zapiski

May 12, 2025

[1]:
```
1 + 1
```

[1]: 2

[2]:
```
5 + 6
```

[2]: 11

[3]:
```
5 * 6
```

[3]: 30

[4]:
```
5+1 * 7
```

[4]: 12

[5]:
```
(5 + 1) * 7
```

[5]: 42

[6]:
```
8 / 2
```

[6]: 4.0

[7]:
```
25 % 4
```

[7]: 1

[8]:
```
26 % 4
```

[8]: 2

[9]:
```
5 ** 2
```

[9]: 25

[10]:
```
2 * 2
```

[10]: 4

[11]:
```
8 / 2
```

[11]: 4.0

[12]:
```
14 / 5
```

[12]: 2.8

[13]:
```
14 // 5
```

[13]: 2

[14]:
```
2.4 / 0.9
```

[14]: 2.6666666666666665

[15]:
```
2.4 // 0.9
```

[15]: 2.0

[16]:
```
9 ** 1000
```

[16]:
```
174787125172265160965997461916466057052906248743518851781188801181068626622727548929148646986468111107560895069614527658877136843587550864751441420209363848187291238008997717938152962847832052351931914268150442405941089021450050064781393581892570190540260548409813795697936855102582523941131864399791652367704476966262864640654033562797532961926424507975047086246247409110544443735530214615147534809075533015326906793309169947988908982465084179556747860639697566455714373765702708040323997775786529684674009371237791577053609422368804910802324413918302796248441107846443951684522796193522126981475341678257645550731607375198537404606459254679604315073780831450168467975805690594875924636864441615186313808527660359581641094515759974207761761891160118515560208077174678595935987949019193338996527127540312792543224796326967591264610315634395437544279268893604704153353752313794131069083394976776429008133390038031040615472315788211244999167381905411044000 1
```

[17]:
```
5 + 3
```

[17]: 8

[18]:
```
_ * 4
```

[18]: 32

[19]:
```
_ + 3
```

[19]:
```
35
```

[20]:
```
_19 + _20
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[20], line 1
----> 1 _19 + _20

NameError: name '_20' is not defined
```

[21]:
```
(8 + 2 * (
    4 + 6 + 8 +
    3 + 4 * 5)
 + 5)
```

[21]:
```
95
```

[22]:
```
ploscina = 5 * 6
```

[23]:
```
visina = 7
```

[24]:
```
prostornina = ploscina * visina
```

[25]:
```
prostornina
```

[25]:
```
210
```

[26]:
```
x = 1
y = 2
```

[27]:
```
x
```

[27]:
```
1
```

[28]:
```
y
```

[28]:
```
2
```

[29]:
```
b = 12
a = b + 1
```

[30]:
```
a = 5
b = a + 1
a = 6
```

[31]:
```
b
```

[31]:
```
6
```

[32]:
```
b = b + 1
```

[33]:
```
b
```

[33]:
```
7
```

[34]:
```
ploščina = 6
```

[35]:
```
ploščina
```

[35]:
```
6
```

[36]:
```
Ploščina
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[36], line 1
----> 1 Ploščina

NameError: name 'Ploščina' is not defined
```

[ ]:
```
     = 16
```

[ ]:
```
  = 3.14
```

[ ]:
```

```

[ ]:
```
stevilo_evglen = 42
```

[ ]:
```
7 = a
```

[ ]:
```
x = 5
y = 7
```

[ ]:
```
if = 6
```

[ ]:
```
x = 5
```

[ ]:
```
x = 6
```

[ ]:
```
abs(-5)
```

[ ]:
```
pow(5, 2)
```

[ ]:
```
sqrt(4)
```

[ ]:
```
from math import *
```

[ ]:
```
sqrt(4)
```

[ ]:
```
sin(3.14)
```

[ ]:
```
pi
```

[ ]:
```
e
```

[ ]:
```
sqrt((5 - 8) * 3 * ploscina) + sin(1)
```

[ ]:
```
1 / (ploščina - 6)
```

[ ]:
```
vreme = "Dež"
```

[ ]:
```
vreme_jutri = 'Dež'
```

[ ]:
```
vreme
```

[ ]:
```
vreme + ploščina
```

[ ]:
```
ploščina + vreme
```

[ ]:
```
napoved = "Danes je "
```

[ ]:
```
napoved + vreme
```

[ ]:
```
vreme * 6
```

[ ]:
```
vreme * 6.5
```

[ ]:
```
vreme * (ploščina / 3)
```

[ ]:
```
ploščina / 3
```

[ ]:
```
vreme * (ploščina // 3)
```

[ ]:
```
print(5 + 2)
```

[ ]:
```
print(vreme)
```

[ ]:
```
print(vreme, 5 + 2, ploščina, sqrt(17))
```

[ ]:
```
x
```

[ ]:
```
print(x + 2)
print(x * 13)
print(x - 4)
```

[ ]:
```
t = input("Temperatura: ")
```

[ ]:
```
t
```

[ ]:
```
c = input("Temperatura: ")
f = float(c) * 9 / 5 + 32
print("To je", f, "Fahrenheitov")
```

[ ]:
```
c * 9 / 5
```

[ ]:
```
while True:
    pass
```

[ ]:
```
str(15)
```

[ ]:
```
float("3.14")
```

[ ]:
```
int("3.14")
```

[ ]:
```
5 + 2
```

[ ]:
```
print("Nekaj")
```

[ ]:
```
teza = float(input("Teža [kg]: "))
visina = float(input("Višina: "))
if visina <= 3:
    print("Priden, vpisal si višino v metrih")
else:
    visina = visina / 100
    print("Prijazno sem pretvoril višino v metre")
bmi = teza / (visina ** 2)
print("BMI:", bmi)
if bmi > 26:
    print("Tole je malo preveč")
if bmi < 12:
    print("Anoreksija")
elif bmi < 18:
    print("Več jest!")
else:
```

```
    print("Le tako naprej")
```

[ ]:
```
print("bla", "bla", "bla")
```

[ ]:
```
5 + 7
```

[ ]:
```
5 < 7
```

[ ]:
```
5 > 7
```

[ ]:
```
x = 5
y = 3
```

[ ]:
```
x > 1 and y > 2
```

[ ]:
```
x > 1
```

[ ]:
```
y > 2
```

[ ]:
```
True and False
```

Če je višina večja od 3 stori naslednje: - deli višino s 100 - povej, da si to storil

[ ]:
```
6 * 7 == 42 or 5934683498498475 % 7 == 3
```

[ ]:
```
6 * 7 == 42 or 1 / 0 > 8
```

[ ]:
```
1 / 0 > 8 or 6 * 7 == 42
```

[ ]:
```
6 * 7 == 42 and 1 / 0 == 2
```

[ ]:
```
n = 2 # 19676545678
izpisanih = 0
najvecje_doslej = n
print(n)
while n != 1:
    if n % 2 == 0:
       n = n // 2
    else:
       n = 3 * n + 1
    if n > najvecje_doslej:
        najvecje_doslej = n
    print(n)
    izpisanih = izpisanih + 1
print("Izpisanih števil:", izpisanih)
print("Največje: ", najvecje_doslej)
```

[ ]:
```
if x > 5:
    print("A")
else:
    if y > 3:
        print("B")
    else:
        print("C")
```

[ ]:
```
if x > 5:
    print("A")
elif y > 3:
    print("B")
else:
    print("C")
```

[38]:
```
s = "1,85"
```

[39]:
```
names = "Ana Berta Cilka Dani"
```

[40]:
```
split(names)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[40], line 1
----> 1 split(names)

NameError: name 'split' is not defined
```

[41]:
```
s.replace(",", ".")
```

[41]:
```
'1.85'
```

[42]:
```
s
```

[42]:
```
'1,85'
```

[44]:
```
names
```

[44]:
```
'Ana Berta Cilka Dani'
```

[45]:
```
names.replace("Berta", "Bertolin")
```

[45]:
```
'Ana Bertolin Cilka Dani'
```

[48]:
```
lower_names = names.lower()
```

[49]:
```
lower_names
```

[49]:
```
'ana berta cilka dani'
```

[51]:
```
lower_names.replace("an", "tu")
```

[51]:
```
'tua berta cilka dtui'
```

[52]:
```
sequence = "acgatagcatcggagactagctagcaggcgtatgcagtcgatgcgatgcagtgactg"
```

[53]:
```
sequence.replace("t", "u")
```

[53]:
```
'acgauagcaucggagacuagcuagcaggcguaugcagucgaugcgaugcagugacug'
```

[55]:
```
sequence.count("t")
```

[55]:
```
11
```

[56]:
```
sequence.count("gg")
```

[56]:
```
2
```

[108]:
```
help(str)
```

```
Help on class str in module builtins:

class str(object)
 |  str(object='') -> str
 |  str(bytes_or_buffer[, encoding[, errors]]) -> str
 |
 |  Create a new string object from the given object. If encoding or
 |  errors is specified, then the object must expose a data buffer
 |  that will be decoded using the given encoding and error handler.
 |  Otherwise, returns the result of object.__str__() (if defined)
 |  or repr(object).
 |  encoding defaults to 'utf-8'.
 |  errors defaults to 'strict'.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return bool(key in self).
 |
 |  __eq__(self, value, /)
 |      Return self==value.
```

```
|
|  __format__(self, format_spec, /)
|      Return a formatted version of the string as described by format_spec.
|
|  __ge__(self, value, /)
|      Return self>=value.
|
|  __getitem__(self, key, /)
|      Return self[key].
|
|  __getnewargs__(self, /)
|
|  __gt__(self, value, /)
|      Return self>value.
|
|  __hash__(self, /)
|      Return hash(self).
|
|  __iter__(self, /)
|      Implement iter(self).
|
|  __le__(self, value, /)
|      Return self<=value.
|
|  __len__(self, /)
|      Return len(self).
|
|  __lt__(self, value, /)
|      Return self<value.
|
|  __mod__(self, value, /)
|      Return self%value.
|
|  __mul__(self, value, /)
|      Return self*value.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __rmod__(self, value, /)
|      Return value%self.
|
|  __rmul__(self, value, /)
|      Return value*self.
|
```

```
 |  __sizeof__(self, /)
 |      Return the size of the string in memory, in bytes.
 |
 |  __str__(self, /)
 |      Return str(self).
 |
 |  capitalize(self, /)
 |      Return a capitalized version of the string.
 |
 |      More specifically, make the first character have upper case and the rest
lower
 |      case.
 |
 |  casefold(self, /)
 |      Return a version of the string suitable for caseless comparisons.
 |
 |  center(self, width, fillchar=' ', /)
 |      Return a centered string of length width.
 |
 |      Padding is done using the specified fill character (default is a space).
 |
 |  count(self, sub[, start[, end]], /)
 |      Return the number of non-overlapping occurrences of substring sub in
string S[start:end].
 |
 |      Optional arguments start and end are interpreted as in slice notation.
 |
 |  encode(self, /, encoding='utf-8', errors='strict')
 |      Encode the string using the codec registered for encoding.
 |
 |      encoding
 |        The encoding in which to encode the string.
 |      errors
 |        The error handling scheme to use for encoding errors.
 |        The default is 'strict' meaning that encoding errors raise a
 |        UnicodeEncodeError.  Other possible values are 'ignore', 'replace' and
 |        'xmlcharrefreplace' as well as any other name registered with
 |        codecs.register_error that can handle UnicodeEncodeErrors.
 |
 |  endswith(self, suffix[, start[, end]], /)
 |      Return True if the string ends with the specified suffix, False
otherwise.
 |
 |      suffix
 |        A string or a tuple of strings to try.
 |      start
 |        Optional start position. Default: start of the string.
 |      end
```

```
 |        Optional stop position. Default: end of the string.
 |
 |  expandtabs(self, /, tabsize=8)
 |      Return a copy where all tab characters are expanded using spaces.
 |
 |      If tabsize is not given, a tab size of 8 characters is assumed.
 |
 |  find(self, sub[, start[, end]], /)
 |      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.
 |
 |  format(self, /, *args, **kwargs)
 |      Return a formatted version of the string, using substitutions from args
and kwargs.
 |      The substitutions are identified by braces ('{' and '}').
 |
 |  format_map(self, mapping, /)
 |      Return a formatted version of the string, using substitutions from
mapping.
 |      The substitutions are identified by braces ('{' and '}').
 |
 |  index(self, sub[, start[, end]], /)
 |      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.
 |
 |  isalnum(self, /)
 |      Return True if the string is an alpha-numeric string, False otherwise.
 |
 |      A string is alpha-numeric if all characters in the string are alpha-
numeric and
 |      there is at least one character in the string.
 |
 |  isalpha(self, /)
 |      Return True if the string is an alphabetic string, False otherwise.
 |
 |      A string is alphabetic if all characters in the string are alphabetic
and there
 |      is at least one character in the string.
 |
 |  isascii(self, /)
 |      Return True if all characters in the string are ASCII, False otherwise.
 |
```

```
 |      ASCII characters have code points in the range U+0000-U+007F.
 |      Empty string is ASCII too.
 |
 |  isdecimal(self, /)
 |      Return True if the string is a decimal string, False otherwise.
 |
 |      A string is a decimal string if all characters in the string are decimal
and
 |      there is at least one character in the string.
 |
 |  isdigit(self, /)
 |      Return True if the string is a digit string, False otherwise.
 |
 |      A string is a digit string if all characters in the string are digits
and there
 |      is at least one character in the string.
 |
 |  isidentifier(self, /)
 |      Return True if the string is a valid Python identifier, False otherwise.
 |
 |      Call keyword.iskeyword(s) to test whether string s is a reserved
identifier,
 |      such as "def" or "class".
 |
 |  islower(self, /)
 |      Return True if the string is a lowercase string, False otherwise.
 |
 |      A string is lowercase if all cased characters in the string are
lowercase and
 |      there is at least one cased character in the string.
 |
 |  isnumeric(self, /)
 |      Return True if the string is a numeric string, False otherwise.
 |
 |      A string is numeric if all characters in the string are numeric and
there is at
 |      least one character in the string.
 |
 |  isprintable(self, /)
 |      Return True if all characters in the string are printable, False
otherwise.
 |
 |      A character is printable if repr() may use it in its output.
 |
 |  isspace(self, /)
 |      Return True if the string is a whitespace string, False otherwise.
 |
 |      A string is whitespace if all characters in the string are whitespace
```

```
and there
 |      is at least one character in the string.
 |
 |  istitle(self, /)
 |      Return True if the string is a title-cased string, False otherwise.
 |
 |      In a title-cased string, upper- and title-case characters may only
 |      follow uncased characters and lowercase characters only cased ones.
 |
 |  isupper(self, /)
 |      Return True if the string is an uppercase string, False otherwise.
 |
 |      A string is uppercase if all cased characters in the string are
uppercase and
 |      there is at least one cased character in the string.
 |
 |  join(self, iterable, /)
 |      Concatenate any number of strings.
 |
 |      The string whose method is called is inserted in between each given
string.
 |      The result is returned as a new string.
 |
 |      Example: '.'.join(['ab', 'pq', 'rs']) -> 'ab.pq.rs'
 |
 |  ljust(self, width, fillchar=' ', /)
 |      Return a left-justified string of length width.
 |
 |      Padding is done using the specified fill character (default is a space).
 |
 |  lower(self, /)
 |      Return a copy of the string converted to lowercase.
 |
 |  lstrip(self, chars=None, /)
 |      Return a copy of the string with leading whitespace removed.
 |
 |      If chars is given and not None, remove characters in chars instead.
 |
 |  partition(self, sep, /)
 |      Partition the string into three parts using the given separator.
 |
 |      This will search for the separator in the string.  If the separator is
found,
 |      returns a 3-tuple containing the part before the separator, the
separator
 |      itself, and the part after it.
 |
 |      If the separator is not found, returns a 3-tuple containing the original
```

```
string
 |      and two empty strings.
 |
 |  removeprefix(self, prefix, /)
 |      Return a str with the given prefix string removed if present.
 |
 |      If the string starts with the prefix string, return
string[len(prefix):].
 |      Otherwise, return a copy of the original string.
 |
 |  removesuffix(self, suffix, /)
 |      Return a str with the given suffix string removed if present.
 |
 |      If the string ends with the suffix string and that suffix is not empty,
 |      return string[:-len(suffix)]. Otherwise, return a copy of the original
 |      string.
 |
 |  replace(self, old, new, /, count=-1)
 |      Return a copy with all occurrences of substring old replaced by new.
 |
 |        count
 |          Maximum number of occurrences to replace.
 |          -1 (the default value) means replace all occurrences.
 |
 |      If the optional argument count is given, only the first count
occurrences are
 |      replaced.
 |
 |  rfind(self, sub[, start[, end]], /)
 |      Return the highest 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.
 |
 |  rindex(self, sub[, start[, end]], /)
 |      Return the highest 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.
 |
 |  rjust(self, width, fillchar=' ', /)
 |      Return a right-justified string of length width.
 |
 |      Padding is done using the specified fill character (default is a space).
 |
 |  rpartition(self, sep, /)
```

```
 |      Partition the string into three parts using the given separator.
 |
 |      This will search for the separator in the string, starting at the end.
If
 |      the separator is found, returns a 3-tuple containing the part before the
 |      separator, the separator itself, and the part after it.
 |
 |      If the separator is not found, returns a 3-tuple containing two empty
strings
 |      and the original string.
 |
 |  rsplit(self, /, sep=None, maxsplit=-1)
 |      Return a list of the substrings in the string, using sep as the
separator string.
 |
 |        sep
 |          The separator used to split the string.
 |
 |          When set to None (the default value), will split on any whitespace
 |          character (including \n \r \t \f and spaces) and will discard
 |          empty strings from the result.
 |        maxsplit
 |          Maximum number of splits.
 |          -1 (the default value) means no limit.
 |
 |      Splitting starts at the end of the string and works to the front.
 |
 |  rstrip(self, chars=None, /)
 |      Return a copy of the string with trailing whitespace removed.
 |
 |      If chars is given and not None, remove characters in chars instead.
 |
 |  split(self, /, sep=None, maxsplit=-1)
 |      Return a list of the substrings in the string, using sep as the
separator string.
 |
 |        sep
 |          The separator used to split the string.
 |
 |          When set to None (the default value), will split on any whitespace
 |          character (including \n \r \t \f and spaces) and will discard
 |          empty strings from the result.
 |        maxsplit
 |          Maximum number of splits.
 |          -1 (the default value) means no limit.
 |
 |      Splitting starts at the front of the string and works to the end.
 |
```

```
 |      Note, str.split() is mainly useful for data that has been intentionally
 |      delimited.  With natural text that includes punctuation, consider using
 |      the regular expression module.
 |
 |  splitlines(self, /, keepends=False)
 |      Return a list of the lines in the string, breaking at line boundaries.
 |
 |      Line breaks are not included in the resulting list unless keepends is
given and
 |      true.
 |
 |  startswith(self, prefix[, start[, end]], /)
 |      Return True if the string starts with the specified prefix, False
otherwise.
 |
 |      prefix
 |        A string or a tuple of strings to try.
 |      start
 |        Optional start position. Default: start of the string.
 |      end
 |        Optional stop position. Default: end of the string.
 |
 |  strip(self, chars=None, /)
 |      Return a copy of the string with leading and trailing whitespace
removed.
 |
 |      If chars is given and not None, remove characters in chars instead.
 |
 |  swapcase(self, /)
 |      Convert uppercase characters to lowercase and lowercase characters to
uppercase.
 |
 |  title(self, /)
 |      Return a version of the string where each word is titlecased.
 |
 |      More specifically, words start with uppercased characters and all
remaining
 |      cased characters have lower case.
 |
 |  translate(self, table, /)
 |      Replace each character in the string using the given translation table.
 |
 |        table
 |          Translation table, which must be a mapping of Unicode ordinals to
 |          Unicode ordinals, strings, or None.
 |
 |      The table must implement lookup/indexing via __getitem__, for instance a
 |      dictionary or list.  If this operation raises LookupError, the character
```

```
is
 |      left untouched.  Characters mapped to None are deleted.
 |
 |  upper(self, /)
 |      Return a copy of the string converted to uppercase.
 |
 |  zfill(self, width, /)
 |      Pad a numeric string with zeros on the left, to fill a field of the
given width.
 |
 |      The string is never truncated.
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs)
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  maketrans(x, y=<unrepresentable>, z=<unrepresentable>, /)
 |      Return a translation table usable for str.translate().
 |
 |      If there is only one argument, it must be a dictionary mapping Unicode
 |      ordinals (integers) or characters to Unicode ordinals, strings or None.
 |      Character keys will be then converted to ordinals.
 |      If there are two arguments, they must be strings of equal length, and
 |      in the resulting dictionary, each character in x will be mapped to the
 |      character at the same position in y. If there is a third argument, it
 |      must be a string, whose characters will be mapped to None in the result.
```

[58]:
```
sequence
```

[58]:
```
'acgatagcatcggagactagctagcaggcgtatgcagtcgatgcgatgcagtgactg'
```

[60]:
```
sequence.endswith("cta")
```

[60]:
```
False
```

[61]:
```
if sequence.endswith("ctg"):
    print("whatever")
```

```
whatever
```

[ ]:
```

```

[62]:
```
names
```

[62]:
```
'Ana Berta Cilka Dani'
```

[63]:
```
first, second, third, fourth = names.split()
```

[64]:
```
first
```

[64]:
```
'Ana'
```

[65]:
```
third
```

[65]:
```
'Cilka'
```

[66]:
```
t = "Benjamin,75,185"
```

[67]:
```
name, weight, height = t.split(",")
```

[68]:
```
name
```

[68]:
```
'Benjamin'
```

[69]:
```
weight
```

[69]:
```
'75'
```

[70]:
```
height
```

[70]:
```
'185'
```

[72]:
```
f = open("data.csv")
```

[75]:
```
print(f)
```

```
<_io.TextIOWrapper name='data.csv' mode='r' encoding='UTF-8'>
```

[128]:
```
highest_bmi = 0
sum = 0
fingers = 0
for line in open("data.txt", encoding="utf-8"):
    name, weight, height = line.strip().split(",")
    bmi = float(weight) / (float(height) / 100) ** 2
    print(name, weight, height, bmi)
    if bmi > highest_bmi:
        highest_bmi = bmi
        highest_bmi_name = name
    sum = sum + bmi
    fingers = fingers + 1

print("Highest BMI:", highest_bmi, highest_bmi_name)
print("Average:", sum / fingers)
```

```
Ana 65 170 22.49134948096886
Berta 80 168 28.344671201814062
Cilka 60 155 24.97398543184183
Highest BMI: 28.344671201814062 Berta
Average: 25.27000203820825
```

[82]:
```
a, b, c = "2342-12415-35".split("-")
```

[83]:
```

```

[83]:
```
'2342'
```

[84]:
```
b
```

[84]:
```
'12415'
```

[85]:
```
c
```

[85]:
```
'35'
```

[90]:
```
from os import *
```

[91]:
```
getcwd()
```

[91]:
```
'/Users/janez/Desktop'
```

[97]:
```
del open
```

[102]:
```
vrstica
```

[102]:
```
'Cilka,60,155'
```

[104]:
```
line
```

[104]:
```
'\ufeffAna,65,170\n'
```

[105]:
```
print("Ana\nBerta")
```

```
Ana
Berta
```

[112]:
```
"      Ana    Berta  Cillka           \n".strip()
```

[112]:
```
'Ana    Berta  Cillka'
```

[113]:
```
s = "####12 14"
```

[114]:
```
s.strip("#")
```

[114]:
```
'12 14'
```

[ ]:
```

```

[129]:
```
from math import *
```

[130]:
```
sin(3.14)
```

[130]:
```
0.0015926529164868282
```

[131]:
```
log(3)
```

[131]:
```
1.0986122886681098
```

[132]:
```
import math
```

[133]:
```
s = "asf"
```

[134]:
```
t = "asoiasjog"
```

[135]:
```
import math
```

[136]:
```
math
```

[136]:
```
<module 'math' from
'/Users/janez/miniforge3/envs/programiranje/lib/python3.13/lib-
dynload/math.cpython-313-darwin.so'>
```

[137]:
```
math.sin(3.14)
```

[137]:
```
0.0015926529164868282
```

[138]:
```
math.exp(8)
```

[138]:
```
2980.9579870417283
```

[139]:
```
import os
```

[140]:
```
os.getcwd()
```

[140]:
```
'/Users/janez/Desktop'
```

[141]:
```
from math import sin, cos
```

[142]:
```
sin(3.14)
```

[142]:
```
0.0015926529164868282
```

[143]:
```
import os
```

[144]:
```
os.open
```

[144]:
```
<function posix.open(path, flags, mode=511, *, dir_fd=None)>
```

[145]:
```
open("data.csv")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[145], line 1
----> 1 open("data.csv")

FileNotFoundError: [Errno 2] No such file or directory: 'data.csv'
```

[146]:
```
del open
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[146], line 1
----> 1 del open

NameError: name 'open' is not defined
```

[148]:
```
print = 17
```

[151]:
```
del print
```

[152]:
```
dir()
```

[152]:
```
['CLD_CONTINUED',
 'CLD_DUMPED',
 'CLD_EXITED',
 'CLD_KILLED',
 'CLD_STOPPED',
 'CLD_TRAPPED',
 'DirEntry',
 'EX_CANTCREAT',
 'EX_CONFIG',
 'EX_DATAERR',
 'EX_IOERR',
 'EX_NOHOST',
 'EX_NOINPUT',
 'EX_NOPERM',
 'EX_NOUSER',
```

```
'EX_OK',
'EX_OSERR',
'EX_OSFILE',
'EX_PROTOCOL',
'EX_SOFTWARE',
'EX_TEMPFAIL',
'EX_UNAVAILABLE',
'EX_USAGE',
'F_LOCK',
'F_OK',
'F_TEST',
'F_TLOCK',
'F_ULOCK',
'In',
'NGROUPS_MAX',
'O_ACCMODE',
'O_APPEND',
'O_ASYNC',
'O_CLOEXEC',
'O_CREAT',
'O_DIRECTORY',
'O_DSYNC',
'O_EVTONLY',
'O_EXCL',
'O_EXLOCK',
'O_FSYNC',
'O_NDELAY',
'O_NOCTTY',
'O_NOFOLLOW',
'O_NOFOLLOW_ANY',
'O_NONBLOCK',
'O_RDONLY',
'O_RDWR',
'O_SHLOCK',
'O_SYMLINK',
'O_SYNC',
'O_TRUNC',
'O_WRONLY',
'Out',
'POSIX_SPAWN_CLOSE',
'POSIX_SPAWN_DUP2',
'POSIX_SPAWN_OPEN',
'PRIO_DARWIN_BG',
'PRIO_DARWIN_NONUI',
'PRIO_DARWIN_PROCESS',
'PRIO_DARWIN_THREAD',
'PRIO_PGRP',
```

```
'PRIO_PROCESS',
'PRIO_USER',
'P_ALL',
'P_NOWAIT',
'P_NOWAITO',
'P_PGID',
'P_PID',
'P_WAIT',
'RTLD_GLOBAL',
'RTLD_LAZY',
'RTLD_LOCAL',
'RTLD_NODELETE',
'RTLD_NOLOAD',
'RTLD_NOW',
'R_OK',
'SCHED_FIFO',
'SCHED_OTHER',
'SCHED_RR',
'SEEK_CUR',
'SEEK_DATA',
'SEEK_END',
'SEEK_HOLE',
'SEEK_SET',
'ST_NOSUID',
'ST_RDONLY',
'TMP_MAX',
'WCONTINUED',
'WCOREDUMP',
'WEXITED',
'WEXITSTATUS',
'WIFCONTINUED',
'WIFEXITED',
'WIFSIGNALED',
'WIFSTOPPED',
'WNOHANG',
'WNOWAIT',
'WSTOPPED',
'WSTOPSIG',
'WTERMSIG',
'WUNTRACED',
'W_OK',
'X_OK',
'_',
'_1',
'_10',
'_102',
'_104',
```

```
'_109',
'_11',
'_110',
'_112',
'_114',
'_12',
'_13',
'_130',
'_131',
'_136',
'_137',
'_138',
'_14',
'_140',
'_142',
'_144',
'_149',
'_15',
'_16',
'_17',
'_18',
'_19',
'_2',
'_21',
'_25',
'_27',
'_28',
'_3',
'_31',
'_33',
'_35',
'_4',
'_41',
'_42',
'_44',
'_45',
'_46',
'_47',
'_49',
'_5',
'_50',
'_51',
'_53',
'_54',
'_55',
'_56',
'_58',
```

```
'_59',
'_6',
'_60',
'_62',
'_64',
'_65',
'_68',
'_69',
'_7',
'_70',
'_71',
'_73',
'_8',
'_83',
'_84',
'_85',
'_9',
'_91',
'__',
'___',
'__builtin__',
'__builtins__',
'__doc__',
'__loader__',
'__name__',
'__package__',
'__session__',
'__spec__',
'_dh',
'_exit',
'_i',
'_i1',
'_i10',
'_i100',
'_i101',
'_i102',
'_i103',
'_i104',
'_i105',
'_i106',
'_i107',
'_i108',
'_i109',
'_i11',
'_i110',
'_i111',
'_i112',
```

```
'_i113',
'_i114',
'_i115',
'_i116',
'_i117',
'_i118',
'_i119',
'_i12',
'_i120',
'_i121',
'_i122',
'_i123',
'_i124',
'_i125',
'_i126',
'_i127',
'_i128',
'_i129',
'_i13',
'_i130',
'_i131',
'_i132',
'_i133',
'_i134',
'_i135',
'_i136',
'_i137',
'_i138',
'_i139',
'_i14',
'_i140',
'_i141',
'_i142',
'_i143',
'_i144',
'_i145',
'_i146',
'_i147',
'_i148',
'_i149',
'_i15',
'_i150',
'_i151',
'_i152',
'_i16',
'_i17',
'_i18',
```

```
'_i19',
'_i2',
'_i20',
'_i21',
'_i22',
'_i23',
'_i24',
'_i25',
'_i26',
'_i27',
'_i28',
'_i29',
'_i3',
'_i30',
'_i31',
'_i32',
'_i33',
'_i34',
'_i35',
'_i36',
'_i37',
'_i38',
'_i39',
'_i4',
'_i40',
'_i41',
'_i42',
'_i43',
'_i44',
'_i45',
'_i46',
'_i47',
'_i48',
'_i49',
'_i5',
'_i50',
'_i51',
'_i52',
'_i53',
'_i54',
'_i55',
'_i56',
'_i57',
'_i58',
'_i59',
'_i6',
'_i60',
```

```
'_i61',
'_i62',
'_i63',
'_i64',
'_i65',
'_i66',
'_i67',
'_i68',
'_i69',
'_i7',
'_i70',
'_i71',
'_i72',
'_i73',
'_i74',
'_i75',
'_i76',
'_i77',
'_i78',
'_i79',
'_i8',
'_i80',
'_i81',
'_i82',
'_i83',
'_i84',
'_i85',
'_i86',
'_i87',
'_i88',
'_i89',
'_i9',
'_i90',
'_i91',
'_i92',
'_i93',
'_i94',
'_i95',
'_i96',
'_i97',
'_i98',
'_i99',
'_ih',
'_ii',
'_iii',
'_oh',
'a',
```

```
'abort',
'access',
'acos',
'acosh',
'altsep',
'asin',
'asinh',
'atan',
'atan2',
'atanh',
'b',
'bmi',
'c',
'cbrt',
'ceil',
'chdir',
'chflags',
'chmod',
'chown',
'chroot',
'close',
'closerange',
'comb',
'confstr',
'confstr_names',
'copysign',
'cos',
'cosh',
'count',
'cpu_count',
'ctermid',
'curdir',
'defpath',
'degrees',
'device_encoding',
'devnull',
'dist',
'dup',
'dup2',
'e',
'environ',
'environb',
'erf',
'erfc',
'error',
'execl',
'execle',
```

```
'execlp',
'execlpe',
'execv',
'execve',
'execvp',
'execvpe',
'exit',
'exp',
'exp2',
'expm1',
'extsep',
'f',
'fabs',
'factorial',
'fchdir',
'fchmod',
'fchown',
'fdopen',
'fingers',
'first',
'floor',
'fma',
'fmod',
'fork',
'forkpty',
'fourth',
'fpathconf',
'frexp',
'fsdecode',
'fsencode',
'fspath',
'fstat',
'fstatvfs',
'fsum',
'fsync',
'ftruncate',
'fwalk',
'gamma',
'gcd',
'get_blocking',
'get_exec_path',
'get_inheritable',
'get_ipython',
'get_terminal_size',
'getcwd',
'getcwdb',
'getegid',
```

```
'getenv',
'getenvb',
'geteuid',
'getgid',
'getgrouplist',
'getgroups',
'getloadavg',
'getlogin',
'getpgid',
'getpgrp',
'getpid',
'getppid',
'getpriority',
'getsid',
'getuid',
'grantpt',
'height',
'highest_bmi',
'highest_bmi_name',
'hypot',
'inf',
'initgroups',
'isatty',
'isclose',
'isfinite',
'isinf',
'isnan',
'isqrt',
'kill',
'killpg',
'lchflags',
'lchmod',
'lchown',
'lcm',
'ldexp',
'lgamma',
'line',
'linesep',
'link',
'listdir',
'lockf',
'log',
'log10',
'log1p',
'log2',
'login_tty',
'lower_names',
```

```
'lseek',
'lstat',
'major',
'makedev',
'makedirs',
'math',
'minor',
'mkdir',
'mkfifo',
'mknod',
'modf',
'name',
'names',
'nan',
'nextafter',
'nice',
'openpty',
'os',
'pardir',
'path',
'pathconf',
'pathconf_names',
'pathsep',
'perm',
'pi',
'pipe',
'ploscina',
'ploščina',
'popen',
'posix_openpt',
'posix_spawn',
'posix_spawnp',
'pow',
'pread',
'preadv',
'prod',
'prostornina',
'ptsname',
'putenv',
'pwrite',
'pwritev',
'quit',
'radians',
'read',
'readlink',
'readv',
'register_at_fork',
```

```
'remainder',
'remove',
'removedirs',
'rename',
'renames',
'replace',
'rmdir',
's',
'scandir',
'sched_get_priority_max',
'sched_get_priority_min',
'sched_yield',
'second',
'sendfile',
'sep',
'sequence',
'set_blocking',
'set_inheritable',
'setegid',
'seteuid',
'setgid',
'setgroups',
'setpgid',
'setpgrp',
'setpriority',
'setregid',
'setreuid',
'setsid',
'setuid',
'sin',
'sinh',
'spawnl',
'spawnle',
'spawnlp',
'spawnlpe',
'spawnv',
'spawnve',
'spawnvp',
'spawnvpe',
'sqrt',
'stat',
'stat_result',
'statvfs',
'statvfs_result',
'strerror',
'sum',
'sumprod',
```

```
 'supports_bytes_environ',
 'symlink',
 'sync',
 'sysconf',
 'sysconf_names',
 'system',
 't',
 'tan',
 'tanh',
 'tau',
 'tcgetpgrp',
 'tcsetpgrp',
 'terminal_size',
 'third',
 'times',
 'times_result',
 'trunc',
 'truncate',
 'ttyname',
 'ulp',
 'umask',
 'uname',
 'uname_result',
 'unlink',
 'unlockpt',
 'unsetenv',
 'urandom',
 'utime',
 'visina',
 'vrstica',
 'wait',
 'wait3',
 'wait4',
 'waitid',
 'waitid_result',
 'waitpid',
 'waitstatus_to_exitcode',
 'walk',
 'weight',
 'write',
 'writev',
 'x',
 'y']
```

[153]:
```
import math as m
```

[155]:
```
m
```

[155]:
```
<module 'math' from
'/Users/janez/miniforge3/envs/programiranje/lib/python3.13/lib-
dynload/math.cpython-313-darwin.so'>
```

[157]:
```
marjeta = 15
```

[183]:
```
f = open("new-file.txt", "w", encoding="utf-8")
highest_bmi = 0
sum = 0
fingers = 0
for line in open("data.txt", encoding="utf-8-sig"):
    name, weight, height = line.strip().split(",")
    bmi = float(weight) / (float(height) / 100) ** 2
    print(f"{name:15}{weight:8}{height:8}{bmi:5.2f}")
    if bmi > highest_bmi:
        highest_bmi = bmi
        highest_bmi_name = name
    sum = sum + bmi
    fingers = fingers + 1

print("Highest BMI:", highest_bmi, highest_bmi_name)
print("Average:", sum / fingers)
f.close()
```

```
Ana             65      170     22.49
Berta           80      168     28.34
Cilka           60      155     24.97
Highest BMI: 28.344671201814062 Berta
Average: 25.27000203820825
```

[184]:
```
x = 7
```

[185]:
```
s = [3, 5, 18, 20, 30]
t = [1, 2, 3, 6, 3]
```

[190]:
```
s + s
```

[190]:
```
[3, 5, 18, 20, 30, 3, 5, 18, 20, 30]
```

[191]:
```
s + t
```

[191]:
```
[3, 5, 18, 20, 30, 1, 2, 3, 6, 3]
```

[193]:
```
sum = 0
for x in s:
    sum += x
```

```
print(sum / len(s))
```

```
15.2
```

[194]:
```
len("Berta")
```

[194]:
```
5
```

[195]:
```
len([1, 2, 3])
```

[195]:
```
3
```

[196]:
```
len([])
```

[196]:
```
0
```

[199]:
```
for x in []:
    print("Nothing")
```

[200]:
```
s
```

[200]:
```
[3, 5, 18, 20, 30]
```

[201]:
```
18 in s
```

[201]:
```
True
```

[202]:
```
19 in s
```

[202]:
```
False
```

[208]:
```
if x in s:
    print(f"Yes, {s} contains {x}")
```

```
Yes, [3, 5, 18, 20, 30] contains 18
```

[207]:
```
x = 18
```

[209]:
```
s
```

[209]:
```
[3, 5, 18, 20, 30]
```

[210]:
```
s.append(5)
```

[211]:
```
s
```

[211]:
```
[3, 5, 18, 20, 30, 5]
```

```
[212]: s.append(5)

[213]: s

[213]: [3, 5, 18, 20, 30, 5, 5]

[214]: s.append(18)

[215]: s

[215]: [3, 5, 18, 20, 30, 5, 5, 18]

[216]: s.count(5)

[216]: 3

[217]: s.count(18)

[217]: 2

[218]: s.count(19)

[218]: 0

[219]: s

[219]: [3, 5, 18, 20, 30, 5, 5, 18]

[220]: s.sort()

[221]: s

[221]: [3, 5, 5, 5, 18, 18, 20, 30]

[222]: s.sort(reverse=True)

[223]: s

[223]: [30, 20, 18, 18, 5, 5, 5, 3]

[224]: names = ["Ana", "Cilka", "Fanči", "Berta", "Ema", "Dani"]

[225]: names

[225]: ['Ana', 'Cilka', 'Fanči', 'Berta', 'Ema', 'Dani']

[226]: names.sort()
```

[227]:
```
names
```

[227]:
```
['Ana', 'Berta', 'Cilka', 'Dani', 'Ema', 'Fanči']
```

[228]:
```
"Dani" < "Ema"
```

[228]:
```
True
```

[229]:
```
"Benjamin" < "Berta"
```

[229]:
```
True
```

[230]:
```
"Črt" < "Dani"
```

[230]:
```
False
```

[238]:
```
n = 19567876543456789
seen = []
while n != 1 and n not in seen:
    seen.append(n)
    if n % 2 == 0:
        n //= 2
    else:
        n = 3 * n + 1


if n == 1:
    print("Janez won")
else:
    print("Janez lost")
```

```
Janez won
```

[237]:
```
len(seen)
```

[237]:
```
1
```

[239]:
```
names
```

[239]:
```
['Ana', 'Berta', 'Cilka', 'Dani', 'Ema', 'Fanči']
```

[240]:
```
for name in names:
    print(name)
```

```
Ana
Berta
Cilka
Dani
```

```
Ema
Fanči
```

[241]:
```
first, sec, thi, fou, fif, six = names
```

[244]:
```
fou
```

[244]:
```
'Dani'
```

[245]:
```
vrstica = "Ana,65,170"
```

[246]:
```
name, weight, height = vrstica.split(",")
```

[247]:
```
vrstica.split(",")
```

[247]:
```
['Ana', '65', '170']
```

[248]:
```
names
```

[248]:
```
['Ana', 'Berta', 'Cilka', 'Dani', 'Ema', 'Fanči']
```

[249]:
```
names[4]
```

[249]:
```
'Ema'
```

[250]:
```
names[0]
```

[250]:
```
'Ana'
```

[252]:
```
len(names)
```

[252]:
```
6
```

[254]:
```
names[5]
```

[254]:
```
'Fanči'
```

[257]:
```
names[-3]
```

[257]:
```
'Dani'
```

[ ]: