Inleiding tot programmeren

Elementary Programming

Double = getal met 2 kommagetallen (-6.90 / 4.45)
Integer = geheel getal (-10000 / 29874)
Float = kommagetal (3.39468735)

Name	Meaning	Example	Result
+	Addition	34 + 1	35
-	Subtraction	34.0 - 0.1	33.9
*	Multiplication	300 * 30	9000
/	Float Division	1 / 2	0.5
//	Integer Division	1 // 2	0
**	Exponentiation	4 ** 0.5	2.0
%	Remainder	20 % 3	2

Even getal -> Getal % 2 = 0
Oneven -> Getal % 2 = 1

Operator	*Example*	*Equivalent*
+=	i += 8	i = i + 8
-=	f -= 8.0	f = f - 8.0
*=	i *= 8	i = i * 8
/=	i /= 8	i = i / 8
//=	i //= 8	i = i // 8
%=	i %= 8	i = i % 8
**=	i **= 8	i = i ** 8

Binair getal omzetten naar decimaal

- Int(binair_getal, 2)

Input vragen

- Naam = input(“Geef je naam: “)
- Getal = int(input(“Geef getal: “)
    - Kan ook float
- Input van meerdere getallen
    - a,b,c = eval(“Geef 3 getallen gesplitst met komma: “)

Kommagetallen weglaten zonder afronden

- int(4.487648764) = 4

2 Kommagetallen behouden zonder afronden

- int(getal*100)/100

Afronden op 2 kommagetallen

- round(3.348943, **2**) = 3.35
    - 2 kan veranderen in aantal getallen achter komma nodig

Control flow

Operator	*Name*
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal to

Controleren voor even getal

- Even = getal % 2 == 0
    - True/False

If-statement

```
if a > 5:
   #expressies wat er wordt uitgevoerd als het juist is
   print("groot")
else:
   print("klein")
   #expressies wat er wordt uitgevoerd als het fout is
```

Elif-statement (=meerdere If achter elkaar ELIF beter voor python)

```
if a > 5: #(Expressie 1)
   #expressies wat er wordt uitgevoerd als het juist is
   print("groot")
elif a > 4: #(Expressie 2)
   print("middelgroot")
   #expressies wat er wordt uitgevoerd als Expressie 1 fout is
elif a > 3:
   print("gemiddeld")
   #expressies wat er wordt uitgevoerd als Expressie 1 en 2 fout is
else:
   print("klein")
   #expressies wat er wordt uitgevoerd als de rest fout is
```

Logical operators

- **and** -> moeten allebei juist zijn
    - if a > 5 and a % 2 == 0:

      #expressies wat er wordt uitgevoerd als het juist is

      print("groot en even")
- **or** -> 1 van de 2 moet juist zijn
    - if a > 5 or a % 2 == 0:

      #expressies wat er wordt uitgevoerd als het juist is

      print("groot of even")
- **not** -> moet fout zijn
    - if not nummer % 2 == 0 :

      #expressies wat er wordt uitgevoerd als het juist is

      print("oneven")

Import random functies

Random getallen (import random)

- **random.randint(a, b)**
  - geeft willekeurig getal tussen a en b, inclusief a en b
- **random.randrange(a, b)**
  - geeft willekeurig getal tussen a en b, inclusief a, exclusief b

- **random.random()**
  - geeft willekeurig kommagetal tussen 0 en 1, inclusief 0 exclusief 1

Function	Description	Example
abs(x)	Returns the absolute value for x.	abs(-2) is 2
max(x1, x2, ...)	Returns the largest among x1, x2, ...	max(1, 5, 2) is 5
min(x1, x2, ...)	Returns the smallest among x1, x2, ...	min(1, 5, 2) is 1
pow(a, b)	Returns a^b. Same as a ** b.	pow(2, 3) is 8
round(x)	Returns an integer nearest to x. If x is equally close to two integers, the even one is returned.	round(5.4) is 5 round(5.5) is 6 round(4.5) is 4
round(x, n)	Returns the float value rounded to n digits after the decimal point.	round(5.466, 2) is 5.47 round(5.463, 2) is 5.46

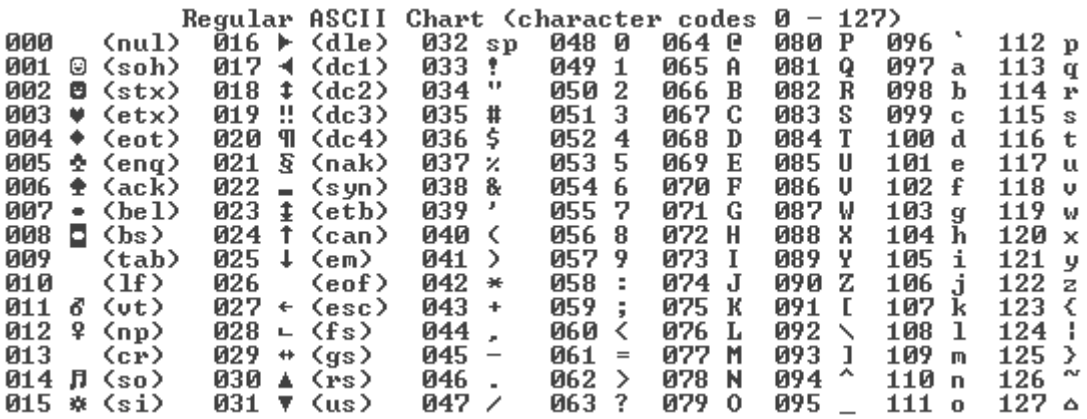
Function	Description	Example
fabs(x)	Returns the absolute value of the argument.	fabs(-2) is 2
ceil(x)	Rounds x up to its nearest integer and returns this integer.	ceil(2.1) is 3 ceil(-2.1) is -2
floor(x)	Rounds x down to its nearest integer and returns this integer.	floor(2.1) is 2 floor(-2.1) is -3
exp(x)	Returns the exponential function of x (e ** x).	exp(1) is 2.71828
log(x)	Returns the natural logarithm of x.	log(2.71828) is 1.0
log(x, base)	Returns the logarithm of x for the specified base.	log10(10, 10) is 1
sqrt(x)	Returns the square root of x.	sqrt(4.0) is 2
sin(x)	Returns the sine of x. x represents an angle in radians.	sin(3.14159 / 2) is 1 sin(3.14159) is 0
asin(x)	Returns the angle in radians for the inverse of sine.	asin(1.0) is 1.57 asin(0.5) is 0.523599
cos(x)	Returns the cosine of x. x represents an angle in radians.	cos(3.14159 / 2) is 0 cos(3.14159) is -1
acos(x)	Returns the angle in radians for the inverse of cosine.	acos(1.0) is 0 acos(0.5) is 1.0472
tan(x)	Returns the tangent of x. x represents an angle in radians.	tan(3.14159 / 4) is 1 tan(0.0) is 0
fmod(x, y)	Returns the remainder of x/y as double.	fmod(2.4, 1.3) is 1.1
degrees(x)	Converts angle x from radians to degrees	degrees(1.57) is 90
radians(x)	Converts angle x from degrees to radians	radians(90) is 1.57

Ascii tabel

- **ord("teken")**
  - geeft het nummer uit Ascii tabel terug
- **chr(nummer)**

- geeft voor het nummer uit Ascii tabel het teken van dat nummer

- **max(“woord”)**
  - geeft letter terug met maximale ord van alle tekens
- **min(“woord”)**
  - geeft letter terug met minimale ord van alle tekens



Regular ASCII Chart (character codes 0 - 127)

000		(nul)	016	►	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001	☺	(soh)	017	◄	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002	☻	(stx)	018	↕	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003	♥	(etx)	019	‼	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	♦	(eot)	020	¶	(dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005	♣	(enq)	021	§	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	♠	(ack)	022	▬	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	•	(bel)	023	↨	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	◘	(bs)	024	↑	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009		(tab)	025	↓	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010		(lf)	026		(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	♂	(vt)	027	←	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012	♀	(np)	028	∟	(fs)	044	,	060	<	076	L	092	\	108	l	124	¦
013		(cr)	029	↔	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014	♫	(so)	030	▲	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	☼	(si)	031	▼	(us)	047	/	063	?	079	O	095	_	111	o	127	⌂

Escape sequence

- sep = wat moet er tussen verschillende delen
- end = wat moet er op het einde
- print(“a”,”b”, end= “\\”) = a b\
- print(“a”,”b”, sep=”\\”) = a\b
- print(“a”,”b”,sep=”\t”, end = “\\)= a	b\

Description	*Escape Sequence*
Backspace	`\b`
Tab	`\t`
Linefeed	`\n`
Carriage return	`\r`
Backslash	`\\`
Single Quote	`\'`
Double Quote	`\"`

Indexing van strings (woord = “tomaten”)

- len(woord)
  - geeft lengte van woord = 7
- woord[0]
  - geeft eerste teken = “t”
- woord[-1]
  - geeft laatste teken = “n”
- woord[2:4]
  - geeft woord van teken 2 tot teken 4, teken 4 er niet bij,teken 2 wel = “ma”
- woord[3:-2]
  - geeft woord van teken 3 tot aan -2(voorlaatste), met teken 3 erbij,het voorlaatste teken niet = “at”
- woord[::-1]
  - geeft het woord omgekeerd = “netamot”

Method	Description
isalnum()	Returns True if all characters in this string are alphanumeric and there is at least one character.
isalpha()	Returns True if all characters in this string are alphabetic and there is at least one character.
isdigit()	Returns True if this string contains only number characters.
isidentifier()	Returns True if this string is a Python identifier.
islower()	Returns True if all characters in this string are lowercase letters and there is at least one character.
isupper()	Returns True if all characters in this string are uppercase letters and there is at least one character.
isspace()	Returns True if this string contains only whitespace characters.

Method	Description
endswith(s1)	Returns True if the string ends with the substring s1.
startswith(s1)	Returns True if the string starts with the substring s1.
find(s1)	Returns the lowest index where s1 starts in this string, or -1 if s1 is not found in this string.
rfind(s1)	Returns the highest index where s1 starts in this string, or -1 if s1 is not found in this string.
count(subtring)	Returns the number of non-overlapping occurrences of this substring.

Formules zijn van de vorm: woord.formule(“”)

Bijvoorbeeld: woord = “tomaten”

- woord.endswith(“en”) = True
- woord.find(“t”) = 0
- woord[::-1].startswith(“n”) = True

Method	Description
capitalize()	Returns a copy of this string with only the first character capitalized.
lower()	Returns a copy of this string with all letters converted to lowercase.
upper()	Returns a copy of this string with all letters converted to uppercase.
title()	Returns a copy of this string with the first letter capitalized in each word.
swapcase()	Returns a copy of this string in which lowercase letters are converted to upper and uppercase to lowercase.
replace(old, new)	Returns a new string that replaces all the occurrence of the old string with a n string.
replace(old, new, n)	Returns a new string that replaces up to n number of the occurrence of the old with a new string.

Method	Description
lstrip()	Returns a string with the leading whitespace characters removed.
rstrip()	Returns a string with the trailing whitespace characters removed.
strip()	Returns a string with the starting and trailing whitespace characters removed.

Formatting

- Print(format(47.349,**"10.2f")** = print(f"{ 47.349**:10.2f**}") = print("**{ :10.2f}**".format(47.349))
  - 10 geeft breedte van output
  - .2f rondt af op 2 getallen na komma
  - Output = 47.35
- print(f"{0.587587**:.2%**}")
  - .2% rondt af op 2 getallen na komma en maakt er een procent van
  - Output = 58.76%
- print(f"{0.587587**:.2e**}")
  - .2e maakt eerst wetenschappelijke notatie dan afronden op 2 kommagetallen
  - Output = 5.88^{e}-01
- :b -> omzetten in binair
- :d -> omzetten in decimaal
- :o -> omzetten in octaal?
- :x -> omzetten in hexadecimaal
- :>20 -> rechts uitlijnen breedte 20
- :<20 -> links uitlijnen breedte 20
- :^20 -> centraal uitlijnen breedte 20
- :^20.2f -> centraal uitlijnen breedte 20 afgerond op 2 kommagetallen

Loops

- **While**

```
i = 0
while i < 100:
    print(i)
    i += 1
```

  - Blijft verdergaan totdat conditie niet meer klopt
- **For-lus**

```
for i in range(0,5):
    print(i)
```

  - Aantal keer is bepaald
  - Aantal keer begint bij 0 en eindigt bij **4** dus eindgrens niet inbegrepen
  - Output = 0 1 2 3 4

- **Break**

```
for i in range(0,5):
    if i == 3:
        break
    print(i)
```

- Break zorgt ervoor dat lus eindigt
- Output = 0 1 2

- **Continue**

```
for i in range(0,5):
    if i == 3:
        continue
    print(i)
```

- Continue zorgt ervoor dat voor deze waarde de lus wordt afgesloten, de lus gaat verder
- Output = 0 1 2 4

Lists and Functions

Functies

```
def max(a,b):
    if a > b:
        return a
    else:
        return b
```

- a en b geen vaste waarde -> **formele paramters**
- max(9,10) = 10 // 9 en 10 zijn **actuele parameters**
- Print niets uit
- Print(max(9,10)) -> print 10

```
def optellen(a,b=4):
    return a+4

print(optellen(4))
```

- Je kan basiswaarde meegeven, dan kan je b leeglaten
- Output = 8

Lijsten

- **Maken**
  - Lijst = list([1,2,3]) = [1,2,3]

- **Methodes**

list	
append(x: object): None	Add an item x to the end of the list.
insert(index: int, x: object): None	Insert an item x at a given index. Note that the first element in the list has index 0.
remove(x: object): None	Remove the first occurrence of the item x from the list.
index(x: object): int	Return the index of the item x in the list.
count(x: object): int	Return the number of times item x appears in the list.
sort(): None	Sort the items in the list.
reverse(): None	Reverse the items in the list.
extend(l: list): None	Append all the items in L to the list.
pop([i]): object	Remove the item at the given position and return it. The square bracket denotes that parameter is optional. If no index is specified, list.pop() removes and returns the last item in the list.

  - In de vorm van lijst.formule()
  - Max(list),min(list),sum(list)
  - Import random -> random.shuffle(list) -> willekeurig element verandert van plaats

- **Indexing van lijsten**
  - Begint bij lijst[0]
  - Lijst [i for i range(0,10)]
    - Lijst met elementen van 0 tot **9**
- “Eend hond tomaat”.split() = [Eend, hond, tomaat]

Objects

Het maken van een klasse

```
class voetballer:
    def __init__(self,leeftijd,ploeg="Barcelona"):
        self.leeftijd=leeftijd
        self.ploeg=ploeg
```

- Initializer zorgt ervoor dat de doorgegeven waarde aan het object wordt gegeven
  - Je kan terug standwaarde instellen
  - Voetballer(30,”Barcelona”)=Voetballer(30)

Set/Get methodes

```
def get_leeftijd(self):
    return self.leeftijd
def set_leeftijd(self,leeftijd):
    self.leeftijd=leeftijd
```

- Speler1 = voetballer(30
- Get methode return leeftijd van het bepaalde object, print niets
- Print(speler1.get_leeftijd())=30
- Set verandert de leeftijd met .set_leeftijd(45)

Assertion errors

```
if isinstance(leeftijd,str):
    raise AssertionError("Geen strings toegelaten")
else:
    self.leeftijd=leeftijd
```

- Isinstance(leeftijd,str) -> True als leeftijd string is

Methodes met andere objecten

```
def is_ouder_dan(self,andere_voetballer):
        return self.get_leeftijd() > andere_voetballer.get_leeftijd()

speler1 = voetballer(45)
speler2 = voetballer(30)
print(speler1.is_ouder_dan(speler2))
```

- De functie print niets
- Print(…) = True