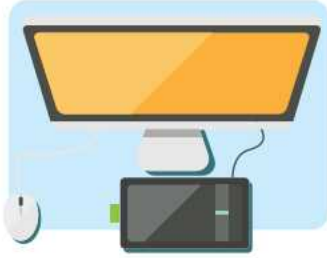


تعارف برائے پائنٹن پروگرامنگ



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



- بنیادی پروگرامنگ کے تصورات کو سمجھ سکیں اور پائنٹن ڈویلپمنٹ انوائرنمنٹ سیٹ اپ کر سکیں۔
- بنیادی پائنٹن سینٹیکس اور ساختہ سے لکھ اور سمجھ سکیں، جس میں ویریبلز، ڈیٹا ٹائپس، اور ان پٹ / آؤٹ پٹ آپریشنز شامل ہیں۔
- پائنٹن میں مختلف آپریٹرز اور اریٹھمٹک، بشمول حسابی (arithmetic)، موازنہ (comparison)، اسائنمنٹ (assignment)، منطقی (logical) آپریٹرز، اور آپریٹرز کی ترجیح کا استعمال کر سکیں۔

تعارف

پائنٹن پروگرامنگ کی دنیا میں خوش آمدید! پائنٹن اپنی سادگی اور پڑھنے میں آسانی کی وجہ سے ایک مقبول لینگویج ہے، جو اسے نئے یوزرز (beginners) اور پیشہ ور افراد دونوں کے لیے مثالی بناتی ہے۔ اس باب میں، ہم بنیادی باتوں سے شروعات کریں گے، جن میں ڈویلپمنٹ ماحول سیٹ اپ کرنا اور بنیادی پروگرامنگ تصورات سیکھنا شامل ہیں۔ اس باب کے اختتام تک، آپ پائنٹن کی ایک جامع سمجھ حاصل کر چکے ہوں گے، جو آپ کو اپنے پروگرام لکھنے، جانچنے اور غلطیاں درست (debug) کرنے کے قابل بنائے گی۔

3.1: پائنٹن پروگرامنگ کا تعارف

پائنٹن ایک وسیع پیمانے پر استعمال ہونے والی ہائی لیول پروگرامنگ لینگویج ہے جو اپنی سادگی اور پڑھنے میں آسانی کے لیے مشہور ہے۔ یہ ایک ہمہ گیر (versatile) لینگویج ہے اور مختلف شعبوں میں لاگو ہوتی ہے، جن میں ویب ڈویلپمنٹ، ڈیٹا کا تجزیہ (data analysis)، اور مصنوعی ذہانت وغیرہ شامل ہیں۔ چاہے آپ ایک سادہ سکرپٹ بنارہے ہوں یا کوئی پیچیدہ سافٹ ویئر تیار کر رہے ہوں، پائنٹن کی صارف دوست فطرت آپ کو تیزی اور مؤثر طریقے سے آغاز کرنے میں مدد کرتی ہے۔

بنیادی پروگرامنگ کے تصورات

کمپیوٹر پروگرامنگ ہدایات کا ایک ایسا سیٹ بنانے کا عمل ہے جو کمپیوٹر کو بتاتا ہے کہ کوئی کام کیسے کرنا ہے۔ یہ ہدایات ایک ایسی پروگرامنگ لینگویج میں لکھی جاتی ہیں جسے کمپیوٹر سمجھ سکتا ہے اور اس پر عمل (execute) کر سکتا ہے۔ کمپیوٹر پروگرامنگ کو ایسے سمجھیں جیسے آپ اپنے دوست کو اپنے گھر پہنچنے کی ہدایات دے رہے ہوں۔ آپ کو واضح اور درست ہونا چاہیے تاکہ آپ کا دوست گم نہ ہو۔

نہ ہو جائے۔ اسی طرح، جب ہم پروگرام لکھتے ہیں، تو ہم مخصوص کاموں کو مکمل کرنے کے لیے کمپیوٹر کو واضح اور درست ہدایات دیتے ہیں۔

پائتھن ڈویلپمنٹ انوائرنمنٹ کا سیٹ اپ

ڈویلپمنٹ انوائرنمنٹ سے مراد کمپیوٹر پائتھن کوڈ کو مؤثر طریقے سے لکھنے، اس پر عمل کرنے / عملدرآمد کرنے اور غلطیاں درست (debug) کرنے کے لیے تیار کرنے کا عمل ہے۔ اس میں ضروری سافٹ ویئر، ٹولز، اور لائبریریز کو انسٹال اور ترتیب دینا شامل ہے جو ڈویلپمنٹ کے عمل کو زیادہ آسان اور مؤثر بناتے ہیں۔

نوٹ:

جب پائتھن انسٹال کریں، تو باکس کو ضرور چیک کریں جس پر لکھا ہو "Add Python to PATH" (پائتھن کو PATH میں شامل کریں)۔ یہ کمانڈ لائن (command line) سے پائتھن پر عمل درآمد آسان بنا دیتا ہے۔ ہم پائتھن پروگرام لکھنے اور اس پر عملدرآمد کرنے کے لیے آن لائن سروسز بھی استعمال کر سکتے ہیں۔



شکل 3.1: پائتھن کا ویب سائٹ

پائتھن انسٹالیشن کے مراحل

پائتھن کی انسٹالیشن (installation) کو مکمل کرنے کے لیے، درج ذیل اقدامات پر عمل کریں:

پائتھن ڈاؤن لوڈ کریں (Download Python)

ویب سائٹ وزٹ کریں:

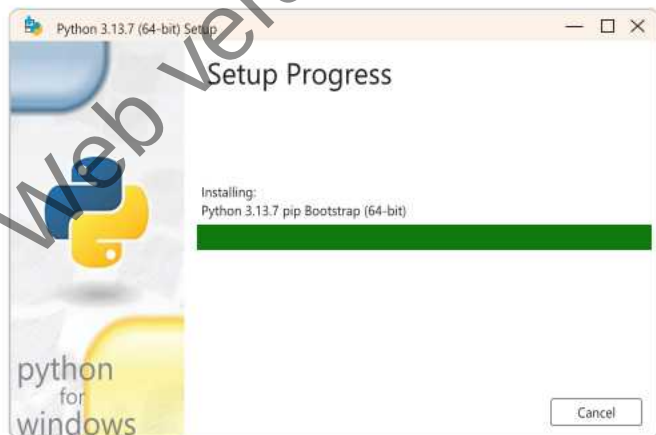
<https://www.python.org/>

ڈاؤن لوڈز سیکشن:

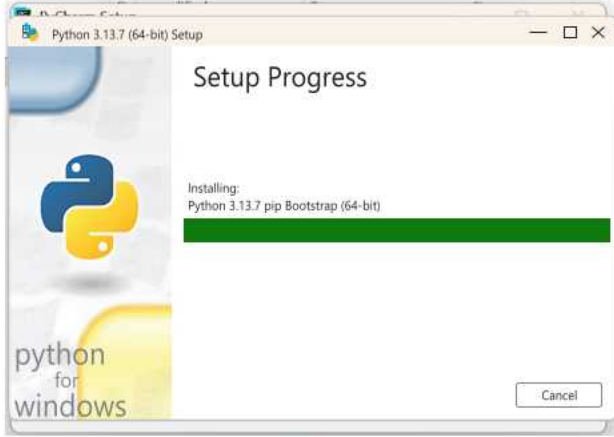
"Downloads" (ڈاؤن لوڈز) کے سیکشن پر جائیں۔

تازہ ترین ورژن:

اپنے آپریٹنگ سسٹم (Windows، macOS، یا Linux) کے لیے پائتھن کے تازہ ترین ورژن (مثلاً، Python 3.13.x) پر کلک کریں۔



شکل 3.2: سیٹ اپ پراسس



شکل 3.3: انسٹالیشن پراسس

ضرورت ہو تو انسٹالیشن کو اپنی مرضی کے مطابق (customize) بنائیں (مثلاً، انسٹالیشن کی ڈائریکٹری کا انتخاب کریں) انسٹالیشن مکمل ہونے کا انتظار کریں، پھر "Close" پر کلک کریں۔

-3 IDE سیٹ اپ کرنا

آپ PyCharm، VS Code، یا IDLE (جو پائتھن کے ساتھ شامل ہوتا ہے) جیسے مشہور IDE میں سے کوئی بھی ڈاؤن لوڈ اور انسٹال کریں۔

PyCharm پائتھن ڈویلپمنٹ کے لیے ایک طاقتور IDE ہے، جو ایک مہلت کمیونٹی ایڈیشن یا ایک پیڈ پرو فینشل ایڈیشن میں دستیاب ہے۔

PyCharm کو ڈاؤن لوڈ کرنے کے لیے <https://www.jetbrains.com/pycharm>

پر جائیں۔ آپریٹنگ سسٹم (Windows، macOS، یا Linux) کے لیے مناسب ورژن ڈاؤن لوڈ کریں۔



شکل 3.4: پائتھن IDE

انسٹالر خود بخود ڈاؤن لوڈ ہو جائے گا۔

-2 پائتھن انسٹال کریں (Install)

(Python)

ڈاؤن لوڈ کی گئی .exe فائل کو چلائیں۔

(انسٹالیشن کے دوران "Add Python

to PATH") (پائتھن کو PATH میں

شامل کریں) کے باکس پر ضرور چیک

لگائیں۔

"Install Now" کو منتخب کریں یا اگر

-4 IDE کو پائتھن کے ساتھ کنفیگر کرنا

PyCharm کو کھولیں اور ایک نیا

پراجیکٹ شروع کریں۔

"Settings" یا "Configure"

(سینکڑوں مینیو میں، "Project

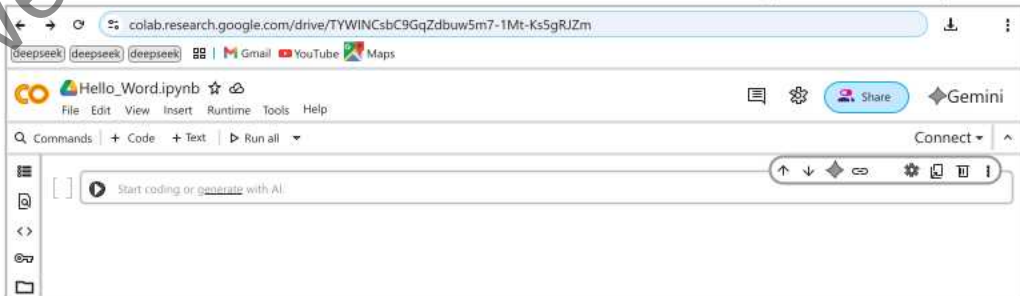
Interpreter" (پراجیکٹ

انٹرپرائز) پر جائیں۔

- گیزر کے آئیکن (Gear Icon) پر کلک کریں، "Add Interpreter" (انٹرپرائیٹر شامل کریں) کو منتخب کریں، اور اپنے پائتھن ایگزیکوٹبل (executable) کا پاتھ منتخب کریں (مثلاً، Windows پر `C:\Python312\python.exe` یا Linux/macOS پر `/usr/bin/python3`)۔
- اگر یہ خود کار طریقے سے نہیں ملتا، تو اپنے انسٹالیشن فولڈر کو براؤز میں تلاش کریں۔
- سینٹر کو اپلائی کریں اور PyCharm کو انٹرپرائیٹر کو انڈیکس کرنے دیں۔

5- کنفیگریشن کی ٹیسٹنگ (Testing the Configuration)

- اپنے IDE میں ایک نئی فائل (مثال کے طور پر `test.py`) بنائیں۔
- ایک سادہ اسکرپٹ لکھیں: `print("Hello, Python!")`
- اس اسکرپٹ کو چلائیں (PyCharm/IDLE میں "Run" بٹن استعمال کریں یا VS Code میں ٹاسک سیٹ اپ کرنے کے بعد F5 دبائیں)۔
- اگر آؤٹ پٹ مثال کے طور پر "Hello, Python!" ظاہر ہوتا ہے، تو آپ کا IDE درست طریقے سے کنفیگر ہو گیا ہے۔
- گوگل کولیب (Google Colab)
- پائتھن استعمال کرنے کا ایک اور بہترین آپشن Google Colab ہے۔
- یہ ایک مفت کلاؤڈ بیسڈ پلیٹ فارم ہے جو Jupyter Notebook کا ماحول فراہم کرتا ہے جس میں پائتھن اور مشہور لائبریریاں جیسے NumPy، Pandas، اور Matplotlib چلنے سے انسٹال ہوتی ہیں۔
- آغاز کرنے کے لیے، <https://colab.research.google.com> پر جائیں، گوگل اکاؤنٹ سے سائن ان کریں، اور ایک نئی نوٹ بک بنائیں۔
- اس کے لیے لوکل انسٹالیشن کی ضرورت نہیں ہے، جو اسے ابتدائی یا محدود وسائل والے کمپیوٹریں پر کام کرنے والوں کے لیے مثالی بناتا ہے۔
- آپ سیلز (cells) میں پائتھن کوڈ لکھ سکتے ہیں اور چلا سکتے ہیں۔



شکل 3.5: گوگل کولیب IDE

3.2 بنیادی پائتھن سینٹیکس اور سٹرکچر

درج ذیل پائتھن پروگرام لیٹگوئج کی سادگی اور پڑھنے میں آسانی کو ظاہر کرتا ہے۔

```

Hello_Word.ipynb
File Edit View Insert Runtime Tools Help
Commands | + Code | + Text | ▶ Run all
[1] print('Hello, Students!')
Hello, Students!

```

اس مثال میں، `print` فنکشن کو "طعام" کے اندر موجود پیغام کو آؤٹ پٹ کرنے کے لیے استعمال کیا گیا ہے۔ یہ پائتھن کے سادہ سینٹیکس کی وضاحت کرتا ہے، جہاں `print` جیسے فنکشنز کو ٹیکسٹ ظاہر کرنے جیسے کام کرنے کے لیے استعمال کیا جاتا ہے۔

پائتھن میں کمنٹس

- کمنٹس وہ لائنیں ہوتی ہیں جو پائتھن انٹرپرائیٹر کے ذریعے چلائی (execute) نہیں جاتیں۔
- ان کا مقصد کوڈ کے بارے میں پروگرام کو وضاحت یا نوٹس فراہم کرنا ہوتا ہے۔
- سنگل لائن کمنٹ لکھنے کے لیے `#` کا نشان استعمال کیا جاتا ہے۔

```

# This is a single - line comment
print ( "K2 is the second-highest mountain in the world " )
'''
This is a multi-line comment.
It can span multiple lines.
'''

print("Edhi Foundation operates the world's largest volunteer
ambulance network.")
Output:
K2 is the second-highest mountain in the world
Edhi Foundation operates the world's largest volunteer ambulance
network.

```



ہم "....." کو ملٹی لائن کمنٹ کے طور پر استعمال کر سکتے ہیں لیکن درحقیقت یہ کمنٹس نہیں ہیں۔ یہ ملٹی لائن سٹرنگ لٹریچر ہیں جن کو رن ٹائم پر نظر انداز کر دیا جاتا ہے۔

پائتھن میں ویری ایبلز (Variables in Python)

پروگرامنگ میں ویری ایبل ایک عارضی کنٹینر کی طرح کام کرتا ہے جو کمپیوٹر کی میموری میں ڈیٹا محفوظ کرتا ہے۔ ویری ایبل کے ذریعے ہم کوئی ویلیو محفوظ کر سکتے ہیں، جسے بعد میں پروگرام میں دوبارہ استعمال یا تبدیل کیا جاسکتا ہے۔ نیچے دی گئی مثال میں age ایک ویری ایبل ہے جو نو میرک ڈیٹا محفوظ کرنے کے لیے استعمال ہوا ہے۔

Iqbal lived for 60 years ویری ایبل کی ویلیو پروگرام کے چلنے کے دوران بدل سکتی ہے، اسی لیے اسے "Variable" کہا جاتا ہے۔

```
age = 71
print ("Azam lived for", age, "years")
age = 60
print ("Iqbal lived for", age, "years")
# Output
# Azam lived for 71 years
# Iqbal lived for 60 years
```

یہ مثال واضح کرتی ہے کہ کس طرح age ویری ایبل کو مختلف ویلیوز تفویض کی جاتی ہیں تاکہ اعظم (Azam) اور اقبال (Iqbal) کی عمروں کی نمائندگی کی جاسکے، جنہیں بعد میں آؤٹ پٹ کے حصے کے طور پر پرنٹ کیا جاتا ہے:

پائتھن میں ویری ایبل کے نام رکھنے کے اصول

ویری ایبل کے نام واضح، بامعنی اور درست (valid) ہونے چاہئیں تاکہ آپ کا کوڈ آسانی سے سمجھا جاسکے۔ پائتھن میں ویری ایبل کے نام رکھتے وقت درج ذیل قواعد پر سختی سے عمل کرنا ضروری ہے:

ویری ایبل کے نام رکھنے کے قوانین:

■ ابتدائی حرف: نام کا آغاز کسی حرف (A-Z، a-z) یا ایک انڈر سکور (_) سے ہونا چاہیے۔ یہ عددی ہندسے (digits) سے شروع نہیں ہو سکتا۔

■ دیگر حروف: پہلے حرف کے بعد، نام میں حروف، ہندسے (0-9)، یا انڈر سکور (_) شامل ہو سکتے ہیں۔

■ ویری ایبل کے نام کیس سینسیٹو ہوتے ہیں، یعنی age اور Age کو دو مختلف ویری ایبلز سمجھا جائے گا۔

■ پائتھن کے ریزروڈ، جیسے کہ if، while، for وغیرہ، کو ویری ایبل کے نام کے طور پر استعمال نہیں کیا جاسکتا۔



کیا آپ جانتے ہیں؟

پائتھن میں مخصوص الفاظ (Reserved Keywords) وہ الفاظ ہوتے ہیں جن کا پائتھن کے اندر مخصوص اور پہلے سے طے شدہ معنی ہوتا ہے۔ انہیں کسی بھی صورت میں ویری ایبل، فنکشن، یا کسی دوسرے شناختی (Identifier) نام کے طور پر استعمال نہیں کیا جاسکتا۔

پانٹھن کے مخصوص الفاظ کی فہرست
درج ذیل پانٹھن کے کچھ مخصوص الفاظ ہیں:

print, if, else, elif, for, while, break, continue, and, or, not, in, is, True, False, None

ہم انہیں ویری ایبل کے نام کے طور پر استعمال نہیں کر سکتے۔

ویری ایبل کی اقسام

پانٹھن میں، آپ مختلف قسم کا ڈیٹا محفوظ کرنے کے لیے مختلف اقسام کے ویری ایبلز بنا سکتے ہیں۔ پانٹھن خود بخود دے کر لیتا ہے کہ ویری ایبل میں رکھی قیمتوں کی بنیاد پر اس کی قسم کیا ہوگی۔
درج ذیل ویری ایبل کی کچھ عام اقسام ہیں:

مثال	وضاحت	قسم
age = 17	یہ مکمل اعداد (whole numbers) کو محفوظ کرتا ہے۔	انٹیجر (Integer) / int
price = 19.99	یہ اعشاریہ والے اعداد (decimal numbers) کو محفوظ کرتا ہے۔	فلوئٹنگ پوائنٹ / (float)
name = "Ali"	یہ متن (text) کو محفوظ کرتا ہے، جسے اقتباس کے نشانات (quotes) میں لکھا جاتا ہے۔	سٹرنگ (String) / str
is_student = True	یہ صرف دو قدروں (درست / True یا غلط / False) کو محفوظ کرتا ہے۔	بُولین / (bool)

پانٹھن میں ان پٹ اور آؤٹ پٹ آپریشنز

ان پٹ اور آؤٹ پٹ آپریشنز آپ کو صارف سے بات چیت کرنے کی اجازت دیتے ہیں۔ آپ صارف کو ان پٹ دینے کا کہہ سکتے ہیں (ان پٹ) اور صارف کو انفارمیشن دیکھا سکتے ہیں (آؤٹ پٹ)۔

ان پٹ فنکشن

ہم صارف سے ان پٹ لینے کے لیے `input()` فنکشن کا استعمال کرتے ہیں۔
`Input()` فنکشن سے لی گئی ان پٹ ہمیشہ سٹرنگ (String) ڈیٹا ٹائپ میں سے ڈیٹا لے گی
`input()` فنکشن سکرین پر ایک پیغام ظاہر کرتا ہے اور صارف کے کچھ ٹائپ کرنے اور Enter Key دبانے کا انتظار کرتا ہے۔
 صارف کی طرف سے داخل کیا گیا ٹیکسٹ پھر ایک متغیر میں محفوظ ہو جاتا ہے۔
 مثال کے طور پر
 کوڈ کی پہلی لائن صارف سے ان کا نام داخل کرنے کے لیے کہتی ہے اور اسے `name` نامی وییری ایبل میں محفوظ کرتی ہے۔

```
name = input("Enter your name: ")
print("Hello, " + name + "!")
# Output
# Enter your name: Maryam Ashraf
# Hello, Maryam Ashraf!
```

آؤٹ پٹ آپریشنز

آؤٹ پٹ آپریشنز
 اسکرین پر معلومات ظاہر کرنے کے لیے `print()` فنکشن استعمال کیا جاتا ہے۔ `print()` فنکشن ایک یا ایک سے زیادہ آرگیومنٹ لیتا ہے اور انہیں دکھاتا ہے۔
 مثال: کوڈ کی دوسری لائن ایک پیغام دکھاتی ہے جس میں صارف کا نام شامل ہوتا ہے۔



کیا آپ جانتے ہیں؟

1. `Print()` فنکشن کے ذریعے ایک ہی وقت میں زیادہ ویلیو کی آؤٹ پٹ لینے کے لیے ان کے درمیان '!' استعمال کرتے ہیں۔
2. پائتھن میں انڈینٹیشن آپ کے کوڈ کا سکوپ (scope) یا وییری ایبلز (structure) کی ساخت بناتی ہے۔ غلط انڈینٹیشن آپ کے کوڈ میں ایرر کا سبب بن سکتی ہے۔

انٹیجر اور فلوٹ ان پٹ کو ہینڈل کرنا:

نو میرک ان پٹ کو ہینڈل کرنے کے لیے، عام طور پر `int()` فنکشن یا `float()` فنکشن استعمال کیا جاتا ہے۔ یہ فنکشنز ان پٹ سٹرنگز کو بالترتیب انٹیجرز (کمل اعداد) یا فلوٹنگ پوائنٹ نمبرز (اعشاریہ والے اعداد) میں تبدیل کرتے ہیں۔

```
# Example : Handling integer input
user_age = int(input("Enter your age: "))
print("Your age is:", user_age)
# Output
# Enter your age: 16
# Your age is : 16
```

مثال: انٹیجر ان پٹس

```
# Example: Handling float input
user_height = float(input("Enter your height in meters:
"))
print("Your height is", user_height, "meters")
# Output
# Enter your height in meters: 1.5
# Your height is 1.5 meters
```

3.3 آپریٹرز اور آپریٹرز

آپریٹرز ایسی علامتیں ہیں جو متغیرات اور مستقل قدروں پر آپریٹرز انجام دیتی ہیں۔ ایک ایکسپریشن متغیرات، آپریٹرز، اور ویلیوز کا ایک مجموعہ ہے جو ایک نتیجہ پیدا کرتا ہے۔ آپریٹرز مختلف قسم کے آپریٹرز کو دیکھتے ہیں۔

حسابی آپریٹرز (Arithmetic Operators)

حسابی آپریٹرز بنیادی ریاضیاتی عوامل انجام دینے کے لیے استعمال ہوتے ہیں جیسے کہ جمع (addition)، نفی (subtraction)، ضرب (multiplication)، تقسیم (division)، ماڈیولس (modulus)، ایکسپونینشل (exponential)، اور فلور تقسیم (floor division) جیسا کہ مندرجہ ذیل کوڈ میں دکھایا گیا ہے۔

```
# Define variables
a= 10
b= 3
# Perform all arithmetic operations on these numeric
variables and print results
print(a, "+", b, "=", a + b) # Output: 10+3=13
print(a, "*", b, "=", a * b) # Output: 10 * 3 = 30
print(a, "/", b, "=", a / b)
# Output: 10 / 3 = 3.333333333333333
print(a, "//", b, "=", a // b) #floor division
# Output: 10 // 3=3
print(a, "%", b, "=", a % b)
# Output: 10 % 3 = 1
print(a, "**", b, "=", a ** b)
# ** represent power operator:
# Output: 10**3= 1000
```

تقابلی آپریٹرز (Comparison Operators)

تقابلی آپریٹرز دو قیمتوں یا ایکسپریشنز کا موازنہ کرنے کے لیے استعمال ہوتے ہیں۔ یہ ان کے درمیان ریشینل لاجک کا تعین کرتے ہیں، جیسے کہ مساوات (equality)، عدم مساوات (inequality)، زیادہ ہونا (greater than)، کم ہونا (less than) یہ آپریٹرز

موازنہ کے نتیجے کی بنیاد پر ایک بولین ویلیو (یعنی True یا False) دیتے ہیں۔ یہاں ایک پانچھن پروگرام ہے جو تمام تقابلی آپریٹرز کے استعمال کو ظاہر کرتا ہے۔

```
# Define variables
X = 10
Y = 5
# Greater than
print (x, ">", y, "=", x > y) # Output: 10 > 5 = True
# Less than
print (x, "<", y, "=", x < y) # Output: 10 < 5 = False
```

```
#Equal to
print (x, "==", y, "=", x == y) # Output: 10 == 5 = False
# Not equal to
print (x, "!=", y, "=", x != y) # Output: 10 != 5 = True
# Greater than or equal to
print (x, ">=", y, "=", x >= y) # Output: 10 >= 5 = True
# Less than or equal to
print (x, "<=", y, "=", x <= y) # Output: 10 <= 5 = False
```

مذکورہ بالا کوڈ پانچھن کے تقابلی آپریٹرز کو ظاہر کرتا ہے، جس میں دو متغیرات x اور y کا موازنہ مختلف آپریٹرز جیسے <, <=, >, >=, !=, == کا استعمال کرتے ہوئے کیا گیا ہے۔

اسائنمنٹ آپریٹرز

اسائنمنٹ آپریٹرز متغیرات (variables) کو ویلیوز (values) تفویض کرنے کے لیے استعمال ہوتے ہیں۔ سب سے عام اسائنمنٹ آپریٹر مساوات کی علامت (=) ہے، جو دائیں طرف کی قدر کو بائیں طرف کے متغیر کو تفویض (assign) کرتی ہے۔ کمپاؤنڈ اسائنمنٹ آپریٹرز بھی ہیں جیسے +=, -=, *=, /= جو حسابی عمل کو اسائنمنٹ کے ساتھ جوڑتے ہیں۔

```
# Define initial values
a = 10
b = 5

# Assignment
x = a; print("a =", x)# Output: a = 10
# Addition assignment
a += b; print("a after addition =", a)# Output: a after addition = 15
# Subtraction assignment
a -= b; print("a after subtraction =", a)# Output: a after subtraction = 10
# Multiplication assignment
a *= b; print("a after multiplication =", a)# Output: a after multiplication = 50
# Division assignment
a /= b; print("a after division =", a)# Output: a after division = 10.0
# Floor division assignment
a //= b; print("a after floor division =", a)# Output: a after floor division = 2
# Modulus assignment
a %= b; print("a after modulus =", a)# Output: a after modulus = 2.0
# Exponentiation assignment
a **= b; print("a after exponentiation =", a)# Output: a after exponentiation = 32.0
```

مذکورہ بالا کوڈ پائتھن میں کمپاؤنڈ اسائنمنٹ آپریٹرز کے استعمال کی وضاحت کرتا ہے۔ یہ ایک متغیر "a" کو ایک ویلیو تفویض کرنے سے شروع ہوتا ہے۔ اس کے بعد یہ مختلف حسابی آپریشنز کو ظاہر کرتا ہے جن میں جمع، نفی، ضرب، تقسیم، فلور تقسیم، ماڈیولس، اور ایکسپونینشل شامل ہیں۔ ہر آپریشن "a" کی ویلیو کو اپ ڈیٹ کرتا ہے، اور نتائج اپ ڈیٹ شدہ ویلیو کے ساتھ پرنٹ کیے جاتے ہیں۔ کوڈ کمنٹس ہر آپریشن کے آؤٹ پٹ کو دکھاتے ہیں۔

منطقی آپریٹرز (Logical Operators)

منطقی آپریٹرز ایک پروگرام میں متعدد شرائط یا ایکسپریشنز کو جوڑنے کے لیے استعمال ہوتے ہیں۔ سب سے عام منطقی آپریٹرز "not" اور "and" ہیں۔ یہ منطقی عمل انجام دینے کے لیے استعمال ہوتے ہیں اور شامل ایکسپریشنز کے جائزہ (evaluation) کی بنیاد پر بولیئن قیمتیں واپس کرتے ہیں۔

```
# Define variables
x = True
y = False
# Logical AND
logical_and = x and y
print(x, "and ", y, "=", logical_and) # Output: True and False = False
# Logical OR
logical_or = x or y
print(x, "or ", y, "=", logical_or) # Output: True or False = True
# Logical NOT
```

```
logical_not_x = not x
print("not", x, " = " , logical_not_x) # Output not True= False
logical_not_y = not y
print("not", y, " = " , logical_not_y)
# Output: not False= True
```

مذکورہ بالا کو پائنتھن میں منطقی آپریٹرز کے استعمال کو ظاہر کرتا ہے۔ یہ دو بولین متغیرات x اور y کی تعریف کرتا ہے اور ان پر "not" اور "and" جیسے منطقی عمل انجام دیتا ہے۔ ان منطقی عمل کے نتائج پرنٹ کیے جاتے ہیں، جو یہ دکھاتے ہیں کہ آیا ایکسپریشنز True ہیں یا False۔ کوڈ میں ہر منطقی عمل کے نتیجے کے ساتھ کمنٹس بھی شامل ہیں۔

پائنتھن میں آپریٹرز کی ترجیح

آپریٹرز کی ترجیح (Operator Precedence) اس ترتیب کا تعین کرتی ہے جس میں ایک ایکسپریشن میں آپریٹرز انجام پاتے ہیں۔ پائنتھن کے ساتھ ساتھ ریاضی میں بھی، کچھ آپریٹرز زیادہ ترجیح رکھتے ہیں اور دوسروں سے پہلے ان کا جائزہ لیا جاتا ہے۔ اس کو سمجھنا یہ یقینی بنانے میں مدد کرتا ہے کہ آپ کے حسابی عمل درست طریقے سے کیے گئے ہیں۔
قوسین '()' سب سے زیادہ فوقیت۔ قوسین کے اندر کے عمل سب سے پہلے کیے جاتے ہیں۔ مثال کے طور پر، $4 * (2 + 3)$ کا نتیجہ 20 آتا ہے۔

- قوت نما (Exponentiation): اس کے بعد پاور آپریٹرز انجام دیے جاتے ہیں۔ مثال کے طور پر، $2 + 2 * 2 * 3 = 18$ ۔
- ضرب (*)، تقسیم (/)، اور ماڈیولس (%): یہ عمل اگلے نمبر پر آتے ہیں۔
- جمع (+) اور نفی (-): ان کی فوقیت ضرب اور تقسیم کے مقابلے میں کم ہوتی ہے۔
- مثال کے طور پر: $4 * 3 + 10 / 2 - 11 \% 3$
- پہلے ضرب، تقسیم، اور ماڈیولس کے عمل کریں:
- $4 * 3$ کا نتیجہ 12 آتا ہے۔
- $10 / 2$ کا نتیجہ 5.0 آتا ہے۔
- $11 \% 3$ کا نتیجہ 2 آتا ہے۔
- پھر جمع اور نفی کے عمل انجام دیں: $12 + 5.0 - 2 = 15.0$
- مثال: آپریٹرز کی ترجیح (Operator precedence) کو ظاہر کرنے کے لیے درج ذیل کوڈ پر غور کریں:

```

print("=== PYTHON OPERATOR PRECEDENCE DEMONSTRATION
===\n")
# 1. Arithmetic Operators
print("1. ARITHMETIC OPERATORS:")
print(f"3 + 2 * 5 = {3 + 2 * 5}") # Multiplication
before addition
print(f"(3 + 2) * 5 = {(3 + 2) * 5}") # Parentheses change
order
print(f"2 ** 3 * 4 = {2 ** 3 * 4}") # Exponentiation
before multiplication
print(f"15 / 3 * 2 = {15 / 3 * 2}") # Division and
multiplication (left to right)
print(f"15 // 4 + 2 = {15 // 4 + 2}") # Floor division
before addition
print(f"11 // 4 + 2 = {11 // 4 + 2}") # Floor division
before addition
print(f"17 % 5 * 2 = {17 % 5 * 2}") # Modulo before
multiplication
# Output
# === PYTHON OPERATOR PRECEDENCE DEMONSTRATION ===

```

```

# 1. ARITHMETIC OPERATORS:

```

```

# 3 + 2 * 5 = 13
# (3 + 2) * 5 = 25
# 2 ** 3 * 4 = 32
# 15 / 3 * 2 = 10.0
# 15 // 4 + 2 = 5
# 11 // 4 + 2 = 4
# 17 % 5 * 2 = 4

```



پانچھن میں، سٹرنگ سے پہلے "f" کا استعمال سٹرنگ انٹرپولیشن (string interpolation) سٹرنگ کے اندر متغیرات یا یکسپریشنز کو شامل کرنے کے لیے کیا جاتا ہے۔

خلاصہ (Summary)

پائتھن ایک اعلیٰ سطحی (ہائی لیول) (high-level)، کثیر المقاصد (versatile) پروگرامنگ لینگویج ہے۔ یہ اپنی سادگی اور پڑھنے میں آسانی کے لیے جانی جاتی ہے۔
پائتھن ویب ڈویلپمنٹ، ڈیٹا اینالیسٹ اور مصنوعی ذہانت (AI) وغیرہ میں استعمال ہوتی ہے۔
پروگرامنگ میں کمپیوٹرز کو درست ہدایات دینا شامل ہے۔ پروگرام کمپیوٹرز کو بتاتے ہیں کہ مخصوص کام کیسے انجام دیے جائیں۔
پائتھن کو python.org سے ڈاؤن لوڈ کریں (تازہ ترین ورژن)۔
انسٹالیشن کے مراحل: ڈاؤن لوڈ کریں انسٹالر چلائیں "Add Python to PATH" (کوچک (ٹک) کریں۔

IDE آپشنز: IDLE، VS Code، PyCharm۔
آن لائن آپشن: Google Colab (کلاؤڈ پر مبنی، انسٹالیشن کی ضرورت نہیں)۔
حسابی آپریٹرز (Arithmetic): +, -, *, /, %, **, //,
تقابلی آپریٹرز (Comparison): >, <, ==, !=, >=, <=,
اسائنمنٹ آپریٹرز (Assignment): =, +=, -=, *=, /=, %=
منطقی آپریٹرز (Logical): not, or, and
کمنٹس آپریٹرز (Comments) کوڈ کو پڑھنے میں آسان بنانے کے لیے استعمال ہوتے ہیں: #



کثیر انتخابی سوالات

1 درج ذیل میں سے کون سا مرحلہ بنیادی پروگرامنگ کے عمل کا حصہ نہیں ہے؟

الف کوڈ لکھنا

ب مرتب کرنا / تشریح کرنا

ج عمل درآمد (Execute) کرنا

د غلطیوں کو نظر انداز کرنا

2 کمانڈ لائن سے آسانی سے چلانے کے لیے پائتھن انسٹال کرتے وقت آپ کو کیا کرنا چاہیے؟

الف Add Python to PATH کو آن چیک کریں

ب ایک مختلف IDE کا انتخاب کریں

ج Add Python to PATH کو چیک کریں

د صرف IDE انسٹال کریں

iii) print(10/3) کا نتیجہ کیا ہوگا؟

3 ب

3.333 الف

4 د

ن

iv) $x \times x = 123$ میں x کا ٹائپ کیا ہے؟

فلوٹ ب

انٹیجر الف

بولین د

سٹرنگ ن

v) پائتھن میں کس آپریٹر کی سب سے زیادہ ترجیح (highest precedence) ہے؟

ضرب (*) ب

جمع (+) الف

قوسین () د

قوت نما (**) ن

vi) $2 < 1$ and $3 > 5$ کا نتیجہ کیا ہے؟

False ب

True الف

None د

Error ن

vii) $+=$ آپریٹر کیا کرتا ہے؟

قدروں کا موازنہ کرتا ہے ب

دو نمبروں کو جمع کرتا ہے الف

قدروں کو ضرب دیتا ہے د

جمع کرتا ہے اور تفویض (assign) کرتا ہے ن

viii) کون سا فنکشن سٹرنگ ان پٹ کو انٹیجر میں تبدیل کرتا ہے؟

int() ب

str() الف

input() د

Float() ن

viii) درج ذیل کوڈ کا نتیجہ کیا ہے؟ `age = 25; print("Age : ", age)`

25 ب

Age: 25 الف

Age د

Age ن

1 پانٹھن میں کمنٹس (comments) کے لیے کون سی علامت استعمال ہوتی ہے؟

/ *

//

-

#

مختصر سوالات

- 1 پروگرامنگ میں کمنٹس کیوں اہم ہیں؟
- 2 پانٹھن میں متغیرات کی تین بنیادی اقسام پر بحث کریں۔
- 3 پانٹھن میں integer اور float ڈیٹا اقسام کے درمیان فرق بیان کریں۔
- 4 منطقی آپریٹرز کیا ہیں اور تینوں کے نام بتائیں؟
- 5 آپ پانٹھن میں صارف سے ان پٹ کیسے حاصل کرتے ہیں؟
- 6 پانٹھن میں print() سٹیٹمنٹ کا مقصد کیا ہے؟
- 7 آپریٹر کی ترجیح کیا ہے؟ ایک مثال دیں۔
- 8 پانٹھن میں متغیرات کے نام رکھنے کے تین اہم اصول کیا ہیں؟

تفصیلی سوالات

- 1 پانٹھنوں میں بنیادی ڈیٹا کی اقسام کیا ہیں؟ مثالوں کے ساتھ integers، floats، strings اور Booleans کی وضاحت کریں۔
- 2 پانٹھنوں میں حسابی آپریٹرز کیا ہیں۔ ہر آپریٹر کی مثالیں فراہم کریں اور وضاحت کریں کہ آپ انہیں کب استعمال کریں گے۔
- 3 بیان کریں کہ پانٹھن میں صارفین سے ان پٹ کیسے حاصل کیا جائے اور آؤٹ پٹ کیسے دکھایا جائے۔ مثالوں کے ساتھ input() اور print() فنکشنز کی وضاحت کریں۔
- 4 وضاحت کریں کہ تقابلی اور منطقی آپریٹرز مشروط سٹیٹمنٹس (conditional statements) میں مل کر کیسے کام کرتے ہیں۔ ایک عملی مثال فراہم کریں۔
- 5 پانٹھن میں مشروط سٹیٹمنٹس کیا ہیں؟ مناسب سینکس اور عملی مثالوں کے ساتھ if، else، if-else سٹیٹمنٹس کی وضاحت کریں۔

6 پانٹھن میں ایک ایسا پروگرام لکھیں جو صارف سے ایک نام لے اور دیے گئے نام کو خوش آمدید (greeting) کا پیغام دکھائے۔

7 پانٹھن میں ایک ایسا پروگرام لکھیں جو صارف سے دو نمبر لے اور ان کا حاصل جمع دکھائے۔

8 پانٹھن میں ایک ایسا پروگرام لکھیں جو صارف سے پانچ نمبر لے اور ان کا اوسط دکھائے۔

9 درج ذیل ایکسپریشنز کو حل کریں:

(a) $8 + 3 * 4 - 2 ** 2$

(b) $20 \% 7 * 3 + 10 // 3$

(c) $2 * 3 ** 2 \% 4 + 10 - 6 / 2$

کثیر انتخابی سوالات کے جوابات

1 د غلطیوں کو نظر انداز کرنا (Ignore Errors)

2 ج "Add Python to PATH" کو چیک کریں

3 ب

4 ج سٹرنگ (String)

5 د توسین ()

6 ب False

7 ج جمع کرتا ہے اور تفویض کرتا ہے (Adds and assigns)

8 ب int()

9 الف Age: 25

10 ج #