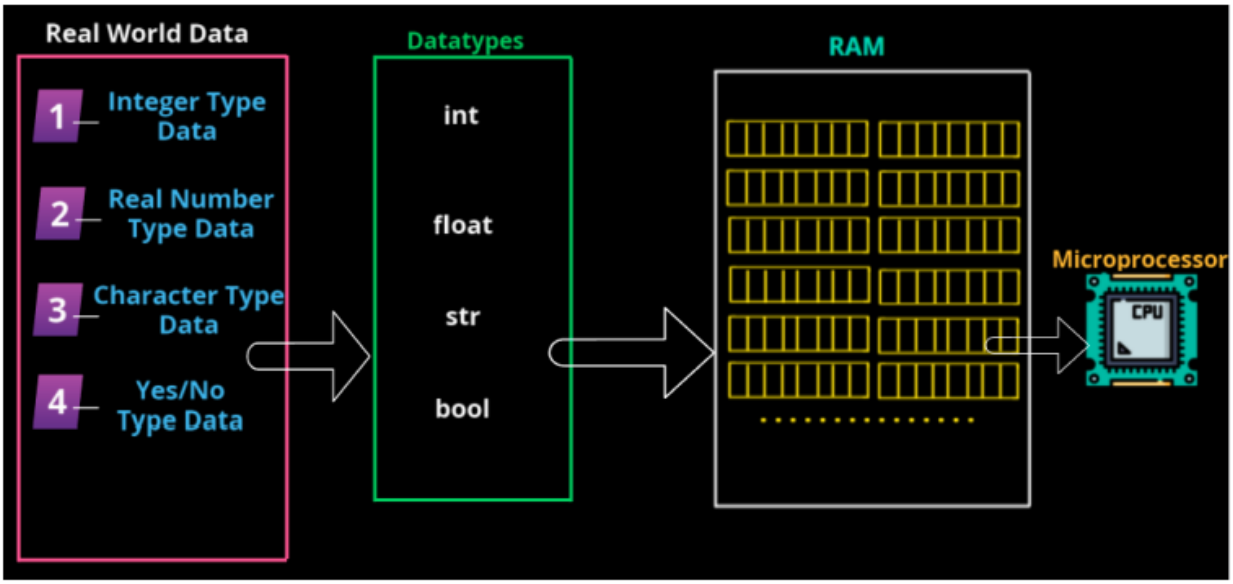
Primitive Data Types

How data is stored in a system?

In every electronic system we have RAM which stores the data temporarily. All the data that we enter is in high level but it is always stored in low level inside the system so that the system/machine can understand. So RAM consists of several bytes, each byte consists of 8 bits, each bit has two transistors which can store high and low value (1's and 0's).

RAM can only take inputs in 0's and 1's.

Apparently the data in real world is not combination of 0's and 1's.
We have basic 4 types of real world data which are integers, real numbers, characters, yes/no types of data. The basic data types in python which represents integers, real world, characters, yes/no type data are int, float, str, bool respectively. These data types help to convert high level form of data to combination of 0's and 1's that is binary format.



Why Do We Need Data Types in Python?

1. Efficient Memory Usage: Python assigns memory based on the type and size of the value. 2. Type-Specific Operations: Each data type supports specific operations (e.g., you can add numbers but not mix strings and integers directly).
3. Error Prevention: Type declarations help catch bugs and invalid operations early.
4. Code Clarity: Knowing variable types makes the code easier to read and maintain.
5. Performance Optimization: Python internally optimizes execution depending on the type of data.

1. int (Integer)

Stores whole numbers without a decimal point.
Example: a = 10
Operations: +, -, *, //, %, **, comparison (==, >, <)
Size: Dynamic in Python (unlimited precision), depends on value

2. float (Floating Point)

Stores numbers with a decimal point.
Example: b = 3.14
Operations: +, -, *, /, //, %, round(), comparison
Size: Typically 24 bytes (platform dependent)

3. str (String)

Stores a character or sequence of characters enclosed in quotes.
Example: name = 'Alice' or “Alice” or ‘‘‘A’’’ , “20” , “True”
Operations: + (concatenation), * (repetition), slicing, indexing, len(), in, not in.
Size: Depends on length; each character ~1 byte (ASCII) or more (Unicode)
[Note : Anything inside double quote will always be treated as String means str data type]

4. bool (Boolean)

Stores either True or False.
Example: flag = True
Operations: and, or, not, comparison operators
Size: 28 bytes in CPython implementation

5. complex (Complex Number)

Stores a complex number in the form a + bj.
Example: z = 2 + 3j
Operations: +, -, *, /, abs(), real, imag
Size: Usually 32 bytes

Memory Mapping

All the operations carried out by computer or any electronic device happens on the RAM. In simple words we can say that RAM is shared by different applications. So when we are trying to execute the python code, it won't occupy all the space in RAM. It'll use only a certain region on the RAM. Within this region there are few divisions or segmentations are present. This region contains something called as stack and private heap.

Objects/Variables are allocated on private heap. References are allocated on stack. And address of the object created on private heap will be present inside the reference, which means reference is pointing to the object.

[Note: As python is dynamically typed programming language, we need not declare the type of object. Based on the value given the interpreter itself decides the type of an object. And in programming languages class means type.]

```python
a=20 # int data type
print(a+a) # addition
print(a*a, a*2, a//2, a/2)
print(type(a), id(a))



b=20.9   #float datatype
print(b+b, b*2, b//2, b/2)
print(type(b))

st="Python learning" # String data type
print(type(st), id(st))
print(st+st) # concatenate means add at the last
print(st*3)

b=60
a=30
# true =1 or 1.0 False =0
boo=True # boolean data type
print(boo, type(boo))
print(b>a)
print(a>b)
boo=False
print(boo, type(boo))
print(True+True)  #1+1
print(True-False) # 1-0
print(True*2)
print(True>False)

boo="False" # string data type not boolean coz it's enclosed in double quote
print(boo, type(boo))

st="20" # string data type
print(st, st*2, type(st))
print(st+st)
st_1="paper20"
print(st_1)
print(type(st_1))
print(st_1*3)
print(st_1+st_1)

comp=20+30j # complex data type
print(comp, type(comp))
```

```
print(comp.real)
print(comp.imag)
```