H.W

* Methods :-

1) Format ()

(method) format : (*args : object, **kwargs : object) -> str

- Return a formatted version of s, using substitutions from args and kwargs.
- The substitutions are identified by braces ('{' and '}')

2) Capitalize ()

(method) capitalize : () -> str
s.capitalize() -> str

- Return a capitalized version of s, i.e. make the first character have upper case and the rest lower case.

3) Casefold ()

(method) casefold : () -> str
s.casefold() -> str

- Return a version of s suitable for caseless comparisons.

4) Center ()

(method) center: (-width: supportsIndex, -fillchar: str = ..., /) -> str
S.center(width[, fillchar]) -> str

- Return S centered in a string of length width.
- Padding is done using the specified fill character (default is a space)

5) count ()

(method) count: (x: str, -start: SupportsIndex | None = ..., -end: supportsIndex | None = ...) -> int
S.count(sub[, start[, end]]) -> int

- Return of number of non-overlapping occurences of substring sub in string S[start:end].
- Optional arguments start and end are interpreted as in slice notation.

6) encode ()

(method) encode: (encoding: str = ..., errors: str = ...) -> bytes

s.encode(encoding='utf-8', errors='strict')->bytes

- Encode s using the codec registered for encoding.
- Default is 'strict' meaning that encoding errors raise a UnicodeEncodeError.
- other possible values are 'ignore', 'replace' and 'xmlcharrefreplace' as well as other name registered with codecs.register_error that can handle UnicodeEncodeError.

7) endswith()

(method) endswith: (_suffix: str | tuple[str, ...], _start: supportsIndex | None = ..., _end: supportsIndex | None = ..., /) -> bool
s.endswith(suffix[, start[, end]]) -> bool

- Return True if s ends with the specified suffix, false otherwise
- With optional start, test s begging at that position.
- With optional end, stop comparing s at that position.
- Suffix can also be a tuple of strings to try.

8) expandtabs()

(method) expandtabs: (tabsize: int = ...) -> str
S.expandtabs(tabsize: s) -> str

- Return a copy of S where all tab characters are expanded using spaces.
- If tabsize is not given, a tab size of 8 characters is assumed.

9) find()

(method) find: (_sub: str, _start: SupportsIndex | None = ..., SupportsIndex | None = ..., /) -> int

S.find(sub[, start[, end]]) -> ~~str~~ int

- Return the lowest index in S where substring sub is found,
- such that sub is contained within S[start:end].
- Optional arguments start and end are interpreted as in slice notation.
- Return -1 on failure.

10) index()

(method) index: (_sub: str, _start: SupportsIndex | None = ..., _end: SupportsIndex | None = ..., /) -> int

S.index(sub[, start[, end]]) -> int

- Return the lowest index in S where substring sub is found.
- such that sub is contained within S[start:end].
- Optional arguments start and end are interpreted as in slice notation.
- Raises ValueError when the substring is not found.

11) isalnum()

(method) isalnum: () -> bool
S.isalnum() -> bool

- Return True if all characters in S are alphanumeric and there is at least one character in S, False otherwise.

12) isalpha()

(method) isalpha: () -> bool
S.isalpha() -> bool

- Return the True if all characters in S are alphabetic and there is at least one character in S, False otherwise.

13) isdecimal()

(method) isdecimal : () -> bool
S.isdecimal() -> bool

- Return True if there are only decimal characters in S, false otherwise.

14) isdigit()

(method) isdigit : () -> bool
S.isdigit() -> bool

- Return True if all characters in S are digits and there is at least one character in S, false otherwise.

15) isidentifier()

(method) isidentifier : () -> bool
S.isidentifier() -> bool

- Return True if S is a valid identifier

according to the language definition.
- Use keyword.iskeyword() to test for reserved identifiers such as "def" and "class".

16) islower()

(method) islower : () -> bool
S.islower() -> bool

- Return True if all cased characters in S are lowercase and there is at least one cased character in S, false otherwise.

17) isnumeric()

(method) isnumeric : () -> bool
S.isnumeric() -> bool

- Return True if there are only numeric characters in S, false otherwise.

18) isprintable()

(method) isprintable : () -> bool
S.isprintable() -> bool

- Return True if all characters in S are considered printable in repr() or S is empty false otherwise.

19) join()

(method) join: (_: Iterable: Iterable[str], /) -> str
s.join(iterable) -> str

- Return a string which is the concatenation of the string in the iterable.
- The seperator is s.

20) Ljust()

(method) Ljust: (_width: SupportsIndex, _fillchar: str = ..., /) -> str
s.Ljust(width[, fillchar]) -> str

- Return s left-justified in a Unicode string of length width.
- Padding is done using the specified fill character (default is a space).