

PH LUZERN PÄDAGOGISCHE HOCHSCHULE

Befehlsübersicht

Beschrieb	Phyton (Textbasiert)	Scratch (Blockbasiert)
Einen Kommentar einfügen	# Ein Kommentar, der bis ans Zeilenende geht """ Ein Kommentar, der über mehrere Zeilen geht. """	Ein Kommentar
Einen Text ausgeben	print ("Hallo Welt")	sage Hallo Welt!
Den Wert einer Variablen ausgeben	name = "Mark" print ("Hallo") print (name) oder	setze name ▼ auf Benutzername sage Hallo sage name
	name = "Mark" print ("Hallo", name)	
Texte kombinieren Texte kann man mit dem '+'-Operator aneinander anfügen.	begruessung = "Hallo" text = begruessung + "Mark" print(text)	setze begruessung auf Hallo setze begruessung auf verbinde begruessung und Mark sage begruessung
Einen Text einlesen	eingabeText = input ("Bitte Text eingeben: ")	frage Bitte Text eingeben: und warte
Eine Zahl einlesen Die Zahl muss mittels int() in eine ganze Zahl oder mittels float() in eine Kommazahl konvertiert werden	eingabeText = input("Bitte eine Zahl eingeben: "); eingabeZahl = int(eingabeText)	frage Bitte eine Zahl eingeben: und warte





Eine Bedingung stellen	bedingung = 10 > 20	40 . 20
Eine Bedingung ist entweder Wahr (True)		10 > 20
oder Falsch (False)		
Eine Bedingung verneinen	bedingung = 10 > 20	nicht 10 > 20
Umdrehen einer Bedingung	gegenteil = not bedingung	110 × 20
for-Schleife	liste = [1, 2, 3, 4]	Gibt es in Scratch nicht
Eine for-Schleife wird verwendet um eine	for zahl in liste:	
Liste, ein Text (ein Text ist nur eine Liste	print("Zahl:", zahl)	
von Buchstaben), o.ä. zu durchzulaufen.	# Hier gehts normal weiter (ACHTUNG: nicht eingerückt)	
Der eingerückte Code wird für jedes		
Listenelment einmal ausgeführt.		
if-Abfragen	wert = 10 # Eine Variable mit einem beliebigen Wert	setze wert ▼ auf 10
Eine if-Abfrage überprüft, ob eine	(hier 10)	Seize Weit V auf 10
Bedingung Wahr oder Falsch ist. Alles was		falls wert > 20 , dann
nach der if-Abfrage mit einem Tab	if wert > 20: # wird NUR ausgeführt, wenn Bedingung	idis Weit 20 , ddill
eingerückt ist gehört zum Körper der if-	wahr ist	sage Der Wert ist grösser als 20!
Abfrage und wird nur ausgeführt, wenn	<pre>print("Der Wert ist größer als 20") # (ACHTUNG:</pre>	Sago Doi Work ist grossor als 201
die Bedingung Wahr ist	Einrückung)	
	# Hier gehts weiter (ACHUTUNG: Nicht eingerückt)	
if-else-Abfragen	wert = 10 # Eine Variable mit einem beliebigen Wert	setze wert ▼ auf 10
Mit einer if-else Abfrage kann man auch	(hier 10)	
auf eine nicht erfüllte Bedingung mit dem		falls wert > 20 , dann
'else'-Zweig reagieren. Es wird immer	if wert > 20: # wird NUR ausgeführt wenn Bedingung	
entweder der if-Zweig oder der else-	erfüllt ist	sage Der Wert ist grösser als 20!
Zweig ausgeführt.	print("Der Wert ist größer als 20")	sonst
	else: # wird NUR ausgeführt wenn Bedingung nicht erfüllt	sage Der Wert ist kleiner oder gleich 20
	ist	
	<pre>print("Wert ist kleiner oder gleich 20.")</pre>	





while-Schleife Eine while-Schleife führt den eingerückten Code aus, solange die Bedingung erfüllt ist.	<pre>zahl = 1 while zahl < 10: # solange die Bedingung wahr ist #wird dieser eingerückte Code wird wiederholt print("Zahl:", zahl) zahl = zahl + 1 # Zähler erhöhen # Hier gehts normal weiter (Achtung: nicht eingerückt)</pre>	setze zahl ▼ auf 1 wiederhole fortlaufend falls zahl < 10 , dann sage zahl āndere zahl ▼ um 1
Eine leere Liste erstellen	liste = []	Neue Liste
Eine Liste mit Einträgen erstellen	liste = [1, 2, 3, 4, 5]	Neue Liste füge 1 zu Liste ▼ hinzu füge 2 zu Liste ▼ hinzu füge 3 zu Liste ▼ hinzu füge 4 zu Liste ▼ hinzu füge 5 zu Liste ▼ hinzu
Einen Eintrag aus der Liste lesen Ein Wert aus der Liste wird über seinen Index, seine Position in der Liste, abgefragt Achtung: Das erste Element hat den Index 0	liste = ["hallo", "test", "welt"] # Liste mit Einträgen erstellen ersterEintrag = liste[0] # der Erste Eintrag aus der Liste wird in der Variablen "ersterEintrag" gespeichert	setze ersterEintrag ▼ auf Element 1 von Liste ▼





		T
Einen Eintrag aus der Liste lesen	liste = ["hallo", "test", "welt"] # Liste mit Einträgen	setze laenge ▼ auf Länge von Liste ▼
Eintrag vom Ende der Liste lesen	erstellen	
Achtung: Das letzte Element hat den	letzterEintrag = liste[-1] # der Letzte Eintrag aus der Liste	setze letzterEintrag ▼ auf Element laenge von Liste ▼
Index -1.	wird in der Variablen "letzterEintrag" gespeichert	
Länge einer Liste ermitteln	liste = [1, 2, 3, 4, 5] # Liste mit Einträgen erstellen	setze laenge ▼ auf Länge von Liste ▼
	laenge = len(liste) # die Länge der Liste wird in der	seize laenge * auf Lange von Liste *
	Variablen "laenge" gespeichert	
Einen Eintrag hinzufügen	liste = [1, 2] # Liste mit Einträgen erstellen	
	liste.append(3) # einen Eintrag anfügen	füge 3 zu Liste ▼ hinzu
Einen Eintrag entfernen	liste = [1, 2, 5, 6] # Liste mit Einträgen erstellen	lösche 6 aus Liste ▼
	liste.remove(6) # löscht den Wert 6 aus der Liste	Justile 6 aus Liste
Eine Funktion definieren	# Neue Funktion mit dem Namen 'sageHallo' definieren	Funktion definieren Funktion aufrufen
Eine Funktion ist ein Block Code, der vom	def sageHallo():	
Haupt-Programm getrennt definiert wird.	<pre>print("Hallo Welt") # Einrückung!</pre>	Definiere sageHallo
Die Funktion wird erst dann ausgeführt,		Dominor Sugariano
wenn man sie über ihren Namen aufruft.	# Funktion aufrufen (keine Einrückung)	
Dafür kann man eine Funktion so oft mal	sageHallo()	sage Hallo Welt sageHallo
will verwenden. Variablen, die innerhalb		
einer Funktion definiert sind, sind nur		
innerhalb der Funktion verwendbar!		
Eine Funktion mit Übergabeparametern	# Funktion definieren	Funktion definieren
definieren	def sageHallo(name, alter):	
Funktion können Parameter (Variablen,	print("Hallo", name)	Definiere sageHallo name alter
die nur innerhalb der Funktion existieren)	<pre>print("Du bist", alter, "Jahre alt")</pre>	
haben. Diese müssen beim Aufrufen		sage verbinde Hallo und name
mitübergeben werden.	# Funktion aufrufen	
	sageHallo("Mark", 22)	sage verbinde verbinde Du bist und alter und Jahre alt
		Funktion aufrufen
		sageHallo name alter
		sageHallo name alter
	l	





Zufallszahl	import random # Modul random importieren	Zufallszahl von 1 bis 100
Will man eine Zufallszahl erzeugen, kann		
man dies mit dem Modul random tun.	# generiert eine Zufallszahl zwischen 1 und 100	
Das Modul random muss einmal	zufallsZahl = random.randint(1,100)	
importiert werden, um verwendet		
werden zu können		
Aktuelles Datum	import datetime # Modul datetime importieren	Gibt es in Scratch nicht
Über das Modul datetime kann man das		
aktuelle Datum abfragen. Das Modul	# aktuelles Datum abfragen	
muss einmal importiert werden, um	heute = datetime.datetime.now()	
verwendet werden zu können.		
	jahr = heute.year	
	monat = heute.month	
	tag = heute.day	
	sekunde = heute.second	
Aktuelle Zeit in Sekunden	import time # Modul time importieren	Gibt es in Scratch nicht
Über das Modul time kann die aktuelle		
Zeit in Sekunden ermittelt werden. Das	# aktuelle Zeit in Sekunden	
Modul muss einmal importiert werden,	sekunden = time.time()	
um verwendet werden zu können.		





Vergleichsmöglichkeiten

Beschrieb	Phyton	Scratch
a ist gleich b	a == b	a = b
a ist nicht gleich b	a != b	nicht a = b
a ist grösser gleich b	a >= b	a > b oder a = b
a ist kleiner gleich b	a <= b	a < b oder a = b
a ist grösser als b	a > b	a > b
a ist kleiner als b	a < b	a < b