Chapter 1: Basics (Syntax, Variables, Data Types, I/O)

Full Notes

Python is an **interpreted, high-level programming language** known for its readability and simplicity. Unlike languages such as C or Java, Python relies on **indentation** (whitespace) to define blocks of code, rather than braces `{}`.

Variables

- Variables in Python are created when a value is assigned to a name.
- Python is **dynamically typed**, meaning you don't declare a variable type in advance.

insert code example below

```
x = 5        # int
y = 3.14     # float
name = "Ali" # string
```

Data Types

- **int** → whole numbers (e.g., 10, -3)
- **float** → decimal numbers (e.g., 3.14, -2.5)
- **str** → text (e.g., `"hello"`)
- **bool** → logical values `True` / `False`
- **NoneType** → absence of a value (`None`)

insert code example below

```
a = True
b = None
print(type(a), type(b))  # <class 'bool'> <class 'NoneType'>
```

Input & Output

- `input()` → reads input as a string.
- `print()` → displays output; supports arguments like `sep` and `end`.

insert code example below

```
name = input("Enter your name: ")
print("Hello,", name)
```

Type Conversion

Convert between types using constructors:

insert code example below

```
age = int("20")      # string → int
pi = float("3.14")   # string → float
s = str(42)          # int → string
```

Strings & Slicing

- Strings are sequences → can be indexed and sliced.
- Negative indexing allows access from the end.

insert code example below

```
s = "Python"
print(s[0])      # P
print(s[-1])     # n
print(s[1:4])    # yth
```

Comments

- Single-line: `# comment`
- Multi-line: triple quotes `""" comment """`

Flashcards (10)

1. **Q:** What defines a block of code in Python?
   A: Indentation
2. **Q:** What type of typing does Python use?
   A: Dynamic typing
3. **Q:** What type does `input()` return?
   A: String (`str`)
4. **Q:** Which function displays output?
   A: `print()`
5. **Q:** How do you convert a float to an integer?
   A: `int(3.5)`
6. **Q:** Which symbol starts a single-line comment?
   A: `#`
7. **Q:** How do you access the last element of a string?
   A: Negative indexing (`s[-1]`)
8. **Q:** What is slicing syntax in Python?
   A: `s[start:stop:step]`
9. **Q:** What keyword represents “no value”?
   A: `None`
10. **Q:** Which string formatting method uses `{}` placeholders?
    A: f-strings (`f"Hello {name}"`)

MCQs (20) with Answers & Hints

Easy (5)

1. Which of the following defines a block of code in Python?

   Hint: Python avoids `{}` unlike Java/C.

   A) Braces `{}`

   B) Indentation

   C) Semicolons

   D) Parentheses

   Answer: B

2. What is the default type of value returned by `input()`?

   Hint: You must convert input into int/float if needed.

   A) int

   B) float

   C) str

   D) bool

   Answer: C

3. Which of the following represents a `NoneType` object?

   Hint: It means "no value".

   A) 0

   B) ""

   C) None

   D) False

   Answer: C

4. What will `print("Hi", "Ali", sep="-")` output?

   Hint: `sep` decides separator between arguments.

   A) Hi Ali

   B) HiAli

   C) Hi-Ali

   D) Error

   Answer: C

5. Which symbol is used for a single-line comment in Python?

   Hint: Similar to shell scripts.

   A) //

   B) <!-- -->

   C) #

   D) --

   Answer: C

Medium (5)

6. What is the output of:

```
x = "Python"
print(x[2:5])
```

Hint: Start index is inclusive, end index is exclusive.

A) Pyt

B) tho

C) yth

D) hon

Answer: C

7. Which function converts `"3.14"` to a float?

   Hint: Use correct constructor.

   A) str()

   B) int()

   C) float()

   D) eval()

   Answer: C

8. Which of the following is falsy in Python?

   Hint: Empty containers are falsy.

   A) [1,2,3]

   B) ""

   C) "False"

   D) 10

   Answer: B

9. What happens if you run:

```
int("3.0")
```

Hint: `int()` can't parse decimals with dots.

A) Returns 3

B) Returns 3.0

C) Error

D) None

Answer: C

10. Which is a correct f-string usage?

Hint: Must start with f and curly braces.

A) print("Age is {age}")

B) print(f"Age is {age}")

C) print(f'Age is age')

D) print(f(Age is age))

Answer: B

Hard (10)

11. Which of the following shows **dynamic typing**?

Hint: Must start with f and curly braces.

A) `int x = 10`

B) `x = 10; x = "hello"`

C) `float y = 5.6`

D) `def func(): pass`

Answer: B

12. What is the output?

```
print("A","B","C", end="*")
```

Hint: `end` replaces newline.

A) A B C *

B) A B C*

C) ABC*

D) A-B-C*

Answer: B

13. Which of the following does NOT create a new string `"abc"`?

Hint: slicing entire string may reuse the object.

A) `"abc"[0:3]`

B) `"a" + "b" + "c"`

C) `"abc".strip()`

D) `str("abc")`

Answer: A

14. Which function shows the exact representation of an object (useful for debugging)?

Hint: Think developer view, not user view.

A) str()

B) repr()

C) print()

D) debug()

Answer: B

15. Which of these is NOT immutable?

Hint: lists can be modified in place.

A) str

B) int

C) float

D) list

Answer: D

16. What will be the result?

```
print("python"[-2:])
```

Hint: Negative index starts from the end.

A) py
B) on
C) th
D) no
Answer: B

17. Which method removes whitespace from both ends of a string?

Hint: Think of peeling outer layers.

A) cut()
B) trim()
C) strip()
D) clean()
Answer: C

18. What happens if you try to convert `float("ten")`?

Hint: Only numeric strings can be cast.

A) 10.0
B) Error
C) None
D) NaN
Answer: B

19. Which of the following correctly swaps two variables `a` and `b`?

Hint: Python supports tuple unpacking.

A) `a = b; b = a`

B) `a, b = b, a`

C) `swap(a,b)`

D) `temp = a; b = a; a = b`

Answer: B

20. Why is `print("Hello", end="")` useful?

Hint: `end` defines what happens after output.

A) It skips spaces

B) It prevents newlines

C) It adds commas

D) It repeats output

Answer: B

Chapter 2: Control Flow (if/elif, loops, comprehensions)

Full Notes

Control flow determines **how Python executes code based on conditions or repetition**.

Conditional Statements

- **if/elif/else** → used for decision making.

insert code example below

```
age = 20
if age >= 18:
    print("Adult")
elif age >= 13:
    print("Teen")
else:
    print("Child")
```

Loops

- **for loop** → iterates over iterables (lists, strings, dicts).

insert code example below

```
for i in range(3):
    print(i)    # 0,1,2
```

- **while loop** → continues until condition is False.

insert code example below

```
n = 3
while n > 0:
    print(n)
    n -= 1
```

Loop Control Keywords

- `break` → exit loop early.
- `continue` → skip current iteration.
- `pass` → placeholder, does nothing.

Truthy and Falsy Values

- Falsy: `0`, `""`, `None`, `[]`, `{}`, `set()`.
- Truthy: most non-empty values.

Comprehensions

- Short syntax for building lists, sets, or dicts.

insert code example below

```
nums = [1,2,3,4,5]
squares = [n*n for n in nums if n % 2 == 0]
print(squares)  # [4, 16]
```

Flashcards (10)

1. **Q:** What keyword starts a conditional block?
   A: `if`
2. **Q:** Which keyword provides an alternative condition?
   A: `elif`
3. **Q:** What keyword is used when all other conditions fail?
   A: `else`
4. **Q:** What operator checks equality?
   A: `==`
5. **Q:** Which keyword starts a while loop?
   A: `while`
6. **Q:** Which keyword starts a for loop?
   A: `for`

7. **Q:** What function generates a sequence of numbers for loops?

   A: `range()`
8. **Q:** Which keyword stops a loop immediately?

   A: `break`
9. **Q:** Which keyword skips to the next loop iteration?

   A: `continue`
10. **Q:** Which keyword defines a placeholder loop with no action?

    A: `pass`

MCQs (20) with Answers & Hints

Easy (5)

1. Which keyword ends a loop immediately?

   Hint: Think of "breaking out".

   A) stop

   B) break

   C) exit

   D) continue

   Answer: B

2. Which of these is **falsy**?

   Hint: Empty containers are always False.

   A) " "

   B) []

   C) [0]

   D) "False"

   Answer: B

3. What does `pass` do?

   Hint: It's like leaving a blank statement.

   A) Skips a loop iteration

   B) Placeholder (does nothing)

   C) Exits the program

   D) Returns from a function

   Answer: B

4. Which loop is best for iterating through a list?

   Hint: Directly goes through elements.

   A) while

   B) repeat

   C) for

   D) goto

   Answer: C

5. What is the output?

```
for i in range(2):
    print("Hi")
```

Hint: Range(2) → 0 and 1 → 2 iterations.

A) Hi

B) HiHi

C) Hi (printed twice)

D) Error

Answer: C

Medium (5)

6. What will this code output?

```
x = 0
while x < 3:
    x += 1
print(x)
```

Hint: Loop stops when x=3.

A) 0
B) 2
C) 3
D) 4
Answer: C

7. Which statement skips the rest of a loop's body but keeps looping?

Hint: It "continues" to the next round.

A) break
B) continue
C) pass
D) skip
Answer: B

8. Which comprehension creates a list of even numbers up to 10?

Hint: Square brackets → list comprehension.

A) `[n for n in range(11) if n % 2 == 0]`
B) `{n for n in range(11) if n % 2 == 0}`
C) `(n for n in range(11) if n % 2 == 0)`
D) `n for n in range(11) if n % 2 == 0`

Answer: A

9. Which of these is NOT valid?

Hint: Missing colon.

A) if condition: print("Hi")

B) for i in [1,2,3]: print(i)

C) while True print("Hi")

D) pass

Answer: C

10. What is output of:

```
for i in range(3):
    if i == 1:
        continue
    print(i)
```

Hint: `continue` skips printing when i=1.

A) 0 1 2

B) 0 2

C) 1 2

D) 2

Answer: B

Hard (10)

11. What will this code print?

```
for i in range(3):
    for j in range(2):
        print(i,j)
```

Hint: Outer 3 × Inner 2 = 6 pairs.

A) 3 pairs
B) 5 pairs
C) 6 pairs
D) 9 pairs
Answer: C

12. Which of these correctly builds a dict mapping numbers to their squares (1–5)?

Hint: Dict → curly braces with key:value.

A) `{n:n**2 for n in range(1,6)}`
B) `[n:n**2 for n in range(1,6)]`
C) `{n,n**2 for n in range(1,6)}`
D) `(n:n**2 for n in range(1,6))`
Answer: A

13. What is output?

```
nums = [1,2,3,4]
print([n for n in nums if n>2])
```

Hint: Only values greater than 2.

A) [1,2]
B) [3,4]
C) [2,3,4]
D) Error
Answer: B

14. Which while loop will run forever?

Hint: 1 is truthy forever.

A) `while False:`
B) `while 0:`
C) `while "":`
D) `while 1:`
Answer: D

15. Which of these statements is TRUE about comprehensions?

Hint: They're concise but not universal.

A) They are always faster.
B) They can replace any loop.
C) They create new collections concisely.
D) They cannot use conditions.
Answer: C

16. What will this output?

```
for i in range(3):
    pass
print(i)
```

Hint: Last value of i = 2.

A) 0
B) 2
C) 3
D) Error
Answer: B

17. Which keyword allows you to handle a "not found" case after loop completion?

Hint: for/while can have an else clause.

A) except
B) else (with loop)
C) finally
D) pass
Answer: B

18. What is output?

```
i=0
while i<3:
    print(i)
    i+=2
```

Hint: i increases by 2 → 0 then 2.

A) 0 1 2
B) 0 2

C) 0 2 4

D) Infinite loop

Answer: B

19. Which comprehension flattens a 2D list `[[1,2],[3,4]]`?

    Hint: Double for → flatten.

    A) `[x for row in lst for x in row]`

    B) `[row for x in lst for row in x]`

    C) `{x for row in lst for x in row}`

    D) `x for row in lst for x in row`

    Answer: A

20. Why should complex comprehensions sometimes be avoided?

    Hint: Too complex = hard to read.

    A) They are slow

    B) They are unreadable

    C) They don't exist

    D) They cause errors

    Answer: B

Chapter 3: Functions & Modules

Full Notes

Functions let you **organise code into reusable blocks**, while modules allow you to **reuse code across programs**.

Defining Functions

- Use `def` keyword:

insert code example below

```
def greet(name):
    return f"Hello, {name}!"
```

Parameters & Arguments

- **Positional arguments** → passed in order.
- **Keyword arguments** → passed by name.
- **Default values** → used when not provided.

insert code example below

```
def area_circle(r, pi=3.14):
    return pi * r * r
print(area_circle(2))        # 12.56
print(area_circle(2, 3.14159))  # 12.56636
```

- `*args` → packs extra positional arguments as a tuple.
- `**kwargs` → packs extra keyword arguments as a dict.

insert code example below

```
def demo(*args, **kwargs):
    print(args, kwargs)
demo(1,2,3, a=10, b=20)
```

Return Values

- Functions may return single or multiple values.

insert code example below

```
def swap(a, b):
    return b, a
x, y = swap(1,2)    # x=2, y=1
```

Scope

- Local: inside function.
- Global: outside any function.
- Use `global` keyword to modify global variables (not recommended often).

Docstrings

- Triple quotes after `def` describe the function.
- Accessible via `help(function)`.

Modules

- A Python file (`.py`) containing code.
- Importing:

insert code example below

```
import math
print(math.sqrt(16))
```

- Aliases: `import math as m`.
- Selective import: `from math import sqrt`.

Virtual Environments & pip

- Virtual environment → isolated Python environment for projects.
- `pip` installs packages:

insert code example below

```
pip install requests
```

Flashcards (10)

1. **Q:** Which keyword defines a function?

   A: `def`
2. **Q:** How do you call a function named `greet`?

   A: `greet()`
3. **Q:** What keyword is used to return a value from a function?

   A: `return`
4. **Q:** What is a parameter?

   A: A variable defined inside the function header
5. **Q:** What is an argument?

   A: The actual value passed to a function
6. **Q:** What are default parameters?

   A: Parameters with preset values (`def func(x=10)`)
7. **Q:** What type of scope do variables inside functions have?

   A: Local scope
8. **Q:** How do you define an anonymous function?

   A: Using `lambda`
9. **Q:** What are docstrings used for?

   A: Documenting functions (`"""description"""`)
10. **Q:** Can a function return multiple values in Python?

    A: Yes, as a tuple

MCQs (20) with Answers & Hints

Easy (5)

1. Which keyword defines a function?

   Hint: Short for define.

   A) func

   B) def

   C) function

   D) define

   Answer: B

2. What is the default return value of a function without `return`?

   Hint: No return → `None`.

   A) 0

   B) None

   C) False

   D) “”

   Answer: B

3. What will this output?

```
def f(x=5): return x
print(f())
```

Hint: Uses default argument.

A) Error

B) None

C) 5

D) "5"

Answer: C

4. Which collects extra positional arguments?

   Hint: Star collects extras.

   A) args

   B) *args

   C) **kwargs

   D) list()

   Answer: B

5. Which library is built-in?

   Hint: Comes with Python by default.

   A) numpy

   B) requests

   C) math

   D) pandas

   Answer: C

Medium (5)

6. What is output?

```
def swap(a,b): return b,a
x,y = swap(3,4)
print(x,y)
```

Hint: Values swapped.

A) 3 4

B) 4 3

C) (4,3)

D) Error

Answer: B

7. Which statement about `global` is TRUE?

   Hint: Accesses outer scope variable.

   A) It creates a new variable.

   B) It modifies a variable outside the function.

   C) It deletes a variable.

   D) It copies a variable.

   Answer: B

8. Which docstring format is correct?

   Hint: Triple quotes.

   A) “comment”

   B) # docstring

   C) “"""This is a docstring."""

   D) /* docstring */

   Answer: C

9. Which of these imports `sqrt` only?

   Hint: `from module import function`.

   A) import math.sqrt

   B) from math import sqrt

   C) import sqrt from math

   D) math sqrt()

   Answer: B

10. Which command installs a package?

    Hint: pip is Python installer.

    A) install pkg

    B) pip install pkg

    C) py install pkg

    D) python get pkg

    Answer: B

Hard (10)

11. Which of these returns multiple values?

    Hint: Tuples, lists, or multiple items all work.

    A) return 1,2

    B) return [1,2]

    C) return (1,2)

    D) All of the above

    Answer: D

12. Which is a higher-order function?

    Hint: "Higher" = takes/returns functions.

    A) Function returning another function.

    B) Function without return.

    C) Function with default parameter.

    D) Function with print only.

    Answer: A

13. Which keyword argument call is valid?

```
def add(x,y): return x+y
```

Hint: Both positional and keyword allowed.

A) add(3,2)

B) add(y=2,x=3)

C) add(3,y=2)

D) All of these

Answer: D

14. Which is dangerous as a default parameter?

    Hint: Mutable defaults can change unexpectedly

    A) int

    B) None

    C) []

    D) 0

    Answer: C

15. Which of these is a module alias import?

    Hint: `as` creates alias.

    A) import math as m

    B) import m = math

    C) math = import m

    D) alias math as m

    Answer: A

16. What happens if you call `help(function_name)`?

    Hint: Used for documentation.

    A) Runs the function.

    B) Prints docstring.

C) Deletes function.

D) Nothing.

Answer: B

17. Which command creates a virtual environment (modern)?

Hint: venv is built-in.

A) python -m venv env

B) pip create env

C) venv install env

D) py env create

Answer: A

18. Which scope does Python check first?

Hint: Follows LEGB rule.

A) Global

B) Built-in

C) Local

D) Module

Answer: C

19. Which rule defines variable resolution order?

Hint: Classic rule in Python.

A) FIFO

B) LIFO

C) LEGB (Local, Enclosing, Global, Built-in)

D) None

Answer: C

20. Why use modules?

Hint: Reuse & structure code.

A) Avoids reusing code

B) Organises and reuses functions

C) Slows execution

D) Creates errors

Answer: B

Chapter 4: Data Structures (Lists, Tuples, Dicts, Sets)

Full Notes

Data structures are **containers** that store and organise data. Python provides powerful built-ins:

Lists

- Ordered, mutable sequences.
- Common operations:

insert code example below

```
nums = [1,2,3]
nums.append(4)        # [1,2,3,4]
nums.pop()            # [1,2,3]
nums.sort()           # in-place
```

Tuples

- Ordered, immutable sequences.
- Used for fixed data or as keys in dictionaries.

insert code example below

```
point = (3,4)
print(point[0])   # 3
```

Dictionaries

- Key-value mappings.
- Fast lookups.

insert code example below

```
ages = {"Ali":20, "Bala":22}
print(ages["Ali"])          # 20
```

```
print(ages.get("Chong",0)) # safe access
```

Sets

- Unordered collection of unique elements.

insert code example below

```
letters = set("balloon")
print(letters)    # {'b','a','l','o','n'}
```

- Operations: union (`|`), intersection (`&`), difference (`-`).

Copying

- `list(a)` / `a[:]` → shallow copy.
- `copy.deepcopy()` → deep copy (nested structures).

Flashcards (10)

1. **Q:** How do you create a list in Python?
   A: With square brackets `[]`
2. **Q:** Are lists mutable or immutable?
   A: Mutable
3. **Q:** How do you create a tuple?
   A: With parentheses `()`
4. **Q:** Are tuples mutable or immutable?
   A: Immutable
5. **Q:** How do you create a set?
   A: With curly braces `{}` or `set()`
6. **Q:** Do sets allow duplicate values?
   A: No
7. **Q:** How do you create a dictionary?
   A: With `{key: value}` pairs

8. **Q:** Which method gets all keys of a dictionary?

   A: `.keys()`
9. **Q:** Which method removes and returns the last list item?

   A: `.pop()`
10. **Q:** Which data structure is best for key-value storage?

    A: Dictionary

MCQs (20) with Answers & Hints

Easy (5)

1. Which of these is mutable?

   Hint: You can append/remove.

   A) tuple

   B) str

   C) list

   D) int

   Answer: C

2. How do you access the 2nd element in `nums=[10,20,30]`?

   Hint: Indexing starts at 0.

   A) nums(2)

   B) nums{1}

   C) nums[1]

   D) nums[2]

   Answer: C

3. Which removes duplicates from a list?

   Hint: Sets only keep unique.

   A) tuple(list)

   B) set(list)

   C) dict(list)

   D) list.remove()

   Answer: B

4. Which method removes and returns last element of list?

   Hint: Think "pop out".

   A) pop()

   B) remove()

   C) del()

   D) cut()

   Answer: A

5. What's the output?

```
ages={"Ali":20}
print(ages.get("Bala",0))
```

Hint: Default value used if key not found.

A) None

B) Error

C) 0

D) 20

Answer: C

Medium (5)

6. Which of these is valid tuple?

   Hint: Comma required for 1-item tuple.

   A) (1)

   B) (1,)

   C) [1,]

   D) {1}

   Answer: B

7. Which method sorts list in-place?

   Hint: Only method of list object.

   A) sorted()

   B) list.sort()

   C) arrange()

   D) order()

   Answer: B

8. Which operation finds common elements of sets A and B?

   Hint: Intersection operator.

   A) A+B

   B) A-B

   C) A&B

   D) A|B

   Answer: C

9. Which is a valid dict?

   Hint: Uses key:value pairs.

   A) {1,2,3}

B) {"a":1, "b":2}

C) [("a",1)]

D) ("a":1)

Answer: B

10. What is output?

```
d = {"x":1}
d["y"] = 2
print(d)
```

Hint: Adding new key.

A) {"x":1}

B) {"x":1,"y":2}

C) [1,2]

D) Error

Answer: B

Hard (10)

11. Which of these can be dict keys?

Hint: Must be immutable/hashable.

A) list

B) set

C) tuple

D) dict

Answer: C

12. What will this print?

```
a=[1,2]; b=a
b.append(3)
print(a)
```

Hint: b points to same list.

A) [1,2]
B) [1,2,3]
C) Error
D) None
Answer: B

13. Which method returns list length?

Hint: Built-in function.

A) length()
B) len()
C) count()
D) size()
Answer: B

14. Which dict comprehension inverts a dict `{1:"a",2:"b"}`?

Hint: Flip keys and values.

A) {v:k for k,v in d.items()}
B) {k:v for k,v in d.items()}
C) [v:k for k,v in d.items()]
D) (v:k for k,v in d.items())
Answer: A

15. Which set operation removes elements in B from A?

Hint: Difference.

A) A|B

B) A-B

C) A&B

D) A^B

Answer: B

16. What is output?

```
s = {1,2,2,3}
print(s)
```

Hint: Sets remove duplicates.

A) {1,2,2,3}

B) {1,2,3}

C) [1,2,3]

D) Error

Answer: B

17. Which slice gives last 3 elements of list `nums`?

Hint: Negative index from end.

A) nums[3]

B) nums[-3:]

C) nums[:3]

D) nums[:-3]

Answer: B

18. Which is TRUE about tuples?

Hint: Immutable = hashable.

A) They are mutable.

B) Can be used as dict keys.

C) Slower than lists.

D) Not indexable.

Answer: B

19. What's output?

```
words=["hi","hello","hey"]
print([w[0] for w in words])
```

Hint: Taking first character.

A) ["hi","hello","hey"]

B) ["h","h","h"]

C) ["i","e","y"]

D) Error

Answer: B

20. Why use deep copy over shallow copy?

Hint: Protects against shared references.

A) It's faster

B) To copy nested structures safely

C) It uses less memory

D) It removes duplicates

Answer: B

Chapter 5: OOP & Exceptions

Full Notes

Object-Oriented Programming (OOP) in Python helps you model real-world entities using **classes and objects**. Exceptions let you handle errors gracefully.

Classes & Objects

- **Class** → blueprint for objects.
- **Object** → instance of a class.

insert code example below

```
class Dog:
    def __init__(self, name):
        self.name = name
    def bark(self):
        print(f"{self.name} says woof!")

d = Dog("Buddy")
d.bark()
```

Methods

- **Instance methods** → take `self`.
- **Class methods** → take `cls`, defined with `@classmethod`.
- **Static methods** → no `self` or `cls`, defined with `@staticmethod`.

Encapsulation

- Hide details using naming conventions:
    - `_var` → protected (convention).
    - `__var` → name-mangled.

Inheritance & Polymorphism

- **Inheritance** → create new classes from existing ones.
- **Polymorphism** → same method name, different behaviours.

insert code example below

```
class Animal:
    def speak(self): print("Generic sound")

class Cat(Animal):
    def speak(self): print("Meow")
```

Exceptions

- **try/except** → catch errors.
- **else** → runs if no error.
- **finally** → always runs.

insert code example below

```
try:
    x = 10/0
except ZeroDivisionError:
    print("You can't divide by zero!")
finally:
    print("Done")
```

- Custom exceptions can be created by inheriting from `Exception`.

Flashcards (10)

1. **Q:** Which function is used to open a file in Python?

   A: `open()`

2. **Q:** What are the two main arguments of `open()`?

   A: Filename and mode

3. **Q:** Which mode opens a file for reading?

   A: `"r"`

4. **Q:** Which mode opens a file for writing (overwrites)?

   A: `"w"`

5. **Q:** Which mode appends data to a file?

   A: `"a"`

6. **Q:** Which mode reads and writes a file?

   A: `"r+"`

7. **Q:** Which statement ensures a file is closed automatically?

   A: `with open(...) as f:`

8. **Q:** Which keyword begins an exception handling block?

   A: `try`

9. **Q:** Which keyword catches an exception?

   A: `except`

10. **Q:** Which keyword runs code no matter what (error or not)?

    A: `finally`

MCQs (20) with Answers & Hints

Easy (5)

1. Which keyword defines a class?

   Hint: Think blueprint keyword.

   A) class

   B) def

   C) object

   D) new

   Answer: A

2. What is `self` in a method?

   Hint: Points to the current object.

   A) Refers to module

   B) Refers to the object instance

   C) Refers to class

   D) Refers to parent

   Answer: B

3. Which method runs when object is created?

   Hint: Constructor.

   A) start()

   B) new()

   C) **init**()

   D) begin()

   Answer: C

4. What does `@staticmethod` mean?

   Hint: Doesn't take self or cls.

   A) Bound to object

   B) Bound to class

   C) No binding needed

   D) Creates static variable

   Answer: C

5. Which block handles exceptions?

   Hint: It "catches" errors.

   A) try

   B) except

   C) finally

D) raise

Answer: B

Medium (5)

6. Which is true about inheritance?

   Hint: Parent → Child.

   A) Child class can't override parent method

   B) Parent inherits from child

   C) Child inherits methods/attributes from parent

   D) Inheritance not supported in Python

   Answer: C

7. What does this code print?

```
class A: pass
a = A()
print(isinstance(a, A))
```

Hint: Object is instance of class.

A) False

B) True

C) Error

D) None

Answer: B

8. Which is the parent of all classes in Python?

   Hint: Everything inherits from it.

   A) class

   B) object

   C) base

   D) None

   Answer: B

9. Which block always runs, regardless of error?

   Hint: Cleans up after try-except.

   A) try

   B) except

   C) else

   D) finally

   Answer: D

10. Which raises an exception manually?

    Hint: Keyword in Python.

    A) throw

    B) raise

    C) except

    D) error

    Answer: B

Hard (10)

11. What is output?

```
class A:
    def show(self): print("A")
class B(A):
    def show(self): print("B")
obj = B()
obj.show()
```

Hint: Child overrides parent.

A) A

B) B

C) Error

D) None

Answer: B

12. Which of these defines a custom exception?

Hint: Must inherit Exception.

A) class MyError: pass

B) class MyError(Exception): pass

C) def MyError(): pass

D) raise MyError

Answer: B

13. What does polymorphism allow?

Hint: Same method, different behaviour.

A) Using multiple classes with same method names

B) Using only one method per class

C) Changing variable type

D) Writing multiple init methods

Answer: A

14. Which method creates object without calling **init**?

    Hint: Lower-level constructor.

    A) **new**

    B) **create**

    C) **obj**

    D) **class**

    Answer: A

15. What happens if error is not handled?

    Hint: Exceptions stop program.

    A) Program ignores it

    B) Program crashes

    C) Error gets logged silently

    D) Nothing

    Answer: B

16. Which OOP concept allows replacing large if/elif with class hierarchy?

    Hint: Override methods instead of ifs.

    A) Inheritance + Polymorphism

    B) Encapsulation

    C) Abstraction

    D) Composition

    Answer: A

17. Which prints “Meow”?

```
class Animal:
    def speak(self): print("Generic")
class Cat(Animal):
    def speak(self): print("Meow")
a = Cat(); a.speak()
```

Hint: Child overrides.

A) Generic

B) Meow

C) Error

D) None

Answer: B

18. What’s the output?

```
try:
    x=5/0
except:
    print("Error")
else:
    print("No error")
```

Hint: Exception goes to except.

A) Error

B) No error

C) Error + No error

D) Crash

Answer: A

19. Why use custom exceptions?

Hint: Custom messages help clarity.

A) To confuse users

B) For meaningful, specific error handling

C) To replace built-in errors

D) To avoid debugging

Answer: B

20. Which method defines behaviour for `str(obj)`?

Hint: User-friendly string representation.

A) **repr**

B) **str**

C) **format**

D) **print**

Answer: B

Chapter 6: File Handling and Exceptions

Full Notes

In Python, you can read and write files easily by using the `open()` function. This allows your programs to store information permanently, instead of losing data each time the program stops. Exception handling helps you prevent your program from crashing when an unexpected error occurs, such as a missing file or invalid input.

Opening Files

The `open()` function is used to open a file. It requires the file name and the mode of operation.

insert code example below

Example:

```
f = open("data.txt", "r")
```

Common Modes:

Mode	Description
`"r"`	Opens the file for reading. The file must already exist.
`"w"`	Opens the file for writing. If the file exists, it will be overwritten.
`"a"`	Opens the file for appending. New data will be added to the end of the file.
`"r+"`	Opens the file for both reading and writing.
`"b"`	Used with other modes for binary files, for example `"rb"` or `"wb"`.

Reading Files

You can read the contents of a file using the `read()` method:

insert code example below

```
f = open("data.txt", "r")
content = f.read()
print(content)
f.close()
```

Other useful methods:

- `f.readline()` reads a single line.
- `f.readlines()` reads all lines and returns them as a list.

Writing to Files

The `"w"` mode allows you to write data to a file. If the file does not exist, it will be created. If it does exist, the contents will be replaced.

insert code example below

```
f = open("data.txt", "w")
f.write("Hello World!\n")
f.close()
```

If you want to add new data without deleting the old data, use the `"a"` mode:

insert code example below

```
with open("log.txt", "a") as f:
    f.write("New entry added.\n")
```

Using "with open()"

The `with` statement is a better and safer way to handle files. It automatically closes the file once you are done, even if an error occurs.

insert code example below

Example:

```
with open("notes.txt", "r") as f:
    data = f.read()
print("File closed automatically.")
```

Working with Binary Files

Binary files store data such as images or audio. To work with them, use binary mode (`"rb"` or `"wb"`).

insert code example below

```
with open("image.png", "rb") as f:
    bytes_data = f.read()
```

Exception Handling

When a Python program faces an error, it normally stops running and shows an error message. This can be avoided using exception handling. The `try` and `except` blocks allow you to "catch" and handle errors gracefully.

insert code example below

Example:

```
try:
    f = open("missing.txt")
except FileNotFoundError:
    print("The file does not exist.")
```

Handling Multiple Exceptions

You can catch different types of errors in the same `try` block by separating them with commas.

insert code example below

```
try:
    x = int("hi")
except (ValueError, TypeError) as e:
    print("Error:", e)
```

Using else and finally

- The **else** block runs only when no errors occur.
- The **finally** block always runs, even if there is an error.

insert code example below

Example:

```
try:
    f = open("data.txt")
except FileNotFoundError:
    print("File missing!")
else:
    print("File opened successfully.")
finally:
    print("Operation completed.")
```

Raising Exceptions Manually

You can create your own error message using the `raise` keyword.

insert code example below

```
x = -1
if x < 0:
    raise ValueError("x cannot be negative")
```

Creating Custom Exceptions

You can create your own exception class by inheriting from the `Exception` class.

insert code example below

```
class MyError(Exception):
    pass

raise MyError("A custom error occurred.")
```

Flashcards (10)

1. **Q:** Which function is used to open a file?

   A: `open()`

2. **Q:** What does the `"r"` mode mean?

   A: It opens a file for reading.

3. **Q:** What happens when you open a file in `"w"` mode?

   A: The file is overwritten or created if it does not exist.

4. **Q:** Which keyword automatically closes a file after use?

   A: `with`

5. **Q:** Which method reads one line from a file?

   A: `readline()`

6. **Q:** What keyword starts an exception-handling block?

   A: `try`

7. **Q:** Which keyword handles an exception if it occurs?

   A: `except`

8. **Q:** Which block always runs, even if there is an error?

   A: `finally`

9. **Q:** How do you raise an exception manually?

   A: By using the `raise` keyword.

10. **Q:** How do you create a custom exception class?

    A: By subclassing the `Exception` class.

Multiple-Choice Questions (20)

Easy (5)

1. Which function is used to open a file?

   A) `file()`

   B) `open()`

   C) `read()`

   D) `load()`

   Answer: B *The `open()` function is used for file operations.*

2. Which mode adds data to the end of a file?

   A) `"r"`

   B) `"w"`

   C) `"a"`

D) `"x"`

Answer: C *The append mode adds new data without deleting existing content.*

3. Which block handles errors?

   A) `catch`

   B) `except`

   C) `error`

   D) `handle`

   Answer: B *Python uses `try` and `except` for error handling.*

4. Which method reads all lines from a file into a list?

   A) `read()`

   B) `readline()`

   C) `readlines()`

   D) `lines()`

   Answer: C *The `readlines()` method returns a list of all lines.*

5. What does the `"w"` mode do to an existing file?

   A) Reads it

   B) Appends to it

   C) Overwrites it

D) Deletes it

Answer: C *Write mode replaces the file's content.*

Medium (5)

6. What will this code print?

insert code example below

```
with open("log.txt", "a") as f:
    f.write("Hi\n")
print(f.closed)
```

A) False

B) True

C) Error

D) None

Answer: B *The file is closed automatically when the block ends.*

7. What happens if you try to open a non-existent file in `"r"` mode?

A) It creates the file

B) It raises an error

C) It returns None

D) It opens an empty file

Answer: B *A `FileNotFoundError` is raised.*

8. What does the `finally` block do?

   A) Runs only when there is an error B) Runs only when no error occurs

   C) Always runs

   D) Never runs

   Answer: C *The `finally` block always executes.*

9. Which statement correctly raises an exception?

   A) `raise "Error"`

   B) `raise Exception("Error")`

   C) `error("Error")`

   D) `throw("Error")`

   Answer: B *The correct syntax uses the `raise` keyword with an Exception object.*

10. Which statement automatically closes a file?

    A) `f.close()`

    B) `auto.open()`

    C) `with open(...) as f:`

    D) `readfile()`

    Answer: C *The `with` statement ensures proper file closure.*

Hard (10)

11. Which mode allows reading and writing of a binary file?

    A) `"rb+"`

    B) `"r+"`

    C) `"wb"`

    D) `"a+"`

    Answer: A *The "+" allows both reading and writing; "b" means binary.*

12. What is the output of this code?

insert code example below

```
try:
    print(1/0)
except ZeroDivisionError:
    print("Oops")
else:
    print("Done")
```

A) Oops

B) Done

C) Oops then Done

D) Program crash

Answer: A *The exception is caught, and "Oops" is printed.*

13. Which exception occurs when executing `int("abc")`?

    A) TypeError

B) NameError

C) ValueError

D) SyntaxError

Answer: C *You cannot convert non-numeric text to an integer.*

14. In `except Exception as e`, what does `e` represent?

A) The line number

B) The exception message

C) The error variable

D) Nothing

Answer: C *It stores the caught exception object.*

15. Which is the correct way to define a custom exception?

A) `class MyError: pass`

B) `class MyError(Exception): pass`

C) `def MyError(): pass`

D) `MyError = Error()`

Answer: B *It must inherit from the `Exception` class.*

16. What will this code output?

insert code example below

```
try:
    f = open("x.txt", "r")
finally:
    print("Check done")
```

A) Only “Check done”

B) Error followed by “Check done”

C) No output

D) Program crash

Answer: B *The `finally` block runs even after an error.*

17. Which keyword is used to manually trigger an error?

    A) `throw`

    B) `raise`

    C) `except`

    D) `error`

    Answer: B *The `raise` keyword is used for this purpose.*

18. Which of the following statements is true about file handling?

    A) Files are automatically closed when using `with`

    B) Files never close

    C) Manual close is mandatory

    D) `open()` is deprecated

Answer: A *Using `with` ensures automatic closure.*

19. What does this code do?

insert code example below

```
try:
    x = 1 / 0
except:
    pass
```

A) Prints the error message

B) Ignores the error C

) Raises the error again

D) Logs the error

Answer: B *The `pass` statement does nothing, effectively ignoring the error.*

20. Why should you handle exceptions during file operations?

A) To prevent the program from crashing when files are missing or unreadable

B) To make the code faster

C) To save memory

D) To delete unused files

Answer: A *Exception handling ensures the program runs smoothly even when errors occur.*

Chapter 7: Using Python Libraries (Math, Random, and NumPy)

Full Notes

Python has many built-in and external libraries that make coding easier and more powerful. A **library** is a collection of ready-made functions and modules that help you perform tasks without writing all the code yourself.
You can import these libraries using the `import` statement.

Using the math Library

The `math` library provides mathematical functions such as square roots, powers, trigonometric functions, and constants like π (pi) and e.

insert code example below

Example:

```
import math

print(math.sqrt(16))      # 4.0
print(math.pow(2, 3))     # 8.0
print(math.pi)            # 3.141592653589793
print(math.ceil(4.2))     # 5
print(math.floor(4.8))    # 4
```

Common math functions:

Function	Description	Example
`math.sqrt(x)`	Returns the square root	√25 → 5
`math.pow(a, b)`	Raises a to the power of b	2^3 → 8
`math.floor(x)`	Rounds down to nearest integer	4.8 → 4
`math.ceil(x)`	Rounds up to nearest integer	4.2 → 5
`math.pi`	Constant for π (3.14159)	

Function	Description	Example
`math.e`	Constant for Euler's number (2.718)	

Using the random Library

The `random` library is used to generate random numbers, shuffle data, and pick random items. It is useful in games, simulations, and data sampling.

insert code example below

Example:

```
import random

print(random.random())          # Random float between 0 and 1
print(random.randint(1, 10))    # Random integer between 1 and 10
print(random.choice(['red', 'green', 'blue']))    # Randomly picks one
element
```

Common random functions:

Function	Description
`random.random()`	Returns a random float between 0 and 1
`random.randint(a, b)`	Returns a random integer between a and b
`random.choice(list)`	Picks a random element from a list
`random.shuffle(list)`	Randomly shuffles the items in a list
`random.uniform(a, b)`	Returns a random float between a and b

insert code example below

Example:

```
colors = ['red', 'green', 'blue']
random.shuffle(colors)
print(colors)
```

Using the NumPy Library

`NumPy` (Numerical Python) is a powerful library for numerical computation and array manipulation. It is widely used in data science, AI, and machine learning.

To use NumPy, it must first be installed using the command:

insert code example below

```
pip install numpy
```

Then import it:

insert code example below

```
import numpy as np
```

Creating Arrays

Arrays in NumPy are similar to lists but faster and more efficient.

insert code example below

```
import numpy as np

arr = np.array([1, 2, 3, 4])
print(arr)
```

Array Operations

NumPy supports mathematical operations directly on arrays:

insert code example below

```
arr = np.array([1, 2, 3])
print(arr + 5)      # [6 7 8]
print(arr * 2)      # [2 4 6]
print(np.sum(arr))  # 6
```

Multi-dimensional Arrays

NumPy supports 2D and 3D arrays:

insert code example below

```
matrix = np.array([[1, 2], [3, 4]])
print(matrix)
print(matrix.shape)    # (2, 2)
print(np.transpose(matrix))  # swaps rows and columns
```

Useful NumPy Functions

Function	Description
`np.zeros(n)`	Creates an array of zeros
`np.ones(n)`	Creates an array of ones
`np.arange(start, stop, step)`	Creates a range of numbers
`np.mean(arr)`	Finds the average value
`np.max(arr)`	Returns the maximum value
`np.min(arr)`	Returns the minimum value

Importing with Aliases

You can rename a library to make it shorter:

insert code example below

```
import numpy as np
import random as rnd
import math as m
```

This makes the code easier to read and type.

Installing External Libraries

Some libraries (like NumPy or Pandas) are not built into Python. You can install them using:

insert code example below

```
pip install library_name
```

Example:

insert code example below

```
pip install numpy
```

Flashcards (10)

1. **Q:** What is a library in Python?

   A: A library is a collection of pre-written code that provides additional functionality.

2. **Q:** Which keyword is used to include a library in Python?

   A: `import`

3. **Q:** What does `math.sqrt(25)` return?

   A: 5.0

4. **Q:** Which function gives a random number between 0 and 1?

   A: `random.random()`

5. **Q:** What does `random.randint(1, 10)` do?

   A: It returns a random integer between 1 and 10.

6. **Q:** What is NumPy mainly used for?

   A: Numerical operations and handling arrays.

7. **Q:** How do you import NumPy with an alias?

   A: `import numpy as np`

8. **Q:** Which NumPy function creates an array of zeros?

   A: `np.zeros(n)`

9. **Q:** Which math function rounds a number up?

   A: `math.ceil()`

10. **Q:** Which command installs external Python libraries?

    A: `pip install library_name`

Multiple-Choice Questions (20)

Easy (5)

1. Which function gives the square root of a number?

   A) `sqrt()`

   B) `math.sqrt()`

   C) `square()`

   D) `root()`

   Answer: B *The `math` library provides the `sqrt()` function.*

2. What is the purpose of the random module?

   A) File operations

   B) Generating random numbers

   C) Sorting data

   D) Creating classes

Answer: B *The `random` module generates random values.*

3. Which command installs NumPy?

   A) `install numpy`

   B) `import numpy`

   C) `pip install numpy`

   D) `setup numpy`

   Answer: C *Use pip to install external libraries.*

4. What is the result of `math.floor(4.9)`?

   A) 5

   B) 4.9

   C) 4

   D) 3

   Answer: C *The floor function rounds down.*

5. What does `random.choice(['a', 'b', 'c'])` return?

   A) Always 'a'

   B) One random element

   C) The whole list

   D) A number

Answer: B *It picks one random element.*

Medium (5)

6. Which of the following gives a random integer between 5 and 15?

   A) `random.number(5,15)`

   B) `random.randint(5,15)`

   C) `random.range(5,15)`

   D) `rand(5,15)`

   Answer: B

7. Which function returns the average of a NumPy array?

   A) `np.sum()`

   B) `np.avg()`

   C) `np.mean()`

   D) `np.total()`

   Answer: C

8. Which line correctly imports NumPy with alias np?

   A) `import np as numpy`

   B) `from numpy import np`

   C) `import numpy as np`

D) `import numpy()`

Answer: C

9. What does `np.arange(1, 6)` produce?

   A) `[1, 2, 3, 4, 5]`

   B) `[1, 2, 3, 4, 5, 6]`

   C) `[0, 1, 2, 3, 4, 5]`

   D) `[2, 3, 4, 5, 6]`

   Answer: A

10. What does `math.pi` represent?

    A) The power function

    B) The circle's radius

    C) The constant π

    D) The value of e

    Answer: C

Hard (10)

11. What does `np.transpose(matrix)` do?

    A) Rotates the matrix

B) Swaps rows and columns

C) Reverses data

D) Sorts the matrix

Answer: B

12. Which function shuffles a list randomly?

A) `random.mix()`

B) `random.shuffle()`

C) `random.sort()`

D) `shuffle.random()`

Answer: B

13. Which NumPy function returns the maximum value of an array?

A) `np.max()`

B) `np.high()`

C) `np.maximum()`

D) `np.maximum_value()`

Answer: A

14. What is the output of `math.pow(2, 4)`?

A) 8

B) 16.0

C) 16

D) 4

Answer: B

15. What is the result of `np.ones(3)`?

A) `[0, 0, 0]`

B) `[1, 1, 1]`

C) `[1.0, 1.0, 1.0]`

D) `[1, 0, 1]`

Answer: C

16. What does `random.uniform(1, 5)` do?

A) Returns random integer

B) Returns random float between 1 and 5

C) Always 1

D) Always 5

Answer: B

17. Which statement correctly rounds 3.2 to 4?

A) `math.floor(3.2)`

B) `math.ceil(3.2)`

C) `round.up(3.2)`

D) `ceil(3.2)`

Answer: B

18. Which NumPy function creates a list of evenly spaced numbers?

A) `np.linspace()`

B) `np.spread()`

C) `np.equal()`

D) `np.fill()`

Answer: A

19. What is the benefit of using NumPy arrays over lists?

A) Slower execution

B) Simpler syntax only

C) Faster and more efficient for calculations

D) Easier file handling

Answer: C

20. What is the output of:

insert code example below

```
import random
print(random.randint(1, 1))
```

A) Error

B) 0

C) 1

D) Random between 0–1

Answer: C

Chapter 8: Advanced Object-Oriented Programming (OOP)

Full Notes

Object-Oriented Programming (OOP) helps organize code by grouping data and behaviors into reusable structures called **classes**.
Objects are created from these classes and can interact with each other, just like real-world entities.

In this chapter, we will look deeper into advanced OOP concepts such as **inheritance**, **polymorphism**, **encapsulation**, and **abstraction**.

---

1. Classes and Objects

A **class** is a blueprint for creating objects. It defines properties (variables) and methods (functions).
An **object** is an instance of a class — a real "thing" based on that blueprint.

insert code example below

Example:

```
class Dog:
    def __init__(self, name):
        self.name = name

    def bark(self):
        print(f"{self.name} says Woof!")

dog1 = Dog("Buddy")
dog1.bark()
```

Output:

```
Buddy says Woof!
```

---

2. Inheritance

Inheritance allows one class to use the properties and methods of another. The new class (called the child class) inherits from the parent class.

insert code example below

Example:

```
class Animal:
    def speak(self):
        print("This animal makes a sound.")

class Dog(Animal):
    def speak(self):
        print("The dog barks.")

d = Dog()
d.speak()
```

Here, `Dog` inherits from `Animal` but overrides the `speak()` method.

Benefits of Inheritance:

- Reduces code repetition.
- Makes it easier to add new types without rewriting old code.

---

3. Polymorphism

Polymorphism means “many forms.” It allows different objects to use the same method name but behave differently.

insert code example below

Example:

```
class Cat:
    def sound(self):
```

```
        print("Meow")

class Cow:
    def sound(self):
        print("Moo")

for animal in [Cat(), Cow()]:
    animal.sound()
```

Even though both classes use the same method name `sound()`, each behaves differently.

---

4. Encapsulation

Encapsulation means hiding the internal details of an object and only exposing what is necessary.
It helps protect data from being modified accidentally.

insert code example below

Example:

```
class BankAccount:
    def __init__(self, balance):
        self.__balance = balance  # private variable

    def deposit(self, amount):
        self.__balance += amount

    def get_balance(self):
        return self.__balance

account = BankAccount(100)
account.deposit(50)
print(account.get_balance())  # 150
```

The double underscore __ before a variable makes it private and cannot be accessed directly outside the class.

---

5. Abstraction

Abstraction means showing only the essential features of an object while hiding the unnecessary details.
It can be achieved using abstract classes and methods.

insert code example below

Example:

```
from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def area(self):
        pass

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius * self.radius

c = Circle(5)
print(c.area())
```

The class `Shape` is abstract and cannot be used directly. The `Circle` class inherits from it and implements the `area()` method.

---

6. Class Methods and Static Methods

- **Instance methods** use `self` and belong to an object.
- **Class methods** use `cls` and belong to the class itself.
- **Static methods** don't use `self` or `cls`; they act like normal functions inside a class.

insert code example below

Example:

```
class Example:
    @classmethod
    def show_class(cls):
        print("This is a class method.")

    @staticmethod
    def greet():
        print("Hello from static method!")

Example.show_class()
Example.greet()
```

---

7. The super() Function

`super()` is used to call methods from the parent class.

insert code example below

Example:

```
class Animal:
    def __init__(self, name):
        self.name = name

class Dog(Animal):
    def __init__(self, name, breed):
        super().__init__(name)
        self.breed = breed

dog = Dog("Buddy", "Beagle")
print(dog.name, dog.breed)
```

Output:

```
Buddy Beagle
```

---

8. Dunder (Magic) Methods

"Dunder" means "double underscore." These are special methods that begin and end with __, like `__init__` or `__str__`.

insert code example below

Example:

```
class Person:
    def __init__(self, name):
        self.name = name

    def __str__(self):
        return f"My name is {self.name}"

p = Person("Ali")
print(p)
```

Output:

```
My name is Ali
```

---

9. Composition

Composition means that one class contains an object of another class. It represents a "has-a" relationship.

insert code example below

Example:

```
class Engine:
    def start(self):
        print("Engine started")

class Car:
    def __init__(self):
        self.engine = Engine()

    def start(self):
        self.engine.start()
        print("Car is running")

my_car = Car()
my_car.start()
```

Flashcards (10)

1. **Q:** What is a class in Python?

   A: A class is a blueprint used to create objects.

2. **Q:** What is inheritance?

   A: It allows a class to reuse and extend another class's code.

3. **Q:** What is polymorphism?

   A: It allows different classes to have methods with the same name but different behavior.

4. **Q:** What does encapsulation mean?

   A: It means protecting data by restricting direct access to it.

5. **Q:** What is abstraction?

A: Showing only important details while hiding internal implementation.

6. **Q:** What does the `super()` function do?

   A: It calls a method from the parent class.

7. **Q:** What is a static method?

   A: A method that does not depend on the object or class instance.

8. **Q:** What is a dunder method?

   A: A special built-in method with double underscores, like `__init__()`.

9. **Q:** How do you make a variable private in a class?

   A: By prefixing it with double underscores (e.g., `__balance`).

10. **Q:** What is composition?

    A: When a class contains an object of another class.

Multiple-Choice Questions (20)

Easy (5)

1. Which keyword defines a class in Python?

   A) `object`

   B) `class`

   C) `define`

   D) `new`

Answer: B

2. Which function is automatically called when an object is created?

   A) `start()`

   B) `run()`

   C) `__init__()`

   D) `begin()`

   Answer: C

3. What is an object?

   A) A built-in function

   B) A variable

   C) An instance of a class

   D) A method

   Answer: C

4. Which concept allows one class to inherit from another?

   A) Polymorphism

   B) Inheritance

   C) Abstraction

   D) Encapsulation

Answer: B

5. What keyword hides a variable in Python?

   A) _

   B) __

   C) `hide`

   D) `lock`

   Answer: B

Medium (5)

6. What is the purpose of `super()` in a subclass?

   A) To skip parent class

   B) To call the parent's method or constructor

   C) To delete the parent

   D) To define a child method

   Answer: B

7. What will this code print?

insert code example below

```
class A:
    def show(self): print("A")
```

```
class B(A):
    def show(self): print("B")

obj = B()
obj.show()
```

A) A

B) B

C) Error

D) None

Answer: B

8. What happens when you try to instantiate an abstract class?

   A) It runs normally

   B) It raises an error

   C) It returns None

   D) It prints “abstract”

   Answer: B

9. What does `__str__()` do in a class?

   A) Creates a string variable

   B) Returns a string representation of the object

   C) Deletes an object

   D) Prints an error

Answer: B

10. What is true about class methods?

    A) They use `self`

    B) They use `cls`

    C) They need no parameters

    D) They can't be used

    Answer: B

Hard (10)

11. What is polymorphism mainly used for?

    A) To rename variables

    B) To allow the same method name with different behaviors

    C) To hide code

    D) To stop inheritance

    Answer: B

12. What does the following print?

insert code example below

```
class A:
    def f(self): print("A")
```

```
class B(A):
    def f(self): print("B")

x = A(); y = B()
for obj in (x, y): obj.f()
```

A) A B

B) B A

C) Error

D) A A

Answer: A

13. Which concept ensures data protection in OOP?

    A) Abstraction

    B) Polymorphism

    C) Encapsulation

    D) Inheritance

    Answer: C

14. What will happen here?

insert code example below

```
class A:
    pass
a = A()
print(a)
```

A) Prints “A”

B) Prints object memory location

C) Prints “None”

D) Error

Answer: B

15. What is the output?

insert code example below

```
class Engine:
    def start(self):
        print("Engine started")

class Car:
    def __init__(self):
        self.engine = Engine()

car = Car()
car.engine.start()
```

A) Engine started

B) Car started

C) Error

D) None

Answer: A

16. Which decorator defines a static method?

A) `@static`

B) `@staticmethod`

C) `@class`

D) `@static_func`

Answer: B

17. What will this output?

insert code example below

```
class A:
    def __str__(self):
        return "Custom"
a = A()
print(a)
```

A) Custom

B) Error

C) A()

D) None

Answer: A

18. Which is not an OOP concept?

A) Polymorphism

B) Encapsulation

C) Compilation

D) Abstraction

Answer: C

19. What is the relationship represented by composition?

A) is-a

B) has-a

C) inherits-from

D) part-of

Answer: B

20. What will this code do?

insert code example below

```
class A:
    pass

class B(A):
    pass

print(issubclass(B, A))
```

A) True

B) False

C) Error

D) None

Answer: A

Chapter 9: Working with Data (JSON, CSV, and APIs)

Full Notes

Modern Python programs often need to **store**, **read**, and **exchange** data between different systems or applications.
The most common formats for this purpose are **JSON (JavaScript Object Notation)**, **CSV (Comma-Separated Values)**, and **APIs (Application Programming Interfaces)**.

These tools allow Python programs to communicate with files, databases, or web servers efficiently.

---

1. Working with JSON

JSON is a lightweight data format used to store and exchange information.
It looks similar to a Python dictionary, with key-value pairs.

insert code example below

Example of JSON data:

```
{
  "name": "Ali",
  "age": 20,
  "city": "Kuala Lumpur"
}
```

Reading JSON in Python

The `json` module allows you to load and save JSON data easily.

insert code example below

Example:

```
import json

# Reading JSON data from a file
```

```
with open("data.json", "r") as file:
    data = json.load(file)

print(data["name"])
```

Explanation:

`json.load()` converts JSON text into a Python dictionary.

Writing JSON in Python

You can also write data to a JSON file:

insert code example below

```
import json

student = {"name": "Aida", "age": 22, "course": "AI"}

with open("student.json", "w") as file:
    json.dump(student, file, indent=4)
```

Explanation:

`json.dump()` converts a Python dictionary into JSON format and saves it to a file.

Converting Between Python and JSON Strings

You can convert Python objects into JSON strings using `json.dumps()`, and convert JSON strings back to Python using `json.loads()`.

insert code example below

Example:

```
import json

data = {"name": "John", "age": 30}
```

```
json_str = json.dumps(data)
print(json_str)  # '{"name": "John", "age": 30}'

python_data = json.loads(json_str)
print(python_data["name"])  # John
```

---

2. Working with CSV Files

CSV files store data in rows and columns separated by commas. They are commonly used for spreadsheets and databases.

insert code example below

Example (data.csv):

```
Name,Age,Course
Ali,21,AI
Aida,22,Data Science
```

Reading CSV Files

You can use the `csv` module to read CSV files easily.

insert code example below

Example:

```
import csv

with open("data.csv", "r") as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

Explanation:

Each line is read as a list of values.

Writing CSV Files

insert code example below

```
import csv

with open("output.csv", "w", newline="") as file:
    writer = csv.writer(file)
    writer.writerow(["Name", "Age"])
    writer.writerow(["Ali", 21])
```

Explanation:

The `writerow()` function writes one row at a time into the CSV file.

Using DictReader and DictWriter

These classes allow reading and writing CSV files as dictionaries.

insert code example below

Example:

```
import csv

with open("students.csv", "r") as file:
    reader = csv.DictReader(file)
    for row in reader:
        print(row["Name"], row["Age"])
```

---

3. Working with APIs

An **API** (Application Programming Interface) allows programs to interact with other software or online services.

For example, a weather API can provide weather information when requested by your program.

To access APIs, Python uses the **requests** library (needs installation):

insert code example below

```
pip install requests
```

Example:

insert code example below

```
import requests

response = requests.get("https://api.github.com")
print(response.status_code)  # 200 means success
print(response.json())       # Converts JSON response to Python dict
```

Sending Data to an API

You can send data to APIs using the `POST` method:

insert code example below

```
import requests

data = {"username": "Ali", "password": "12345"}
response = requests.post("https://example.com/login", data=data)
print(response.status_code)
```

Explanation:

`requests.post()` sends data to a web service. If the API returns a response, you can access it using `.json()` or `.text`.

Common HTTP Response Codes

Code Meaning

200 OK – The request succeeded

Code Meaning

404 Not Found – The resource doesn't exist

401 Unauthorized – Login required

500 Internal Server Error – Problem on the server side

Summary

- **JSON** → Used for structured data and easy exchange between systems.
- **CSV** → Used for tables, rows, and columns.
- **APIs** → Allow data to be retrieved or sent over the internet. Together, these tools help Python interact with the real world beyond your computer.

Flashcards (10)

1. **Q:** What does JSON stand for?

   A: JavaScript Object Notation.

2. **Q:** Which module is used to handle JSON in Python?

   A: The `json` module.

3. **Q:** What function reads JSON data from a file?

   A: `json.load()`

4. **Q:** What function writes Python data to a JSON file?

   A: `json.dump()`

5. **Q:** What is a CSV file used for?

   A: Storing data in a table format with commas separating values.

6. **Q:** Which module is used to handle CSV files?

A: The `csv` module.

7. **Q:** What function is used to make API requests?

   A: `requests.get()` or `requests.post()`

8. **Q:** What does an HTTP status code of 200 mean?

   A: The request was successful.

9. **Q:** How do you convert a Python dictionary into a JSON string?

   A: Using `json.dumps()`.

10. **Q:** Which library allows communication with web APIs?

    A: The `requests` library.

Multiple-Choice Questions (20)

Easy (5)

1. What is the full form of JSON?

   A) Java Syntax Object Notation

   B) JavaScript Object Notation

   C) Joint System Output Node

   D) Java Source Output Name

   Answer: B

2. Which function converts JSON data into a Python dictionary?

A) `json.load()`

B) `json.dumps()`

C) `json.write()`

D) `json.export()`

Answer: A

3. What separates values in a CSV file?

A) Tabs

B) Spaces

C) Commas

D) Semicolons

Answer: C

4. Which module is used to handle CSV data?

A) `csv`

B) `excel`

C) `spreadsheet`

D) `file`

Answer: A

5. What does an API do?

A) Edits files

B) Connects software systems

C) Runs loops

D) Handles only local data

Answer: B

Medium (5)

6. Which command installs the requests module?

   A) `import requests`

   B) `install requests`

   C) `pip install requests`

   D) `setup requests`

   Answer: C

7. What does `json.dump()` do?

   A) Loads JSON into a variable

   B) Writes Python data to a JSON file

   C) Deletes JSON data

   D) Prints a JSON string

Answer: B

8. What is the purpose of `csv.DictReader()`?

   A) Reads CSV data as dictionaries

   B) Reads only numeric data

   C) Writes CSV rows

   D) Formats text

   Answer: A

9. What does a 404 HTTP code mean?

   A) Success

   B) Not Found

   C) Unauthorized

   D) Error on client side

   Answer: B

10. Which of the following returns JSON from an API response?

    A) `response.text()`

    B) `response.data()`

    C) `response.json()`

    D) `response.load()`

Answer: C

Hard (10)

11. What does `json.dumps()` return?

    A) A Python dictionary

    B) A JSON string

    C) A list

    D) None

    Answer: B

12. What will happen if you open a non-existent CSV file in read mode?

    A) It will create a new file

    B) It will raise an error

    C) It will skip it

    D) It will write a blank row

    Answer: B

13. Which keyword ensures a file closes automatically?

    A) `with`

    B) `as`

C) `close`

D) `finally`

Answer: A

14. Which module must be imported to handle APIs?

A) `http`

B) `web`

C) `requests`

D) `network`

Answer: C

15. What will `csv.writer(file).writerow(["Ali", 21])` do?

A) Append data to the file

B) Write a single row with two values

C) Write column headers

D) Do nothing

Answer: B

16. Which HTTP code shows a server-side error?

A) 200

B) 404

C) 401

D) 500

Answer: D

17. What does the `newline=""` argument in `open()` do for CSV files?

A) Adds blank lines

B) Prevents extra blank lines in output

C) Closes the file early

D) Adds commas automatically

Answer: B

18. What type of data structure does `json.load()` return?

A) List

B) Dictionary

C) Tuple

D) String

Answer: B

19. Which of the following correctly sends data to an API?

A) `requests.post(url, data=data)`

B) `requests.send(data)`

C) `requests.push(url)`

D) `requests.api(url)`

Answer: A

20. What is a common reason to use JSON instead of CSV?

A) JSON is better for structured, nested data

B) CSV is always faster

C) CSV cannot be read

D) JSON is simpler but less readable

Answer: A

Chapter 10: Graphical User Interface (GUI) and Game Basics using Tkinter and Pygame

Full Notes

Python is not limited to text-based programs — you can also build visual applications and games.
Two popular tools for this are **Tkinter**, which is used for creating Graphical User Interfaces (GUIs), and **Pygame**, which is used for making simple 2D games.

---

1. Introduction to GUI

A **Graphical User Interface (GUI)** allows users to interact with programs using buttons, text boxes, menus, and other visual elements instead of typing commands.

In Python, the most common library for GUI development is **Tkinter**. Tkinter comes built into Python, so no installation is needed.

insert code example below

To start:

```
import tkinter as tk

window = tk.Tk()
window.title("My First GUI")
window.geometry("300x200")

label = tk.Label(window, text="Hello, World!")
label.pack()

window.mainloop()
```

Explanation:

- `tk.Tk()` creates the main window.
- `Label` displays text.

- `pack()` arranges the widget on the screen.
- `mainloop()` keeps the window open until the user closes it.

---

2. Common Tkinter Widgets

Widget	Description	Example
`Label`	Displays text or images	`Label(window, text="Hi!")`
`Button`	Adds a clickable button	`Button(window, text="Click")`
`Entry`	Creates a text input field	`Entry(window)`
`Frame`	Organizes other widgets	`Frame(window)`
`Canvas`	Draws shapes or graphics	`Canvas(window)`

insert code example below

Example:

```
import tkinter as tk

def greet():
    print("Button clicked!")

window = tk.Tk()
button = tk.Button(window, text="Click Me", command=greet)
button.pack()
window.mainloop()
```

When the button is clicked, the `greet()` function runs.

---

3. Adding Input and Output

You can collect user input with the `Entry` widget:

insert code example below

```
import tkinter as tk

def show_name():
    name = entry.get()
    label.config(text=f"Hello, {name}!")

window = tk.Tk()
entry = tk.Entry(window)
entry.pack()

label = tk.Label(window)
label.pack()

button = tk.Button(window, text="Submit", command=show_name)
button.pack()

window.mainloop()
```

When the user types their name and clicks the button, the label updates with a greeting.

---

4. Introduction to Pygame

Pygame is a Python library used for developing 2D games. It provides tools to display graphics, handle keyboard and mouse input, and add sound effects.

To install Pygame:

insert code example below

```
pip install pygame
```

---

5. Starting a Basic Game Window

Example:

insert code example below

```
import pygame
pygame.init()

screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption("My First Game")

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    screen.fill((0, 0, 255))  # Fills screen with blue
    pygame.display.update()

pygame.quit()
```

Explanation:

- `pygame.init()` starts Pygame.
- `set_mode()` creates the game window.
- The **game loop** runs continuously until the player exits.
- `event.type == pygame.QUIT` lets you close the window.
- `fill()` changes the background color.
- `update()` refreshes the screen.

---

6. Drawing Shapes

You can draw rectangles, circles, and lines using built-in functions:

insert code example below

```
pygame.draw.rect(screen, (255, 0, 0), (50, 50, 100, 60))  # Red rectangle
pygame.draw.circle(screen, (0, 255, 0), (200, 150), 40)   # Green circle
pygame.display.update()
```

---

7. Handling Keyboard Input

Example:

insert code example below

```
keys = pygame.key.get_pressed()
if keys[pygame.K_LEFT]:
    print("Left arrow pressed")
```

---

8. Displaying Images and Sounds

Images can be loaded using:

insert code example below

```
player = pygame.image.load("player.png")
screen.blit(player, (100, 100))
pygame.display.update()
```

Sound effects can be added using:

insert code example below

```
sound = pygame.mixer.Sound("click.wav")
sound.play()
```

---

9. Game Loop Concept

Every Pygame program runs inside a **main loop**. This loop handles:

1. Events (keyboard, mouse)
2. Updates (movement, collision)
3. Drawing (redraw the screen)

Example (simplified):

insert code example below

```
while running:
    handle_input()
    update_game_objects()
    draw_screen()
```

This makes the game interactive and dynamic.

---

10. Combining GUI and Game Development

You can use **Tkinter** for menus or settings and **Pygame** for the actual gameplay. For example, a Tkinter window could collect player names before starting a Pygame level.

Flashcards (10)

1. **Q:** What is a GUI?

   A: A Graphical User Interface that allows users to interact visually with a program.

2. **Q:** Which library is used for GUI development in Python?

   A: Tkinter.

3. **Q:** What is the function of `mainloop()` in Tkinter?

   A: It keeps the window open and running.

4. **Q:** Which widget is used to display text?

   A: `Label`.

5. **Q:** Which command installs Pygame?

   A: `pip install pygame`.

6. **Q:** What does `pygame.init()` do?

A: It initializes all Pygame modules.

7. **Q:** What is the main purpose of the game loop?

   A: To continuously update and refresh the game window.

8. **Q:** Which Pygame function changes the background color?

   A: `screen.fill((R,G,B))`.

9. **Q:** Which function loads an image in Pygame?

   A: `pygame.image.load()`.

10. **Q:** Which function plays a sound effect in Pygame?

    A: `pygame.mixer.Sound().play()`.

Multiple-Choice Questions (20)

Easy (5)

1. Which Python library is used for GUI applications?

   A) `pygame`

   B) `tkinter`

   C) `pandas`

   D) `math`

   Answer: B

2. What does `window.mainloop()` do in a Tkinter program?

A) Closes the window

B) Keeps the window open

C) Starts the computer

D) Refreshes widgets only

Answer: B

3. Which widget allows user input?

   A) `Label`

   B) `Button`

   C) `Entry`

   D) `Canvas`

   Answer: C

4. What is the function of `pygame.display.update()`?

   A) Loads music

   B) Redraws the game window

   C) Closes the screen

   D) Plays sound

   Answer: B

5. Which method closes a Pygame window safely?

A) `pygame.stop()`

B) `pygame.quit()`

C) `exit()`

D) `close()`

Answer: B

Medium (5)

6. What is the output of this code?

insert code example below

```
import tkinter as tk
win = tk.Tk()
win.title("Hello")
print(type(win))
```

A) Tkinter window object

B) String

C) Error

D) List

Answer: A

7. Which function draws a rectangle in Pygame?

A) `pygame.rect()`

B) `pygame.draw.rect()`

C) `pygame.rectangle()`

D) `pygame.draw()`

Answer: B

8. What does this code do?

insert code example below

```
pygame.display.set_mode((500, 400))
```

A) Creates a game window of 500x400 pixels

B) Draws a rectangle

C) Sets the FPS

D) Creates a sprite

Answer: A

9. In Tkinter, which method arranges widgets on the screen?

   A) `display()`

   B) `arrange()`

   C) `pack()`

   D) `set()`

   Answer: C

10. Which of the following runs continuously in a game?

A) Function

B) Loop

C) Class

D) Event

Answer: B

Hard (10)

11. What does the `command` argument in a Button do?

    A) Calls a function when the button is clicked

    B) Creates a label

    C) Defines text

    D) Starts the game

    Answer: A

12. Which Pygame function checks for key presses?

    A) `pygame.keys()`

    B) `pygame.get_key()`

    C) `pygame.key.get_pressed()`

    D) `pygame.check_key()`

Answer: C

13. Which Pygame event indicates the window was closed?

A) `pygame.CLOSE`

B) `pygame.EXIT`

C) `pygame.QUIT`

D) `pygame.END`

Answer: C

14. What color will this line create?

insert code example below

```
screen.fill((255, 255, 0))
```

A) Blue

B) Yellow

C) Red

D) White

Answer: B

15. Which of these functions loads an image file in Pygame?

A) `pygame.image.load("file.png")`

B) `pygame.load.image("file.png")`

C) `pygame.img("file.png")`

D) `pygame.display.image()`

Answer: A

16. Which of these is a correct way to play a sound?

A) `sound.play()`

B) `pygame.play(sound)`

C) `pygame.mixer(sound)`

D) `pygame.start_sound()`

Answer: A

17. Which part of a game handles inputs and updates the display?

A) Game loop

B) Event handler only

C) Initialization

D) Quit section

Answer: A

18. Which Tkinter widget can display shapes?

A) `Label`

B) `Canvas`

C) `Frame`

D) `Text`

Answer: B

19. Which of the following closes the Pygame safely after running?

A) `pygame.close()`

B) `pygame.end()`

C) `pygame.quit()`

D) `exit()`

Answer: C

20. What happens if `pygame.init()` is not called before creating a window?

A) It still runs

B) Some modules may not work properly

C) It prints “error”

D) It installs missing files

Answer: B