Basis

Invoer	Uitvoer	Beschrijving
`print("één", 1, end="\n")`	'één 1'	Print waarden gescheiden door spatie. Optionele end- attribuut bepaalt waar volgende print komt, bv. end=\n of end=""
`print("one", "two", sep="X")`	'oneXtwo'	Print waarden gescheiden door optionele parameter sep="".
`print('mango\'s')`	'mango's'	Bakslash behandelt volgend teken als string.
`+ - * / **`		Optelling, aftrekking, vermenigvuldiging, deling, machtsverheffing.
`5 // 2`	2	Integer deling: rondt af naar beneden.
`5 % 2`	1	Modulo: rest na deling.
`int() | float() | str()`		Converteert naar type int (afronden naar beneden) | float (eventueel .0 erachter plaasen) | string.
`abs(-1) | round(4.56, 1)`	1 | 4.6	Absolute waarde | afronden, optionele parameter is aantal cijfers na of voor de komma.
`max(1, 2, 5)`	5	Maximum van 2 of meer waarden. Optionele parameter key=func().
`min("x", "ab", key=len)`	'x'	Minimum van 2 of meer waarden. Optionele parameter key=func().
`len("abc")`	3	Lengte van een object.
`sorted(['ab', 'b'], key=len)`	['b', 'ab']	Geeft een gesorteerde versie weer van een sequentie. Optionele parameter key.
`x = input("Tekst: ")`	Tekst:	Slaagt de input (string) van de gebruiker op in variabele x.
`a, b = b, a`		Wisselt de waarden van variabelen a en b om.
`type(a)`		Geeft weer tot welke datatype of klasse een object behoort, bv. str, int, float, ...
`isinstance('5', int)`	False	Geeft True of False. Checkt of object behoort tot het gegeven type of class.
`range(start=0, einde, stap=1)`		Iterable van optionele start (incl.) tot einde (excl.) met optionele stap.
`# Commentaar | """Commentaar"""`		Commentaar op 1 lijn | over meerdere lijnen.
`assert var > 0, "Foutmelding"`		Checkt in programma of aan een voorwaarde voldaan wordt. Zo niet, wordt de optionele boodschap geprint en stopt het programma.

Functions

Invoer	Uitvoer	Beschrijving

`def hello():`
`  print("Hello")`
`hello()`
	Hello	Functie "hello"

`def repeat(function, x):`
`  for i in range(x):`
`  function()`
`repeat(hello, 3)`
	Hello
	Hello
	Hello
		Functie als parameter in een functie

`def functie1(user):`
`  def functie2(naam):`
`  return f"Welkom {naam}"`
`  print(functie2(user.naam))`
`functie1(user)`
	Welkom User1	Nested functies

Format

Invoer	Uitvoer	Beschrijving
`f"Getal: {12}"`	'Getal: 12'	String format.
`f"{'a':<3},{'b':^3},{'c':*>3}"`	'a , b ,**c'	Neemt een minimale aantal karakters als ruimte, wordt opgevuld met witruimte of zelfgekozen karakter. Uitlijning links, midden of rechts.
`f"{1.235:.2f}"`	'1.24'	Specifieert format getal als int (d) of float (f). Specifieert aantal cijfers na de komma, afgerond.
`f"{1.235:^7.2f}"`	' 1.24 '	Combinatie van voorgaande.

Strings

Invoer	Uitvoer	Beschrijving
`s1 = "aardbei" | s2 = "bes"`	'aardbei' | 'Bes'	Initialiseert een string.
`len(s1)`	7	Geeft lengte van string.
`max(s1) | min(s)`	'r' | 'a'	Geeft grootste | kleinste karakter.
`ord('a') | chr(97)`	97 | 'a'	Geeft overeenkomstige ASCII-ordening of -teken
`s[2] | s[2:5:1]`	'r' | 'rdb'	Substring via index of slice (excl. bovengrens, optionele stap).
`s[::-1]`	'iebdraa'	String inverteren.
`s1 + s2`	'aardbeiBes'	Strings samenvoegen (concateneren).
`LET OP: Onderstaande methodes veranderen de originele string niet, maar geven eventueel een bewerkte kopie.`		
`s.strip() | s1.strip('a')`	'rdbei'	Verwijdert witruime of optionele zelfgekozen karakters aan het begin en einde van de string.
`s.lstrip() | s.rstrip()`		Idem als strip(), maar enkel aan begin of einde.
`s2.upper() | s2.lower()`	'BES' | 'bes'	Creëert een versie van een string met alle letters als hoofdletters of kleine letters.
`s1.title() | s1.capitalize()`	'Aardbei'	Maakt eerste letter van elk woord hoofdletter. | Maakt enkel eerste letter hoofdletter.
`s2.swapcase()`	'bES'	Inverteert hoofdletters en kleine letters.
`s1.count('a')`	2	Geeft weer hoe vaak een substring voorkomt.
`s1.find("a", 0, 5)`	0	Geeft laagste index van gegeven substring in de string. -1 indien niet gevonden. Optionele start en einde waar het moet zoeken (excl. einde).
`s1.index("a", 2, 5)`	ValueError	Idem als find(), maar geeft ValueError als substring niet wordt gevonden.
`s.rfind() | s.rindex()`		Idem als find() en index() maar geeft hoogste index.
`s1.replace("a", "X") |` `s1.replace("a", "X", 1)`	'XXrdbei' | 'Xardbei'	Kopie van string waarbij alle instanties van een substring vervangen wordt. Optionele parameter geeft aan hoeveel instanties er vervangen moeten worden.
`s1 in s2`	False	Geeft True of False. Checkt of s1 substring is van s2.
`s.islower() | s.isupper() |` `s.istitle()`		Geeft True of False. Checkt of string enkel kleine letters bevat | enkel hoofdletters bevat | elke woord begint met een hoofdletter.
`s1.startswith('aard') |` `s1.endswith('ei')`	True	Geeft True of False. Checkt of strings start | eindigt met substring.
`s1.startswith(('a', 'b', 'c')) |` `s1.endswith(('x', 'e'))`	True	Geeft True of False. Checkt of string start | eindigt met één van de substrings in de tuple.

`s.isalpha()`		Geeft True of False. Check of string enkel bestaat uit letters A-Za-z (spaties of leestekens -> False).
`s.isnumeric()`		Geeft True of False. Check of string enkel bestaat uit cijfer.
`s.isalnum()`		Geeft True of False. Checkt of string enkel bestaat letters en/of cijfers.
`" ".join([s1, s2]) | s2.join("123")`	'aardbei bes' | '1bes2bes3'	Joint string met sequentie.
`"a b cde".split()`	['a', 'b', 'cde']	Geeft lijst van woorden in string, gesplitst op witruimte.
`"a b cde".split('c')`	['a b ', 'de']	Geeft lijst van woorden in string, gesplitst op optionele parameter.
`s.splitlines()`		Geeft lijst van substrings, gesplitst op newlines.

Lists

Invoer	Uitvoer	Beschrijving
`l = ['1', '23', '456'] | l = list(sequentie)`	['1', '23', '456']	Initialiseert een lijst.
`l_2 = [x for x in range(5) if x % 2 == 0]`	[0, 2, 4]	List comprehension.
`len(l)`	3	Geeft lengte van de lijst (aantal elementen).
`max(l) | min(l, key=len)`	'456' | '1'	Geeft grootste | kleinste element. Optionele key.
`sum(l_2)`	6	Som van elementen in de li jst.
`list(enumerate(["a", "b", "c"]))`	[(0, "a"), (1, "b"), (2,"c")]	Geeft tuples van index en value van een list
`l[2] | l[:2:1]`	'456' | ['1', '23']	Element of sublijst (slice) uit de lijst (excl. bovengrens). Optionele stap.
`l[::-1]`	['456', '23', '1']	Lijst inverteren.
`l + l_2`	['1', '23', '456', 0, 2, 4]	Lijsten samenvoegen (concateneren).
`'1' in l`	True	Geeft True of False. Checkt of de lijst een element bevat.
LET OP: Onderstaande methodes passen de originele lijst aan, in tegenstelling tot bij strings.		
`l.append('7')`	['1', '23', '456', '7']	Voegt nieuw element achteraan toe aan de lijst.
`l.extend(l_2)`	['1', '23', '456', 0, 2, 4]	Maakt list langer door een sequentie aan toe te voegen.
`l.insert(1, 'a')`	['1', 'a', '23', '456']	Voegt element toe aan lijst op specifieke plaats.
`l.remove('23')`	['1', '456']	Verwijdert eerste instantie van element uit de lijst.
`var = l.pop(1)`	'23'	Verwijdert laatste element of optioneel element met gegeven index uit de lijst. Retourneert het element.
`del l[1] | del l[:2]`	['1', '456'] | ['456']	Verwijdert element of sublijst (via slice) uit lijst.
`l.clear()`	[]	Maakt lijst leeg
`l.index('23')`	1	Geeft de index van de eerste instantie van een element uit de lijst.
`l.count('456')`	1	Geeft weer hoe vaak een element voorkomt in de lijst.
`l.sort(key=len, reverse=True)`	['456', '23', '1']	Sorteert elementen in de lijst van laag naar hoog of via optionele key, of optioneel in omgekeerde volgorde.
`l_2.reverse()`	[4, 2, 0]	Lijst inverteren.
`copy.copy(l)`		(import copy) Maakt een ondiepe kopie van de lijst.

`copy.deepcopy(l)`		(import copy) Maakt een diepe kopie van de lijst.
`" ".join(l)`	'1 23 456'	Joint string met elementen uit de lijst.
`"a b cde".split()`	['a', 'b', 'cde']	Geeft lijst van woorden in string, gesplitst op witruimte.
`"a b cde".split('c')`	['a b ', 'de']	Geeft lijst van woorden in string, gesplitst op optionele parameter.
`s.splitlines()`		Geeft lijst van substrings, gesplitst op newlines.
`list(zip([1,2,3],[4,5,6]))`	[(1,4), (2,5), (3,6)]	Maakt tuples (x,y) van waarde x in list1 en y in list 2

Tuples

Invoer	Uitvoer	Beschrijving
`t = ('1', '23', '456')` | `t = tuple(sequentie)`	('1', '23', '456')	Initialiseert een tuple.
`len(t)`	3	Geeft lengte van de tuple (aantal elementen).
`max(t) | min(t, key=len)`	'456' | '1'	Geeft grootste | kleinste element. Optionele key.
`sum((1, 2, 3))`	6	Som van elementen in de tuple.
`t[2] | t[:2:1]`	'456' | ('1', '23')	Element of subtuple (slice) uit de tuple (excl. bovengrens). Optionele stap.
`t[::-1]`	('456', '23', '1')	Tuple inverteren.

Sets

Invoer	Uitvoer	Beschrijving
`s1 = {'1', '23', '456'}` | `s2 = set('156')`	{'1', '23', '456'} | {'6', '1', '5'}	Initialiseert een set.
`len(s1)`	3	Geeft lengte van de set (aantal elementen).
`max(s1) | min(s1, key=len)`	'456' | '1'	Geeft grootste | kleinste element. Optionele key.
`sum({1, 2, 3})`	6	Som van elementen in de set.
`s1.add('a')`	{'a', '1', '23', '456'}	Voegt element toe aan de set.
`s1.update('bbq')`	{'b', '1', 'q', '23', '456'}	Voegt alle elementen van een sequentie toe aan de set.
`s1.remove('1') | s1.discard('1')`	{'23', '456'}	Verwijders een element uit de set. Discard() negeert niet-bestaande elementen. Remove() geeft een error.
`s1.pop()`	'1'	Verwijdert een willekeurig element uit de set en retourneert de waarde.
`s.clear()`	set()	Verwijdert alle elementen uit de set.
`copy.copy(s)`		(import copy) Maakt een ondiepe kopie van de set.
`copy.deepcopy(s)`		(import copy) Maakt een diepe kopie van de set.
`s1.union(s2)`	{'23', '456', '6', '1', '5'}	Unie / alle elementen van beide set.
`s1.intersection(s2)`	{'1'}	Doorsnede / gemeenschappelijke elementen van beide sets.
`s1.difference(s2)`	{'23', '456'}	Geeft een set van alle elementen van s1 , waarvan gemeenschappelijke elementen met s2 verwijderd zijn.

`s1.symmetric_difference(s2)`	{'23', '456', '6', '5'}	Geeft alle elementen die s1 en s2 niet gemeenschappelijk hebben.
`s1.isdisjoint(s2)`	False	Geeft True of False. Checkt of de twee sets enkel verschillende elementen hebben.
`s1.issubset({'1', '23', '456', '7'})`	True	Geeft True of False. Checkt of s1 een subset is van s2.
`s1.issuperset({'1'})`	True	Geeft True of False. Checkt of s2 een subset is van s1.
`s3 = frozenset(('a', 'b', 'c'))`	frozenset({'c', 'a', 'b'})	Creëert een onveranderbare set.

Dictionaries

Invoer	Uitvoer	Beschrijving
`dict = dict()`		Initialiseert een dictionary
`dict = {"Key1": "Value1"}`		Initialiseert een dictionary
`dict["Key1"]`	"Value1"	Returned de value van bepaalde key (key != index).
`"Key1" in dict`	True	Checkt of de key in dictionary is.
`dict["Key2"] = "Value2"`		Geeft een value voor bepaalde key. (Maakt key aan als niet bestaat).
`dict.pop("Key2")`	"Value 2"	Verwijdert de key en de value van dictionary en returned de value.
`len(dict)`	1	Geeft lengte van de set (aantal elementen).
`dict.keys()`	dict_keys(["Key1"])	Geeft een dict_keys object met een list van de keys.
`dict.values()`	dict_values(["Value1"])	Geeft een dict_values object met een list van de values.
`dict.items()`	dict_items([("Key1", "Value1")])	Geeft een dict_items object met een list van tuples (key, value)
`dict.setdefault("Key1", "Value2")`	"Value1"	Checkt of key bestaat en returned de value als het bestaat, anders maakt hij de key en geeft hij de gespecifieerde value aan de key.
`dict = {**dict1, **dict2}`		Maakt van 2 dictionaries één dictionary.

Comprehensions

Invoer	Uitvoer	Beschrijving
`[ x for x in [1, 2, 3] if x % 2 == 0 ]`	[2]	Maakt een list met for loop en if-statement
`( x for x in [1, 2, 3] if x % 2 == 0 )`	(2)	Maakt een tuple met for loop en if-statement
`{ x for x in [1, 2, 3] if x % 2 == 0 }`	{2}	Maakt een set met for loop en if-statement
`{ k: v for k, v in dict.items() if v == "Value1" }`	{"Key1": "Value1"}	Maakt een dictionary met for loop en if-statement

`all([True, True, False])`	False	Return True if all booleans in iterable object are True
`any([True, False, False])`	True	Return True if one boolean in iterable object is True

Classes

Invoer	Uitvoer	Beschrijving
`class Klassenaam:`		Initialiseert een klasse.
`class Klassenaam(Klassenaam):`		Initialiseert een child klasse. Methodes van childs kunnen de parent methode overschrijven. (Niet als parent een abstractclass is)
`def __init__(self, p1, p2):`		Initialisatie / constructor van de klasse. Optioneel parameters p1, p2, ... definiëren die je moet meegeven bij het creëren van een object.
`def __repr__(self):`		Hiermee kan je regelen wat er getoond wordt als je een object van een bepaalde klasse aanroept.
`def __str__(self):`		Hiermee kan je regelen wat er getoond wordt als je een object van een bepaalde klasse print.
`def __eq__(self, other):`		Hiermee kan je bepalen wanneer twee objecten hetzelfde zijn.
`def __add__(self, other):`		Hiermee kan je twee objecten optellen en maakt een nieuwe object als result.
`def __sub__(self, other):`		Hiermee kan je twee objecten aftrekken en maakt een nieuwe object als result.
`def __mul__(self, other):`		Hiermee kan je twee objecten vermenigvuldigen en maakt een nieuwe object als result.
`@property` `def value(self):`		Dit is een getter om een value te krijgen van een object van een bepaalde klasse. Variabele: self.__value | Call: object.value
`@value.setter` `def value(self, value):`		Dit is een setter om een value te veranderen van een object van een bepaalde klasse. Variabele: self.__value | Call: object.value = value
`@staticmethod` `def method():`		Dit is een static methode die opgroept kan worden zonder een object te moeten creëren. Call: klassenaam.method()
`var1 = Klassenaam()`		Creëert object van een klasse.
`type(var1)`	<class 'Klassenaam'>	Geeft weer tot welke datatype of klasse een object behoort.
`var1.__class__`	<class 'Klassenaam'>	Geeft weer tot welke klasse een object behoort.
`isinstance(var1, Klassenaam)`	True	Geeft True of False. Checkt of object behoort tot het gegeven type of klasse.
`var2 = copy(var1)`		(from copy import copy) Maakt een ondiepe kopie van het object zonder onderlinge relatie.
`var3 = deepcopy(var1)`		(from copy import deepcopy) Maakt diepe kopie van object zonder onderlinge relatie.
`from abc import ABC, abstractmethod`		Importeer de abstractclass en abstractmethod.
`class Klassenaam(ABC):`		Maak een klasse een abstractclass.
`@abstractmethode` `def method(self):`		Maakt van een method een abstractmethod.
`super().__init__()`		Gebruikt constructor van parent class.

Iterables

Invoer	Uitvoer	Beschrijving

`def __iter__(self):`	Wordt gebruikt om de class een iterable te maken. iter(Klassenaam)
`def __next__(self):`	Wordt gebruikt om volgende element van een iterable te returnen. next(iterable)
`raise StopIteration()`	Wordt gebruikt in een iterable om aan te tonen dat de iterable op het einde is en dat er geen elementen meer zijn.

Generators

Invoer	Uitvoer	Beschrijving

`def repeat(value):`
`  while True:`
`  yield value`
		Generator functie die gaat onthouden waar hij gestopt is en terug gaan na een yield.

`def __next__(self):`		Wordt gebruikt om volgende element van een iterable te returnen. next(iterable)
`iterable = iter(list)`		Make from a list an iterable
`next(iterable)`		Gives the next element of an iterable
`raise StopIteration()`		Wordt gebruikt in een iterable om aan te tonen dat de iterable op het einde is en dat er geen elementen meer zijn.

Error Handling

Invoer	Beschrijving
`raise ValueError("Error Message")`	Stopt het programma en geeft een ValueError met een message in de terminal.

Testing (pytest module)

Invoer	Beschrijving
`def test_name_function():`	Test functie die altijd met "test_" start.
`assert expected == actual, f"message"`	Geeft een error als expected niet gelijk is aan actual.

`@pytest.mark.parametrize('var1, var2', [`
`  ("value1", 1),`
`  ("value2", 2)`
`])`
`def test_name_function(var1, var2):`
	Setup variables with different values to test each value

`def setup_function ():` `  global var1, var2`	Setup common variables for each test in a function name setup_function

`@pytest.fixture`
`def var1():`
`  return value`
 	
`def test_name_function(var1):`
	Setup variables that can be used by functions.

`def create_task(*, description = "Default Description", due_date = tomorrow(), finished = False):`	Variable factory to make variable with needed values. * is used to force the caller to mention the parameter names.
`pytest.approx(value, abs=value)`	Om een bereik van tolerantie toe te passen op een value.
`with pytest.raises(RuntimeError):`	Checkt of er effectief een RuntimeError gegooid wordt.
`# Arrange`	Comment where you setup your test.
`# Act`	Comment where you do an action you need to test.
`# Assert`	Comment where you test if result of your action does the right thing.

Handige modules (import …)

re module

Invoer	Beschrijving
re.fullmatch(expression, string)	Check if entire string matches expression.
re.match(expression, string)	Check if beginning of string matches expression.
re.findall(expression, string)	Checks all matches in a string.
re.search(expression, string)	Checks for a match in a string.
re.sub(expression, value, string[, iterations])	Replace match with specified value.
re.split(expression, string[, iterations])	Return a list where string has been split at each match.
[] | [1-9] | [A-Za-z] | [abc]	Set of characters.
.	Any Character.
^ | ^[Aa]	Starts with.
$ | [Aa]$	Ends with.
* | a*	Zero or more occurences.
+ | a+	One or more occurences.
? | a?	Zero or one occurences.
{min} | a{1}	Exactly the specified number of occurences.
{min,} | a{1,}	Minimum specified number of occurences or more.
{min,max} | a{1,3}	Minimum to maximum specified number of occurences.
| | this|that	OR.
()	Capture a group.
\d	Digit [0-9].
\D	No digit [^0-9].
\s	White space character.
\S	No white space character.
\w	Word character [A-Za-z1-9_].
\W	No word character [^A-Za-z1-9_].
\n	Newline.
\t	Tab.
\1 | (.+)\1	Match content of first group
\2 | (.+)(.+)\2	Match content of second group
expression = r"\d"	With "r" prefix it means the string is raw. And you don't need to double backslash to point out a escaping like \d

`match.group(number)`	Get the specified group of a match
`flags = re.MULTILINE`	Indicate that regex expressiion is active on multiple line

util module

Invoer	Beschrijving
`group_by(list, key_function)`	Sorteer in een dictionary de list aan de hand van de key_function.
`partition(list, key_function)`	Sorteer de list in een tuple van 2 lists aan de hand van de key_function ([True] , [False]).
`count(list, key_function)`	Telt op hoeveel mensen aan de key_function voorwaarde voldoen.
`indices_of(list, key_function)`	Geeft de indices van de waardes die voldoen aan de key_function
`function(cards, lambda card: card.suit)`	Voorbeeld van een key_function dat kan filteren

itertools module

Invoer	Uitvoer	Beschrijving
`count(start[, step])`	0, 1, 2, 3, 4, 5, 6, 7, 8, ...	Iterator list thats starts at given value.
`pairwise([0,1,2,3,4,5])`	(0, 1), (1, 2), (2, 3), (3, 4), ...	From a list it makes tuples of 2 values from the list.
`cycle("abcde")`	a, b, c, d, e, a, b, c, d, ...	Cycles through an iterable.
`repeat(element[, times])`	element, element, ...	Repeat the element. Can be a certain amount to repeat.
`permutations(iterable, length)`	012, 120, 201, 021, ...	Gives all the possible order of an iterable
`groupby(iterable)`	(key, value), (key, value), ...	Groups the same values with the same key if neighbors

math module

Invoer	Uitvoer	Beschrijving
`sqrt(4)`	2.0	Vierkantswortel
`exp(1)`	2.71828...	Exponentiële / e to de macht.
`log(2)`	0.69314...	Natuurlijk logaritme.
`log10(100)`	2.0	Logartime met 10 als basis.

random module

Invoer	Uitvoer	Beschrijving
`random()`	0.10542...	Random float in bereik [0, 1) dus excl. 1.
`uniform(2.5, 3.7)`	2.8654...	Random float in bereik incl. beide grenzen.

`randint(2, 5)`	2	Random int in bereik incl. beide grenzen.
`randrange(0, 50, 10)`	20	Random int in range exl. bovengrens, optionele stap.
`choice(['a', 'b', 'c'])`	'a'	Random element uit sequentie (lijst, tuple, string, range, ...).
`choices([1, 2, 3], weights=[5, 2, 2], k=2)`	[2, 1]	Random lijst van elementen uit gegeven sequentie. Optionele weights- end k-parameter geven gewichten van keuzes en lengte van genereerde lijst.
`shuffle([1, 2, 3])`	[1, 3, 2]	Shuffelt een sequentie random door elkaar. Wordt uitgevoerd op de originele sequentie!
`seed() | seed(1)`		Maakt de generator volledig random | Genereert telkens dezelfde waarden.

copy module

Invoer	Beschrijving
`b = copy(a)`	Maakt ondiepe kopie van een object (bv. lijst, klasse, ...) zonder onderlinge relatie.
`b = deepcopy(a)`	Maakt diepe kopie van een object (bv. lijst, klasse, ...) zonder onderlinge relatie.