڈ کا مجموعی طور پر آبجیکٹ کوڈ میں ترجمہ کرتا ہے۔ پورے پروگرام کو اسکین کرنے کے بعد اگر اس میں کہیں کوئی غلطی ہے تو غلطی کا میسیج ظاہر کر دیتا ہے۔

5.1.1 پائتھن کی خصوصیات

- پائتھن ایک اعلیٰ سطحی لینگویج ہے۔ یہ مفت اور اوپن سورس لینگویج ہے۔
- یہ ایک Interpreted زبان ہے کیوں کہ انٹرپریٹر پائتھن پروگراموں پر عمل درآمد کرتا ہے۔

- پائتھن پروگراموں کو سمجھنا آسان ہے کیوں کہ ان میں واضح طور پر متعینہ نحوی ترکیب (Syntax) ہوتی ہے اور ساخت بھی نسبتاً سادہ ہے۔
- پائتھن Case-sensitive ہے۔ مثال کے طور پر پائتھن میں NUMBER اور Number ایک جیسے نہیں ہیں۔
- پائتھن قابل منتقل (Portable) اور پلیٹ فارم مبرا ہے، یعنی اسے مختلف آپریٹنگ سسٹم اور ہارڈ ویئر پلیٹ فارم پر چلایا جاسکتا ہے۔
- پائتھن میں پہلے سے متعینہ فنکشن کی ایک زرخیز لائبریری موجود ہے۔
- پائتھن ویب ڈویلپمنٹ میں بھی معاون ہے۔ پائتھن کو استعمال کرکے متعدد معروف ویب سروسز اور اپلی کیشن بنائی گئی ہیں۔
- پائتھن میں بلاک اور نیسٹڈ بلاک (بلاک کے اندر بلاک) کے لیے انڈنٹیشن کا استعمال ہوتا ہے۔

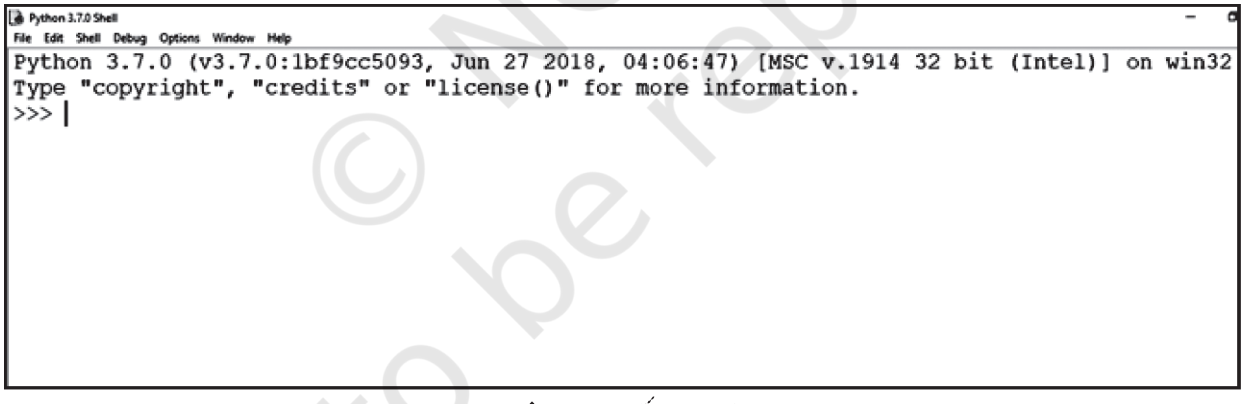
پائتھن کو ڈاؤن لوڈ کرنا

پائتھن 3 کا جدید ترین ورژن آفیشل ویب سائٹ پر دست یاب ہے:

https://www.python.org/

5.1.2 پائتھن کے ساتھ کام کرنا (Working with Python)

پائتھن پروگرام کو تحریر کرنے اور اسے چلانے (Execute) کے لیے ہمیں اپنے کمپیوٹر پر پائتھن انٹرپریٹر انسٹال کرنا ہوگا یا ہم کسی بھی آن لائن پائتھن انٹرپریٹر کا استعمال کرسکتے ہیں۔ انٹرپریٹر کو پائتھن شیل (Python shell) بھی کہتے ہیں۔ پائتھن انٹرپریٹر کا ایک سیمپل اسکرین شکل 5.1 میں دکھایا گیا ہے۔



شکل 5.1: پائتھن انٹرپریٹر یا شیل

اوپر دکھائے گئے اسکرین میں علامت >>> پائتھن پرومپٹ (Python prompt) ہے جو اس بات کا اشارہ ہے کہ انٹرپریٹر ہدایات حاصل کرنے کے لیے تیار ہے۔ ہم پائتھن انٹرپریٹر کا استعمال کرکے کمانڈ یا بیانات پر عمل درآمد کرنے کے لیے انھیں اس پرومپٹ پر ٹائپ کرسکتے ہیں۔

5.1.3 عمل درآمد کے طریقے (Execution Modes)

پائتھن انٹرپریٹر کو استعمال کرنے کے دو طریقے ہیں:

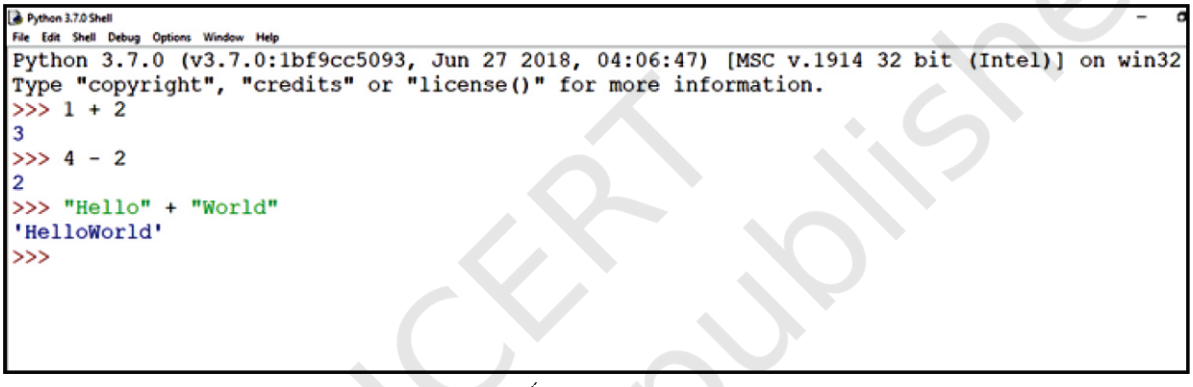
a) انٹرایکٹیو طریقہ (Interactive mode)

b) اسکرپٹ طریقہ (Script mode)

انٹرایکٹیو طریقے میں انفرادی بیان پر فوری عمل درآمد ہوتا ہے جب کہ اسکرپٹ طریقے میں ہم ایک سے زیادہ ہدایات کو ایک فائل میں لکھتے ہیں جسے پائتھن سورس کوڈ فائل کہتے ہیں جس پر عمل درآمد کیا جاسکتا ہے۔

(A) انٹرایکٹیو طریقہ (Interactive Mode)

انٹرایکٹیو طریقے میں کام کرنے کے لیے ہم >>> پرومپٹ پر براہ راست پائتھن اسٹیٹمنٹ تحریر کرسکتے ہیں۔ جیسے ہی ہم اینٹر دباتے ہیں، انٹرپریٹر اسٹیٹمنٹ پر عمل درآمد کرتا ہے اور نتیجہ ظاہر کردیتا ہے جیسا کہ شکل 5.2 میں دکھایا گیا ہے۔



شکل 5.2: انٹرایکٹیو طریقے میں پائتھن انٹرپریٹر

صرف ایک لائن والے کوڈ کی جانچ کرنے کے فوری عمل درآمد کے لیے انٹرایکٹیو موڈ میں کام کرنا آسان ہوتا ہے۔ لیکن انٹرایکٹیو موڈ میں ہم اسٹیٹمنٹ کو مستقبل میں استعمال کے لیے محفوظ نہیں کرسکتے ہیں اور اگر ہم اسے بعد میں چلانا چاہتے ہیں تو دوبارہ ٹائپ کرنا پڑے گا۔

(B) اسکرپٹ طریقہ (Script Mode)

اسکرپٹ طریقے میں ہم پائتھن پروگرام کو ایک فائل میں لکھتے ہیں، اسے محفوظ کرتے ہیں اور ایگزیکیوٹ کرنے کے لیے انٹرپریٹر کا استعمال کرتے ہیں۔ پائتھن اسکرپٹ کو ایسی فائلوں میں محفوظ کیا جاتا ہے جہاں فائل کے نام میں ایکسٹینشن ".py" ہوتا ہے، اگر صارف کوئی متبادل پیش نہیں کرتا ہے تو پائتھن اسکرپٹ، پائتھن انسٹالیشن فولڈر میں محفوظ ہوجاتی ہیں۔ اسکرپٹ کو ایگزیکیوٹ کرنے کے لیے ہم مندرجہ ذیل میں سے کوئی ایک کام کرسکتے ہیں:

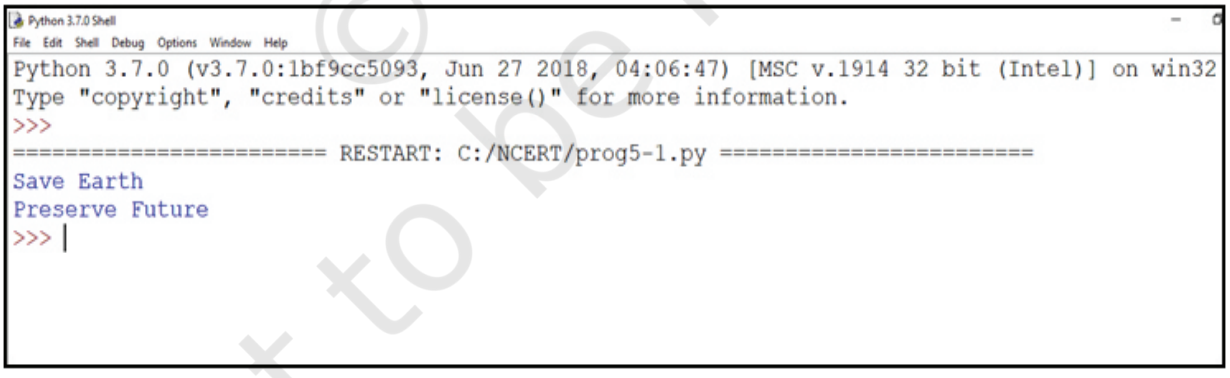
a) پرومپٹ پر پاتھ کے ساتھ فائل کا نام لکھیے۔ مثال کے طور پر اگر فائل کا نام prog5-1.py ہے تو ہم prog5-1.py ٹائپ کرتے ہیں۔ بصورت دیگر ہم پروگرام کو IDLE سے براہ راست کھول سکتے ہیں جیسا کہ شکل 5.3 میں دکھایا گیا ہے۔

b) اگر ہم اسکرپٹ موڈ میں کام کرتے ہیں تو فائل کو محفوظ (Save) کرنے کے بعد مینو سے [Run]->[Run Module] کلک کریں جیسا کہ شکل 5.4 میں دکھایا گیا ہے۔

پروگرام 5-1 اسکرپٹ موڈ میں پرنٹ اسٹیٹمنٹ کو دکھانے کے لیے پروگرام لکھیے۔

شکل 5.3: پائتھن سورس کوڈ فائل (prog5-1.py)

شکل 5.4: اسکرپٹ موڈ میں IDLE کا استعمال کرکے پائتھن پروگرام پر عمل درآمد



شکل 5.5: اسکرپٹ موڈ پر عمل درآمد کیے گئے پروگرام کا نتیجہ

5.2 پائتھن کے کلیدی الفاظ (Python Keywords)

کلیدی الفاظ (Keywords) دراصل مخصوص الفاظ (Reserved Words) ہیں۔ پائتھن انٹرپریٹر ہر ایک کلیدی لفظ کو ایک مخصوص معنی میں استعمال کرتا ہے اور ہم اپنے پروگرام میں ایک کلیدی لفظ کا استعمال صرف اسی مقصد کے لیے کر سکتے ہیں جس کے لیے اسے متعین کیا گیا ہے۔ کیوں کہ پائتھن Case Sensitive ہے

نوٹ

چنانچہ کلیدی الفاظ کو بالکل اسی طرز پر لکھا جانا چاہیے جیسا کہ جدول 5.1 میں دیا گیا ہے۔

جدول 5.1 : پائتھن سے متعلق کلیدی الفاظ

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

5.3 شناخت کنندہ (IDENTIFIER)

پروگرامنگ کی لینگویجز میں شناخت کنندگان ایسے نام ہیں جن کا استعمال پروگرام میں کسی متغیر (Variable)، فنکشن یا کسی اور شے کی شناخت کے لیے کیا جاتا ہے۔ پائتھن میں شناخت کنندگان کے تسمیہ کے ضابطے ذیل میں دیے گئے ہیں۔

- نام کی ابتدا بڑے حرف یا چھوٹے حرف یا انڈر اسکور کی علامت (_) سے ہونی چاہیے۔ اس کے بعد کیریکٹر کا کوئی بھی اتحاد a-z، A-Z، 0-9 یا انڈر اسکور (_) ہوسکتا ہے۔ لہٰذا شناخت کنندہ کی ابتدا کسی ہندسہ (Digit) سے نہیں کی جاسکتی۔
- یہ کسی بھی لمبائی کا ہوسکتا ہے۔ (تاہم اگر یہ مختصر اور بامعنی ہو تو بہتر ہے)
- یہ کوئی کلیدی لفظ یا جدول 5.1 میں دکھایا گیا مخصوص لفظ نہیں ہونا چاہیے۔
- ہم شناخت کنندگان میں مخصوص علامات مثلاً !، @، #، $، %، وغیرہ کا استعمال نہیں کر سکتے ہیں۔

مثال کے طور پر کسی طالب علم کے ذریعے تین مضامین میں حاصل کردہ نمبروں کا اوسط معلوم کرنے کے لیے ہم شناخت کنندگان کا انتخاب a, b, c, or A, B, C کے بجائے marks1، marks2، marks3 اور avg کے طور پر کرتے ہیں۔

```
avg = (marks1 + marks2 + marks3)/3
```

اسی طرح، مستطیل کا رقبہ معلوم کرنے کے لیے ہم شناخت کنندگان کے نام کے لیے واحد حرف کو استعمال کرنے کے بجائے وضاحت اور پڑھنے میں سہولت کے پیش نظر area، length، breadth کا استعمال کر سکتے ہیں۔

```
area = length * breadth
```

5.4 متغیرات (VARIABLES)

پروگرام میں کسی متغیر کی شناخت نام (شناخت کنندہ) کے ذریعے منفرد طور پر کی جاتی ہے۔ پائتھن میں متغیر سے مراد ایک آبجیکٹ ہے جو میموری میں ذخیرہ شدہ کوئی آئٹم یا ایلیمنٹ ہوسکتا ہے۔ متغیر کی قدر ایک اسٹرنگ (مثلاً 'Global Citizen'، 'b')، عددی (مثلاً 345) یا الفانیومیرک کیریکٹر (CD67) کا کوئی بھی مجموعہ ہوسکتی ہے۔

پائتھن میں ہم نئے متغیر کی تشکیل اور انھیں مخصوص قدریں تفویض کرنے کے لیے اسائنمینٹ اسٹیٹمینٹ کا استعمال کرسکتے ہیں۔

```
gender    = 'M'
message   = "Keep Smiling"
price     = 987.9
```

پروگرام 5-2 پائتھن میں متغیروں کی قدریں ظاہر کرنے کے لیے ایک پروگرام لکھیے۔

```
#Program 5-2
#To display values of variables
message = "Keep Smiling"
print(message)
userNo = 101
print('User Number is', userNo)
```

نتیجہ:

```
Keep Smiling
User Number is 101
```

پروگرام 5-2 میں متغیر 'Message' کی قدر اسٹرنگ قسم کی ہے چنانچہ اس کے مواد کو دہرے ''واوین'' میں لکھا گیا ہے (یہ اکہرے واوین میں بھی ہوسکتا ہے) جب کہ متغیر 'userNo' کی قدر واوین میں نہیں ہے کیوں کہ یہ عددی قدر ہے۔

پائتھن میں متغیر کا اظہار مضمر ہے یعنی متغیرات کو جب پہلی مرتبہ قدر تفویض کی جاتی ہے تو ان کا اظہار اور تعین خود کار انداز میں ہوجاتا ہے۔ عبارتوں میں استعمال کرنے سے پہلے متغیروں کو ہمیشہ قدریں تفویض کی جانی چاہئیں بصورت دیگر پروگرام میں غلطی ظاہر ہوسکتی ہے۔ ایک عبارت (expression) میں جہاں کہیں بھی متغیر کا نام آتا ہے انٹرپریٹر اسے اس مخصوص متغیر کی قدر سے بدل دیتا ہے۔

پروگرام 5-3 مستطیل کا رقبہ معلوم کرنے کے لیے پائتھن میں ایک پروگرام لکھیے جب کہ اس کی لمبائی 10 اکائیاں اور چوڑائی 20 اکائیاں دی گئی ہیں۔

```
#Program 5-3
#To find the area of a rectangle
length = 10
```

```
breadth = 20
area = length * breadth
print(area)
```

نتیجہ:

```
200
```

5.5 تبصرے (COMMENTS)

تبصروں کا استعمال سورس کوڈ میں ریمارک یا نوٹ شامل کرنے کے لیے کیا جاتا ہے۔ انٹرپریٹر تبصروں پر عمل درآمد نہیں کرتا ہے۔ انھیں صرف اس مقصد کے تحت شامل کیا جاتا ہے کہ لوگوں کے لیے سورس کوڈ کو سمجھنا آسان ہو جائے۔ انھیں بنیادی طور پر سورس کوڈ کا مفہوم اور مقصد نیز اس کی ان پُٹ اور آؤٹ پُٹ ضروریات کو رقم کرنے کے لیے استعمال کیا جاتا ہے تاکہ بعد میں ہمیں یہ یاد رہے کہ یہ کس طرح کام کرتا ہے اور اسے کس طرح استعمال کرنا ہے۔ بڑے اور پیچیدہ قسم کے سافٹ ویئر کے معاملے میں یہ ممکن ہے کہ پروگرامرس کو ٹیموں کی شکل میں کام کرنا پڑے اور بعض اوقات، ایک پروگرامر کے ذریعے تحریر کردہ پروگرام کسی دوسرے پروگرامر کو استعمال کرنا پڑ سکتا ہے۔ ان حالات میں پروگرام کی ورکنگ کو سمجھنے کے لیے تبصروں کی شکل میں دستاویز سازی کی ضرورت ہوتی ہے۔ پائتھن میں تبصرہ کی شروعات # (ہیش کی علامت) سے ہوتی ہے # کے بعد آخر تک جو کچھ رقم کیا جاتا ہے اسے تبصرہ مانا جاتا ہے اور انٹرپریٹر اسٹیٹمینٹ کے دَل درآمد کے دوران اسے نظر انداز کر دیتا ہے۔

مثال 5.1

```
#Variable amount is the total spending on
#grocery
amount = 3400
#totalMarks is sum of marks in all the tests
#of Mathematics
totalMarks = test1 + test2 + finalTest
```

پروگرام 5-4 دو اعداد کا حاصل جمع معلوم کرنے کے لیے پائتھن پروگرام لکھیے۔

```
#Program 5-4
#To find the sum of two numbers
num1 = 10
num2 = 20
result = num1 + num2
print(result)
```

نتیجہ:

```
30
```

آبجیکٹ پر مبنی پروگرامنگ (OOP) کے سیاق میں آبجیکٹ حقیقی دنیا کی نمائندگی کرتی ہیں مثلاً ملازم، طالب علم، موٹر گاڑی، باکس، کتاب وغیرہ۔ C++، جاوا وغیرہ جیسی کسی بھی آبجیکٹ پر مبنی پروگرامنگ لینگویج میں ہر ایک آبجیکٹ سے وابستہ دو چیزیں ہوتی ہیں:(i) ڈیٹا یا خصوصیات اور (ii) طرز عمل یا طریقے (میتھڈ)۔ علاوہ ازیں کلاس اور کلاس سے متعلق نظام مراتب کے تصورات ہیں جن سے آبجیکٹ کو ٹھوس شکل میں پیش کیا جا سکتا ہے۔ تاہم OOP سے متعلق تصورات ہماری موجودہ بحث کے دائرے سے باہر ہیں۔

پائتھن بھی آبجیکٹ پر مبنی پروگرامنگ کے زمرے میں آتا ہے۔ حالاں کہ پائتھن میں آبجیکٹ کی تعریف کھلے طور پر کی جاتی ہے کیوں کہ ممکن ہے کچھ آبجیکٹ میں خصوصیات نہ ہوں اور کچھ آبجیکٹ میں میتھڈ نہ ہوں۔

5.6 ہر ایک چیز آبجیکٹ ہے (Everything is an Object)

پائتھن ہر ایک قدر (ویلیو) یا ڈیٹا آئٹم کو خواہ وہ عددی ہو، اسٹرنگ ہو یا کسی اور قسم کا ہو (اگلے باب میں بحث کی گئی ہے) اس معنی میں آبجیکٹ تصور کرتا ہے کہ اسے کسی متغیر کو تفویض کیا جا سکتا ہے یا کسی فنکشن کو آرگیومینٹ کے طور پر پاس کیا جا سکتا ہے۔

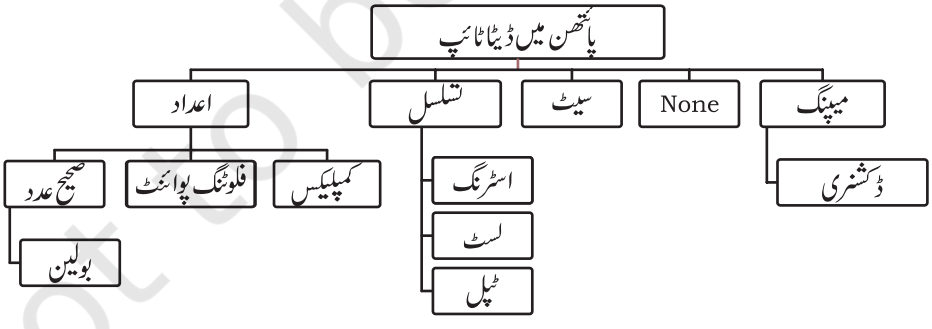
پائتھن میں ہر ایک آبجیکٹ کو ایک منفرد شناخت (ID) تفویض کی جاتی ہے جو اس آبجیکٹ کی تمام عمر کے لیے وہی رہتی ہے۔ یہ ID آبجیکٹ کے میموری ایڈریس کے مماثل ہے۔ فنکشن id() آبجیکٹ کی شناخت کو ظاہر کرتا ہے۔

مثال 5.2

```
>>> num1 = 20
>>> id(num1)
1433920576          #identity of num1
>>> num2 = 30 - 10
>>> id(num2)
1433920576          #identity of num2 and num1
                         #are same as both
refers to                              #object 20
```

5.7 ڈیٹا کی اقسام (Data Types)

پائتھن میں ہر ایک قدر (ویلیو) کا تعلق ایک مخصوص ڈیٹا ٹائپ سے ہے۔ ڈیٹا ٹائپ، ڈیٹا ویلیو کی اس قسم کی شناخت کرتا ہے جو ایک متغیر کو تفویض کی جا سکتی ہیں اور وہ عمل جس کو اس ڈیٹا پر انجام دیا جا سکتا ہے۔ پائتھن میں دست یاب ڈیٹا ٹائپ کی فہرست شکل 5.6 میں دی گئی ہے۔



شکل 5.6: پائتھن میں مختلف ڈیٹا ٹائپ

5.7.1 نمبر (Number)

نمبر ڈیٹا ٹائپ صرف عددی قدروں کو ہی اسٹور کرتا ہے۔ اسے مزید تین مختلف اقسام میں تقسیم کیا گیا ہے :

int، float اور complex۔

نوٹ

جدول 5.2 نیومیرک ڈیٹا ٹائپ

ٹائپ/کلاس	وضاحت	مثالیں
int	صحیح اعداد	–12, –3, 0, 125, 2
float	حقیقی یا فلوٹنگ پوائنٹ اعداد	–2.04, 4.0, 14.23
complex	پیچیدہ (ملتف) اعداد	3 + 4j, 2 – 2j

بولین ڈیٹا ٹائپ (Bool) صحیح عدد کی ذیلی قسم ہے۔ یہ ایک منفرد ڈیٹا ٹائپ ہے جو دو مستقلوں پر مشتمل ہوتا ہے یعنی صحیح (True) اور غلط (False)۔ بولین True ایک غیر صفر، غیر معدوم، غیر خالی قدر ہے۔ بولین False ایک صفر قدر ہے۔

آیئے اب بلٹ ان فنکشن Type() کا استعمال کرتے ہوئے متغیر کے ڈیٹا ٹائپ کو متعین کرنے کے لیے انٹرایکٹیو موڈ میں کچھ بیانات کو ایگزیکیوٹ کرنے کی کوشش کریں۔

مثال 5.3

```
>>> num1 = 10
>>> type(num1)
<class 'int'>

>>> num2 = -1210
>>> type(num2)
<class 'int'>

>>> var1 = True
>>> type(var1)
<class 'bool'>

>>> float1 = -1921.9
>>> type(float1)
<class 'float'>
>>> float2 = -9.8*10**2
>>> print(float2, type(float2))
-980.0000000000001 <class 'float'>

>>> var2 = -3+7.2j
>>> print(var2, type(var2))
(-3+7.2j) <class 'complex'>
```

صحیح اعداد، بولین، فلوٹ، وغیرہ جیسے سادہ ڈیٹا ٹائپ کے متغیرات واحد قدروں کے حامل ہوتے ہیں۔ لیکن اس قسم کے متغیرات کو اطلاعات کی طویل فہرست تفویض کرنا کارگر ثابت نہیں ہوتا مثلاً ایک سال میں مہینوں کے نام، کلاس میں طلبا کے نام فون بک میں درج نام اور نمبر یا عجائب گھر میں موجود فن پاروں اور نوادرات کی فہرست، وغیرہ۔ اس مقصد کے لیے پائتھن میں ٹپل، لسٹ، ڈکشنری اور سیٹ جیسے ڈیٹا ٹائپ دست یاب ہیں۔

نوٹ

5.7.2 تسلسل (Sequence)

پائتھن تسلسل کا ایک مرتب مجموعہ ہے جہاں ہر ایک آئٹم کی انڈیکسنگ صحیح عدد کے ذریعے کی جاتی ہے۔ پائتھن میں دستیاب تین قسم کے تسلسل ڈیٹا ٹائپ اسٹرنگ (Strings)، لسٹ (Lists) اور ٹپل (Tuple) ہیں ۔ہم ان میں سے ہر ایک کے بارے میں تفصیل سے آئندہ ابواب میں پڑھیں گے۔ ان ڈیٹا ٹائپ کا مختصر تعارف ذیل میں پیش کیا گیا ہے۔

(A) اسٹرنگ (String)

اسٹرنگ کیریکٹر کا مجموعہ ہے۔ یہ کیریکٹر حروف تہجی، ہندسے (Digit) یا اسپیشل کیریکٹر ہو سکتے ہیں جن میں اسپیس بھی شامل ہیں۔ اسٹرنگ ویلیو کو یا تو اکہرے واوین (مثلاً 'Hello') یا دوہرے واوین (مثلاً "Hello") میں لکھا جاتا ہے۔ واوین اسٹرنگ کا حصہ نہیں ہیں، ان کا استعمال اسٹرنگ کی ابتدا اور اختتام کی نشاندہی کرنے کے لیے کیا جاتا ہے مثال کے طور پر

```
>>> str1 = 'Hello Friend'
>>> str2 = "452"
```

ہم اسٹرنگ پر ریاضیاتی عمل انجام نہیں دے سکتے ہیں حتیٰ کہ اسٹرنگ میں عددی قدریں موجود ہوں جیسا کہ str2 میں ہے۔

(B) لسٹ (List)

لسٹ آئٹموں کا ایسا تسلسل ہے جو ایک دوسرے سے کوما کے ذریعے علاحدہ رہتے ہیں اور یہ آئٹم مربع بریکٹ [ ] میں بند رہتے ہیں۔

مثال 5.4

```
#To create a list
>>> list1 = [5, 3.4, "New Delhi", "20C", 45]
#print the elements of the list list1
>>> print(list1)
[5, 3.4, 'New Delhi', '20C', 45]
```

(C) ٹپل (Tuples)

ٹپل آئٹموں کا ایسا تسلسل ہے جو ایک دوسرے سے کوما کے ذریعے علاحدہ رہتے ہیں اور یہ آئٹم قوسین ( ) میں بند رہتے ہیں۔ یہ لسٹ کے برعکس ہے جس میں قدریں مربع بریکٹ [ ] میں بند رہتی ہیں۔ ایک مرتبہ تشکیل ہو جانے کے بعد ٹپل کو تبدیل نہیں کیا جا سکتا ہے۔

مثال 5.5

```
#create a tuple tuple1
```

نوٹ

```
>>> tuple1 = (10, 20, "Apple", 3.4, 'a')
#print the elements of the tuple tuple1
>>> print(tuple1)
(10, 20, "Apple", 3.4, 'a')
```

5.7.3 سیٹ (Set)

سیٹ آئٹموں کا غیر مرتب مجموعہ ہے جو ایک دوسرے سے کوما کے ذریعے علاحدہ رہتے ہیں اور یہ آئٹم منجھلے بریکٹ {} میں بند رہتے ہیں۔ یہ لسٹ کی طرح ہی ہیں سوائے اس کے کہ اس میں مکرر (Duplicate) اندراجات نہیں ہو سکتے ہیں۔ ایک مرتبہ تشکیل ہو جانے کے بعد سیٹ کے عناصر کو تبدیل نہیں کیا جا سکتا ہے۔

مثال 5.6

```
#create a set
>>> set1 = {10,20,3.14,"New Delhi"}
>>> print(type(set1))
<class 'set'>
>>> print(set1)
{10, 20, 3.14, "New Delhi"}
#duplicate elements are not included in set
>>> set2 = {1,2,1,3}
>>> print(set2)
{1, 2, 3}
```

5.7.4 معدوم (None)

None واحد قدر والا ایک مخصوص ڈیٹا ٹائپ ہے۔ اس کا استعمال کسی صورت حال میں قدر کی عدم موجودگی کو ظاہر کرنے کے لیے کیا جاتا ہے۔ None کسی مخصوص عمل کو سپورٹ نہیں کرتا ہے اور یہ نہ تو False ہے نہ ہی صفر (0)۔

مثال 5.7

```
>>> myVar = None
>>> print(type(myVar))
<class 'NoneType'>
>>> print(myVar)
None
```

5.7.5 میپنگ (Mapping)

میپنگ، پائتھن میں ایک غیر مرتب ڈیٹا ٹائپ ہے۔ فی الحال پائتھن میں صرف ایک معیاری میپنگ ڈیٹا ٹائپ ہے جسے ڈکشنری کہتے ہیں۔

(A) ڈکشنری (Dictionary)

پائتھن میں ڈکشنری کے اندر ڈیٹا آئٹم کی-ویلیو (Key-Value) جوڑی میں ہوتے ہیں۔ ڈکشنری کے

اندر بند ہوتے ہیں۔ڈکشنری کی مدد سے ڈیٹا کو تیزی کے ساتھ ایکسس کیا جاسکتا ہے۔ہر ایک **کی** اور اس کی ویلیو کے درمیان رابطہ کی علامت (:) کا استعمال کیا جاتا ہے۔ڈکشنری کی **کی**:ویلیو جوڑی کو ایکسس کرنے کے لیے **کی** کو استعمال کیا جاتا ہے۔کیز عام طور سے اسٹرنگ ہوتی ہیں اور ان کی ویلیو کوئی بھی ڈیٹا ٹائپ ہوسکتی ہے۔ڈکشنری میں موجود کسی بھی ویلیو کو ایکسس کرنے کے لیے ہمیں اس سے متعلق **کی** کی وضاحت مربع بریکٹ [ ] میں کرنی ہوگی۔

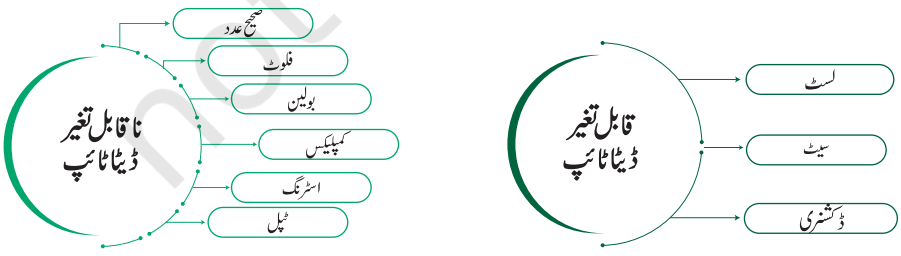
مثال 5.8

```
#create a dictionary
>>> dict1 = {'Fruit':'Apple',
'Climate':'Cold', 'Price(kg)':120}
>>> print(dict1)
{'Fruit': 'Apple', 'Climate': 'Cold',
'Price(kg)': 120}
>>> print(dict1['Price(kg)'])
120
```

5.7.6 قابل تغیر اور نا قابل تغیر ڈیٹا ٹائپ

بعض اوقات ہمیں پروگرام میں استعمال کیے جانے والے کچھ متغیرات کی قدروں کو تبدیل کرنے یا اپڈیٹ کرنے کی ضرورت پیش آجاتی ہے۔حالاں کہ پائتھن میں کچھ ایسے ڈیٹا ٹائپ ہیں کہ اگر ایک مرتبہ ان کی تشکیل ہوجائے اور انھیں قدریں تفویض کردی جائیں تو ہمیں ان کی قدروں کو تبدیل کرنے کی اجازت نہیں ہے۔ایسے متغیرات جن کی تشکیل ہوجانے اور انھیں قدر تفویض کرنے کے بعد تبدیل کیا جاسکے قابل تغیر (Mutable) کہلاتے ہیں۔ایسے متغیرات جن کی تشکیل ہوجانے اور انھیں قدر تفویض کرنے کے بعد تبدیل نہ کیا جاسکے نا قابل تغیر (Immutable) کہلاتے ہیں۔جب ایک نا قابل تغیر متغیر کی قدر کو تبدیل (Update) کرنے کی کوشش کی جاتی ہے تو پرانا متغیر ضائع ہوجاتا ہے اور میموری میں اسی نام سے نئے متغیر کی تشکیل ہوجاتی ہے۔

پائتھن ڈیٹا ٹائپ کی درجہ بندی قابل تغیر اور نا قابل تغیر کے تحت کی جاسکتی ہے جیسا کہ شکل 5.7 میں دکھایا گیا ہے۔

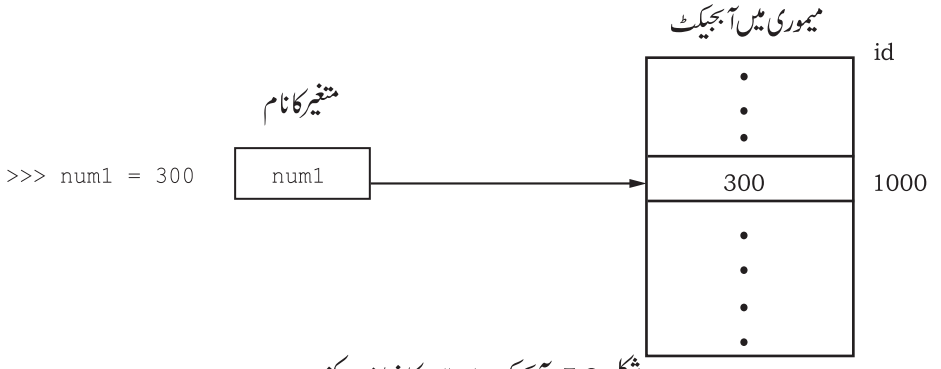


شکل 5.7 : ڈیٹا ٹائپ کی درجہ بندی

آیئے دیکھیں کہ جب کسی متغیر کی قدر کو تبدیل کرنے کی کوشش کی جاتی ہے تو کیا ہوتا ہے۔

```
>>> num1 = 300
```

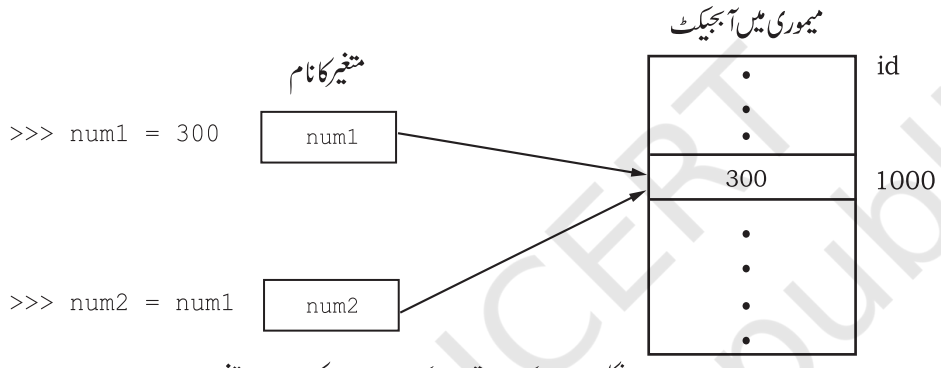
یہ بیان ایک آبجیکٹ کی تشکیل کرے گا جس کی قدر 300 ہے اور اس آبجیکٹ کا حوالہ شناخت کنندہ num1 کے ذریعے دیا جاتا ہے جیسا کہ شکل 5.8 میں دکھایا گیا ہے۔



شکل 5.8: آبجیکٹ اور اس کا شناخت کنندہ

```
>>> num2 = num1
```

بیان num2 = num1 میں num2 کو قدر 300 کا حوالہ دیا جائے گا اور ساتھ ہی اسے num1 کا بھی حوالہ دیا جا رہا ہے اور میموری لوکیشن نمبر فرض کیجیے 1000 پر اسٹور کیا گیا ہے۔ چنانچہ num1 حوالہ جاتی لوکیشن کو num2 کے ساتھ شیئر کرتا ہے جیسا کہ شکل 5.9 میں دکھایا گیا ہے۔

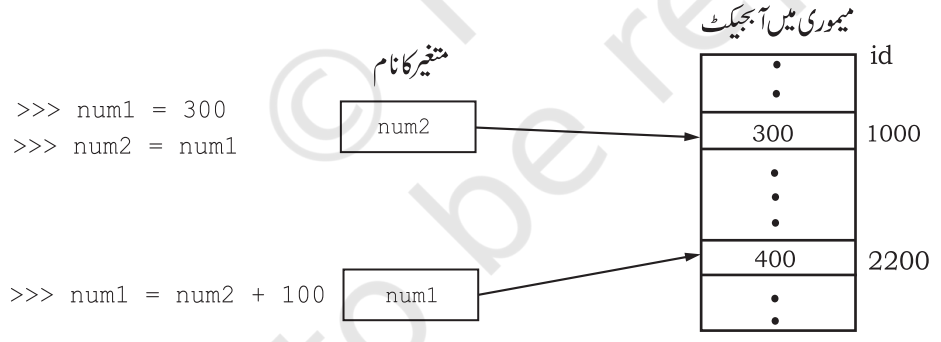


شکل 5.9: ایک ہی قدر اور ایک ہی شناخت کنندہ والے متغیرات

اس طرح، پائتھن ڈیٹا کو چھوڑ کر صرف حوالہ کی نقل تیار کر کے اسائنمینٹ کو موثر بناتا ہے:

```
>>>num1=num2+100
```

یہ بیان num1 = num2 + 100 متغیر num1 کو میموری لوکیشن نمبر مثلاً 2200 پر اسٹور کیے گئے آبجیکٹ کے ساتھ مربوط کر دیتا ہے جس کی قدر 400 ہے۔ کیوں کہ num1 ایک صحیح عدد ہے جو نا قابل تغیر قسم کا ہے چنانچہ اس کی تشکیل دوبارہ کی گئی ہے جیسا کہ شکل 5.10 میں دکھایا گیا ہے۔



شکل 5.10: مختلف قدروں اور مختلف شناخت کنندہ والے متغیرات

پائتھن اسٹرنگ کا موازنہ کیریکٹر کی ASCII قدر کا استعمال کر کے لغت نگاری کے انداز میں کرتا ہے۔ اگر دونوں اسٹرنگ کا پہلا کیریکٹر یکساں ہے تو دوسرے کیریکٹر کا موازنہ کیا جاتا ہے اور اسی طرح آگے بھی۔

5.7.7 پائتھن ڈیٹا ٹائپ کے استعمال کا فیصلہ کرنا

جب ہمیں قابل تکرار ڈیٹا کے ایسے مجموعے کی ضرورت ہوتی ہے جس میں بار بار ترمیم کرنے کی ضرورت پیش آتی ہے تو لسٹ کے استعمال کو ترجیح دی جاتی ہے۔ مثال کے طور پر اگر ہم کسی کلاس کے طلبا کے نام ایک فہرست

میں اسٹور کرتے ہیں تو جب نئے طلبا کورس میں شامل ہوتے ہیں یا کوئی چھوڑ کر چلا جاتا ہے تو لسٹ کو تبدیل کرنا آسان ہوتا ہے۔ ٹپل کا استعمال اس وقت کیا جاتا ہے جب ہمیں ڈیٹا میں کسی قسم کی تبدیلی مطلوب نہیں ہوتی ہے۔ مثال کے طور پر سال میں مہینوں کے نام۔ اگر ہمیں منفرد نوعیت کے عناصر مطلوب ہیں اور ہم چاہتے ہیں کہ ایک ہی عنصر دو مرتبہ نہ آئے تو سیٹ کے استعمال کو ترجیح دی جاتی ہے، مثال کے طور پر عجائب گھر میں فن پاروں اور نوادرات کی فہرست۔ اگر ہمارے ڈیٹا میں مسلسل طور پر ترمیم کی جارہی ہے یا ہمیں کسٹم کی بنیاد پر ڈیٹا کو تیزی کے ساتھ تلاش کرنا ہے یا ہمیں کی: ویلیو جوڑی کے درمیان منطقی ربط کی ضرورت ہے تو ڈکشنری کے استعمال کی صلاح دی جاتی ہے۔ موبائل فون بک ڈکشنری کا اچھا استعمال ہے۔

5.8 آپریٹرز (OPERATORS)

قدروں پر مخصوص ریاضیاتی یا منطقی عملوں کو انجام دینے کے لیے آپریٹر کا استعمال کیا جاتا ہے۔ وہ قدریں جن پر آپریٹر عمل انجام دیتا ہے آپرینڈ (Operand) کہلاتی ہیں۔ مثلاً num + 10 میں قدر 10 اور متغیر num آپرینڈ ہیں جب کہ جمع کا نشان (+) آپریٹر ہے۔ پائتھن میں مختلف قسم کے آپریٹر بروئے کار لائے جاتے ہیں جن کی زمرہ بندی کو اس سیکشن میں مختصراً بیان کیا گیا ہے۔

5.8.1 حسابی آپریٹرز (Arithmetic Operators)

پائتھن حسابی آپریٹر کے استعمال کی اجازت دیتا ہے۔ ان آپریٹر کو حساب کے چار بنیادی عملوں کے ساتھ ساتھ ماڈیولر تقسیم، فلور تقسیم اور قوت نما سے متعلق عملوں کو انجام دینے کے لیے استعمال کیا جاتا ہے۔

جدول 5.3 پائتھن میں حسابی آپریٹر

آپریٹر	عمل	وضاحت	مثال (لیب میں آزمائیے)

+	جمع	آپریٹر کے دونوں جانب کی عددی قدروں کو جمع کرتا ہے
		اس آپریٹر کا استعمال آپریٹر کے دونوں جانب موجود دو اسٹرنگ کو یکجا کرنے کے لیے بھی استعمال کیا جاسکتا ہے
			`>>> num1 = 5`
			`>>> num2 = 6`
			`>>> num1 + num2`
			`11`
			`>>> str1 = "Hello"`
			`>>> str2 = "India"`
			`>>> str1 + str2`
			`'HelloIndia'`

-	تفریق	آپریٹر کے دائیں جانب والی قدر کو بائیں جانب والی قدر میں سے گھٹا دیتا ہے۔	`>>> num1 = 5`
			`>>> num2 = 6`
			`>>> num1 - num2`
			`-1`

*	ضرب	آپریٹر کے دونوں جانب کی عددی قدروں کو ضرب کرتا ہے۔ اگر آپریٹر کے بائیں جانب اسٹرنگ اور دائیں جانب صحیح عددی قدر ہے تو یہ اسٹرنگ کو دہراتا ہے۔	`>>> num1 = 5`
			`>>> num2 = 6`
			`>>> num1 * num2`
			`30`
			`>>> str1 = 'India'`
			`>>> str1 * 2`
			`'IndiaIndia'`

/	تقسیم	آپریٹر کے بائیں جانب والی قدر کو دائیں جانب والی قدر سے تقسیم کرتا ہے اور خارج قسمت کو ظاہر کر دیتا ہے۔	`>>> num1 = 8`
			`>>> num2 = 4`
			`>>> num2 / num1`
			`0.5`

%	ماڈیولس (مقیاس)	آپریٹر کے بائیں جانب والی قدر کو دائیں جانب والی قدر سے تقسیم کرتا ہے اور باقی کو ظاہر کر دیتا ہے۔	`>>> num1 = 13`
			`>>> num2 = 5`
			`>>> num1 % num2`
			`3`

//	فلور تقسیم	آپریٹر کے بائیں جانب والی قدر کو دائیں جانب والی قدر سے تقسیم کرتا ہے اور اعشاریہ والے حصے کو مسترد کرتے ہوئے خارج قسمت کو ظاہر کر دیتا ہے۔ بعض اوقات اسے صحیح عددی تقسیم بھی کہتے ہیں۔	`>>> num1 = 13`
			`>>> num2 = 4`
			`>>> num1 // num2`
			`3`
			`>>> num2 // num1`
			`0`

**	قوت نما (Exponent)	آپریٹر پر قوت نما سے متعلق تحسیبات کو انجام دیتا ہے یعنی بائیں جانب والے آپریٹر پر دائیں جانب والے آپریٹر کو قوت کے طور پر استعمال کرتا ہے۔	`>>> num1 = 3`
			`>>> num2 = 4`
			`>>> num1 ** num2`
			`81`

5.8.2 نسبتی آپریٹرز (Relational Operators)

نسبتی آپریٹر اپنے دونوں جانب موجود آپریٹر کی قدروں کا موازنہ کرتا ہے اور ان کے درمیان تعلق کو متعین کرتا ہے۔ مندرجہ ذیل مثالوں کے لیے پائتھن متغیرات num1 = 10، num2 = 0، num3 = 10، str1 = "Good" اور str2 = "Afternoon" کو فرض کیجیے:

جدول 5.4 : پائتھن میں نسبتی آپریٹرز

آپریٹر	عمل	وضاحت	مثال (لیب میں آزمائیے)

==	مساوی ہے	اگر دونوں آپریٹر (زیر عمل قدریں) مساوی ہیں تو یہ True ظاہر کرتا ہے بصورت دیگر False ظاہر کرتا ہے۔	`>>> num1 == num2`
			`False`
			`>> str1 == str2`
			`False`

!=	مساوی نہیں ہے	اگر دونوں آپریٹر (زیر عمل قدریں) مساوی نہیں ہیں تو یہ True ظاہر کرتا ہے بصورت دیگر False ظاہر کرتا ہے۔	`>>> num1 != num2`
			`True`
			`>>> str1 != str2`
			`True`
			`>>> num1 != num3`
			`False`

>	سے بڑا	اگر بائیں جانب والے آپریٹر کی قدر دائیں جانب والے آپریٹر سے زیادہ ہے تو یہ True ظاہر کرتا ہے بصورت دیگر False ظاہر کرتا ہے۔	`>>> num1 > num2`
			`True`
			`>>> str1 > str2`
			`True`

<	سے چھوٹا	اگر بائیں جانب والے آپریںڈ کی قدر دائیں جانب والے آپریںڈ سے کم ہے تو یہ True ظاہر کرتا ہے بصورت دیگر False ظاہر کرتا ہے۔	>>> num1 < num3
			False
			>>> str2 < str1
			True

>=	سے بڑا یا مساوی	اگر بائیں جانب والے آپریںڈ کی قدر دائیں جانب والے آپریںڈ سے زیادہ یا اس کے مساوی ہے تو یہ True ظاہر کرتا ہے بصورت دیگر False ظاہر کرتا ہے۔	>>> num1 >= num2
			True
			>>> num2 >= num3
			False
			>>> str1 >= str2
			True

<=	سے چھوٹا یا مساوی	اگر بائیں جانب والے آپریںڈ کی قدر دائیں جانب والے آپریںڈ سے کم یا اس کے مساوی ہے تو یہ True ظاہر کرتا ہے بصورت دیگر False ظاہر کرتا ہے۔	>>> num1 <= num2
			False
			>>> num2 <= num3
			True
			>>> str1 <= str2
			False

5.8.3 اسائنمینٹ آپریٹرز (Assignment Operators)

اسائنمینٹ آپریٹر اپنے بائیں جانب موجود متغیر کو قدر تفویض کرتا ہے یا اس کی قدر کو تبدیل کرتا ہے۔

جدول 5.5 : پائتھن میں اسائنمنٹ آپریٹر

آپریٹر	وضاحت	مثال (لیب میں آزمایئے)

=	دائیں جانب والے آپریںڈ کی قدر بائیں جانب والے آپریںڈ کو تفویض کرتا ہے۔	>>> num1 = 2
		>>> num2 = num1
		>>> num2
		2
		>>> country = 'India'
		>>> country
		'India'

+=	یہ دائیں جانب والے آپریںڈ کی قدر بائیں جانب والے آپریںڈ میں جمع کرتا ہے اور نتیجہ بائیں جانب والے آپریںڈ کو تفویض کر دیتا ہے۔
	نوٹ: x += y کی قدر وہی ہے جو x = x + y کی ہے۔
		>>> num1 = 10
		>>> num2 = 2
		>>> num1 += num2
		>>> num1
		12
		>>> num2
		2
		>>> str1 = 'Hello'
		>>> str2 = 'India'
		>>> str1 += str2
		>>> str1
		'HelloIndia'

-=	یہ دائیں جانب والے آپریںڈ کی قدر کو بائیں جانب والے آپریںڈ میں سے گھٹاتا ہے اور نتیجہ بائیں جانب والے آپریںڈ کو تفویض کر دیتا ہے۔
	نوٹ: x -= y کی قدر وہی ہے جو x = x - y کی ہے۔
		>>> num1 = 10
		>>> num2 = 2
		>>> num1 -= num2
		>>> num1
		8

*=	یہ دائیں جانب والے آپریںڈ کی قدر کو بائیں جانب والے آپریںڈ سے ضرب کرتا ہے اور نتیجہ بائیں جانب والے آپریںڈ کو تفویض کر دیتا ہے۔
	نوٹ: x *= y کی قدر وہی ہے جو x = x * y کی ہے۔
		>>> num1 = 2
		>>> num2 = 3
		>>> num1 *= 3
		>>> num1
		6
		>>> a = 'India'
		>>> a *= 3
		>>> a
		'IndiaIndiaIndia'

/=		>>> num1 = 6
		>>> num2 = 3
		>>> num1 /= num2
		>>> num1
		2.0

%=	یہ بائیں جانب والے آپریںڈ کی قدر کو دائیں جانب والے آپریںڈ سے تقسیم کرتا ہے اور نتیجہ بائیں جانب والے آپریںڈ کو تفویض کر دیتا ہے۔
	نوٹ: x/= y کی قدر وہی ہے جو x = x / y کی ہے۔
		>>> num1 = 7
		>>> num2 = 3
		>>> num1 %= num2
		>>> num1
		1

//=	یہ دو آپریںڈ کو استعمال کر کے مقیاس (ماڈیولس) کا عمل انجام دیتا ہے اور نتیجہ بائیں جانب والے آپریںڈ کو تفویض کر دیتا ہے۔
	نوٹ: x%= y کی قدر وہی ہے جو x = x % y کی ہے۔
		>>> num1 = 7
		>>> num2 = 3
		>>> num1 //= num2
		>>> num1
		2

**=	یہ آپریںڈ پر قوت نما سے متعلق تحسیب کا عمل انجام دیتا ہے اور نتیجہ بائیں جانب والے آپریںڈ کو تفویض کر دیتا ہے۔
	نوٹ: x**= y کی قدر وہی ہے جو x = x ** y کی ہے۔
		>>> num1 = 2
		>>> num2 = 3
		>>> num1 **= num2
		>>> num1
		8

5.8.4 منطقی آپریٹرز (Logical Operators)

ایسے تین منطقی آپریٹر ہیں جنھیں پائتھن میں استعمال کیا جا سکتا ہے۔ یہ آپریٹر (and, or, not) صرف چھوٹے حروف میں لکھے جاتے ہیں۔ منطقی آپریٹر دونوں جانب کے منطقی آپریںڈ کی بنیاد پر True یا False کے لیے تحسیب کرتا ہے۔ ہر ایک قدر منطقی طور پر یا تو True ہوتی ہے یا False۔ اگر صارف کی طرف سے کوئی انتخاب نہ کیا جائے (By default) تو سبھی قدریں True ہوتی ہیں سوائے None، False، 0 (صفر)، خالی مجموعے "" ، () ، [] ، {} اور دیگر چند مخصوص قدروں کے۔ چنانچہ اگر ہم یہ کہیں کہ num1 = 10، num2 = -20 تو num1 اور num2 منطقی طور پر True ہیں۔

جدول 5.6 : پائتھن میں منطقی آپریٹر

آپریٹر	عمل	وضاحت	مثال (لیب میں آزمایئے)

and	منطقی AND	اگر دونوں آپرینڈ True ہیں تو شرط True ہوجاتی ہے۔	>>> True and True
			True
			>>> num1 = 10
			>>> num2 = -20
			>>> bool(num1 and num2)
			True
			>>> True and False
			False
			>>> num3 = 0
			>>> bool(num1 and num3)
			False
			>>> False and False
			False

or	منطقی OR	اگر دونوں میں سے کوئی ایک آپرینڈ True ہے تو شرط True ہوجاتی ہے۔	>>> True or True
			True
			>>> True or False
			True
			>>> bool(num1 or num3)
			True
			>>> False or False
			False

not	منطقی NOT	اپنے آپرینڈ کی منطقی حالت کو الٹنے کے لیے استعمال ہوتا ہے۔	>>> num1 = 10
			>>> bool(num1)
			True
			>>> not num1
			>>> bool(num1)
			False

5.8.5 شناختی آپریٹرز (Identity Operators)

شناختی آپریٹر کا استعمال یہ متعین کرنے کے لیے کیا جاتا ہے کہ آیا کسی متغیر کی قدر ایک مخصوص قسم کی ہے یا نہیں۔
شناختی آپریٹر کا استعمال یہ متعین کرنے کے لیے بھی کیا جاسکتا ہے کہ دو متغیر ایک ہی آبجیکٹ سے متعلق ہیں یا نہیں۔ دو قسم کے شناختی آپریٹر ہیں۔

جدول 5.7: پائتھن میں شناختی آپریٹرز

آپریٹر	وضاحت	مثال (لیب میں آزمایئے)

is	اگر آپریٹر کے دونوں طرف والے متغیر ایک ہی میموری لوکیشن کی طرف اشارہ کرتے ہیں تو یہ True ظاہر کرتا ہے بصورت دیگر False ظاہر کرتا ہے۔ var1 is var2 کا نتیجہ True ہوگا اگر id(var1) اور id(var2) دونوں مساوی ہیں۔	`>>> num1 = 5`
		`>>> type(num1) is int`
		`True`
		`>>> num2 = num1`
		`>>> id(num1)`
		`1433920576`
		`>>> id(num2)`
		`1433920576`
		`>>> num1 is num2`
		`True`

is not	اگر آپریٹر کے دونوں طرف والے متغیر ایک ہی میموری لوکیشن کی طرف اشارہ کرتے ہیں تو یہ False ظاہر کرتا ہے بصورت دیگر True ظاہر کرتا ہے۔ var1 is not var2 کا نتیجہ True ہوگا اگر id(var1) اور id(var2) دونوں مساوی نہیں ہیں۔	`>>> num1 is not num2` `False`

5.8.6 ممبرشپ آپریٹرز (Membership Operators)

ممبرشپ آپریٹر کا استعمال اس بات کی جانچ کرنے کے لیے کیا جاتا ہے کہ آیا قدر دیے ہوئے تسلسل کا ممبر (رکن) ہے یا نہیں۔

جدول 5.8 : پائتھن میں ممبرشپ آپریٹر

آپریٹر	وضاحت	مثال (لیب میں آزمایئے)

in	اگر متغیر/قدر متعینہ تسلسل میں موجود ہے تو یہ True ظاہر کرتا ہے بصورت دیگر False ظاہر کرتا ہے۔	`>>> a = [1,2,3]`
		`>>> 2 in a`
		`True`
		`>>> '1' in a`
		`False`

not in	اگر متغیر/قدر متعینہ تسلسل میں موجود نہیں ہے تو یہ True ظاہر کرتا ہے بصورت دیگر False ظاہر کرتا ہے۔	`>>> a = [1,2,3]`
		`>>> 10 not in a`
		`True`
		`>>> 1 not in a`
		`False`

5.9 عبارتیں (EXPRESSIONS)

عبارت کی تعریف مستقلوں (Constants)، متغیروں (Variables) اور آپریٹروں (Operators) کے مجموعے کے طور پر کی جاتی ہے۔ ایک عبارت ہمیشہ قدر کی تحسیب کرتی ہے۔ ایک قدر یا اکیلے متغیر کو بھی

عبارت تصور کیا جاسکتا ہے لیکن اکیلے آپریٹر کو عبارت تصور نہیں کیا جاتا۔ درست عبارتوں کی کچھ مثالیں ذیل میں دی گئی ہیں۔

```
  (i) 100                (iv) 3.0 + 3.14
 (ii) num                 (v) 23/3 -5 * 7(14 -2)
(iii) num - 20.4         (vi) "Global" + "Citizen"
```

5.9.1 آپریٹر کی پیش روی (Precedence of Operator)

عبارت کی تحسیب کا دارومدار آپریٹر کی پیش روی پر ہوتا ہے۔ اگر کسی عبارت میں مختلف قسم کے آپریٹر موجود ہیں تو پیش روی اس بات کا تعین کرتی ہے کہ کون سے آپریٹر کا اطلاق پہلے ہوگا۔ زیادہ پیش روی والے آپریٹر کی تحسیب کم پیش روی والے آپریٹر سے پہلے کی جاتی ہے۔ ابھی تک جتنے بھی آپریٹروں کا مطالعہ کیا گیا ہے ان میں سے بیشتر بائنری آپریٹر ہیں۔ بائنری آپریٹر دو آپرینڈ والے آپریٹر ہیں۔ یونیری آپریٹر (Unary operator) کو صرف ایک آپرینڈ کی ضرورت ہوتی ہے اور بائنری آپریٹروں کے مقابلے میں ان کی پیش روی زیادہ ہوتی ہے۔ نفی (-) کے ساتھ ساتھ جمع (+) آپریٹر یونیری اور بائنری دونوں قسم کے آپریٹر کے طور پر کام کرتے ہیں، لیکن not یونیری منطقی آپریٹر ہے۔

```
#Depth is using - (minus) as unary operator
Value = -Depth
#not is a unary operator, negates True
print(not(True))
```

مندرجہ ذیل جدول میں سبھی آپریٹر کی پیش روی کو زیادہ سے کم کی ترتیب میں دیا گیا ہے۔

جدول 5.9 : پائتھن میں سبھی آپریٹروں کی پیش روی

پیش روی کی ترتیب	آپریٹرز	وضاحت
1	**	قوت نما سے متعلق عمل (کی قوت)
2	~ ,+, -	اتمامی، یونیری جمع اور یونیری نفی
3	* ,/, %, //	ضرب، تقسیم، ماڈیولس اور فلور تقسیم
4	+, -	جمع اور تفریق
5	<= , < , > , >=, == , !=	نسبتی آپریٹرز اور مساویت آپریٹرز
6	=, %=, /=, //=, -=, +=, *=, **=	اسائنمینٹ آپریٹرز
7	is, is not	شناختی آپریٹرز
8	in, not in	ممبرشپ آپریٹرز
9	not	منطقی آپریٹرز
10	and	
11	or	

نوٹ

نوٹ:

a) آپریٹروں کی پیش روی کو منسوخ کرنے کے لیے قوسین کا استعمال کیا جاسکتا ہے۔ ( ) کے اندر موجود عبارت کی تحسیب پہلے کی جاتی ہے۔

b) یکساں پیش روی والے آپریٹروں کے معاملے میں عبارت کی تحسیب بائیں سے دائیں طرف کی جاتی ہے۔

مثال 5.9 پائتھن مندرجہ ذیل عبارت کی تحسیب کس طرح کرے گا؟

```
20 + 30 * 40
```

حل:

```
= 20 + (30 * 40)          #Step 1
```

+ کی پیش روی + کے مقابلے زیادہ ہے

```
= 20 + 1200               #Step 2
= 1220                    #Step 3
```

مثال 5.10 پائتھن مندرجہ ذیل عبارت کی تحسیب کس طرح کرے گا؟

```
20 - 30 + 40
```

حل:

(-) اور (+) دونوں آپریٹروں کی پیش روی مساوی ہے۔ چنانچہ پہلا آپریٹر یعنی تفریق کا اطلاق دوسرے آپریٹر یعنی جمع سے پہلے ہوگا (بائیں سے دائیں)

```
= (20 - 30) + 40      #Step 1
= -10 + 40            #Step 2
= 30                  #Step 3
```

مثال 5.11 پائتھن مندرجہ ذیل عبارت کی تحسیب کس طرح کرے گا؟

```
(20 + 30) * 40
```

حل:

```
= (20 + 30) * 40          # Step 1
```

قوسین ( ) کا استعمال کرکے ہم + کی پیش روی کو * کے مقابلے زیادہ کرسکتے ہیں۔

```
= 50 * 40                 # Step 2
= 2000                    # Step 3
```

مثال 5.12 پائتھن مندرجہ ذیل عبارت کی تحسیب کس طرح کرے گا؟

```
15.0 / 4 + (8 + 3.0)
```

نوٹ

حل:

```
= 15.0 / 4 + (8.0 + 3.0)   #Step 1
= 15.0 / 4.0 + 11.0        #Step 2
= 3.75 + 11.0              #Step 3
= 14.75                    #Step 4
```

5.10 بیان (STATEMENT)

پائتھن میں 'بیان' اس کوڈ کی اکائی ہے جسے پائتھن انٹرپریٹر ایگزیکیوٹ کرسکتا ہے۔

مثال 5.13

```
>>> x = 4              #assignment statement
>>> cube = x ** 3      #assignment statement
>>> print (x, cube)    #print statement
4 64
```

5.11 ان پُٹ اور آؤٹ پُٹ (INPUT AND OUTPUT)

بعض اوقات کسی پروگرام کو آخری استعمال کنندہ سے کچھ ان پُٹ ڈیٹا یا اطلاع حاصل کرنے اور اسے پروسیس کرکے مطلوبہ نتیجہ فراہم کرنے کے لیے استعمال کنندہ سے تعامل کرنے کی ضرورت پیش آتی ہے۔ پائتھن میں، استعمال کنندہ سے ان پُٹ حاصل کرنے کے لیے ہمارے پاس input() فنکشن ہے۔ input() فنکشن استعمال کنندہ کو ڈیٹا داخل کرنے کی تجویز پیش کرتا ہے۔ یہ استعمال کنندہ کے سبھی ان پُٹ کو اسٹرنگ کے طور پر حاصل کرتا ہے۔ استعمال کنندہ خواہ کوئی عدد داخل کرے یا اسٹرنگ لیکن input() فنکشن اس ڈیٹا کو اسٹرنگ کے طور پر لیتا ہے۔ input() کے لیے سنٹیکس مندرجہ ذیل ہے:

```
input ([Prompt])
```

prompt وہ اسٹرنگ ہے جسے ہم ان پُٹ لینے سے پہلے اسکرین پر ظاہر کرسکتے ہیں اور یہ اختیاری ہے۔ جب پرومپٹ (prompt) کا تعین کرلیا جاتا ہے تو پہلے اسے اسکرین پر ظاہر کیا جاتا ہے جس کے بعد استعمال کنندہ ڈیٹا داخل یا درج کرسکتا ہے۔ جو کچھ کی بورڈ سے ٹائپ کیا جاتا ہے input( ) اسے حاصل کرتا ہے، اسٹرنگ میں تبدیل کرتا ہے اور اسے اسائنمینٹ آپریٹر (=) کے بائیں طرف موجود متغیر کو تفویض کردیتا ہے۔ اینٹر کی دبانے پر ان پُٹ فنکشن کے لیے ڈیٹا کو داخل کرنے کا کام ختم ہوجاتا ہے۔

مثال 5.14

```
>>> fname = input("Enter your first name: ")
Enter your first name: Arnab
>>> age = input("Enter your age: ")
Enter your age: 19
>>> type(age)
<class 'str'>
```

متغیر fname استعمال کنندہ کے ذریعے داخل کی گئی اسٹرنگ 'Arnab' کو حاصل کرے گا۔ اسی طرح متغیر age اسٹرنگ '19' کو حاصل کرے گا۔ ہم استعمال کنندہ سے حاصل کیے گئے اسٹرنگ ڈیٹا کے ڈیٹا ٹائپ کو مناسب عددی قدر میں تبدیل کر سکتے ہیں۔ مثال کے طور پر مندرجہ ذیل بیان حاصل کی گئی اسٹرنگ کو صحیح عدد میں تبدیل کر دے گا۔ اگر استعمال کنندہ کوئی غیر عددی قدر داخل کرتا ہے تو غلطی سرزد ہونے کا پیغام ظاہر ہو جائے گا۔

مثال 5.15

```
#function int() to convert string to integer
>>> age = int( input("Enter your age:"))
Enter your age: 19
>>> type(age)
<class 'int'>
```

ان پُٹ ڈیٹا کو معیاری آؤٹ پُٹ ڈیوائس مثلاً اسکرین پر ظاہر کرنے کے لیے پائتھن print() فنکشن کا استعمال کرتا ہے۔ ہم فنکشن کا مطالعہ باب 7 میں کریں گے۔ فنکشن print() عبارت کو اسکرین پر ظاہر کرنے سے پہلے اس کی قدر پیمائی کرتا ہے۔ print() ایک مکمل لائن کو آؤٹ پُٹ کے طور پر پیش کرتا ہے اور مابعد آؤٹ پُٹ کے لیے اگلی لائن کا رخ کرتا ہے۔ print() کے لیے سنٹیکس مندرجہ ذیل ہے۔

```
print(value [, ..., sep = ' ', end = '\n'])
```

- sep: اختیاری پیرامیٹر sep آؤٹ پُٹ قدروں کے درمیان علاحدگی کی علامت ہے۔ ہم علاحدگی کے لیے کسی کیریکٹر، صحیح عدد یا اسٹرنگ کا استعمال کر سکتے ہیں۔ اگر استعمال کنندہ کوئی متبادل پیش نہیں کرتا ہے تو پروگرام علاحدگی کے لیے اسپیس (space) کا استعمال کرتا ہے۔
- end: یہ بھی اختیاری ہے اور اس کی مدد سے ہم آخری قدر کے بعد شامل کی جانے والی کسی بھی اسٹرنگ کو متعین کر سکتے ہیں۔ ڈیفالٹ نئی سطر ہے۔

مشاہدہ کیجیے کہ جمع کا نشان دو اسٹرنگ کے درمیان خالی جگہ کو شامل نہیں کرتا ہے جب کہ کوما پرنٹ اسٹیٹمنٹ میں دو اسٹرنگ کے درمیان خالی جگہ داخل کر دیتا ہے۔

مثال 5.16

بیان	نتیجہ
print("Hello")	Hello
print(10*2.5)	25.0
print("I" + "love" + "my" + "country")	Ilove my country
print("I'm", 16, "years old")	I'm 16 years old

مذکورہ بالا مثال میں تیسرا پرنٹ فنکشن اسٹرنگس کو جوڑ رہا ہے اور ہم اسٹرنگس کو جوڑنے کے لیے ان کے درمیان جمع کی علامت (+) کا استعمال کرتے ہیں۔ چوتھا پرنٹ فنکشن بھی اسٹرنگس کو جوڑتا ہوا نظر آتا ہے لیکن یہاں اسٹرنگس کے درمیان کوما (،) کا استعمال کیا گیا ہے۔ دراصل یہاں ہم پرنٹ فنکشن کو کئی آرگیومینٹ پاس

نوٹ

کر رہے ہیں جنھیں ایک دوسرے سے کوما کے ذریعے علاحدہ کیا گیا ہے۔ کیوں کہ آرگیومینٹ مختلف قسم کے ہوسکتے ہیں چنانچہ پرنٹ فنکشن یہاں اسٹرنگ کے ساتھ صحیح عدد (16) کو حاصل کرتا ہے۔ لیکن پرنٹ اسٹیٹمنٹ میں اگر مختلف قسم کی قدریں ہیں اور کوما کی جگہ '+' کا استعمال کیا جاتا ہے تو یہ غلطی سرزد ہونے کا پیغام ظاہر کردے گا جیسا کہ اگلے سیکشن میں صریح تبدیلی کے تحت بحث کی گئی ہے۔

5.12 ٹائپ کی تبدیلی (TYPE CONVERSION)

مندرجہ ذیل پروگرام پر غور کیجیے

```
num1 = input("Enter a number and I'll double
it: ")
num1 = num1 * 2
print(num1)
```

پروگرام سے یہ توقع کی جاتی ہے کہ وہ حاصل کیے گئے عدد کی دوگنی قدر کو ظاہر اور اسے متغیر num1 میں اسٹور کرے گا۔ لہٰذا اگر صارف عدد 2 داخل کرتا ہے تو وہ یہ توقع کرتا ہے کہ پروگرام آؤٹ پُٹ کے طور پر عدد 4 کو ظاہر کرے گا لیکن پروگرام مندرجہ ذیل کو ظاہر کرتا ہے۔

```
Enter a number and I'll double it: 2
22
```

اس کی وجہ یہ ہے کہ ان پٹ فنکشن کے ذریعے پیش کی گئی قدر ایک اسٹرنگ ("2") ہے۔ نتیجتاً اسٹیٹمنٹ num1 = num1 * 2 میں num1 اسٹرنگ ویلیو کا حامل ہے اور * تکراری آپریٹر کے طور پر کام کرتا ہے چنانچہ آؤٹ پُٹ "22" ہوجاتا ہے۔ آؤٹ پُٹ کے طور پر عدد 4 حاصل کرنے کے لیے استعمال کنندہ کے ذریعے داخل کی گئی قدر کے ڈیٹا ٹائپ کو صحیح عدد (Integer) میں تبدیل کرنا ہوگا۔ چنانچہ ہم پروگرام میں مندرجہ ذیل ترمیمات کرتے ہیں۔

```
num1 = input("Enter a number and I'll double
it: ")
num1 = int(num1)   #convert string input to
                   #integer
num1 = num1 * 2
print(num1)
```

اب، پروگرام متوقع آؤٹ پُٹ ظاہر کرے گا جیسا کہ ذیل میں دکھایا گیا ہے۔

```
Enter a number and I'll double it: 2
4
```

آیئے اب یہ سمجھنے کی کوشش کریں کہ ڈیٹا ٹائپ کی تبدیلی کیا ہے اور یہ کس طرح کام کرتی ہے۔ جب بھی ضرورت ہو ہم پائتھن میں متغیر کے ڈیٹا ٹائپ کو دوسرے ڈیٹا ٹائپ میں تبدیل کرسکتے ہیں۔ ڈیٹا ٹائپ کی تبدیلی کے عمل کو دو طریقوں سے انجام دیا جاسکتا ہے: یا تو صریحاً (جبری)، اس وقت جب پروگرامر ایک ڈیٹا ٹائپ کو دوسرے ڈیٹا ٹائپ میں تبدیل کرنے کے لیے انٹرپریٹر کو اس کی صراحت کرتا ہے یا مضمر طور پر، جب

نوٹ

انٹرپریٹر خود اس ضرورت کو محسوس کرتا ہے اور تبدیلی کے عمل کو خود کار انداز میں انجام دیتا ہے۔

5.12.1 صریحی تبدیلی (Explicit Conversion)

صریحی تبدیلی کو ٹائپ کاسٹنگ بھی کہتے ہیں اور یہ اس وقت ہوتا ہے جب ڈیٹا ٹائپ کی تبدیلی واقع ہوتی ہے کیوں کہ یہ تبدیلی پروگرام کے اندر پروگرامر کے ذریعے جبری طور پر کی گئی ہے۔ ڈیٹا ٹائپ صریحی تبدیلی کی عمومی شکل ذیل میں دی گئی ہے:

```
(new_data_type) (expression)
```

ڈیٹا ٹائپ کی صریحی تبدیلی کے دوران ڈیٹا یا اطلاعات کے ضائع ہونے کا خطرہ بنا رہتا ہے کیوں کہ ہم کسی عبارت کو ایک مخصوص ڈیٹا ٹائپ ہونے کے لیے مجبور کرتے ہیں۔ مثال کے طور پر x = 20.67 کی فلوٹنگ ویلیو کو صحیح عددی ٹائپ میں تبدیل کرنا، یعنی int(x) کسری حصے 67. کو مسترد کر دے گا۔ ذیل میں پائتھن کے کچھ ایسے فنکشن دیے جا رہے ہیں جن کا استعمال کسی عبارت یا متغیر کو صریحی طور پر مختلف ڈیٹا ٹائپ میں تبدیل کرنے کے لیے کیا جاتا ہے۔

جدول 5.10 : پائتھن میں ڈیٹا ٹائپ کی صریحی تبدیلی کے لیے فنکشن

فنکشن	وضاحت
`int(x)`	x کو صحیح عدد میں تبدیل کرتا ہے۔
`float(x)`	x کو فلوٹنگ پوائنٹ نمبر میں تبدیل کرتا ہے۔
`str(x)`	x کو اسٹرنگ میں تبدیل کرتا ہے۔
`chr(x)`	x کو کیریکٹر میں تبدیل کرتا ہے۔
`ord(x)`	x کو یونیکوڈ کیریکٹر میں تبدیل کرتا ہے۔

پروگرام 5-5 صحیح عدد (int) سے فلوٹ میں صریحی ڈیٹا ٹائپ تبدیلی کے لیے پروگرام

```
#Program 5-5
#Explicit type conversion from int to float
num1 = 10
num2 = 20
num3 = num1 + num2
print(num3)
print(type(num3))
num4 = float(num1 + num2)
print(num4)
print(type(num4))
```

نتیجہ:

```
30
<class 'int'>
30.0
<class 'float'>
```

پروگرام 5-6 فلوٹ سے صحیح عدد (int) میں ڈیٹا ٹائپ کی صریحی تبدیلی کے لیے پروگرام

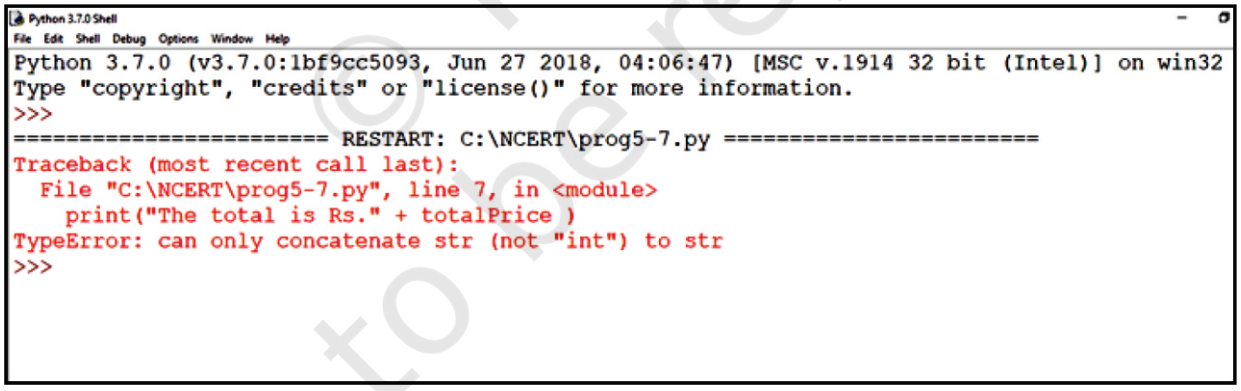
```
#Program 5-6
#Explicit type conversion from float to int
num1 = 10.2
num2 = 20.6
num3 = (num1 + num2)
print(num3)
print(type(num3))
num4 = int(num1 + num2)
print(num4)
print(type(num4))
```

نتیجہ:

```
30.8
<class 'float'>
30
<class 'int'>
```

پروگرام 5-7 اعداد اور اسٹرنگ کے درمیان ڈیٹا ٹائپ کی تبدیلی کی مثال

```
#Program 5-7
#Type Conversion between Numbers and Strings
priceIcecream = 25
priceBrownie = 45
totalPrice = priceIcecream + priceBrownie
print("The total is Rs." + totalPrice )
```



شکل 5.11 : پروگرام 5-7 کا آؤٹ پُٹ

جیسا کہ شکل 5.11 میں دکھایا گیا ہے ایگزیکیوشن کے دوران پروگرام 5-7 ایک غلطی کو ظاہر کرتا ہے جس میں یہ اطلاع دی جاتی ہے کہ انٹرپریٹر صحیح عددی ڈیٹا ٹائپ کو صریحی طور پر اسٹرنگ میں تبدیل نہیں کرسکتا۔ یہ کافی آسان محسوس ہوتا ہے کہ پروگرام کو استعمال کی بنیاد پر صحیح عددی قدر کو اسٹرنگ میں تبدیل کر دینا چاہیے۔ حالاں کہ انٹرپریٹر خود یہ فیصلہ نہیں کرسکتا کہ تبدیلی کا عمل کب انجام دیا جائے کیوں کہ اس میں اطلاعات کے ضائع ہونے کا خدشہ ہے۔ پائتھن صریحی تبدیلی کے لیے ایک طریقہ کار مہیا کرتا ہے تاکہ کوئی بھی

نوٹ

مطلوبہ نتیجے کو واضح طور پر بیان کر سکے۔ پروگرام 5-8 صریحی ٹائپ کاسٹنگ کا استعمال کرتے ہوئے بہترین طریقے سے کام کرتا ہے۔

پروگرام 5-8 صریحی ٹائپ کاسٹنگ کو ظاہر کرنے کے لیے پروگرام

```
#Program 5-8
#Explicit type casting
priceIcecream = 25
priceBrownie = 45
totalPrice = priceIcecream + priceBrownie
print("The total in Rs." + str(totalPrice))
```

نتیجہ:

```
The total in Rs.70
```

اسی طرح فلوٹ کو اسٹرنگ میں تبدیل کرنے کے لیے ٹائپ کاسٹنگ کی ضرورت ہوتی ہے۔ پائتھن میں کوئی بھی حسب ضرورت اسٹرنگ کو صحیح عدد یا فلوٹ ویلیو میں تبدیل کر سکتا ہے۔

پروگرام 5-9 صریحی تبدیلی کو ظاہر کرنے کے لیے پروگرام

```
#Program 5-9
#Explicit type conversion
icecream = '25'
brownie = '45'
#String concatenation
price = icecream + brownie
print("Total Price Rs." + price)
#Explicit type conversion - string to integer
price = int(icecream)+int(brownie)
print("Total Price Rs." + str(price))
```

نتیجہ:

```
Total Price Rs.2545
Total Price Rs.70
```

5.12.2 مضمر تبدیلی (Implicit Conversion)

مضمر تبدیلی جسے غیر ارادی تبدیلی بھی کہا جاتا ہے اس وقت واقع ہوتی ہے جب ڈیٹا ٹائپ کی تبدیلی پائتھن کے ذریعے خودکار انداز میں کی جاتی ہے پروگرامر کے ذریعے نہیں۔

پروگرام 5-10 صحیح عدد (int) سے فلوٹ میں مضمر ڈیٹا ٹائپ تبدیلی کو ظاہر کرنے کے لیے پروگرام

```
#Program 5-10
#Implicit type conversion from int to float

num1 = 10              #num1 is an integer
num2 = 20.0            #num2 is a float
```

نوٹ

```
sum1 = num1 + num2  #sum1 is sum of a float
and an integer
print(sum1)
print(type(sum1))
```

نتیجہ:

```
30.0
<class 'float'>
```

مذکورہ بالا مثال میں متغیر num1 میں ذخیرہ شدہ صحیح عددی قدر متغیر num2 میں ذخیرہ شدہ فلوٹ قدر میں شامل ہوجاتی ہے اور نتیجہ انٹرپریٹر کو صریحی طور پر بتائے بغیر خود بخود متغیر sum1 میں ذخیرہ شدہ فلوٹ قدر میں تبدیل ہوجاتا ہے۔ یہ مضمر ڈیٹا تبدیلی کی ایک مثال ہے۔ آپ کو شاید اس بات پر حیرت ہوسکتی ہے کہ فلوٹ قدر کو صحیح عددی قدر میں کیوں نہیں تبدیل کیا گیا؟ ایسا ٹائپ کو فروغ دینے کی وجہ سے ہوتا ہے جو اطلاعات کے زیاں کے بغیر ڈیٹا کو وسیع جسامت کے ڈیٹا ٹائپ میں تبدیل کرکے عملوں کی انجام دہی (جب بھی ممکن ہو) میں مدد کرتا ہے۔

5.13 ڈی بگنگ (DEBUGGING)

پروگرام تحریر کرتے وقت پروگرامر سے غلطیاں سرزد ہوسکتی ہیں چنانچہ یہ ممکن ہے کہ پروگرام ایگزیکیوٹ نہ ہوپائے یا غلط نتیجہ برآمد ہوسکتا ہے۔ اس قسم کی اغلاط (جنھیں بگ یا سہو بھی کہا جاتا ہے) کی نشاندہی کرنے اور انھیں پروگرام سے دور کرنے کا عمل ڈی بگنگ کہلاتا ہے۔ پروگرام میں پیش آنے والی غلطیوں کو مندرجہ ذیل زمروں میں تقسیم کیا جاسکتا ہے:

i) نحوی اغلاط
ii) منطقی اغلاط
iii) رن ٹائم اغلاط

5.13.1 نحوی اغلاط (Syntax Errors)

پروگرامنگ کی دیگر زبانوں کی طرح پائتھن کے بھی اپنے ضابطے ہیں جو اس کی نحوی ترکیب (سنٹیکس) کا تعین کرتے ہیں۔ انٹرپریٹر بیانات کی ترجمانی صرف اسی صورت میں کرتا ہے جب یہ نحو کے اعتبار سے (پائتھن کے ضابطوں کے مطابق) درست ہو۔ اگر نحوی غلطی موجود ہے تو انٹرپریٹر غلطی کا پیغام ظاہر کرتا ہے اور پروگرام کے ایگزیکیوشن کو روک دیتا ہے۔ مثال کے طور پر قوسین جوڑیوں میں ہونے چاہئیں چنانچہ عبارت (10 + 12) نحوی اعتبار سے درست ہے لیکن (7 + 11 دائیں قوسین کی عدم موجودگی کی وجہ سے درست نہیں ہے۔ پروگرام کے ایگزیکیوشن سے پہلے اس قسم کی اغلاط کو دور کرنا ضروری ہے۔

نوٹ

5.13.2 منطقی اغلاط (Logical Errors)

منطقی غلطی پروگرام میں موجود ایک بگ (bug) ہے جو اس کے غلط طرز عمل کا سبب ہے۔ منطقی غلطی کے نتیجے میں غیر مطلوب آؤٹ پُٹ حاصل ہوتا ہے لیکن پروگرام کا ایگزیکیوشن یکا یک بند نہیں ہوتا۔ کیوں کہ پروگرام منطقی اغلاط موجود ہونے کے باوجود بھی کامیابی کے ساتھ انجام پاتا ہے چنانچہ بعض اوقات ان اغلاط کی نشاندہی کرنا مشکل ہوتا ہے۔ منطقی اغلاط کی موجودگی کا واحد ثبوت غلط آؤٹ پُٹ ہے۔ پروگرام کے آؤٹ پُٹ سے پیچھے کی طرف چلتے ہوئے کوئی بھی اس بات کی شناخت کرسکتا ہے کہ کیا غلط ہوا ہے۔

مثال کے طور پر، اگر ہم دو اعداد 10 اور 12 کا اوسط معلوم کرنا چاہتے ہیں اور اس کے لیے کوڈ 10 + 12/2 لکھتے ہیں تو یہ کامیابی کے ساتھ انجام پذیر ہوگا اور ہمیں آؤٹ پُٹ کے طور پر 16 حاصل ہوگا۔ لیکن 16، اعداد 10 اور 12 کا اوسط نہیں ہے۔ اوسط معلوم کرنے کے لیے درست کوڈ (10 + 12)/2 ہونا چاہیے تھا تاکہ 11 کی شکل میں صحیح آؤٹ پُٹ حاصل ہوسکے۔

منطقی اغلاط کو معنوی اغلاط (Semantic errors) بھی کہا جاتا ہے کیوں کہ یہ اس وقت واقع ہوتی ہیں جب پروگرام کے معنی (اس کا مفہوم) درست نہیں ہوتے۔

5.13.3 رن ٹائم اغلاط (Runtime Error)

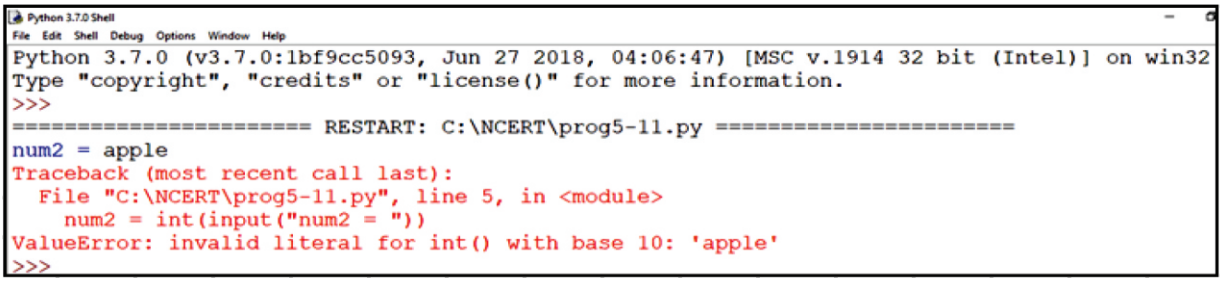
رن ٹائم اغلاط کے نتیجے میں پروگرام، ایگزیکیوشن کے دوران بے قاعدہ انداز میں اختتام پذیر ہوجاتا ہے۔ رن ٹائم غلطی اس وقت ظاہر ہوتی ہے جب بیان نحوی اعتبار سے تو درست ہوتا ہے لیکن انٹرپریٹر اسے ایگزیکیوٹ نہیں کرسکتا ہے۔ پروگرام کے چلنے یا اگزیکیوٹ ہونے کے بعد تک رن ٹائم اغلاط ظاہر نہیں ہوتی ہیں۔

مثال کے طور پر، ہمارے پاس پروگرام میں تقسیم کے عمل سے متعلق ایک بیان ہے۔ غلطی سے اگر داخل کیا گیا نسب نما 0 ہے تو یہ ایک رن ٹائم غلطی ظاہر کرے گا جیسے ''division by zero''

آیئے پروگرام 5-11 پر غور کریں جس میں دو قسم کی رن ٹائم اغلاط کو دکھایا گیا ہے۔ یہ اس وقت ظاہر ہوتی ہیں جب استعمال کنندہ کوئی غیر صحیح عددی قدر یا 0 داخل کردیتا ہے۔ جب استعمال کنندہ num2 کے لیے صحیح عددی قدر کو داخل کرتا ہے تو پروگرام صحیح آؤٹ پُٹ فراہم کرتا ہے۔

پروگرام 5-11 رن ٹائم اغلاط کو ظاہر کرنے والے پروگرام کی مثال

```
#Program 5-11
#Runtime Errors Example
num1 = 10.0
num2 = int(input("num2 = "))
#if user inputs a string or a zero, it leads
to runtime error
print(num1/num2)
```



شکل 5.11 : پروگرام 11-5 کا آؤٹ پُٹ

خلاصہ

- پائتھن ایک اوپن سورس، اعلیٰ سطحی انٹرپریٹر پر مبنی زبان ہے جس کا استعمال متعدد سائنسی اور غیر سائنسی کمپیوٹنگ مقاصد کے لیے کیا جاسکتا ہے۔
- تبصرے دراصل ایسے بیانات ہیں جو پروگرام میں ایگزیکیوٹ نہیں ہوتے ہیں۔
- شناخت کنندہ (Identifier) استعمال کنندہ کے ذریعے متعین کیا گیا نام ہے جو کسی پروگرام میں متغیر یا مستقلہ کو دیا جاتا ہے۔
- کمپیوٹر پروگرام میں اغلاط کی نشاندہی کرنے اور انھیں پروگرام سے دور کرنے کا عمل ڈی بگنگ کہلاتا ہے۔
- ایک ایسے متغیر کو استعمال کرنے کی کوشش کرنا جسے کوئی قدر تفویض نہیں کی گئی ہے، غلطی کے ظاہر ہونے کا سبب ہے۔
- پائتھن میں متعدد ڈیٹا ٹائپ ہیں جیسے صحیح عدد، بولین، فلوٹ، ملتف، اسٹرنگ، لسٹ، ٹپل، سیٹ، نن (None) اور ڈکشنری۔
- ڈیٹا کی تبدیلی یا تو صریحی یا مضمر ہوسکتی ہے۔
- آپریٹر ایسی ترکیب ہے جس کا استعمال آپرینڈ کی قدروں میں ردو بدل کے لیے کیا جاتا ہے۔ آپریٹر یونری یا بائنری ہوسکتے ہیں۔

نوٹ

- ایک عبارت قدروں، متغیروں اور آپریٹروں کا مجموعہ ہے۔
- پائتھن میں استعمال کنندہ سے ان پُٹ حاصل کرنے کے لیے input() فنکشن موجود ہوتا ہے۔
- ان پُٹ ڈیٹا کو معیاری آؤٹ پُٹ ڈیوائس پر ظاہر کرنے کے لیے پائتھن print() فنکشن کا استعمال کرتا ہے۔

مشق

1۔ مندرجہ ذیل میں سے کون سے شناخت کنندہ کے نام درست نہیں ہیں اور کیوں؟

i	Serial_no.	v	Total_Marks
ii	1st_Room	vi	total-Marks
iii	Hundred$	vii	_Percentage
iv	Total Marks	viii	True

2۔ مندرجہ ذیل میں ہر ایک سے متعلق اسائنمینٹ اسٹیٹمنٹ لکھیے۔

(a) متغیر length کو 10 اور متغیر breadth کو 20 تفویض کرنا۔

(b) length اور breadth متغیروں کی قدروں کا اوسط متغیر sum کو تفویض کرنا۔

(c) اسٹرنگ 'Paper'، 'Gel Pen' اور 'Eraser' پر مشتمل لسٹ متغیر stationery کو تفویض کرنا۔

(d) اسٹرنگ 'Mohandas'، 'Karamchand' اور 'Gandhi' متغیر first، middle اور last کو تفویض کرنا۔

(e) اسٹرنگ متغیرات first، middle اور last کی باہم مربوط قدر متغیر fullname کو تفویض کرنا۔ ناموں کے مختلف حصوں کے درمیان خالی جگہوں کی مناسب موجودگی کو یقینی بنائیں۔

3۔ پائتھن میں مندرجہ ذیل بیانات کے نظیری منطقی عبارتیں تحریر کیجیے اور عبارتوں (فرض کیجیے کہ num1، num2، num3، first، middle، last پہلے ہی سے بامعنی قدریں ہیں) کی تحسیب کیجیے:

(a) 20 اور 10- کا حاصل جمع 12 سے کم ہے۔

(b) num3 کی قدر 24 سے زیادہ نہیں ہے۔

(c) 6.75 صحیح اعداد num1 اور num2 کی قدروں کے درمیان میں ہے۔

(d) اسٹرنگ 'middle' اسٹرنگ 'first' سے بڑی اور اسٹرنگ 'last' سے چھوٹی ہے۔

(e) لسٹ Stationery خالی ہے۔

نوٹ

4۔ ہر ایک عبارت میں اس طرح قوسین لگائیے کہ اس کی تحسیب کا نتیجہ True ہو۔

a) `0 == 1 == 2`

b) `2 + 3 == 4 + 5 == 7`

c) `1 < -1 == 3 > 4`

5۔ مندرجہ ذیل کا آؤٹ پُٹ لکھیے:

a)
```
num1 = 4
num2 = num1 + 1
num1 = 2
print (num1, num2)
```

b)
```
num1, num2 = 2, 6
num1, num2 = num2, num1 + 2
print (num1, num2)
```

c)
```
num1, num2 = 2, 3
num3, num2 = num1, num3 + 1
print (num1, num2, num3)
```

6۔ مندرجہ ذیل ڈیٹا قدروں کو ظاہر کرنے کے لیے کون سے ڈیٹا ٹائپ کا استعمال کیا جائے گا اور کیوں؟

a) سال میں مہینوں کی تعداد

b) دہلی کا باشندہ ہے یا نہیں

c) موبائل نمبر

d) جیب خرچ

e) کرہ کا حجم

f) مربع کا احاطہ

g) طالب علم کا نام

h) طالب علم کا پتہ

7۔ مندرجہ ذیل کا آؤٹ پُٹ بتائیے اگر num1 = 4، num2 = 3، num3 = 2 ہے۔

a)
```
num1 += num2 + num3
print (num1)
```

b)
```
num1 = num1 ** (num2 + num3)
print (num1)
```

c)
```
num1 **= num2 + num3
```

d)
```
num1 = '5' + '5'
print(num1)
```

e)
```
print(4.00/(2.0+2.0))
```

f)
```
num1 = 2+9*((3*12)-8)/10
print(num1)
```

نوٹ

g)
```
num1 = 24 // 4 // 2
print(num1)
```
h)
```
num1 = float(10)
print (num1)
```
i)
```
num1 = int('3.14')
print (num1)
```
j)
```
print('Bye'  == 'BYE')
```
k)
```
print(10 != 9 and 20 >= 20)
```
l)
```
print(10 + 6 * 2 ** 2 != 9//4 -3 and 29
>= 29/9)
```
m)
```
print(5 % 10 + 10 < 50 and 29 <= 29)
```
n)
```
print((0 < 6) or (not(10 == 6) and
(10<0)))
```

8۔ مندرجہ ذیل کی درجہ بندی نحوی غلطی، منطقی غلطی یا رن ٹائم غلطی کے تحت کیجیے۔

a)
```
25 / 0
```
b)
```
num1 = 25; num2 = 0; num1/num2
```

9۔ 10 اکائی والا ایک ڈارٹ بورڈ اور وہ دیوار جس پر ڈارٹ بورڈ لٹکا ہوا ہے دونوں کو دو ابعادی مختص نظام کا استعمال کر کے ظاہر کیا گیا ہے جس میں بورڈ کا مرکز مختص (0,0) پر ہے۔ متغیر x اور y ڈاٹ بورڈ پر مارے جانے والے ڈارٹ کے x- مختص اور y- مختص کو اسٹور کرتے ہیں۔ متغیر x اور y کا استعمال کر کے ایک پائتھن پروگرام لکھیے جو ڈارٹ کے ڈارٹ بورڈ (اس کے اندر) پر مارے جانے کی صورت میں True کی تحسیب کرتا ہے اور اس کے بعد ان ڈارٹ مختصات کے لیے عبارت کی تحسیب کرتا ہے۔

a)
```
(0,0)
```
b)
```
(10,10)
```
c)
```
(6, 6)
```
d)
```
(7,8)
```

10۔ درجہ حرارت کی اکائی ڈگری سیلسیس کو ڈگری فاہرینائٹ میں تبدیل کرنے کے لیے ایک پائتھن پروگرام لکھیے۔ اگر پانی 100 ڈگری سیلسیس پر ابلتا ہے اور 0 ڈگری سیلسیس پر جم جاتا ہے تو فاہرینائٹ اسکیل پر پانی کا نقطۂ جوش اور نقطۂ انجماد معلوم کرنے کے لیے پروگرام کا استعمال کیجیے۔

(Hint: T(°F) = T(°C) × 9/5 + 32)

11۔ قابل ادائیگی رقم کی تحسیب کرنے کے لیے ایک پائتھن پروگرام لکھیے اگر رقم سادہ سود پر بطور قرض دی گئی ہے۔ اصل زر یا بطور قرض دی گئی رقم = P، سود کی شرح = R % سالانہ اور مدت = T سال۔

(SI) = (P x R x T)/ 100

قابل ادائیگی رقم = اصل زر + SI

P، R اور T کو پروگرام میں ان پُٹ کے طور پر دیا گیا ہے۔

12۔ تین افراد A، B اور C کے ذریعے کسی کام کو مکمل کرنے میں لگنے والے دنوں کی تعداد کی تحسیب کے

نوٹ

لیے ایک پروگرام لکھیے۔ A، B، C اکیلے اس کام کو بالترتیب x دن میں، y دن میں، z دن میں مکمل کرتے ہیں۔ اگر تینوں ایک ساتھ مل کر کام کرتے ہیں تو دنوں کی تعداد کی تحسیب کے لیے فارمولا xyz/ (xy + yz + xz) ہے جہاں x، y اور z کو پروگرام میں ان پُٹ کے طور پر دیا گیا ہے۔

13۔ دو صحیح اعداد کو داخل کرنے اور ان پر سبھی حسابی عملوں کو انجام دینے کے لیے ایک پروگرام لکھیے۔

14۔ تیسرے متغیر کا استعمال کر کے دو اعداد کو آپس میں بدلنے کے لیے ایک پروگرام لکھیے۔

15۔ تیسرے متغیر کا استعمال کیے بغیر دو اعداد کو آپس میں بدلنے کے لیے ایک پروگرام لکھیے۔

16۔ اسٹرنگ ''GOOD MORNING'' کو n مرتبہ دہرانے کے لیے ایک پروگرام لکھیے۔ یہاں n استعمال کنندہ کے ذریعے داخل کیا گیا صحیح عدد ہے۔

17۔ تین اعداد کا اوسط معلوم کرنے کے لیے ایک پروگرام لکھیے۔

18۔ نصف قطر r والے ایک کرہ کا حجم $\frac{4}{3}\pi r^3$ ہے۔ 7cm، 12cm، 16cm نصف قطر والے کروں کے حجم معلوم کرنے کے لیے ایک پائتھن پروگرام تحریر کیجیے۔

19۔ ایک پروگرام لکھیے جو استعمال کنندہ سے اس کا نام اور عمر داخل کرنے کے لیے کہتا ہے۔ ایک پیغام پرنٹ کیجیے جس میں استعمال کنندہ کو یہ بتایا گیا ہو کہ کون سے سال میں ان کی عمر 100 سال ہو جائے گی۔

20۔ فارمولا $E = mc^2$ یہ بتاتا ہے کہ کمیت (m) کو روشنی کی چال (c) کے مربع سے ضرب کر کے معادل توانائی (E) کی تحسیب کی جا سکتی ہے ($c = 3\times10^8$ m/s)۔ ایک پروگرام لکھیے جو استعمال کنندہ سے شے کی کمیت کو حاصل کرتا ہے اور اس کی کمیت کی تحسیب کرتا ہے۔

21۔ فرض کیجیے کہ ایک سیڑھی کو دیوار سے لگا کر رکھا گیا ہے۔ متغیر length اور angle سیڑھی کی لمبائی اور اس زاویہ کو اسٹور کرتے ہیں جو زاویہ یہ سیڑھی زمین کے ساتھ بناتی ہے کیوں کہ سیڑھی کو دیوار کے ساتھ جھکا کر لگایا گیا ہے۔ لمبائی اور زاویوں کی مندرجہ ذیل قدروں کے لیے دیوار کی جس اونچائی تک سیڑھی پہنچتی ہے اس کی تحسیب کے لیے ایک پائتھن پروگرام لکھیے۔

a) 16 فٹ اور 75 ڈگری

b) 20 فٹ اور 0 ڈگری

c) 24 فٹ اور 45 ڈگری

d) 24 فٹ اور 80 ڈگری

کیس اسٹڈی پر مبنی سوال (CASE STUDY-BASED QUESTION)

اسکول طلبا سے متعلق ڈیٹا کے رکھ رکھاؤ کے لیے ''اسٹوڈینٹ مینجمنٹ انفارمیشن سسٹم'' (SMIS) کا استعمال کرتا ہے۔ یہ نظام مندرجہ ذیل سہولیات فراہم کرتا ہے:

نوٹ

- طلبا کی ذاتی تفصیلات کو درج کرنا اور ان کا رکھ رکھاؤ
- مختلف امتحانات میں حاصل کردہ نمبروں کا رکھ رکھاؤ اور طلبا کے نتائج کی تحسیب
- طلبا کی حاضری پر نظر رکھنا
- طلبا سے متعلق دیگر ڈیٹا کو منظّم کرنا۔ آیئے اس عمل (پروسیس) کو خودکار انداز میں مرحلہ وار انجام دیتے ہیں۔

اپنے اسکول کے شناختی کارڈ کی مدد سے طلبا کی ذاتی تفصیلات کی شناخت کیجیے اور اپنے اسکول کے سبھی طلبا کے لیے ان تفصیلات کو حاصل کرنے کے لیے ایک پروگرام لکھیے اور انھیں مندرجہ ذیل فارمیٹ میں ظاہر کیجیے۔

Name of School

Student Name: PQR Roll No: 99
Class: XI Section: A
Address : Address Line 1
Address Line 2

City: ABC Pin Code: 999999
Parent's/ Guardian's Contact No: 9999999999

دستاویز سازی کے لیے مشورہ (DOCUMENTATION TIPS)

یہ حقیقت ہے کہ مناسب طریقے سے تحریر کیے گئے پروگرام کو پڑھنا اور سمجھنا آسان ہوتا ہے نیز مستقبل میں فروغ دینے کے لیے اس میں گنجائش موجود ہوتی ہے۔ چنانچہ کوڈنگ کے دوران دستاویز سازی پر خصوصی توجہ بہت اہم ہے۔ آیئے ہم نے اپنے کیس اسٹڈی پروگرام میں جو دستاویز سازی کی ہے اس کا جائزہ لیں اور یہ بھی معلوم کریں کی ہمارے دوست بھی دستاویز سازی پر اسی طرح توجہ دیتے ہیں یا نہیں۔

اچھی دستاویز سازی کو پرکھنے کے لیے نکات ذیل میں دیے گئے ہیں:

- پروگرام کا مقصد ابتدا میں واضح طور پر بیان کیا گیا ہے۔
- ہر ایک فنکشن کا مقصد ہر ایک فنکشن کی ابتدا میں واضح طور پر بیان کیا گیا ہے۔
- تبصروں کو مناسب جگہوں پر داخل کیا گیا ہے تاکہ پروگرام کو پڑھنے اور سمجھنے میں مزید آسانی ہو۔

(نوٹ: ضرورت سے زیادہ تبصروں سے کوئی فائدہ حاصل نہیں ہوتا ہے)

- متغیروں اور فنکشن کے نام موزوں اور بامعنی ہیں۔
- ایسے متغیروں کا استعمال نہیں کیا گیا ہے جن کے نام واحد حرف پر مشتمل ہیں۔
- پروگرام کا نام بامعنی نہیں ہے۔

(نوٹ: اپنے پروگرام کوڈ کو ظاہر کرنے کے لیے پروگرام کے نام کے طور پر اپنے نام کو استعمال کرنا

نوٹ

مناسب نہیں ہے مثال کے طور پر 'raman. py' یا 'namya.py'۔ بینکنگ سے متعلق پروگرام کے لیے پروگرام کا نام 'bankingProject.py' یا داخلہ سے متعلق پروگرام کے لیے پروگرام کا نام 'admProcess' زیادہ موزوں ہے۔

- پروگرام کوڈ کو صحیح طریقے سے انڈینٹ کیا گیا ہے۔
- پورے پروگرام میں تسمیہ کا ایک ہی طریقہ اختیار کیا گیا ہے۔

(نوٹ: پہلے عدد کو ظاہر کرنے والے متغیر کے لیے کچھ نام اس طرح ہیں: firstNum, first_num)

آیئے اس مشق کو ہم اپنے ساتھیوں کی کیس اسٹڈیز کے لیے بھی کریں اور انھیں بازرسی فراہم کریں۔ ساتھیوں کے حوالے سے ایک مناسب بازرسی پروجیکٹ کی دستاویز سازی کو بہتر بنانے میں معاون ثابت ہوتی ہے۔ اس سے ہمیں اپنی غلطیوں کو پہچاننے میں بھی مدد ملتی ہے۔