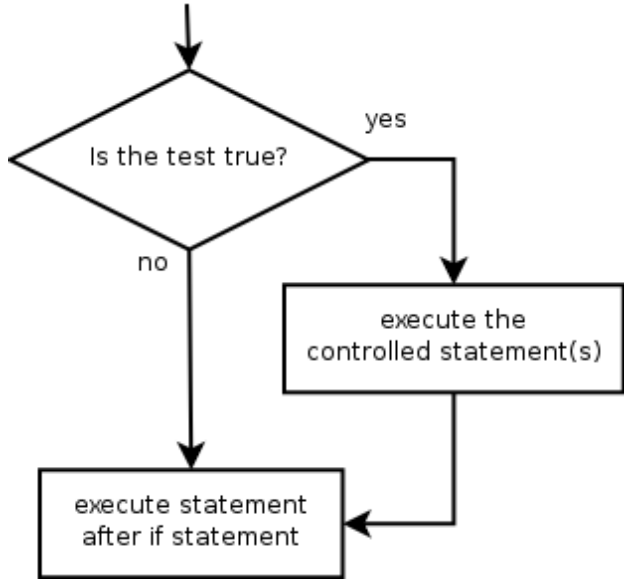
Python Branching

Panchatcharam M

BRANCHING

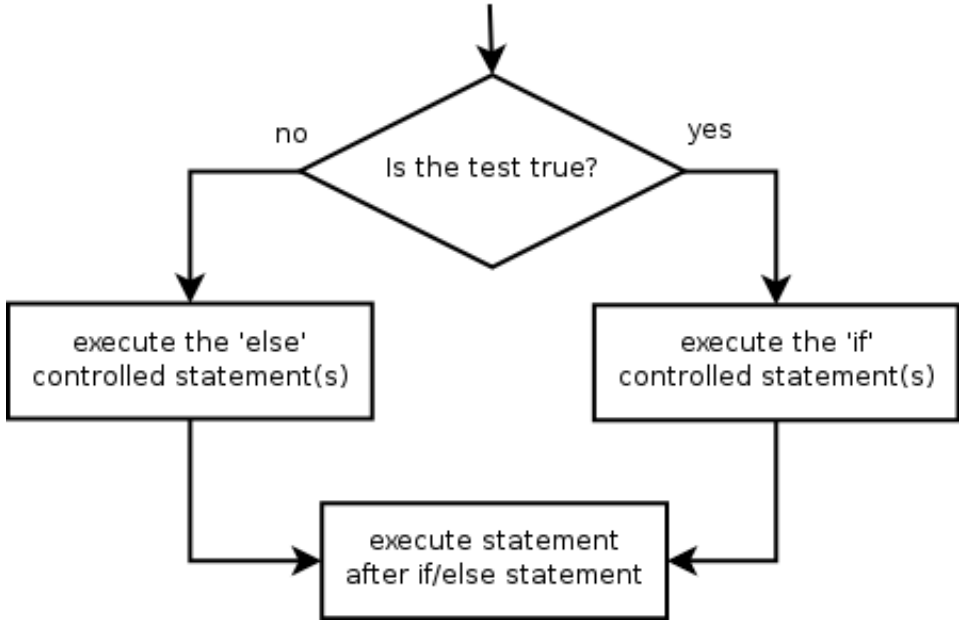
- **`if` statement**: Executes a group of statements only if a certain condition is true. Otherwise, the statements are skipped.
  - Syntax:
    ```
    if condition:
        statements
    ```
- Example:

```
gpa = 3.4
if gpa > 2.0:
    print("Your application is accepted.")
```



- **`if/else` statement**: Executes one block of statements if a certain condition is True, and a second block of statements if it is False.
  - Syntax:

```
if condition:
    statements
else:
    statements
```

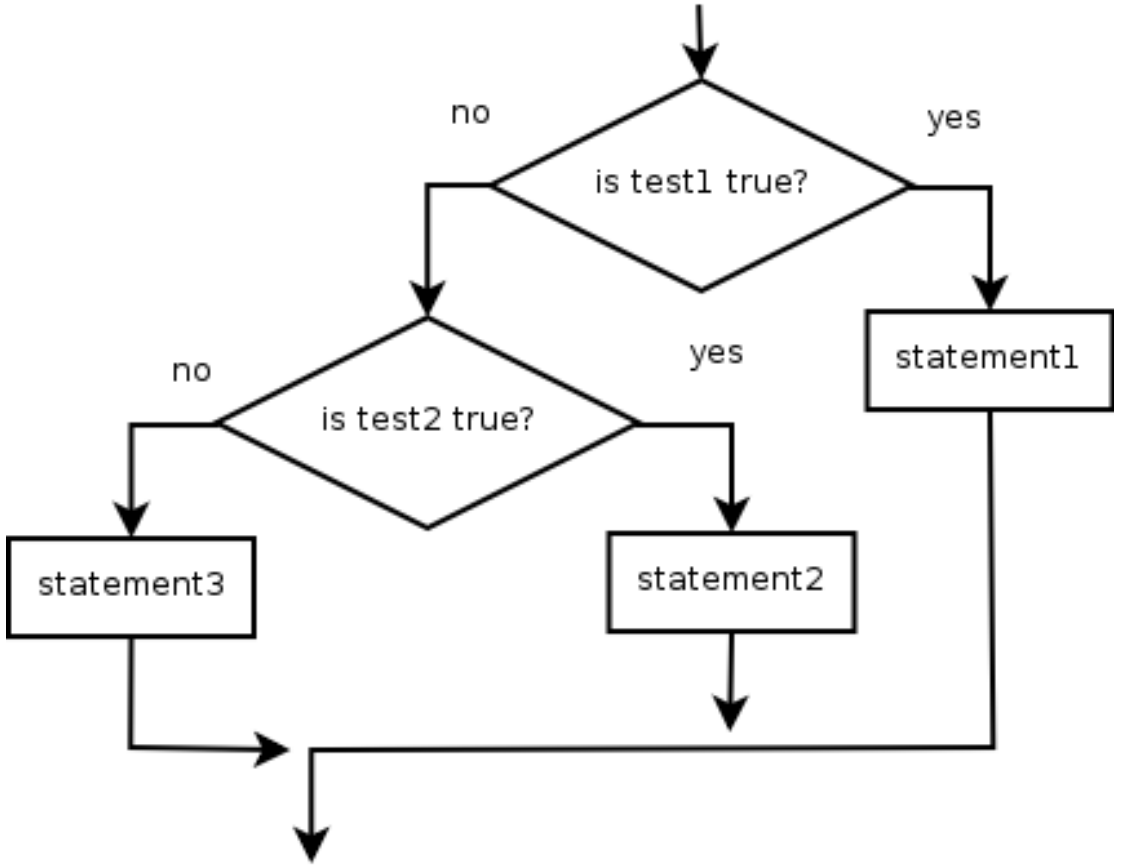


- Example:

```
gpa = 8.4
if gpa > 7.0:
    print("Eligible For Project!")
else:
    print("Your application is denied.")
```

- Multiple conditions can be chained with `elif` ("else if"):

```
if condition :
        statements
elif condition :
        statements
else:
        statements
```



Relational Operators

Operator	Meaning	Example	Result
==	equals	1 + 1 == 2	True
!=	does not equal	3.2 != 2.5	True
<	less than	10 < 5	False
>	greater than	10 > 5	True
<=	less than or equal to	126 <= 100	False
>=	greater than or equal to	5.0 >= 5.0	True

Logical Operators

Operator	Example	Result
and	9 != 6 and 2 < 3	True
or	2 == 3 or -1 < 5	True
not	not 7 > 0	False

Python Loops

Panchatcharam M

LOOPS

- **`for` loop**: Repeats a set of statements over a group of values.
  - Syntax:

```
for variableName in groupOfValues:
    statements
```

    - We indent the statements to be repeated with tabs or spaces.
    - ***variableName*** gives a name to each value, so you can refer to it in the ***statements***.
    - ***groupOfValues*** can be a range of integers, specified with the `range` function.

- Example:

```
for x in range(1, 6):
    print(x, "squared is", x * x)
```

Output:

```
1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
5 squared is 25
```

- The `range` function specifies a range of integers:
  - `range(`***start***`,` ***stop***`)` - the integers between ***start*** (inclusive)
    and ***stop*** (exclusive)

- It can also accept a third value specifying the change between values.
  - `range(`***start***`,` ***stop***`,` ***step***`)` - the integers between ***start*** (inclusive)
    and ***stop*** (exclusive) by ***step***

- Example:

```
for x in range(5, 0, -1):
    print(x)
print("Blastoff!")
```

Output:

```
5
4
3
2
1
Blastoff!
```

```
sum = 0
for i in range(1, 11):
    sum = sum + (i * i)
print("sum of first 10 squares is", sum)
```

Output:

```
sum of first 10 squares is 385
```

- Example:

```
course="Data Science"

for x in course:
    print(x)
```

Output:

```
D
a
t
a

S
c
i
e
n
c
e
```

Break, Pass and Continue

- Example:

```
course="Data Science"

for x in course:
    if x=='S':
      break
    print(x)
```

Output:

```
D
a
t
a
```

```
for x in course:
    pass
```

Output:

```
for x in course:
    if x=='S':
      continue
    print(x)
```

Output:

```
D
a
t
a

c
i
e
n
c
e
```

- Example:

```
for i in range(1,5):
    for j in range(1,5):
        print("(",i,",",j,")",end=",")
```

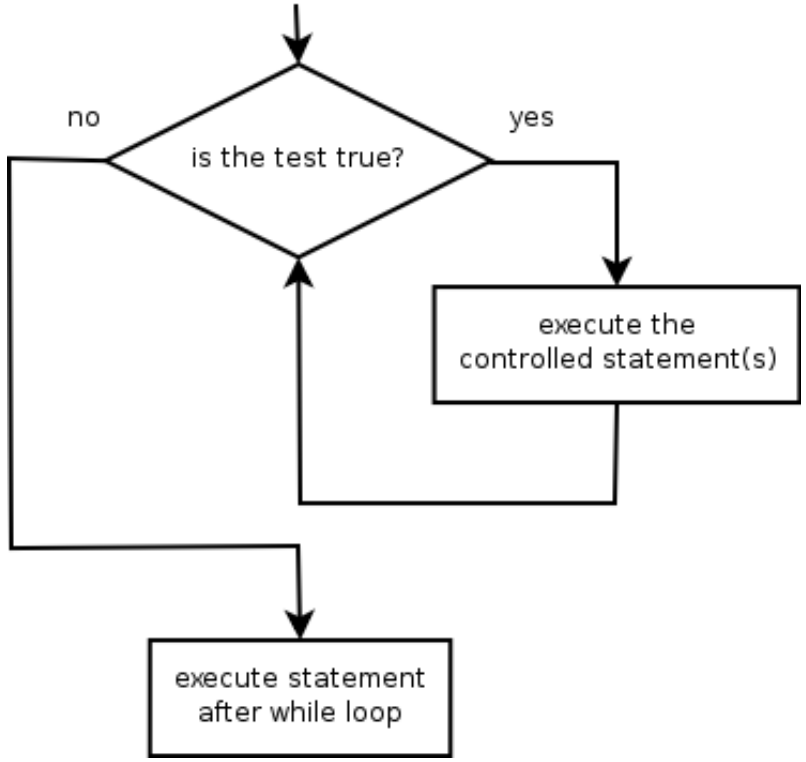
Output:

```
( 1 , 1 ),( 1 , 2 ),( 1 , 3 ),( 1 , 4 ),( 2 , 1 ),( 2 , 2 ),( 2 , 3 ),( 2 , 4 ),
( 3 , 1 ),( 3 , 2 ),( 3 , 3 ),( 3 , 4 ),( 4 , 1 ),( 4 , 2 ),( 4 , 3 ),( 4 , 4 ),
```

- **`while` loop**: Executes a group of statements as long as a condition is True.
  - good for *indefinite loops* (repeat an unknown number of times)

- Syntax:

```
while condition :
        statements
```

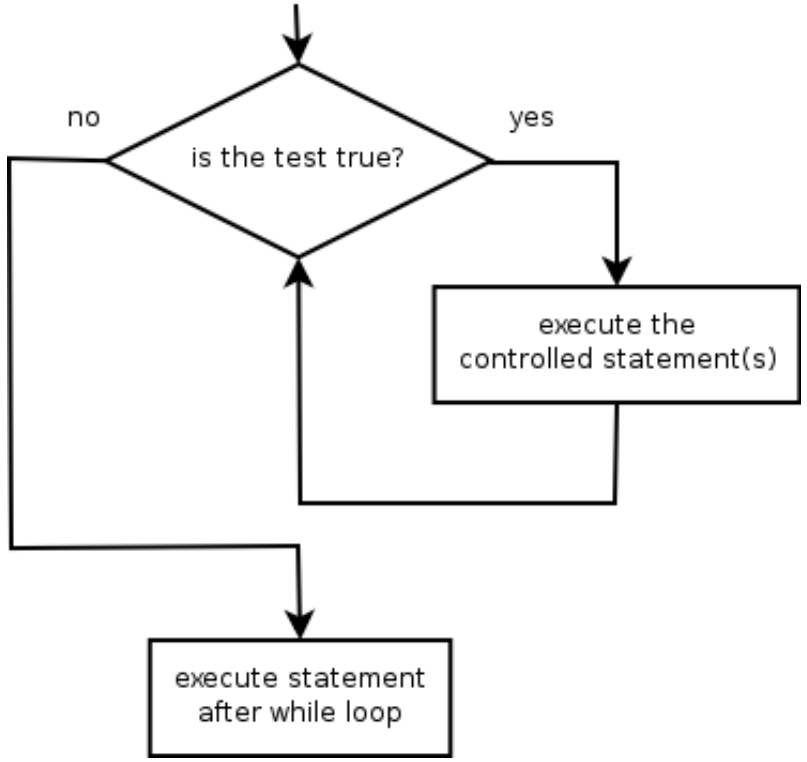


- Example:

```
number = 1
while number < 200:
    print(number)
    number = number * 2
```

  - Output:

```
1 2 4 8 16 32 64 128
```



- Check whether given number is prime or composite

SUMMARY

- ✓ **How to write Simple Python Code**
- ✓ **Python Language Mechanism**
- ✓ **Each code is sequence of instructions**

- ✗ **Not easy for larger problems**
- ✗ **Difficult to keep track of details**
- ✗ **How do check whether you have given correct input to section of the code**

End of Python Branching