Amar Panchal
amarpanchal.education

Generations of Programming languages:
Gen1 Machine Language
Gen2 Assembly language
Gen3 High Level Language
Gen4 Structured Query Language (SQL)
Gen5 Knowledge based Language

→ Art of making computer do what exactly you want

→ WORM - Write Once Run Many

A programming language has three parts:

a. Sequences
b. Selections
c. loops

→ OOPS Object oriented Programming is a thinking methodology, where we first think about output needed, then process to produce output & then input to run the process.
→ before going to interview, prepare these questions too:

1. Who r u ?
2. From where r u ?
3. Family values ????/////??
4. Last degree , what best you did?
5. Why jumped to this domain ?
6. Interests?
7. future?

→ dynamic, high-level, free, open-source, interpreted programming language
→ python is not pure object-oriented, but capable of achieving object-oriented programming
→ guido van rossumand feb 20, 1991, based on BBC MOnty's python Flying Circus
→ syntax is grammar of programming
→ python has

a. Loose syntax - not having a specific syntax, having multiple options for one syntax
b. OOPs
c. Eazy
d. Dynamic casting - on the go data type, auto-casting

→ IDLE - Integrated Development & Learning Environment
→ Python supports dynamic casting, it means data type of a variable will be decided by the data given to it
→ in python, we have five primitive basic data types:

a. int

b. float
c. bool
d. string
e. complex

→ type() command would specify the data type of a variable
→ comments in python:
a. ‘#’ is used to put single line comments
b. “ ‘’’ ‘’’ “ three-single quotes are used to put multiple-line comments
→ character sets are basically of two types:
a. ASCII 8-bit
b. UNICODE 16-bit
→ python follows UNICODE character set
→ Tokens are
a. keywords : reserved meaning words
b. Identifiers : name to identify a variable, function, etc.
c. Operators :

→ Variable assignments in python:
a. Normal a=10; b=20; c=30
b. Chained a=b=c=100
c. Pair based assignment a, b, c = 10, 20.23, “amar”
→ Memory handling in python:
a. Same data share memory
b. Change in data updates at new memory location
c. id()
→ AGC (Auto-Garbage Collection) : collects & clears garbage values at regular intervals, to free up garbage locations

→ Multiple data with same value will share memory space
→ Python will allocate new memory location for every change done to a variable
→ Python has AGC(Auto-Garbage Collection)
→ command used for memory reference is “id()”
→ Python I/O:
a. Output: print()

```
I. print('/"/'''/""" message """/'''/"/')
II. print(var)
III. print('message', var)
IV. print(end="\n", sep=" ") # end: end-of_line ; sep:
separator

a=10
print("a is", a)
# each comma induces a space
# after each print() it puts a newline char
```

b. Input
    I. input("Prompt message:") # default read str, can be cast to int, float, etc

Default mode for I/O in python is 'str'

→ Python operators:

a. Arithmetic operators

'+' add	lowest
'-' subtract	lower
'*' multiplication	low
'/' division	high
'%' modulus , supports 'float' values	higher
'//' floor division / gives 'int' based results	second highest
'**' exponent / power	highest

b. Comparison operators

==	checks if two operands are equal
!=	checks if two operands are not equal
<>	checks if two operands are not equal
>	checks if left operand is greater than right operand
<	checks if left operand is lesser than right operand
>=	checks if the left operands is gerater than or equal to right operand
<=	checks if the left operands is lesser than or equal to right operand

c. Assignment operators :

I. In assignment operators, we have short-circuit (or short-hand) operator, which is a replacement to increment/decrement operator

II. Short-hand operators gave an ability to provide fractional & Non-unit increment/decrement

'=' assign
'+=' add and assign short-hand
'-'= subtract and assign short-hand
'*=' multiply and assign short-hand
'/=' divide and assign short-hand
'%=' modulus and assign short-hand
'**=' exponent and assign short-hand
'//=' floor division and assign short-hand

d. Logical operators

not	logical NOT
and	logical AND
or	logical OR

e. Bitwise operators
   << bitwise left shift
   >> bitwise right shift
   & bitwise AND highest
   ^ bitwise XOR higher
   | bitwise OR high

   ~ binary ones complement
   << binary left shift
   >> binary right shift

f. Identity operators
   is
   is not

g. Membership operators
   in
   not in

h. asas

→ shorthand operators can be used in place of increment/decrement operators, but increment/decrement operators are not allowed